ADOBE® DREAMWEAVER® CS5 및 CS5.5

확장

마지막 업데이트 2011 년 6 월 27 일

법적 고지 사항

법적 고지 사항에 대해서는 http://help.adobe.com/ko_KR/legalnotices/index.html을 참조하십시오.

목차

6장: Dreamweaver 문서 객체 모델

7장: 삽입 막대 객체

8장: 브라우저 호환성 확인 문제 API

9장: 명령

10장: 메뉴 및 메뉴 명령

11장: 툴바

12장: 보고서

13장: 태그 라이브러리 및 편집기

14장: 속성 관리자

15장: 부동 패널

16장: 비헤이비어

17장: 서버 비헤이비어

18장: 데이터 소스

19장: 서버 서식

20장: 구성 요소

21장: 서버 모델

22장: 데이터 변환기

23장: C 레벨 확장성

24장: Shared 폴더

1장: 소개

본 **Dreamweaver CS5 확장** 설명서에서는 Dreamweaver의 Extension을 만드는 데 사용할 수 있는 Adobe® Dreamweaver® CS5 프레임워크 및 API(응용 프로그램 프로그래밍 인터페이스)에 대해 설명합니다. 본 **Extending Dreamweaver CS5** 설명서는 다음과 같은 정보를 제공합니다.

- 각 Extension 유형의 작동 방식
- Dreamweaver에서 다양한 객체를 구현하기 위해 호출하는 API 함수
- Dreamweaver의 기능을 구성하는 메뉴, 부동 패널, 서버 비헤이비어 등
- 각 Extension 유형에 대한 간단한 예제
- 다양한 HTML 및 XML 파일에서 태그를 편집하여 명령이나 문서 형식을 추가함으로써 Dreamweaver를 사용자 정의하는 방법

Dreamweaver Extension에서 다양한 지원 작업을 수행하는 데 사용할 수 있는 유틸리티 및 범용 JavaScript™ API에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**를 참조하십시오. 데이터베이스를 사용하는 Extension을 만들려는 경우에는 **Dreamweaver 사용 설명서**에서 데이터베이스 연결 만들기에 대한 항목을 참조하십시오.

Extension 정보

대부분의 Dreamweaver Extension은 HTML과 JavaScript로 작성됩니다. 이 설명서는 Dreamweaver, HTML, XML 및 JavaScript 프로그래밍에 익숙한 사용자를 대상으로 합니다. C Extension을 구현하려는 개발자의 경우에는 C DLL(동적 링크 라이브러리)을 만들고 사용하는 방법에 대해 알고 있어야 합니다. 웹 응용 프로그램을 만들기 위한 Extension을 작성하려면 하나 이상의 플랫폼에서 ASP(Active Server Pages), ASP.NET, PHP(PHP: Hypertext Preprocessor), Adobe® ColdFusion® 또는 JSP(Java Server Pages) 등의 서버측 스크립팅도 능숙하게 수행할 수 있어야 합니다.

Extension 설치

Extension 작성 절차에 익숙해지기 위해 Adobe Exchange 웹 사이트(http://www.adobe.com/go/exchange_kr)에서 제공하는 Extension 및 리소스를 탐색할 수도 있습니다. 기존 Extension을 설치하면 사용자 자신의 Extension을 만드는 데 필요한 몇 가지 도구를 얻을 수 있습니다.

1. Adobe 다운로드 웹 사이트(http://www.adobe.com/go/downloads_kr)에서 Adobe® Extension Manager를 다운로드하여 설치합니다.
2. Adobe Exchange 웹 사이트(http://www.adobe.com/go/exchange_kr)에 로그온합니다.
3. 사용 가능한 Extension 중에서 원하는 것을 선택한 다음 [Download] 링크를 클릭하여 Extension 패키지를 다운로드합니다.
4. Extension 패키지를 Dreamweaver가 설치된 폴더의 Dreamweaver/Downloaded Extensions 폴더에 저장합니다.
5. Extension Manager에서 [파일] > [확장 설치]를 선택합니다. Dreamweaver에서 [명령] > [Extension 관리]를 선택하여 Extension Manager를 시작합니다.

Extension Manager는 Downloaded Extension 폴더에 있는 Extension을 Dreamweaver에 자동으로 설치합니다.

일부 Extension의 경우에는 Extension을 사용하기 전에 Dreamweaver를 다시 시작해야 합니다. Extension을 설치할 때 Dreamweaver가 실행 중이면 응용 프로그램을 종료했다가 다시 시작하라는 메시지가 표시됩니다.

설치 후 Extension에 대한 기본 정보를 보려면 Dreamweaver에서 [명령] > [Extension 관리]를 선택하여 Extension Manager로 이동합니다.

Extension 만들기

Dreamweaver Extension을 만들기 전에 Adobe Exchange 웹 사이트(http://www.adobe.com/go/exchange_kr)에서 만들려는 Extension이 이미 존재하는지 확인하십시오. 필요에 맞는 Extension이 없으면 다음 단계에 따라 Extension을 만듭니다.

- 만들려는 Extension 유형을 결정합니다. Extension 유형에 대한 자세한 내용은 70페이지의 "Dreamweaver Extension의 유형"을 참조하십시오.
- 만들려는 Extension 유형에 대한 설명서를 검토합니다. 해당 Extension 유형을 만드는 데 익숙해지려면 적절한 항목에서 간단한 Extension 예제를 만들어 보는 것이 좋습니다.
- 수정하거나 만들어야 하는 파일을 결정합니다.
- Extension의 UI(사용자 인터페이스)가 있는 경우 UI를 계획합니다.
- 필요한 파일을 만들어 적절한 폴더에 저장합니다.
- Dreamweaver를 다시 시작하여 새로운 Extension을 인식하도록 합니다.
- Extension을 테스트합니다.
- 다른 사람과 공유할 수 있도록 Extension을 패키지화합니다. 자세한 내용은 76페이지의 "Extension Manager 작업"을 참조하십시오.

Extension 개발자를 위한 추가 리소스

다른 개발자와 Extension 작성에 관한 의견을 나누려면 Dreamweaver Extensibility 뉴스 그룹에 가입하십시오. Adobe 웹 사이트(http://www.adobe.com/cfusion/webforums/forum/categories.cfm?forumid=12&catid=190&entercat=y)에서 이 뉴스 그룹에 액세스할 수 있습니다.

Dreamweaver CS5의 새로운 기능

새로운 기능

이러한 각 기능과 관련된 새로운 함수가 유틸리티 API와 JavaScript API에 추가되었습니다. 새로운 함수에 대한 자세한 내용은 Dreamweaver CS5에 새로 추가된 함수를 참조하십시오.

설명서 리소스 센터
Adobe의 설명서를 통해 Dreamweaver 기술을 향상시키십시오.
http://www.adobe.com/support/documentation/buy_books.html에서 전문가가 작성한 최신 내용을 확인할 수 있습니다.

사용되지 않는 함수
Dreamweaver에서는 몇 가지 함수가 사용되지 않습니다. 유틸리티 및 JavaScript API에서 제거된 함수에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**를 참조하십시오.

이 설명서에서 사용된 규칙

이 설명서에는 다음과 같은 인쇄 규칙이 사용되었습니다.

- 코드 글꼴은 코드 부분과 API 리터럴을 나타냅니다. 여기에는 클래스 이름, 메서드 이름, 함수 이름, 유형 이름, 스크립트, SQL 문, HTML과 XML의 태그 및 속성 이름 등이 포함됩니다.
- *기울임체 코드 글꼴*은 코드에서 대체 가능한 항목을 나타냅니다.
- 연속 기호(¬)는 긴 코드 행이 둘 이상의 행으로 분리되었음을 나타냅니다. 이 설명서에서는 여백 제한으로 인해 원래는 연속되어야 하는 코드 행이 나뉘어져 표시됩니다. 코드 행을 복사할 경우에는 연속 기호를 제거한 후 하나의 행으로 입력하십시오.
- 함수의 인수가 중괄호({ })로 묶여 있으면 해당 인수가 선택적이라는 것을 나타냅니다.
- 함수 이름에 접두어 dreamweaver.이 있으면(예: dreamweaver.***funcname***) 코드를 작성할 때 dw.***funcname***으로 줄여 쓸 수 있습니다. 이 설명서에서는 함수를 정의할 때나 색인에서 완전한 dreamweaver. 접두어를 사용합니다. 그러나 많은 예제에서는 보다 짧은 dw. 접두어가 사용됩니다.

이 설명서에는 다음과 같은 명명 규칙이 사용됩니다.

- 개발자—Extension 작성을 담당하는 개발자
- 사용자—Dreamweaver를 사용하는 사람
- 방문자—사용자가 만든 웹 페이지를 보는 사람

2장: Dreamweaver 사용자 정의

Adobe Dreamweaver Extension을 만들고 사용할 수 있을 뿐 아니라 다양한 방법으로 Dreamweaver를 사용자 정의하여 보다 익숙하고 편안한 환경에서 효율적으로 작업할 수 있습니다.

Dreamweaver를 사용자 정의하는 방법

몇 가지 일반적인 방법으로 Dreamweaver를 사용자 정의할 수 있습니다. 이 중 일부는 **Dreamweaver 사용 설명서**에 설명되어 있습니다. 다양한 영역에서 Dreamweaver 환경을 설정할 수 있습니다. [편집] > [환경 설정] 또는 [Dreamweaver] > [환경 설정](Mac OS X의 경우)을 선택하면 나타나는 [환경 설정] 패널에서 액세스 가능성, 코드 색상 표시, 글꼴, 강조 표시, 브라우저에서 미리 보기 등의 기능을 사용자 정의할 수 있습니다. 키보드 단축키 편집기([편집]> [키보드 단축키])를 사용하거나 구성 파일을 편집하여 키보드 단축키를 변경할 수도 있습니다.

기본 문서 사용자 정의

DocumentTypes/NewDocuments 폴더에는 Dreamweaver를 사용하여 만들 수 있는 각 형식의 기본 문서(빈 문서)가 들어 있습니다. [파일] > [새 파일]을 선택하고 [기본 페이지], [동적 페이지] 또는 [기타] 범주에서 항목을 선택하여 빈 문서를 새로 만들면 이 폴더에 있는 적절한 기본 문서를 기반으로 새 문서가 만들어집니다. 특정 형식의 기본 문서에 표시되는 내용을 변경하려면 이 폴더에 있는 해당 문서를 편집하십시오.

참고: 작성 중인 사이트의 모든 페이지에 저작권 정보와 같은 공통 요소를 포함하거나 공통 레이아웃을 적용하려는 경우에는 기본 문서를 변경하는 것보다 템플릿과 라이브러리 항목을 사용하는 것이 좋습니다. 템플릿 및 라이브러리 항목에 대한 자세한 내용은 Dreamweaver 사용 설명서를 참조하십시오.

페이지 디자인 사용자 정의

Dreamweaver에서는 미리 디자인된 CSS(Cascading Style Sheet), 프레임세트 및 페이지 디자인을 다양하게 제공합니다. [파일] > [새 파일]을 선택하여 이러한 디자인을 기반으로 한 페이지를 만들 수 있습니다.

사용 가능한 디자인을 사용자 정의하려면 BuiltIn/css, BuiltIn/framesets, BuiltIn/Templates 및 BuiltIn/TemplatesAccessible 폴더의 파일을 편집하십시오.

참고: [페이지 디자인] 및 [페이지 디자인(액세스 가능성)] 범주에 있는 디자인은 Dreamweaver 템플릿 파일입니다. 템플릿에 대한 자세한 내용은 Dreamweaver 사용 설명서를 참조하십시오.

또한 BuiltIn 폴더의 하위 폴더에 페이지 디자인 파일을 추가하여 사용자 정의 페이지 디자인을 만들 수도 있습니다. [새 문서] 대화 상자에 해당 파일에 대한 설명을 표시하려면 페이지 디자인 파일에 해당하는 디자인 노트 파일을 해당 _notes 폴더에 만듭니다.

대화 상자 모양 사용자 정의

객체, 명령 및 비헤이비어에 대한 대화 상자 레이아웃은 HTML 형식으로 지정되어 Dreamweaver 응용 프로그램 폴더 내의 Configuration 폴더에 HTML 파일로 저장되어 있습니다. Dreamweaver에서 양식을 편집할 때와 같은 방식으로 이러한 양식을 편집할 수 있습니다. 자세한 내용은 **Dreamweaver 사용 설명서**를 참조하십시오.

참고: 다중 사용자 운영 체제에서는 Dreamweaver 구성 파일이 아니라 사용자의 Configuration 폴더에 있는 구성 파일 사본을 편집해야 합니다. 자세한 내용은 72페이지의 “다중 사용자 Configuration 폴더”를 참조하십시오.

대화 상자 모양 변경

1 Dreamweaver에서 [편집] > [환경 설정]을 선택한 다음 [코드 다시 작성] 범주를 선택합니다.

2 [붙여넣기시 양식 항목의 이름 변경] 옵션의 선택을 취소합니다.

이 옵션을 선택하지 않으면 양식의 항목을 복사하여 붙여넣을 때 원래 이름을 유지할 수 있습니다.

3 [확인]을 클릭하여 [환경 설정] 대화 상자를 닫습니다.

4 하드 디스크의 Configuration/Objects, Configuration/Commands 또는 Configuration/Behaviors 폴더에서 적절한 HTM 파일을 찾습니다.

5 Configuration 폴더 이외의 다른 위치에 파일을 복사합니다.

6 이 사본을 Dreamweaver에서 열고 양식을 편집한 후 저장합니다.

7 Dreamweaver를 종료합니다.

8 변경된 파일을 원본이 있는 Configuration 폴더에 다시 복사합니다. 이 경우 원본 파일은 나중에 필요할 때 다시 복원할 수 있도록 백업해 두는 것이 좋습니다.

9 Dreamweaver를 다시 시작하여 변경된 내용을 확인합니다.

대화 상자의 작동 방식은 변경할 수 없고 모양만 변경할 수 있습니다. 또한 대화 상자에 포함되는 양식 요소의 유형과 이름은 계속 동일하게 유지해야 Dreamweaver가 대화 상자에서 얻는 정보를 원래와 동일하게 계속 사용할 수 있습니다.

예를 들어 주석 객체는 대화 상자의 텍스트 영역에서 텍스트 입력 정보를 얻고 간단한 JavaScript 함수를 사용하여 해당 텍스트를 HTML 주석으로 바꾼 다음 문서에 삽입합니다. 이 대화 상자를 표현하는 양식은 Configuration/Objects/Invisibles 폴더의 Comment.htm 파일에 있습니다. 이 파일을 열어 텍스트 영역의 크기 및 기타 속성을 변경할 수 있지만 textarea 태그 전체를 제거하거나 name 속성의 값을 변경하면 주석 객체가 제대로 동작하지 않습니다.

기본 파일 유형 변경

기본적으로 Dreamweaver에서는 인식된 모든 파일 유형이 [파일] > [열기] 대화 상자에 표시됩니다. 이 대화 상자의 팝업 메뉴를 사용하여 특정 유형의 파일만 표시되도록 할 수 있습니다. 대부분의 작업에서 특정 파일 유형(예: ASP 파일)만 주로 사용하는 경우에는 기본적으로 표시되는 파일 유형을 변경할 수 있습니다. Dreamweaver Extensions.txt 파일의 첫 번째 행에 지정된 파일 유형이 기본 유형으로 설정됩니다.

참고: [파일] > [열기] 대화 상자에 Dreamweaver에서 열 수 없는 파일을 포함하여 모든 파일 유형이 나타나도록 하려면 [모든 파일(*.*)]을 선택합니다. [모든 문서]는 이와 달리 Dreamweaver에서 열 수 있는 파일만 표시합니다.

Dreamweaver 파일 > 열기 대화 상자의 기본 파일 유형 변경

1 Configuration 폴더에 있는 Extensions.txt 파일의 백업 사본을 만듭니다.

2 텍스트 편집기에서 Extensions.txt를 엽니다.

3 새 기본값으로 사용할 파일 유형에 해당하는 행을 잘라낸 다음 해당 행이 파일의 첫 번째 행이 되도록 파일의 처음에 붙여넣습니다.

4 파일을 저장합니다.

5 Dreamweaver를 다시 시작합니다.

새 기본값을 확인하려면 [파일] > [열기]를 선택하고 파일 유형 팝업 메뉴를 확인합니다.

파일 > 열기 대화 상자의 메뉴에 새 파일 유형 추가

1 Configuration 폴더에 있는 Extensions.txt 파일의 백업 사본을 만듭니다.

2 텍스트 편집기에서 Extensions.txt를 엽니다.

3 각각의 새 파일 유형에 대한 새 행을 추가합니다. 새 파일 유형의 파일 확장명을 쉼표로 구분하여 대문자로 입력합니다. 그런 다음 콜론을 추가하고 [파일] > [열기] 대화 상자에 나타나는 파일 유형에 대해 팝업 메뉴에 표시할 설명을 간단히 입력합니다.

예(JPEG 파일의 경우): JPG,JPEG,JFIF:JPEG Image Files

4 파일을 저장합니다.

5 Dreamweaver를 다시 시작합니다.

변경된 내용을 확인하려면 [파일] > [열기]를 선택하고 파일 유형 팝업 메뉴를 클릭합니다.

타사 태그 해석 방식 사용자 정의

ASP, Adobe ColdFusion, JSP 및 PHP와 같은 서버측 기술은 HTML 파일에 HTML이 아닌 특수한 코드를 사용합니다. 서버에서는 이 코드를 기반으로 HTML 내용을 만들어 제공합니다. Dreamweaver에서는 HTML이 아닌 태그가 있을 경우 Dreamweaver에서 HTML이 아닌 태그를 읽고 표시하는 방법이 정의된 타사 태그 파일의 정보와 해당 태그를 비교합니다.

예를 들어 ASP 파일은 일반 HTML 외에도 서버에서 해석할 ASP 코드를 포함합니다. ASP 코드는 HTML 태그와 유사하지만 <%로 시작하고 %>로 끝나는 한 쌍의 구분 기호로 표시됩니다. Dreamweaver Configuration/ThirdPartyTags 폴더에는 ASP 코드를 비롯한 다양한 타사 태그의 형식을 설명하고 Dreamweaver에서 해당 코드가 표시되는 방식을 정의하는 Tags.xml이라는 파일이 포함되어 있습니다. 이 Tags.xml에 ASP 코드가 지정된 방식 때문에 Dreamweaver에서는 구분 기호 사이에 있는 내용이 해석되지 않습니다. 대신 [디자인] 뷰에는 해당 내용이 ASP 코드임을 나타내는 아이콘이 표시됩니다. 사용자 고유의 태그 데이터베이스 파일을 사용하여 Dreamweaver에서 태그를 읽고 표시하는 방식을 정의할 수 있습니다. Dreamweaver에서의 태그 표시 방식을 지정하려면 각 태그 세트에 대해 새 태그 데이터베이스 파일을 만듭니다.

참고: 이 단원에서는 Dreamweaver에서 사용자 정의 태그가 표시되는 방식을 정의하는 방법에 대해 설명합니다. 그러나 사용자 정의 태그의 내용이나 속성을 편집할 수 있도록 하는 방법에 대해서는 설명하지 않습니다. 속성 관리자를 만들어 사용자 정의 태그의 속성을 검사하고 변경하는 방법에 대한 자세한 내용은 201페이지의 "속성 관리자"를 참조하십시오.

각 태그 데이터베이스 파일은 하나 이상의 사용자 정의 태그에 대해 이름, 유형, 내용 모델, 렌더링 체계 및 아이콘을 정의합니다. 만들 수 있는 태그 데이터베이스 파일의 수에는 제한이 없지만 해당 파일은 모두 Configuration/ThirdPartyTags 폴더에 있어야 Dreamweaver에서 읽고 처리할 수 있습니다. 태그 데이터베이스 파일의 확장명은 .xml입니다.

서로 관련되지 않은 여러 사이트를 동시에 작업하고 있는 프리랜서 개발자의 경우 특정 사이트에 대한 모든 태그 사양을 한 파일에 저장할 수 있습니다. 그런 다음 사이트 관리자에게 전달할 사용자 정의 아이콘 및 속성 관리자와 함께 이 태그 데이터베이스 파일을 포함하기만 하면 됩니다.

태그 사양은 tagspec이라는 XML 태그로 정의합니다. 예를 들어 다음 코드에서는 happy라는 태그에 대한 사양을 설명합니다.

```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false" content_model=
"marker_model" icon="happy.gif" icon_width="18" icon_height="18"></tagspec>
```

tagspec을 사용하여 다음 두 종류의 태그를 정의할 수 있습니다.

- 일반 HTML 스타일 태그

  happy 태그 예제는 일반 HTML 스타일 태그입니다. 이 예제는 열기 태그인 <happy>로 시작하여 닫기 태그인 </happy>로 끝나며, 열기 태그와 닫기 태그 사이에는 데이터가 포함되어 있습니다.

- 문자열 구분 태그

  문자열 구분 태그는 시작 문자열과 종료 문자열로 구성되며, 태그 사이에 내용이 없고 닫기 태그가 없다는 점에서 빈 HTML 태그(예: img)와 비슷합니다. happy 태그를 문자열 구분 태그로 사용할 경우에는 태그 사양에 start_string 및 end_string 속성이 포함됩니다. ASP 태그는 문자열 구분 태그입니다. 이 태그는 <% 문자열로 시작하고 %> 문자열로 끝나며 닫기 태그가 없습니다.

아래에서는 tagspec 태그에 사용할 수 있는 속성과 올바른 값에 대해 설명합니다. 별표(*)로 표시된 속성은 문자열 구분 태그에서는 무시됩니다. 선택적 속성은 속성 목록에서 중괄호({})로 표시됩니다. 중괄호로 표시되지 않은 모든 속성은 필수 속성입니다.

<tagspec>

설명

타사 태그에 대한 정보를 제공합니다.

속성

tag_name, {tag_type}, {render_contents}, {content_model}, {start_string}, {end_string}, {detect_in_attribute}, {parse_attributes}, icon, icon_width, icon_height, {equivalent_tag}, {is_visual}, {server_model}

- tag_name은 사용자 정의 태그의 이름입니다. 문자열 구분 태그의 경우 tag_name은 지정된 속성 관리자를 해당 태그에 사용할 수 있는지 여부를 결정하는 데에만 사용됩니다. 속성 관리자의 첫 번째 행에 이 태그 이름이 포함되어 있고 양쪽에 별표가 표시되어 있다면 해당 유형의 태그에 속성 관리자를 사용할 수 있습니다. 예를 들어 ASP 코드에 대한 태그 이름이 ASP인 경우 ASP 코드를 검사할 수 있는 속성 관리자의 첫 번째 행에는 *ASP*가 포함되어야 합니다. 속성 관리자 API에 대한 자세한 내용은 201페이지의 “속성 관리자”를 참조하십시오.
- tag_type은 해당 태그가 빈 태그(예: img 태그)인지, 아니면 열기 태그와 닫기 태그 사이에 내용을 포함하는 태그(예: code 태그)인지를 결정합니다. 문자열 구분 태그가 아닌 일반 태그에는 이 속성이 필수 사항입니다. 문자열 구분 태그의 경우에는 태그가 항상 비어 있기 때문에 이 속성이 무시됩니다. 유효한 값은 "empty" 및 "nonempty"입니다.
- render_contents는 [디자인] 뷰에 태그의 내용을 표시할 것인지, 아니면 지정된 아이콘을 대신 표시할 것인지를 결정합니다. 이 속성은 비어 있지 않은 태그의 경우 필수 사항이지만 비어 있는 태그의 경우에는 무시됩니다. 비어 있는 태그에는 내용이 없기 때문입니다. 이 속성은 속성 외부에 나타나는 태그에만 적용됩니다. 다른 태그의 속성 값 내부에 나타나는 태그에 대한 내용은 렌더링되지 않습니다. 유효한 값은 "true" 및 "false"입니다.
- content_model은 태그에 포함될 수 있는 내용의 종류와 태그가 나타날 수 있는 HTML 파일 내의 위치를 지정합니다. 유효한 값은 "block_model", "head_model", "marker_model" 및 "script_model"입니다.
  - block_model은 태그가 div 및 p와 같은 블록 수준 요소를 포함할 수 있으며 body 섹션이나 div, layer 또는 td 등의 다른 본문 내용 태그 내에만 나타날 수 있도록 지정합니다.
  - head_model은 태그가 텍스트 내용을 포함할 수 있으며 head 섹션에만 나타날 수 있도록 지정합니다.
  - marker_model은 태그가 유효한 HTML 코드를 포함할 수 있으며 HTML 파일의 모든 위치에 나타날 수 있도록 지정합니다. Dreamweaver의 HTML 유효성 검사기에서는 marker_model로 지정된 태그가 무시됩니다. 그러나 이러한 태그의 내용은 유효성 검사기에서 무시되지 않습니다. 따라서 이 태그 자체는 어디에나 나타날 수 있지만 태그의 내용은 태그의 위치에 따라 잘못된 HTML로 평가될 수 있습니다. 예를 들어 일반 텍스트는 문서의 head 섹션에서 유효한 head 요소의 외부에는 나타날 수 없으므로 일반 텍스트를 포함하는 marker_model 태그는 head 섹션에 포함할 수 없습니다. 일반 텍스트가 있는 사용자 정의 태그를 head 섹션에 포함하려면 태그의 내용 모델을 marker_model 대신 head_model로 지정하십시오. marker_model은 p 또는 div와 같은 블록 수준 요소 내(예: 단락 내)에 인라인으로 표시되어야 하는 태그에 사용합니다. 태그 자체가 고유의 단락으로 표시되어 태그의 앞과 뒤 행에서 줄이 바뀌어야 하는 경우에는 이 모델을 사용하면 안 됩니다.
  - script_model은 태그가 문서의 열기 HTML 태그와 닫기 HTML 태그 사이의 어디에나 존재할 수 있도록 합니다. Dreamweaver에서는 이 모델을 사용하는 태그가 있을 경우 태그의 내용이 모두 무시됩니다. 이 태그는 Dreamweaver에서 파싱할 수 없는 일부 ColdFusion 태그와 같은 마크업에 사용됩니다.
- start_string은 문자열 구분 태그의 시작을 표시하는 구분 기호를 지정합니다. 문자열 구분 태그는 문서에서 주석이 나타날 수 있는 모든 위치에 나타날 수 있습니다. Dreamweaver에서는 이 태그를 파싱하거나 start_string과 end_string 사이의 엔터티 또는 URL을 디코딩하지 않습니다. end_string이 지정된 경우 이 속성은 필수 사항입니다.
- end_string은 문자열 구분 태그의 끝을 표시하는 구분 기호를 지정합니다. start_string이 지정된 경우 이 속성은 필수 사항입니다.
- detect_in_attribute는 문자열이 속성 이름 또는 값 내부에 나타나는 경우에도 start_string과 end_string 사이(또는 두 문자열이 정의되지 않은 경우 열기 태그와 닫기 태그 사이)의 모든 내용을 무시할지 여부를 지정합니다. 문자열 구분 태그의 경우에는 일반적으로 이 속성을 "true"로 설정해야 합니다. 기본값은 "false"입니다. 예를 들어 ASP 태그는 속성 값 내부에 나타나기도 하

고 인용 부호(")를 포함하기도 합니다. ASP 태그 사양에는 detect_in_attribute="true"로 지정되기 때문에 Dreamweaver에서는 ASP 태그가 속성 값 내부에 나타날 경우 내부 인용 부호를 포함하여 ASP 태그를 무시합니다.

- parse_attributes는 태그의 속성을 파싱할지 여부를 지정합니다. 이 속성이 "true"(기본값)로 설정되어 있으면 Dreamweaver에서 이 속성이 파싱되고, "false"로 설정되어 있으면 인용 부호 외부에서 다음에 나타나는 닫기 꺾쇠 괄호까지의 모든 내용이 무시됩니다. 예를 들어 Dreamweaver에서 속성 이름/값 쌍으로 파싱할 수 없는 <cfif a is 1>의 cfif와 같은 태그에 대해서는 이 속성을 "false"로 설정해야 합니다.
- icon은 태그와 연관된 아이콘 파일의 경로 및 이름을 지정합니다. 이 속성은 빈 태그와 빈 태그는 아니지만 [문서] 윈도우의 [디자인] 뷰에 내용이 표시되지 않는 태그의 경우에 필수 사항입니다.
- icon_width는 아이콘의 폭을 픽셀 단위로 지정합니다.
- icon_height는 아이콘의 높이를 픽셀 단위로 지정합니다.
- equivalent_tag는 특정 ColdFusion 양식 관련 태그에 해당하는 간단한 HTML 태그를 지정합니다. 이 태그는 다른 태그와 함께 사용할 수 없습니다.
- is_visual은 태그가 페이지에 직접적인 시각적 영향을 주는지 여부를 지정합니다. 예를 들어 ColdFusion 태그 cfgraph는 is_visual에 대한 값을 지정하지 않습니다. 따라서 기본적으로 "true"로 설정됩니다. ColdFusion 태그 cfset의 경우에는 is_visual이 "false"로 설정되도록 지정됩니다. 서버 마크업 태그의 가시성은 [환경 설정] 대화 상자의 [보이지 않는 요소] 범주에 의해 제어되며, 가시적 서버 마크업 태그의 가시성은 비가시적 서버 마크업 태그의 가시성과는 별개로 설정될 수 있습니다.
- server_model은 선택 사항으로, 특정 서버 모델에 속한 페이지에만 tagspec 태그가 적용되도록 지정합니다. server_model을 지정하지 않으면 tagspec 태그가 모든 페이지에 적용됩니다. 예를 들어 ASP 및 JSP 태그의 구분 기호는 동일하지만 JSP에 대한 tagspec 태그는 "JSP"의 server_model을 지정하므로 Dreamweaver에서는 JSP 페이지에 적절한 구분 기호가 포함된 코드가 있으면 JSP 아이콘이 표시됩니다. JSP가 아닌 페이지에 이러한 코드가 있으면 ASP 아이콘이 표시됩니다.

내용

없음(빈 태그)

컨테이너

없음

예제

```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false" content_model=
"marker_model" icon="happy.gif" icon_width="18" icon_height="18"></tagspec>
```

디자인 뷰에 사용자 정의 태그가 표시되는 방식

[문서] 윈도우의 [디자인] 뷰에 사용자 정의 태그가 표시되는 방식은 tagspec 태그의 tag_type 및 render_contents 속성 값에 따라 달라집니다. tag_type의 값이 "empty"이면 icon 속성에 지정된 아이콘이 표시됩니다. tag_type의 값이 "nonempty"이지만 render_contents의 값이 "false"이면 아이콘은 빈 태그의 경우와 동일하게 표시됩니다. 다음 예제에서는 앞에서 정의한 happy 태그의 인스턴스가 HTML에 어떻게 나타나는지 보여 줍니다.

```
<p>This is a paragraph that includes an instance of the <code>happy</code>
tag (<happy>Joe</happy>).</p>
```

태그 사양에 render_contents가 "false"로 설정되어 있으므로 happy 태그의 내용(Joe)이 렌더링되지 않습니다. 대신 시작 태그, 종료 태그 및 태그 내용이 단일 아이콘으로 나타납니다.

render_contents 값이 "true"인 비어 있지 않은 태그의 경우에는 [디자인] 뷰에 아이콘이 나타나지 않습니다. 대신 열기 태그와 닫기 태그 사이의 내용(예: <mytag>This is the content between the opening and closing tags</mytag>에서 태그 사이의 텍스트)이 표시됩니다. [보기] > [시각 도구] > [보이지 않는 요소]가 활성화되어 있는 경우, [강조 표시] 환경 설정에 지정된 타사 태그 색상으로 내용이 강조 표시됩니다. 강조 표시는 태그 데이터베이스 파일에 정의된 태그에만 적용됩니다.

타사 태그의 강조 표시 색상 변경

1 [편집] > [환경 설정]을 선택하고 [강조 표시] 범주를 선택합니다.

2 [타사 태그] 색상 상자를 클릭하여 색상 선택기를 표시합니다.

3 색상을 선택하고 [확인]을 클릭하여 [환경 설정] 대화 상자를 닫습니다. 색상 선택에 대한 자세한 내용은 **Dreamweaver 사용 설명서**를 참조하십시오.

타사 태그 덮어쓰기 방지

Dreamweaver에서는 특정 종류의 HTML 코드 오류가 수정됩니다. 자세한 내용은 **Dreamweaver 사용 설명서**를 참조하십시오. 기본적으로 Dreamweaver에서는 .asp(ASP), .cfm(ColdFusion), .jsp(JSP) 및 .php(PHP)를 포함하여 특정 파일 확장명을 가진 파일에서 HTML이 변경되지 않도록 합니다. 이 기본 설정은 Dreamweaver에서 HTML이 아닌 태그에 포함된 코드를 실수로 수정하지 않도록 하기 위한 것입니다. Dreamweaver의 기본적인 재작성 비헤이비어를 변경하면 HTML 파일을 열 때 HTML을 다시 작성할 수 있습니다. 또한 Dreamweaver에서 다시 작성하지 않는 파일 유형 목록에 다른 파일 유형을 추가할 수 있습니다.

Dreamweaver에서는 속성 관리자에 특정 특수 문자를 입력하면 이 문자가 숫자 값으로 바뀌어 인코딩됩니다. 일반적으로 Dreamweaver에서 이와 같이 인코딩하면 특수 문자가 플랫폼과 브라우저에 관계없이 올바르게 표시될 가능성이 크므로 이 방법을 사용하는 것이 가장 좋습니다. 그러나 이러한 인코딩은 타사 태그와 충돌할 수 있으므로 타사 태그가 있는 파일을 처리할 때에는 Dreamweaver의 인코딩 비헤이비어를 변경해야 하는 경우도 있습니다.

Dreamweaver가 보다 많은 파일 유형에서 HTML을 다시 작성할 수 있도록 설정

1 [편집] > [환경 설정]을 선택하고 [코드 다시 작성] 범주를 선택합니다.

2 다음 옵션 중 하나를 선택합니다.

- 잘못 중첩되거나 닫히지 않은 태그 수정
- 불필요한 닫기 태그 제거

3 다음 중 하나를 수행합니다.

- [코드 재작성 안함: 다음 확장명이 있는 파일] 옵션에 있는 확장명 목록에서 확장명을 하나 이상 삭제합니다.
- [코드 재작성 안함: 다음 확장명이 있는 파일] 옵션의 선택을 취소합니다. 이 옵션의 선택을 취소하면 Dreamweaver가 모든 유형의 파일에서 HTML을 다시 작성할 수 있습니다.

Dreamweaver가 다시 작성하지 못하는 파일 유형 추가

1 [편집] > [환경 설정]을 선택하고 [코드 다시 작성] 범주를 선택합니다.

2 다음 옵션 중 하나를 선택합니다.

- 잘못 중첩되거나 닫히지 않은 태그 수정
- 불필요한 닫기 태그 제거

3 [코드 재작성 안함: 다음 확장명이 있는 파일] 옵션이 선택되어 있는지 확인하고 텍스트 필드의 목록에 새 파일 확장명을 추가합니다.

새 파일 유형이 [파일] > [열기] 대화 상자의 파일 유형 팝업 메뉴에 나타나지 않는 경우 Configuration/Extensions.txt 파일에 해당 확장명을 추가할 수 있습니다. 자세한 내용은 5페이지의 “기본 파일 유형 변경”을 참조하십시오.

Dreamweaver 인코딩 옵션 해제

1 [편집] > [환경 설정]을 선택하고 [코드 다시 작성] 범주를 선택합니다.

2 [특수 문자] 옵션 중 하나 또는 둘 모두의 선택을 취소합니다.

그 밖의 [코드 다시 작성] 환경 설정에 대한 자세한 내용은 **Dreamweaver 사용 설명서**를 참조하십시오.

다중 사용자 환경에서 Dreamweaver 사용자 정의

Microsoft® Windows® XP, Windows Vista 또는 Mac OS® X와 같은 다중 사용자 운영 체제에서도 Dreamweaver를 사용자 정의할 수 있습니다. Dreamweaver에서는 각 사용자의 사용자 정의 구성이 다른 사용자의 구성에 영향을 주지 않도록 합니다. 다중 사용자 운영 체제에서 Dreamweaver를 처음 실행하면 사용자의 Configuration 폴더에 구성 파일이 복사됩니다. 대화 상자와 패널을 사용하여 Dreamweaver를 사용자 정의하면 응용 프로그램에서는 Dreamweaver 구성 파일 대신 사용자의 구성 파일을 수정합니다. 다중 사용자 환경에서 Dreamweaver를 사용자 정의하려면 Dreamweaver 구성 파일이 아니라 해당 사용자의 구성 파일을 편집합니다. 대부분의 사용자에게 적용되는 변경을 하려면 Dreamweaver 구성 파일을 편집합니다. 그러나 해당되는 사용자 구성 파일이 이미 있는 사용자에게는 이 변경 내용이 적용되지 않습니다. 따라서 모든 사용자에게 영향을 주는 변경 작업을 수행하려면 Extension Manager를 사용하여 Extension을 만들고 설치합니다.

참고: 이전의 다중 사용자 운영 체제(Windows 98, Windows ME 및 Mac OS 9.x)에서는 모든 사용자가 단일 세트의 Dreamweaver 구성 파일을 공유합니다.

사용자의 Configuration 폴더 위치는 사용자의 플랫폼에 따라 다릅니다.

Windows XP 플랫폼에서는 다음 위치가 사용됩니다.

하드 디스크: \Documents and Settings\username\Application Data\Adobe\Dreamweaver CS5\Configuration

참고: 이 폴더는 숨겨진 폴더 내에 있을 수 있습니다.

Windows Vista 플랫폼에서는 다음 위치가 사용됩니다.

하드 디스크: \Users***username***\AppData\Roaming\Adobe\Dreamweaver CS5\Configuration

Mac OS X 플랫폼에서는 다음 위치가 사용됩니다.

하드 디스크: \Users/username/Library/Application Support/Adobe/Dreamweaver CS5/Configuration

참고: 다중 사용자 운영 체제의 모든 사용자가 사용할 수 있는 Extension을 설치하려면 Administrator(Windows) 또는 root(Mac OS X)로 로그인해야 합니다.

Dreamweaver를 처음 실행하면 일부 구성 파일만 사용자의 Configuration 폴더에 복사됩니다. 이때 복사되는 파일은 Configuration 폴더의 version.xml 파일에 지정되어 있습니다. 응용 프로그램 내에서 Dreamweaver를 사용자 정의하면 구성 파일이 사용자의 Configuration 폴더에 자동으로 복사됩니다. 예를 들어 [코드 단편] 패널에서 미리 디자인된 코드 단편 중 하나를 수정하면 구성 파일이 자동으로 복사됩니다. 사용자의 Configuration 폴더에 있는 파일 버전은 항상 Dreamweaver Configuration 폴더의 버전보다 우선합니다. 구성 파일을 사용자 정의하려면 해당 구성 파일이 사용자의 Configuration 폴더에 있어야 합니다. Dreamweaver에서 해당 파일이 아직 복사되지 않은 경우에는 파일을 사용자의 Configuration 폴더에 직접 복사하여 편집해야 합니다.

다중 사용자 환경에서 구성 파일 삭제

다중 사용자 운영 체제에서 작업 중 Dreamweaver 내의 [코드 단편] 패널에서 미리 디자인된 코드 단편을 삭제하는 경우와 같이 구성 파일 삭제를 수반하는 작업을 수행하면 사용자의 Configuration 폴더에 mm_deleted_files.xml이라는 파일이 자동으로 만들어집니다. 파일이 mm_deleted_files.xml에 나열되어 있으면 Dreamweaver는 해당 파일이 존재하지 않는 것처럼 동작합니다.

구성 파일 비활성화

1. Dreamweaver를 종료합니다.
2. 텍스트 편집기를 사용하여 사용자의 Configuration 폴더에 있는 mm_deleted_files.xml을 편집합니다. 해당 파일에 item 태그를 추가하고 비활성화할 구성 파일의 경로를 Dreamweaver Configuration 폴더에 대한 상대 경로로 지정합니다.

   참고: Dreamweaver 내에서 mm_deleted_files.xml을 편집하지 마십시오.
3. mm_deleted_files.xml을 저장하고 닫습니다.

4 Dreamweaver를 다시 시작합니다.

mm_deleted_files.xml 태그 구문

mm_deleted_files.xml 파일에는 Dreamweaver에서 무시해야 할 구성 파일을 지정하는 구조적 항목 목록이 들어 있습니다. 이러한 항목은 XML 태그로 지정되며 텍스트 편집기로 편집할 수 있습니다.

다음에 나오는 mm_deleted_files.xml 태그의 구문 설명에서 선택적 속성은 속성 목록에 중괄호({})로 표시됩니다. 중괄호로 표시되지 않은 모든 속성은 필수 속성입니다.

<deleteditems>

설명

Dreamweaver에서 삭제된 것으로 처리해야 하는 항목의 목록이 저장되는 컨테이너 태그입니다.

속성

없음

내용

이 태그에는 item 태그가 하나 이상 들어 있어야 합니다.

컨테이너

없음

예제

```
<deleteditems>
<!-- item tags here -->
</deleteditems>
```

<item>

설명

Dreamweaver에서 무시해야 할 구성 파일을 지정합니다.

속성

name

name 속성은 구성 파일 경로를 Configuration 폴더에 대한 상대 경로로 지정합니다. Windows에서는 경로의 각 요소를 구분하는 데 백슬래시(\)를 사용하지만 Macintosh®에서는 콜론(:)을 사용합니다.

내용

없음(빈 태그)

컨테이너

이 태그는 deleteditems 태그에 포함되어야 합니다.

예제

```
<item name="snippets\headers\5columnwith4links.csn" />
```

다중 사용자 환경에서 Dreamweaver 다시 설치 및 제거

Dreamweaver를 설치한 후 나중에 다시 설치하거나 최신 버전으로 업그레이드하는 경우, Dreamweaver에서는 기존 사용자 구성 파일이 자동으로 백업되므로 해당 파일을 사용자 정의했더라도 변경 내용에 액세스할 수 있습니다. 다중 사용자 시스템에서 관리자 권한이 있는 사용자만 수행할 수 있는 Dreamweaver 제거 작업을 수행하면 각 사용자의 Configuration 폴더도 삭제됩니다.

FTP 매핑 변경

FTPExtensionMap.txt 파일(Windows)과 FTPExtensionMapMac.txt 파일(Macintosh)은 파일 확장명을 FTP 전송 모드(ASCII 또는 BINARY)에 매핑합니다.

이 두 파일의 각 행에는 GIF와 같은 파일 확장명과 ASCII 또는 BINARY 하나가 지정되어, 해당 확장명을 가진 파일을 전송할 때 두 FTP 전송 모드 중에서 사용해야 할 모드를 지정합니다. Macintosh의 경우 각 행에는 작성기 코드(예: DmWr)와 파일 유형(예: TEXT)도 포함되어 있습니다. Macintosh에서 지정된 파일 확장명을 가진 파일을 다운로드하면 지정된 작성기와 파일 유형이 해당 파일에 할당됩니다.

전송할 파일에 파일 확장명이 없으면 BINARY 전송 모드가 사용됩니다.

참고: Dreamweaver에서는 파일을 Macbinary 모드로 전송할 수 없습니다. 파일을 Macbinary 모드로 전송하려면 다른 FTP 클라이언트를 사용해야 합니다.

다음 예제에서는 Macintosh 파일에서 확장명이 .html인 파일은 ASCII 모드로 전송해야 함을 나타내는 행을 보여 줍니다.

```
HTML DmWr TEXT ASCII
```

FTPExtensionMap.txt 파일과 FTPExtensionMapMac.txt 파일(Macintosh)에서 특정 행의 모든 요소는 탭으로 구분됩니다. 확장명과 전송 모드는 대문자로 나타냅니다.

기본 설정을 변경하려면 텍스트 편집기에서 파일을 편집합니다.

새 파일 확장명에 대한 정보 추가

1. 텍스트 편집기에서 확장명 맵 파일을 편집합니다.
2. 빈 행에서 파일 확장명을 대문자로 입력하고 Tab 키를 누릅니다.
3. Macintosh의 경우 작성기 코드를 추가하고 Tab 키를 누른 다음 파일 유형을 입력하고 Tab 키를 다시 한 번 누릅니다.
4. **ASCII** 또는 **BINARY**를 입력하여 FTP 전송 모드를 설정합니다.
5. 파일을 저장합니다.

Dreamweaver의 확장 가능한 문서 형식

XML은 복잡한 문서 및 데이터 구조를 정의하는 데 유용합니다. Dreamweaver에서는 몇 가지 XML 스키마를 사용하여 서버 비헤이비어, 태그 및 태그 라이브러리, 구성 요소, 문서 형식, 참조 정보 등에 대한 정보를 구성합니다.

Dreamweaver에서 Extension을 만들어 사용할 때는 XML 파일을 만들거나 기존 XML 파일을 수정하여 Extension에서 사용하는 데이터를 관리해야 하는 경우가 많습니다. 대부분의 경우 Configuration 폴더 내의 적절한 하위 폴더에서 기존 파일을 복사하여 템플릿으로 사용할 수 있습니다.

문서 형식 정의 파일

확장 가능한 문서 형식의 핵심 구성 요소는 문서 형식 정의 파일입니다. 정의 파일은 여러 개가 있을 수 있으며 모든 정의 파일은 Configuration/DocumentTypes 폴더에 있습니다. 각 정의 파일에는 적어도 하나의 문서 형식에 대한 정보가 들어 있습니다. 각 문서 형식에 대해 서버 모델, 색상 코딩 스타일, 설명 등의 필수 정보가 설명되어 있습니다.

참고: Dreamweaver 문서 형식 정의 파일과 XML DTD(문서 형식 정의)를 혼동하지 마십시오. Dreamweaver의 문서 형식 정의 파일에는 여러 개의 documenttype 요소가 포함되어 있으며 각 요소는 하나의 문서 형식과 연관된 태그 및 속성의 미리 정의된 컬렉션을 정의합니다. Dreamweaver에서는 시작 시 문서 형식 정의 파일을 파싱하고 정의된 모든 문서 형식에 대한 정보가 포함된 데이터베이스를 메모리에 만듭니다.

Dreamweaver에서는 초기 문서 형식 정의 파일을 제공합니다. MMDocumentTypes.xml이라는 이 파일에는 Adobe에서 제공하는 문서 형식 정의가 들어 있습니다.

문서 형식	서버 모델	내부 유형	파일 확장명	이전 서버 모델
ASP.NET C#	ASP.NET-Csharp	Dynamic	aspx, ascx	
ASP.NET VB	ASP.NET-VB	Dynamic	aspx, ascx	
ASP JavaScript	ASP-JS	Dynamic	asp	
ASP VBScript	ASP-VB	Dynamic	asp	
ColdFusion	ColdFusion	Dynamic	cfm, cfml	UltraDev 4 ColdFusion
ColdFusion 구성 요소		Dynamic	cfc	
JSP	JSP	Dynamic	jsp	
PHP	PHP	Dynamic	php, php3	
라이브러리 항목		DWExtension	lbi	
ASP.NET C# 템플릿		DWTemplate	axcs.dwt	
ASP.NET VB 템플릿		DWTemplate	axvb.dwt	
ASP JavaScript 템플릿		DWTemplate	aspjs.dwt	
ASP VBScript 템플릿		DWTemplate	aspvb.dwt	
ColdFusion 템플릿		DWTemplate	cfm.dwt	
HTML 템플릿		DWTemplate	dwt	
JSP 템플릿		DWTemplate	jsp.dwt	
PHP 템플릿		DWTemplate	php.dwt	
HTML		HTML	htm, html	
ActionScript		Text	as	
CSharp		Text	cs	
CSS		Text	css	
Java		Text	java	
JavaScript		Text	js	
VB		Text	vb	
VBScript		Text	vbs	

문서 형식	서버 모델	내부 유형	파일 확장명	이전 서버 모델
Text		Text	txt	
EDML		XML	edml	
TLD		XML	tld	
VTML		XML	vtm, vtml	
WML		XML	wml	
XML		XML	xml	

새 문서 형식을 만들어야 하는 경우 Adobe에서 제공하는 문서 정의 파일(MMDocumentTypes.xml)에 항목을 추가하거나 Configuration/DocumentTypes 폴더에 사용자 정의된 문서 정의 파일을 추가할 수 있습니다.

참고: NewDocuments 하위 폴더는 Configuration/DocumentTypes 폴더에 있습니다. 이 하위 폴더에는 각 문서 형식에 대한 기본 페이지(템플릿)가 포함되어 있습니다.

문서 형식 정의 파일의 구조

다음 예제에서는 일반적인 문서 형식 정의 파일의 구조를 보여 줍니다.

```
<?xml version="1.0" encoding="utf-8"?>
<documenttypes     xmlns:MMString="http://www.adobe.com/schemes/data/string/">
    <documenttype
        id="dt-ASP-JS"
        servermodel="ASP-JS"
        internaltype="Dynamic"
        winfileextension="asp,htm, html"
        macfileextension="asp, html"
        previewfile="default_aspjs_preview.htm"
        file="default_aspjs.htm"
        priorversionservermodel="UD4-ASP-JS" >
        <title>
            <loadString id="mmdocumenttypes_0title" />
        </title>
        <description>
            <loadString id="mmdocumenttypes_0descr" />
        </description>
    </documenttype>
    ...
</documenttypes>
```

참고: 문서 형식에 대한 색상 코딩은 Configuration/CodeColoring 폴더에 있는 XML 파일에 지정되어 있습니다.

앞의 예제에서 loadString 요소는 Dreamweaver에서 ASP-JS 유형 문서의 제목 및 설명에 사용되는 지역화 문자열을 지정합니다. 지역화 문자열에 대한 자세한 내용은 19페이지의 “지역화 문자열 제공”을 참조하십시오.

다음 표에서는 문서 형식 정의 파일 내에서 사용할 수 있는 태그 및 속성에 대해 설명합니다.

태그	속성	필수	설명
documenttype (root)		예	부모 노드입니다.
	id	예	모든 문서 형식 정의 파일에서 고유한 식별자입니다.
	servermodel	아니오	연관된 서버 모델을 대/소문자를 구분하여 지정합니다. 기본적으로 다음 값을 사용할 수 있습니다. ASP.NET C# ASP.NET VB ASP VBScript ASP JavaScript ColdFusion JSP PHP MySQL getServerModelDisplayName() 함수를 호출하면 이러한 이름이 반환됩니다. 서버 모델 구현 파일은 Configuration/ServerModels 폴더에 있습니다. Extension 개발자는 이 목록을 확장하여 새로운 서버 모델을 만들 수 있습니다.
	internaltype	예	Dreamweaver의 파일 처리 방법에 대한 광범위한 분류입니다. internaltype은 [디자인] 뷰가 해당 문서에 대해 활성화되며 Dreamweaver 템플릿 또는 Extension과 같은 특수한 경우를 처리하는지 여부를 나타냅니다. 유효한 값은 다음과 같습니다. Dynamic DWExtension(특별한 표시 영역이 있음) DWTemplate(특별한 표시 영역이 있음) HTML HTML4 Text([코드] 뷰만 해당) XHTML1 XML([코드] 뷰만 해당) 모든 서버 모델 관련 문서 형식은 Dynamic에 매핑되고 HTML은 HTML에 매핑됩니다. 스크립트 파일(예: CSS, JS, VB 및 CS)은 Text에 매핑됩니다. internaltype이 DWTemplate인 경우에는 dynamicid를 지정합니다. 그러지 않으면 [새 문서] 대화 상자에서 만드는 새로운 빈 템플릿이 [서버 비헤이비어] 또는 [바인딩] 패널에서 인식되지 않습니다. 이 템플릿의 인스턴스는 단순히 HTML 템플릿입니다.

태그	속성	필수	설명
	dynamicid	아니오	동적 문서 형식의 고유 식별자에 대한 참조입니다. 이 속성은 internaltype이 DWTemplate인 경우에만 의미가 있습니다. 이 속성을 사용하여 동적 템플릿을 동적 문서 형식과 연관시킬 수 있습니다.
	winfileextension	예	Windows에서 문서 형식과 연관된 파일 이름 확장명입니다. 여러 개의 파일 이름 확장명은 쉼표로 구분된 목록을 사용하여 지정하십시오. 이 목록의 첫 번째 확장명이 documenttype 문서를 저장할 때 Dreamweaver에서 사용되는 확장명입니다. 두 개의 비서버 모델 연관 문서 형식의 파일 이름 확장명이 동일한 경우 Dreamweaver에서는 첫 번째 문서 형식을 해당 확장명의 문서 형식으로 인식합니다.
	macfileextension	예	Macintosh에서 문서 형식과 연관된 파일 이름 확장명입니다. 여러 개의 파일 이름 확장명은 쉼표로 구분된 목록을 사용하여 지정하십시오. 이 목록의 첫 번째 확장명이 documenttype 문서를 저장할 때 Dreamweaver에서 사용되는 확장명입니다. 두 개의 비서버 모델 연관 문서 형식의 파일 이름 확장명이 동일한 경우 Dreamweaver에서는 첫 번째 문서 형식을 해당 확장명의 문서 형식으로 인식합니다.
	previewfile	아니오	[새 문서] 대화 상자의 [미리 보기] 영역에 렌더링되는 파일입니다.
	file	예	DocumentTypes/NewDocuments 폴더에 있으며 새 documenttype 문서에 대한 템플릿 내용이 포함된 파일입니다.
	priorversionservermodel	아니오	이 문서의 서버 모델에 Dreamweaver UltraDev 4에 해당하는 항목이 있으면 이전 버전의 서버 모델 이름을 지정합니다. UltraDev 4 ColdFusion은 유효한 이전 서버 모델입니다.
title (하위 태그)		예	[새 문서] 대화 상자의 빈 문서 아래에 범주 항목으로 나타나는 문자열입니다. 정의 파일에 직접 이 문자열을 포함하거나 지역화를 위해 간접적으로 이 문자열을 가리킬 수 있습니다. 이 문자열을 지역화하는 방법에 대한 자세한 내용은 19페이지의 "지역화 문자열 제공"을 참조하십시오. 서식을 지정할 수 없으므로 HTML 태그는 지정할 수 없습니다.
description (하위 태그)		아니오	문서 형식을 설명하는 문자열입니다. 정의 파일에 직접 이 문자열을 포함하거나 지역화를 위해 간접적으로 이 문자열을 가리킬 수 있습니다. 이 문자열을 지역화하는 방법에 대한 자세한 내용은 19페이지의 "지역화 문자열 제공"을 참조하십시오. 서식을 지정할 수 있으므로 HTML 태그를 지정할 수 있습니다.

참고: 사용자가 새 문서를 저장하면 Dreamweaver에서는 해당 문서 형식과 연관된 현재 플랫폼의 확장명 목록(예: winfileextension 및 macfileextension)을 검사합니다. 그런 다음 이 목록의 첫 번째 문자열을 기본 파일 이름 확장명으로 사용합니다. 이 기본 파일 이름 확장명을 변경하려면 쉼표로 구분된 목록에서 새 기본값이 먼저 나열되도록 확장명 순서를 변경합니다.

Dreamweaver는 시작 시 모든 문서 형식 정의 파일을 읽고 유효한 문서 형식 목록을 만듭니다. 존재하지 않는 서버 모델이 있는 정의 파일 내의 모든 항목은 비서버 모델 문서 형식으로 처리됩니다. 잘못된 내용이나 고유하지 않은 ID가 있는 항목은 무시됩니다.

문서 형식 정의 파일이 손상되었거나 Configuration/DocumentTypes 폴더에서 사용할 수 없는 경우에는 오류 메시지가 나타나며 Dreamweaver가 종료됩니다.

동적 템플릿 정의

동적 문서 형식을 기반으로 하는 템플릿을 만들 수 있으며, 이러한 템플릿을 동적 템플릿이라고 합니다. 동적 템플릿을 정의할 때는 다음 두 요소가 반드시 필요합니다.

- 새 문서 형식에 대한 internaltype 속성의 값은 DWTemplate이어야 합니다.
- dynamicid 속성은 반드시 설정해야 하며 해당 값은 기존 동적 문서 형식의 식별자에 대한 참조여야 합니다.

  다음 예제에서는 동적 문서 형식을 정의합니다.

```
<documenttype
    id="PHP_MySQL"
    servermodel="PHP MySQL"
    internaltype="Dynamic"
    winfileextension="php,php3"
    macfileextension="php,php3"
    file="Default.php">
    <title>PHP</title>
    <description><![CDATA[PHP document]]></description>
</documenttype>
```

  그러면 이 동적 문서 형식 PHP_MySQL을 기반으로 하는 다음의 동적 템플릿을 정의할 수 있습니다.

```
<documenttype
    id="DWTemplate_PHP"
    internaltype="DWTemplate"
    dynamicid="PHP_MySQL"
    winfileextension="php.dwt"
    macfileextension="php.dwt"
    file="Default.php.dwt">
    <title>PHP Template</title>
    <description><![CDATA[Dreamweaver PHP Template document]]></description>
</documenttype>
```

Dreamweaver 사용자가 DWTemplate_PHP 유형의 빈 템플릿을 새로 만들면 해당 파일에 PHP 서버 비헤이비어를 만들 수 있게 됩니다. 또한 사용자가 새 템플릿의 인스턴스를 만들 때는 이 인스턴스에서 PHP 서버 비헤이비어를 만들 수 있습니다.

앞의 예제에서 사용자가 템플릿을 저장하면 Dreamweaver에서는 파일에 .php.dwt 확장명을 자동으로 추가합니다. 사용자가 템플릿 인스턴스를 저장하면 Dreamweaver에서는 파일에 .php 확장명을 추가합니다.

문서 확장명과 파일 유형 추가 및 수정

기본적으로 Dreamweaver에서는 인식된 모든 파일 유형이 [파일] > [열기] 대화 상자에 표시됩니다. 문서 형식을 만든 후 Extension 개발자는 해당 Extensions.txt 파일을 업데이트해야 합니다. 사용자가 Windows XP, Windows Vista 또는 Mac OS X 등의 다중 사용자 시스템을 사용하는 경우도 있습니다. 이러한 경우 또 다른 Extensions.txt 파일이 사용자 Configuration 폴더에 있습니다. Dreamweaver에서는 이 Extensions.txt 파일을 찾아 파싱하므로 사용자가 이 파일을 업데이트해야 합니다.

사용자의 Configuration 폴더 위치는 사용자의 플랫폼에 따라 다릅니다.

Windows XP 플랫폼에서는 다음 위치가 사용됩니다.

하드 디스크: \Documents and Settings\사용자 이름\Application Data\Adobe\Dreamweaver CS5\Configuration

참고: 이 폴더는 숨겨진 폴더 내에 있을 수 있습니다.

Windows Vista 플랫폼에서는 다음 위치가 사용됩니다.

하드 디스크: \Users**username**\AppData\Roaming\Adobe\Dreamweaver CS5\Configuration

Mac OS X 플랫폼에서는 다음 위치가 사용됩니다.

hard disk:\Users/사용자 이름/Library/Application Support/Adobe/Dreamweaver CS5/Configuration

사용자의 Configuration 폴더에서 Extensions.txt 파일을 찾을 수 없는 경우 Dreamweaver에서는 Dreamweaver Configuration 폴더에서 이 파일을 찾습니다.

참고: 다중 사용자 플랫폼의 경우 Dreamweaver에서는 Dreamweaver Configuration 폴더가 아니라 사용자의 Configuration 폴더에 있는 Extensions.txt의 사본을 파싱합니다. 따라서 Dreamweaver Configuration 폴더에 있는 Extensions.txt의 사본을 편집하면 Dreamweaver에서는 변경 내용을 인식하지 못합니다.

문서 확장명을 만들려면 기존 문서 형식에 새 확장명을 추가하거나 문서 형식을 만듭니다.

기존 문서 형식에 새 확장명 추가

1. MMDocumentTypes.xml을 편집합니다.
2. 기존 문서 형식의 winfileextension 및 macfileextension 속성에 새 확장명을 추가합니다.

새 문서 형식 추가

1. Configuration 폴더에 있는 Extensions.txt 파일의 백업 사본을 만듭니다.
2. 텍스트 편집기에서 Extensions.txt를 엽니다.
3. 각각의 새 파일 유형에 대한 새 행을 추가합니다. 새 파일 유형의 파일 확장명을 쉼표로 구분하여 대문자로 입력합니다. 그런 다음 콜론을 추가하고 파일 유형의 팝업 메뉴에 표시할 설명을 간단히 입력합니다. 팝업 메뉴는 [파일] > [열기] 대화 상자에 나타납니다.

   예를 들어 JPEG 파일의 경우 JPG,JPEG,JFIF:JPEG Image Files라고 입력합니다.
4. Extensions.txt 파일을 저장합니다.
5. Dreamweaver를 다시 시작합니다.

변경된 내용을 확인하려면 [파일] > [열기]를 선택하고 파일 유형의 팝업 메뉴를 클릭합니다.

Dreamweaver 파일 > 열기 대화 상자의 기본 파일 유형 변경

1. Configuration 폴더에 있는 Extensions.txt 파일의 백업 사본을 만듭니다.
2. 텍스트 편집기에서 Extensions.txt를 엽니다.
3. 새 기본값에 해당하는 행을 잘라내어 해당 행이 파일의 첫 번째 행이 되도록 파일의 처음에 붙여넣습니다.
4. Extensions.txt 파일을 저장합니다.
5. Dreamweaver를 다시 시작합니다.

변경된 내용을 확인하려면 [파일] > [열기]를 선택하고 파일 유형의 팝업 메뉴를 클릭합니다.

기타 도움말 항목

http://www.adobe.com/go/16410_kr

지역화 문자열 제공

문서 형식 정의 파일 내에서 <title> 및 <description> 하위 태그는 문서 형식의 표시 제목 및 설명을 지정합니다. 이 두 하위 태그에서 하위 태그의 지역화 문자열을 지정하기 위한 자리 표시자로 MMString:loadstring 지시문을 사용할 수 있습니다. 이 과정은 사용자가 문자열 식별자를 자리 표시자로 사용하여 페이지에서 사용할 특정 문자열을 지정하는 서버측 스크립팅과 유사합니다. 자리 표시자에는 특정 태그를 사용하거나 값을 바꿀 수 있는 태그 속성을 지정할 수 있습니다.

지역화 문자열 제공

1 문서 형식 정의 파일의 맨 처음에 다음 명령문을 입력합니다.

```
<?xml version="1.0" encoding="utf-8"?>
```

2 <documenttypes> 태그에서 MMString 네임스페이스를 선언합니다.

```
<documenttypes
    xmlns:MMString="http://www.adobe.com/schemes/data/string/">
```

3 문서 형식 정의 파일에서 지역화 문자열을 제공할 위치에 MMString:loadstring 지시문을 사용하여 지역화 문자열에 대한 자리 표시자를 정의합니다. 이 자리 표시자는 다음 두 가지 방법 중 하나를 사용하여 지정할 수 있습니다.

```
<description>
    <loadstring>myJSPDocType/Description</loadstring>
</description>
```

또는

```
<description>
    <loadstring id="myJSPDocType/Description" />
</description>
```

이러한 예제에서 myJSPDocType/Description은 지역화 문자열의 자리 표시자 역할을 하는 고유한 문자열 식별자입니다. 지역화 문자열은 다음 단계에서 정의합니다.

4 Configuration/Strings 폴더에서 지역화 문자열을 정의하는 새 XML 파일을 만들거나 기존 파일을 편집합니다. 예를 들어 다음 코드는 Configuration/Strings/strings.xml 파일에 있을 때 myJSPDocType/Description 문자열을 정의합니다.

```
<strings>
...
    <string id="myJSPDocType/Description"
        value=
        "<![CDATA[JavaServer Page with <em>special</em> features]]>"
    />
...
</strings>
```

참고: 앞의 예제에서 myJSPDocType/Description과 같은 문자열 식별자는 응용 프로그램 내에서 고유해야 합니다. Dreamweaver는 시작 시 Configuration/Strings 폴더 내의 모든 XML 파일을 파싱하고 이러한 고유 문자열을 로드합니다.

문서 형식 정의 파일의 규칙

Dreamweaver에서는 서버 모델과 연관된 여러 문서 형식이 파일 확장명을 공유할 수 있습니다. 예를 들어 ASP-JS와 ASP-VB는 .asp를 해당 파일 확장명으로 사용할 수 있습니다. 우선 순위를 갖는 서버 모델에 대한 자세한 내용은 300페이지의 “canRecognizeDocument()”를 참조하십시오.

Dreamweaver에서 서버 모델과 연관되지 않은 문서 형식은 파일 확장명을 공유할 수 없습니다.

두 문서 형식에서 하나의 파일 확장명을 사용하는데 한 유형은 서버 모델과 연관되어 있고 다른 유형은 그렇지 않은 경우 서버 모델과 연관되지 않은 문서 형식이 우선 순위를 갖습니다. 예를 들어 서버 모델과 연관되지 않은 SAM이라는 문서 형식이 있고 이 문서 형식의 파일 확장명이 .sam인 경우 ASP-JS 문서 형식에 이 파일 확장명을 추가한다고 가정합니다. Dreamweaver 사용자가 확장명이 .sam인 파일을 열면 Dreamweaver에서는 이 파일에 ASP-JS가 아니라 SAM 문서 형식을 지정합니다.

문서 선언 정의

Dreamweaver를 사용하면 Configuration/DocumentTypes 폴더에 있는 MMDocumentTypeDeclarations.xml 파일을 통해 문서에 DTD를 설정할 수 있습니다. 적용되는 사용 가능한 DTD 및 문서의 목록은 MMDocumentTypeDeclarations.xml 파일에 정의되어 있습니다.

Dreamweaver에서 문서 열기

사용자가 문서 파일을 열면 Dreamweaver에서는 파일 확장명을 기준으로 일련의 단계에 따라 문서 형식을 식별합니다.

고유한 문서 형식을 찾으면 Dreamweaver에서는 해당 유형을 사용하고 사용자가 여는 문서의 연관된 서버 모델(있는 경우)을 로드합니다. 사용자가 Dreamweaver UltraDev 4 서버 비헤이비어를 사용하기로 선택하면 Dreamweaver에서는 해당하는 UltraDev 4 서버 모델을 로드합니다.

파일 확장명이 둘 이상의 문서 형식에 매핑되면 Dreamweaver에서는 다음 작업을 수행합니다.

- 문서 형식 목록에 정적 문서 형식이 있으면 해당 문서 형식이 우선합니다.
- 모든 문서 형식이 동적이면 Dreamweaver에서는 이러한 문서 형식과 연관된 서버 모델의 사전순 목록을 만든 다음 각 서버 모델에서 canRecognizeDocument() 함수를 호출합니다. 자세한 내용은 300페이지의 “canRecognizeDocument()”를 참조하십시오. 그런 다음 반환값을 수집하여 가장 큰 양의 정수 값을 반환한 서버 모델을 확인합니다. 가장 큰 정수를 반환하는 서버 모델의 문서 형식은 Dreamweaver에서 열려는 문서에 지정하는 문서 형식이 됩니다. 그러나 둘 이상의 서버 모델이 동일한 정수를 반환하는 경우 Dreamweaver에서는 해당 서버 모델의 사전순 목록을 검색한 후 목록의 첫 번째 문서 형식을 선택하여 사용합니다. 예를 들어 ASP-JS와 ASP-VB가 ASP 문서를 사용하고 각각의 canRecognizeDocument() 함수가 같은 값을 반환하는 경우, ASP-JS가 사전순에 따라 먼저 표시되므로 Dreamweaver에서는 ASP-JS에 해당 문서를 지정합니다.

Dreamweaver에서는 파일 확장명을 문서 형식에 매핑할 수 없는 경우 문서를 텍스트 파일로 엽니다.

작업 영역 레이아웃 사용자 정의

Dreamweaver에서는 작업 영역 레이아웃을 사용자 정의할 수 있습니다. 지정된 레이아웃에 있는 패널뿐 아니라, 패널의 위치 및 크기, 패널의 축소/확장 상태, 응용 프로그램 윈도우의 위치 및 크기, [문서] 윈도우의 위치 및 크기를 비롯한 다른 속성도 사용자 정의할 수 있습니다.

작업 영역 레이아웃은 Configuration/Workspace 레이아웃 폴더에 저장된 XML 파일에 지정되어 있습니다. 다음 단원에서는 XML 태그 구문에 대해 설명합니다. 선택적 속성은 속성 목록에서 중괄호({})로 표시됩니다. 중괄호로 표시되지 않은 모든 속성은 필수 속성입니다.

<panelset>

설명

패널 세트에 대한 설명이 시작되는 부분을 나타내는 가장 바깥쪽 태그입니다.

속성

없음

내용

이 태그에는 application, document 또는 panelset 태그가 하나 이상 포함될 수 있습니다.

컨테이너

없음

예제

```
<panelset>
    <!-- panelset tags here -->
</panelset>
```

<application>

설명

응용 프로그램 윈도우의 초기 위치 및 크기를 지정합니다.

속성

rect, maximize

- rect는 응용 프로그램 윈도우의 위치 및 크기를 지정합니다. 이 문자열은 "왼쪽 위쪽 오른쪽 아래쪽" 형식의 정수로 지정됩니다.
- maximize는 부울 값으로서, 응용 프로그램이 시작될 때 최대화되면 true이고, 그렇지 않으면 false입니다. 기본값은 true입니다.

내용

없음

컨테이너

이 태그는 panelset 태그에 포함되어야 합니다.

예제

```
<panelset>
    <application rect="0 0 1000 1200" maximize="false">
    </application>
</panelset>
```

<document>

설명

[문서] 윈도우의 초기 위치 및 크기를 지정합니다.

속성

rect, maximize

- rect는 [문서] 윈도우의 위치 및 크기를 지정합니다. 이 문자열은 "왼쪽 위쪽 오른쪽 아래쪽" 형식의 정수로 지정됩니다. maximize 값이 true인 경우 rect 값은 무시됩니다.
- maximize는 부울 값으로서, [문서] 윈도우가 시작될 때 최대화되면 true이고, 그렇지 않으면 false입니다. 기본값은 true입니다.

내용

없음

컨테이너

이 태그는 panelset 태그에 포함되어야 합니다.

예제

```
<panelset>
    <document rect="100 257 1043 1200" maximize="false">
    </document>
</panelset>
```

<panelframe>

설명

전체 패널 그룹에 대해 설명합니다.

속성

x, y, {width, height}, dock, collapse

- x는 패널 그룹의 왼쪽 위치를 지정하며 해당 값으로는 정수나 화면을 기준으로 한 값을 사용할 수 있습니다. 화면에 표시되지 않는 정수 값일 경우 패널 그룹은 화면에 표시할 수 있는 가장 가까운 화면 위치에 표시됩니다. 상대 값은 "left" 또는 "right"가 될 수 있는데, 이 값은 패널 그룹의 가장자리를 실제 화면의 가장자리에 맞출 방식을 나타냅니다.
- y는 패널 그룹의 위쪽 위치를 지정하며, 해당 값으로는 정수나 화면을 기준으로 한 값을 사용할 수 있습니다. 화면에 표시되지 않는 정수 값일 경우 패널 그룹은 화면에 표시할 수 있는 가장 가까운 화면 위치에 표시됩니다. 상대 값은 "top" 또는 "bottom"이 될 수 있는데, 이 값은 패널 그룹의 가장자리를 실제 화면의 가장자리에 맞출 방식을 나타냅니다.
- width는 패널 그룹의 폭(픽셀)을 나타내며, 선택적 속성입니다. width를 지정하지 않으면 내장된 패널 그룹 기본값이 사용됩니다.
- height는 패널 그룹의 높이(픽셀)를 나타내며, 선택적 속성입니다. height를 지정하지 않으면 내장된 패널 그룹 기본값이 사용됩니다.
- dock은 패널 그룹이 결합될 응용 프로그램 프레임의 가장자리를 지정하는 문자열 값입니다. Macintosh에서는 패널 그룹을 결합할 수 없으므로 이 속성이 무시됩니다.
- collapse는 부울 값으로서, 패널 그룹이 축소되면 true이고 확장되면 false입니다. Macintosh에서는 부동 패널이 사용되므로 이 속성이 무시됩니다.

내용

이 태그에는 panelcontainer 태그가 하나 이상 포함되어야 합니다.

컨테이너

이 태그는 panelset 태그에 포함되어야 합니다.

예제

```
<panelset>
    <panelframe rect="196 453 661 987" visible="true" dock="floating">
        <!-- panelcontainer tags here -->
    </panelframe>
</panelset>
```

<panelcontainer>

설명

전체 패널 그룹에 대해 설명합니다.

속성

expanded, title,{height}, activepanel, visible, maximize, maxRestorePanel, maxRestoreIndex, maxRect, tabsinheader

- expanded는 부울 값으로서, 패널이 확장되면 true이고 그렇지 않으면 false입니다.
- title은 패널 제목을 지정하는 문자열입니다.
- height는 패널 높이(픽셀)를 지정하는 정수로서, 선택적 속성입니다. height를 지정하지 않으면 각 패널에 대해 내장된 기본 값이 사용됩니다.

  참고: 폭은 부모로부터 상속됩니다.
- activepanel은 프론트 패널의 ID를 나타내는 숫자입니다.
- visible은 부울 값으로서, 패널이 표시되면 true이고 그렇지 않으면 false입니다.
- maximize는 부울 값으로서, 패널이 처음 표시될 때 최대화되면 true이고 그렇지 않으면 false입니다.
- maxRestorePanel은 복원할 패널의 ID를 나타내는 숫자입니다.
- maxRect는 최대화될 때의 패널 위치 및 크기를 나타내는 문자열입니다. 이 문자열은 "왼쪽 위쪽 오른쪽 아래쪽" 형식의 정수로 지정됩니다.
- tabsinheader는 부울 값으로서, 탭이 머리글 막대의 아래쪽이 아닌 머리글에 있으면 true이고 그렇지 않으면 false입니다.

내용

이 태그에는 panel 태그가 하나 이상 포함되어야 합니다.

컨테이너

이 태그는 panelframe 태그에 포함되어야 합니다.

예제

```
<panelset>
    <panelframe rect="196 453 661 987" visible="true" dock="floating">
        <panelcontainer title="Color" height="250" visible="true" expanded="true"
        activepanel="20">
            <!-- panel tags here --->
        </panelcontainer>
    </panelframe>
</panelset>
```

<panel>

설명

패널 컨테이너에 나타날 패널을 지정합니다.

속성

id, visibleTab

- id는 패널 ID를 나타내는 숫자입니다. 다음 표에서는 값 목록을 보여 줍니다.

소프트웨어	ID	패널
Adobe® Flash®	1	속성
	2	액션
	3	정렬

소프트웨어	ID	패널
	4	비헤이비어
	5	구성 요소
	6	구성 요소 관리자
	7	색상 혼합기
	8	색상 견본
	9	작업 내역
	10	정보
	11	라이브러리
	12	무비 탐색기
	13	출력
	14	속성
	15	프로젝트
	16	변환
	17	장면
	18	문자열
	19	디버거
	101–110	라이브러리
Dreamweaver	1	속성
Flex Builder	1	속성

- visibleTab은 부울 값으로서, 탭과 패널이 표시되면 true이고 그렇지 않으면 false입니다.

내용

없음

컨테이너

이 태그는 panelcontainer 태그에 포함되어야 합니다.

예제

```
<panelset>
    <panelframe rect="196 453 661 987" visible="true" dock="floating">
        <panelcontainer title="Color" height="250" visible="true" expanded="true"
        activepanel="20">
            <panel id="20"></panel>
        </panelcontainer>
    </panelframe>
</panelset>
```

코딩 툴바 사용자 정의

[코딩] 툴바에는 처음에 15개의 버튼이 표시되는데, 이는 사용할 수 있는 버튼 중 일부입니다. Configuration/Toolbars/Toolbars.xml 파일을 편집하여 툴바에 표시할 버튼이나 버튼이 나타나는 순서를 변경하는 방법으로 코딩 툴바를 사용자 정의할 수 있습니다. 또한 Extension Manager를 통해 툴바에 고유한 버튼을 삽입할 수도 있습니다.

버튼 순서 변경

1. Configuration/Toolbars/toolbars.xml 파일을 엽니다.
2. 다음 주석을 검색하여 [코드] 뷰 툴바 섹션을 찾습니다.

```
<!-- Code view toolbar -->
```

3. 버튼이 툴바에 원하는 순서대로 표시되도록 버튼 태그를 복사하여 붙여넣습니다.
4. 파일을 저장합니다.

버튼 제거

1. Configuration/Toolbars/toolbars.xml 파일을 엽니다.
2. 다음 주석을 검색하여 코딩 툴바 섹션을 찾습니다.

```
<!-- Code view toolbar -->
```

3. 제거할 버튼을 주석으로 처리합니다.

   다음 예제에서는 툴바에 표시되지 않도록 주석으로 처리된 버튼을 보여 줍니다.

```
<!-- remove button from Coding toolbar
    <button id="DW_ExpandAll"
        image="Toolbars/images/MM/T_ExpandAll_Sm_N.png"
        disabledImage="Toolbars/images/MM/T_ExpandAll_Sm_D.png"
        tooltip="Expand All"
        domRequired="false"
        enabled="dw.getFocus(true) == 'textView' || dw.getFocus(true) == 'html'¬
            "command="if (dw.getFocus(true) == 'textView' || dw.getFocus(true) ¬
            == 'html') dw.getDocumentDOM().source.expandAllCodeFragments();"
        update="onViewChange" />
-->
```

4. 파일을 저장합니다.

툴바에 표시되지 않는 버튼을 다시 나타나게 하려면 XML 파일에서 버튼 주위의 주석을 제거합니다.

키보드 단축키 매핑 변경

Dreamweaver에는 Dreamweaver 기능에 대한 다양한 키보드 단축키가 있습니다. 기본 키보드 단축키는 menus.xml 파일에 나열되어 있으며 미국 키보드용으로 만들어져 있습니다. Dreamweaver에 제공된 단축키 수는 제한되어 있으므로 다국어 키보드의 경우 영숫자가 아닌 특정 단축키(a-z 또는 0-9 이외의 문자)를 사용하려면 매핑을 다시 해야 합니다. 이를 위하여 Dreamweaver에서는 다국어 키보드용 키보드 단축키 매핑을 정의하는 여러 개의 xml 파일이 제공됩니다. 이러한 파일은 Configuration\Menus\Adaptive Sets 폴더에 있습니다. Dreamweaver에서는 컴퓨터에 연결된 다국어 키보드가 감지되면 자동으로 해당 키보드의 매핑 파일에 따라 키보드 단축키가 다시 설정됩니다. 키보드 레이아웃에 사용 가능한 적절한 파일이 없는 경우 해당 키보드 레이아웃에서 작동하지 않는 단축키는 제거됩니다.

키보드 단축키 매핑 파일의 이름은 해당 파일이 나타내는 키보드 레이아웃에 대한 두 자로 된 언어 코드를 사용하여 지정됩니다. 예를 들어 독일어 키보드 레이아웃용 파일은 de.xml입니다. 같은 언어라도 국가별로 다른 키보드 레이아웃을 사용하는 경우에는 두 자로 된 언어 코드 뒤에 대시("-")와 두 자로 된 국가 코드를 붙여 매핑 파일의 이름을 지정합니다. 예를 들어 fr-ca.xml은 캐

나다 프랑스어 키보드 레이아웃의 매핑 파일 이름입니다. 두 자로 된 언어 코드는 ISO 639(http://en.wikipedia.org/wiki/List_of_ISO_639_codes)에 정의되어 있으며 국가 코드는 ISO 3166(http://en.wikipedia.org/wiki/ISO_3166-1_alpha-2)에 정의되어 있습니다.

컴퓨터의 활성 키보드 언어 설정이 변경되면 Dreamweaver에서는 해당 키보드의 국가 및 언어에 맞는 키보드 단축키 매핑 파일이 있는지 확인합니다. 먼저 국가에 맞는 매핑 파일이 있는지 확인한 다음, 이 파일이 없으면 해당 언어에 대한 매핑 파일을 확인합니다. 예를 들어 컴퓨터에 캐나다 프랑스어 키보드를 연결한 경우 Dreamweaver에서는 먼저 캐나다 프랑스어 키보드 레이아웃용 fr-ca.xml 파일이 있는지 찾아보고, 이 파일이 없으면 fr.xml을 찾습니다. 다음 표에서는 Dreamweaver에서 제공하는 매핑 파일 목록을 보여 줍니다.

파일 이름	Windows 플랫폼	Macintosh 플랫폼
ca.xml	카탈로니아어	카탈로니아 스페인어
de.xml	독일어(독일, 오스트리아)	오스트리아, 독일어
de-ch.xml	독일어(스위스)	스위스 독일어
es.xml	스페인어(국제 정렬), 스페인어(전통 정렬)	스페인어 - ISO
fr.xml	프랑스어(프랑스)	프랑스어
fr-ca.xml	프랑스어(캐나다)	캐나다 프랑스어 - CSA
fr-ch.xml	프랑스어(스위스)	스위스 프랑스어
it.xml	이탈리아어(이탈리아), 이탈리아어(스위스)	이탈리아어 - Pro
it-mac.xml	N/A	이탈리아어
ja.xml	일본어	일본어
nl-be.xml	네덜란드어(벨기에), 프랑스어(벨기에)	벨기에어
zh-cn.xml	중국어(중국), 중국어(싱가포르)	중국어 간체

Dreamweaver에서 제공되는 것 이외의 키보드 레이아웃을 사용하는 경우, 사용하는 특정 키보드에 맞는 매핑 파일을 만들어 Configuration\Menus\Adaptive Sets 폴더에 저장할 수 있습니다.

키보드 매핑 파일 만들기

1 Configuration\Menus\Adaptive Sets 폴더에 있는 키보드 매핑 파일 중 하나의 사본을 만들고 사용하는 키보드 레이아웃에 해당하는 2자 언어 코드와 .xml 확장명을 사용하여 파일 이름을 지정합니다.

기존 파일의 사본을 만들 때는 사용하는 키보드 레이아웃과 유사한 키보드 매핑 파일을 복사해야 합니다. 예를 들어 스웨덴어 키보드용 키보드 매핑 파일을 만드는 경우 스웨덴어 키보드 레이아웃은 독일어 키보드 레이아웃과 비슷하므로 de.xml을 복사할 수 있습니다.

2 새로 만든 키보드 매핑 파일을 Configuration\Menus\Adaptive Sets 이외의 다른 폴더로 이동합니다.

3 이동한 키보드 매핑 파일을 Dreamweaver에서 엽니다.

4 사용하는 키보드 레이아웃에 맞게 단축키 태그를 제거하거나 추가합니다.

미국용 키보드 단축키 세트와 사용하는 언어용 단축키 세트를 비교하면 수정할 키보드 단축키 태그를 결정하는 데 도움이 됩니다. 다음 절차에서는 두 키보드 레이아웃의 단축키를 비교하는 방법에 대해 설명합니다.

5 키보드 단축키를 변경한 후 파일을 저장하고 이 파일을 다시 Configuration\Menus\Adaptive Sets 폴더로 이동합니다.

수정할 키보드 단축키 태그 결정

1 컴퓨터의 활성 키보드 언어 설정이 사용하려는 언어로 설정되지 않은 경우 해당 언어로 전환합니다. 이 작업은 컴퓨터의 운영 체제를 통해 수행할 수 있습니다. 예를 들어 Windows의 경우 [제어판]에서 언어를 선택할 수 있습니다.

2 [편집] > [키보드 단축키]를 선택하여 Dreamweaver [키보드 단축키 편집기]를 시작합니다.

3 편집기 대화 상자의 오른쪽 위에 있는 세 번째 이미지 버튼을 클릭합니다. 이 버튼 위에 포인터를 놓으면 "세트를 HTML로 보내기"라는 도구 설명이 나타납니다.

[HTML 파일로 저장] 대화 상자가 나타나면 여기에서 현재 키보드 레이아웃에 대한 키보드 단축키 요약 파일의 이름을 입력해야 합니다.

4 요약 파일을 저장한 다음 [키보드 단축키 편집기] 대화 상자를 닫습니다.

5 컴퓨터의 운영 체제를 통해 키보드 레이아웃을 미국용 키보드로 전환합니다.

6 [편집] > [키보드 단축키]를 선택하여 Dreamweaver [키보드 단축키 편집기]를 시작합니다.

7 편집기 대화 상자의 오른쪽 위에 있는 세 번째 이미지 버튼을 클릭하여 설정을 HTML 파일로 내보냅니다.

8 요약 파일을 저장한 다음 [키보드 단축키 편집기] 대화 상자를 닫습니다.

9 두 개의 키보드 단축키 요약 파일을 인쇄하여 비교하면 사용하는 언어 키보드 레이아웃에 맞게 Dreamweaver에서 제거한 모든 단축키를 확인할 수 있습니다. 이러한 단축키의 경우에는 사용하는 키보드 레이아웃에서만 사용 가능한 키를 사용하여 새로운 단축키를 다시 할당해야 합니다.

두 파일을 비교하여 얻은 정보를 이용하여 다시 할당할 각 단축키에 대한 단축키 태그를 추가하거나 제거하는 방법으로 키보드 레이아웃 매핑 파일을 업데이트할 수 있습니다.

키보드 레이아웃 매핑 XML 파일

다음 예제에서는 프랑스어 키보드 매핑 파일(fr.xml)의 형식을 보여 줍니다.

```
<shortcutset language="French">
<shortcut key="Cmd+[" newkey="Cmd+&ugrave;"/>
<shortcut key="Cmd+]" newkey="Cmd+)"/>
<shortcut platform="win" key="Cmd+Shift+>" newkey="Cmd+Opt+Shift+,"/>
<shortcut platform="mac" key="Cmd+Shift+>" newkey="Cmd+<"/>
<shortcut platform="win" key="Cmd+Shift+<" newkey="Cmd+Shift+,"/>
<shortcut key="Cmd+'" newkey="Cmd+Shift+=" />
...
</shortcutset>
```

키보드 레이아웃 XML 파일의 구문은 다음과 같습니다.

```
<shortcutset language="language_name">
<shortcut key="key_combination" newkey="key_combination"/>
<shortcut platform="op_system" key="key_combination" newkey="key_combination"/>
</shortcutset>
```

위치:

- **language_name**은 키보드의 언어(예: French, Spanish, German 등)입니다.
- **key_combination**은 키보드 단축키(예: Cmd+[, Cmd+Shift+>, Ctrl+$)입니다.
- key는 대체할 키보드 단축키를 지정합니다.
- newkey는 key를 대체할 키보드 단축키를 지정합니다.
- platform=**op_system**은 해당 단축키가 적용되는 시스템입니다. win 또는 mac 중 하나를 지정합니다. 플랫폼을 지정하지 않으면 단축키가 두 플랫폼 모두에 적용됩니다.

3장: 코드 뷰 사용자 정의

Adobe Dreamweaver의 [코드] 뷰에서는 코드를 신속하게 입력하고 코드의 가독성과 정확성을 높이는 데 도움이 되는 두 가지 기능을 제공합니다. 이 두 기능은 코드 힌트와 코드 색상 표시입니다. 또한 Dreamweaver에는 코드가 지정된 대상 브라우저에 맞는지 확인하고 기본 HTML 서식을 바꿀 수 있는 기능이 있습니다.

코드 힌트 및 코드 색상 표시 기능을 사용자 정의하려면 해당 기능이 구현되어 있는 XML 파일을 수정하면 됩니다. CodeHints.xml 또는 SpryCodeHints.xml 파일에 항목을 추가하여 코드 힌트 메뉴에 항목을 추가할 수 있습니다. 코드 색상 표시 스타일 파일인 Colors.xml을 수정하여 색상 체계를 수정하거나, CodeColoring.xml과 같은 코드 색상 표시 구문 파일 중 하나를 수정하여 코드 색상 표시 체계를 변경하거나 새로 추가할 수 있습니다. 대상 브라우저에 대한 CSS(Cascading Style Sheet) 프로파일 파일을 수정하여 Dreamweaver에서 CSS 속성 및 값을 검사하는 방법을 바꿀 수도 있습니다. 또한 [환경 설정] 대화 상자에서 Dreamweaver의 기본 HTML 서식을 바꿀 수 있습니다. 다음 단원에서는 이러한 기능을 사용자 정의하는 방법에 대해 설명합니다.

코드 힌트

코드 힌트는 [코드] 뷰에서 특정 문자 패턴을 입력할 때 자동으로 열리는 메뉴입니다. 코드 힌트는 현재 입력하고 있는 문자열을 완성할 수 있는 문자열 목록을 제공하여 입력 시간을 단축시켜 줍니다. 입력하려는 문자열이 메뉴에 나타나면 해당 메뉴로 스크롤하고 Enter 또는 Return 키를 눌러 입력을 마칩니다. 예를 들어 <를 입력하면 팝업 메뉴에 태그 이름 목록이 나타나므로 나머지 태그 이름을 모두 입력할 필요 없이 메뉴에서 태그를 선택하여 텍스트에 넣을 수 있습니다. Dreamweaver에서는 Spry 프레임워크에 대한 코드 힌트도 제공합니다.

Dreamweaver는 Configuration/CodeHints 폴더의 CodeHints.xml 파일 및 기타 XML 파일에서 코드 힌트 메뉴를 로드합니다. 이 항목에 설명된 XML 스키마 형식을 사용하여 XML 파일에서 코드 힌트 메뉴를 정의하고 이를 Configuration/CodeHints 폴더에 저장하면 Dreamweaver에 코드 힌트 메뉴를 추가할 수 있습니다.

Dreamweaver에 코드 힌트 파일의 내용이 로드된 후에는 JavaScript를 통해 새로운 코드 힌트 메뉴를 동적으로 추가할 수도 있습니다. 예를 들어 JavaScript 코드는 [바인딩] 패널의 세션 변수 목록을 채웁니다. 같은 코드로 코드 힌트 메뉴를 추가하여 사용자가 [코드] 뷰에서 **"Session."**을 입력하면 세션 변수의 메뉴가 표시되게 할 수 있습니다. JavaScript를 사용하여 코드 힌트 메뉴를 추가하거나 수정하는 방법에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**에서 "코드 함수"를 참조하십시오.

일부 유형의 코드 힌트 메뉴는 Dreamweaver에서 XML 파일이나 JavaScript API를 통해 표현할 수 없습니다. CodeHints.xml 파일, SpryCodeHints.xml 파일 및 JavaScript API는 코드 힌트 엔진의 유용한 하위 세트를 표시하지만 일부 Dreamweaver 기능에 액세스할 수 없습니다. 예를 들어 색상 선택기를 열기 위한 JavaScript 후크가 없으므로 Dreamweaver에서 JavaScript를 사용하여 [속성 값] 메뉴를 표시할 수 없습니다. 이 경우 텍스트를 삽입할 수 있는 텍스트 항목 메뉴만 열립니다.

참고: 텍스트를 삽입하면 삽입 포인터는 삽입된 문자열 뒤에 옵니다.

CodeHints.xml 파일

CodeHints.xml 파일에는 다음 엔터티가 포함됩니다.

- 모든 메뉴 그룹의 목록

  [환경 설정] 대화 상자에서 [코드 힌트] 범주를 선택하면 메뉴 그룹의 목록이 표시됩니다. [편집] > [환경 설정]을 선택하여 [환경 설정] 대화 상자를 열 수 있습니다. Dreamweaver에서 제공하는 메뉴 그룹 또는 코드 힌트 메뉴 유형에는 태그 이름, 속성 이름, 속성 값, 함수 인수, 객체 메서드 및 변수, HTML 엔터티 등이 있습니다.

- 각 메뉴 그룹에 대한 설명

목록에서 메뉴 그룹을 선택하면 해당 코드 힌트 범주에 대한 설명이 [환경 설정] 대화 상자에 나타납니다. 선택한 항목에 대한 설명은 메뉴 그룹 목록 아래에 나타납니다.

- 코드 힌트 메뉴

  코드 힌트 메뉴는 해당 메뉴가 표시되도록 하는 패턴과 명령 목록으로 구성됩니다. 예를 들어 & 같은 패턴은 &, >, < 등의 메뉴를 표시할 수 있습니다.

  다음 예제에서는 CodeHints.xml 파일의 형식을 보여 줍니다. 굵게 표시된 태그에 대한 자세한 내용은 33페이지의 "코드 힌트 태그"를 참조하십시오.

```
<codehints>
<menugroup name="HTML Entities" enabled="true" id="CodeHints_HTML_Entities">
    <description>
    <![CDATA[ When you type a '&', a pop-up menu shows
        a list of HTML entities. The list of HTML entities
        is stored in Configuration/CodeHints.xml. ]]>
    </description>
    <menu pattern="&amp;amp;">
            <menuitem value="&amp;amp;amp;" texticon="&amp;amp;"/>
            <menuitem value="&amp;amp;lt;" icon="lessThan.gif"/>
    </menu>
</menugroup>

<menugroup name="Tag Names" enabled="true" id="CodeHints_Tag_Names">
    <description>
            <![CDATA[ When you type '<', a pop-up menu shows
            all possible tag names.0You can edit the list of tag
            names using the
            <a href="javascript:dw.popupTagLibraryEditor()"> Tag Library
                 Editor </a>]]>
        </description>
</menugroup>

<menugroup name="Function Arguments" enabled="true"
        id="CodeHints_Function_Arguments">
    <description>
        ...
    </description>
    <function pattern="ArraySort(array, sort_type, sort_order)"
            doctypes="CFML"/>
    <function pattern="Response.addCookie(Cookie cookie)"
            doctypes="JSP"/>
</menugroup>
</codehints>
```

JavaScript 코드 힌트

Dreamweaver에서는 Spry 프레임워크에 대한 코드 힌트를 지원합니다. Spry 코드 힌트 파일(SpryCodeHints.xml)은 기본 형식은 CodeHints.xml과 동일하지만 method 등의 새로운 키워드가 사용되며 클래스와 클래스 멤버 목록을 연결하기 위한 새로운 classpattern 속성(예: Spry.Data.XMLDataSet)이 있습니다. 각 클래스에 대한 클래스 멤버 목록은 메뉴(메서드, 속성, 이벤트) 내에 중첩되어 있습니다.

<method> 태그 및 해당 속성은 function 태그 및 해당 속성과 유사하지만 부모 menu 태그에는 연결에 필요한 classpattern 속성이 있어야 합니다. 또한 속성을 지정하기 위한 property 태그와 이벤트를 지정하기 위한 event 태그가 있으며 이러한 태그는 [코드 힌트] 팝업 메뉴에서 해당 아이콘으로 표시됩니다. 매개 변수 힌트를 지원하기 위한 parammenu 및 parammenuitem 태그도 있습니다.

다음 예제에서는 SpryCodeHints.xml 파일의 형식을 보여 줍니다. 굵게 표시된 태그에 대한 자세한 내용은 33페이지의 "코드 힌트 태그"를 참조하십시오.

```
<function pattern="XMLDataSet(xmlsource, xpath, {options})"
    caseSensitive="true" />
<menu classpattern="Spry.Data.XMLDataSet">
    <property pattern="url" icon="shared/mm/images/hintMisc.gif" />
    <property pattern="xpath" icon="shared/mm/images/hintMisc.gif" />
    ...
    ...
    <method pattern="getData()" icon="shared/mm/images/hintMisc.gif" />
    <method pattern="getData()" icon="shared/mm/images/hintMisc.gif" />
    <method pattern="loadData()" icon="shared/mm/images/hintMisc.gif" />
    <method pattern="getCurrentRow()" icon= ".../hintMisc.gif" />
    <method pattern="setCurrentRow(rowID)" icon= ".../hintMisc.gif" />
    <method pattern="setCurrentRowNumber(rowNumber)" icon= ".../hintMisc.gif" />
    <method pattern="getRowNumber(rowObj)" icon= ".../hintMisc.gif" />
    <method pattern="setColumnType(columnName, columnType)" icon= ".../hintMisc.gif" />
        <parammenu pattern='"' name="columnName" index="0" type="spryDataReferences">
        </parammenu>
        <parammenu pattern='"' name="columnType" index="1" type="enumerated">
            <parammenuitem label="integer" value="integer" icon=".../hintMisc.gif"/>
            <parammenuitem label="image" value="image" icon=".../hintMisc.gif"/>
            <parammenuitem label="date" value="date" icon=".../hintMisc.gif"/>
            <parammenuitem label="string" value="string" icon=".../hintMisc.gif"/>
        </parammenu>
    </method>
    <method pattern="getColumnType(columnName)" icon= ".../hintMisc.gif" />
    <method pattern="distinct()" icon= ".../hintMisc.gif" />
    <method pattern="getSortColumn()" icon= ".../hintMisc.gif" />
    <method pattern="sort()" icon= ".../hintMisc.gif" />
     ...
     ...
    <event pattern="onCurrentRowChanged" icon= ".../hintMisc.gif" />
    <event pattern="onDataChanged" icon= ".../hintMisc.gif" />
     ...
     ...
</menu>
<function pattern="Accordion(element,{options})" caseSensitive="true" />
<menu classpattern="Spry.Widget.Accordion">
    <method pattern="openNextPanel()" icon= ".../hintMisc.gif" />
    <method pattern="openPreviousPanel()" icon= ".../hintMisc.gif" />
    <method pattern="openFirstPanel()" icon= ".../hintMisc.gif" />
     ...
     ...
</menu>
</function>
<function pattern="XMLDataSet(xmlsource, xpath, {options})" caseSensitive="true">
    <parammenu pattern='{,' name="options" index="2" type="optionArray"
                allowwhitespaceprefix="true">
        <parammenuitem label="sortOnLoad" value="sortOnLoad:"
                icon="shared/mm/images/hintMisc.gif" datatype="string"/>
            <optionparammenu pattern="sortOrderOnLoad" label="sortOrderOnLoad"
                    value="sortOrderOnLoad:" icon="shared/mm/images/hintMisc.gif"
                    type="enumerated" datatype="string">
                <optionparammenuitem label="ascending" value="ascending"
                    icon="shared/mm/images/hintMisc.gif"/>
                <optionparammenuitem label="descending" value="descending"
                    icon="shared/mm/images/hintMisc.gif"/>
            </optionparammenu>
    </parammenu>
</function>
```

클래스 선언

다음과 같은 형식으로 변수와 클래스를 연결하여 클래스를 선언합니다.

```
<variablename>[space][= operator][new keyword][space]<classname>
```

예:

```
var dsFoo = new Spry.Data.XMLDataSet("products.xml", "products/product");
var fooAccordion = new Spry.Widget.Accordion("accordionDivId");
```

클래스 이름 Spry.XML.DataSet의 경우에는 ColorCoding.xml 파일에서 이 클래스를 다시 선언해야 합니다. 그러면 색상 표시 상태 엔진에서는 이 이름이 클래스의 인스턴스임을 인식하고 선언의 왼쪽에 정의된 변수 이름을 가져와서 페이지에 대한 해당 클래스 유형과 함께 변수 목록에 저장합니다(예: 위의 예제에서 변수 fooAccordion과 클래스 유형 Spry.Widget.Accordion).

CodeColoring.xml에서 클래스 이름을 다시 선언하는 구문은 다음과 같습니다.

```
<classlibrary name="Spry Keywords" id="CodeColor_JavascriptSpry">
    <class>Spry.Data.XMLDataSet</class>
    <class>Spry.Widget.Accordion</class>
</classlibrary>
```

위치:

- classlibrary는 클래스를 색상 ID "CodeColor_JavascriptSpry"로 그룹화하는 새 태그입니다.

  class는 클래스 라이브러리의 사용 가능한 개별 클래스를 나열하는 데 사용됩니다. 클래스 목록에는 Debug, Data, Util, Widget, Effect 등의 다양한 spry 패키지에 있는 다른 spry 클래스뿐 아니라 그 밖의 Ajax(비동기 JavaScript 및 XML) 도구 키트나 JavaScript 라이브러리도 포함될 수 있습니다.

Crosstag 속성 코드 힌트

Dreamweaver에서는 Spry 속성 이름 및 값에 대한 코드 힌트를 제공합니다. 이러한 속성은 여러 태그에 공통적으로 사용되므로 각 **tag**.vtm 파일을 열고 Spry 속성 목록을 추가할 필요가 없습니다. 대신 Dreamweaver에서는 Configuration/TagLibraries 디렉토리의 Spry.vtm이라는 단일 VTM 파일에 적용할 수 있는 속성 그룹(예: spry:region, spry:repeat) 및 태그 그룹에 대한 새로운 xml 형식을 제공합니다.

Spry 속성 그룹화 형식

다음 코드에서는 .vtm 파일의 형식을 보여 줍니다. 이 형식을 사용하면 특정 태그에 적용할 속성을 지정할 수 있습니다.

참고: Spry 속성 그룹화 형식을 Spry 프레임워크 외부에도 사용할 수 있습니다.

```
<crosstag_attributes>
    <attributegroup id="group_id_1" name="group_name_1">
        <attrib name = "fooAttr1">
        <attrib name = "barAttr1">
        ...
        <taggroup>
            <tag name = "fooTag1">
            <tag name = "barTag1">
            ...
        </taggroup>
    </attribgroup>
    <attributegroup id="group_id_2" name="group_name_2">
        <attrib name = "fooAttr2">
        <attrib name = "barAttr2">
        ...
        <taggroup>
            <tag name = "fooTag2">
            <tag name = "barTag2">
            ...
        </taggroup>
    </attribgroup>
</crosstag_attributes>
```

위치:

- attributegroup은 다음에 나오는 태그 그룹의 속성을 나열합니다.
- taggroup은 앞에 나오는 속성이 적용되는 태그를 나열합니다.

예제

```
<crosstag_attributes>
    <attribgroup id="spryRegionAttrs" name="Spry1.2">
        <attrib name="spry:region"  type="spryDataSet" allowmultiplevalues="yes"/>
        <attrib name="spry:detailregion"  type="spryDataSet" allowmultiplevalues="yes"/>
        <attrib name="spry:content"/>
        <attrib name="spry:if"/>
        <attrib name="spry:choose">
        <attrib name="spry:when"/>
        <attrib name="spry:default"/>
        <attrib name="spry:state" type="Enumerated">
            <attriboption value="read"   caption="read"/>
            <attriboption value="loading" caption="loading"/>
            <attriboption value="failed"  caption="failed"/>
        </attrib>
        <taggroup>
            <tag name="div"/>
            <tag name="span"/>
            ...
        </taggroup>
    </attribgroup>
    <attribgroup id="spryBehaviorAttrs" name="Spry1.2">
        <attrib name="spry:hover"  type="cssStyle"/>
        <attrib name="spry:select" type="cssStyle"/>
        <attrib name="spry:odd"   type="cssStyle"/>
        <attrib name="spry:even"   type="cssStyle"/>
        <taggroup>
            <tag name="a"/>
            <tag name="abbr"/>
            <tag name="acronym"/>
            ...
        </taggroup>
    </attribgroup>
</crosstag_attributes>
```

코드 힌트 태그

코드 힌트 XML 파일에서는 코드 힌트 메뉴를 정의하는 다음 태그를 사용합니다. 이러한 태그를 사용하여 코드 힌트 메뉴를 추가로 정의할 수 있습니다.

<codehints>

설명

codehints 태그는 CodeHints.xml 및 SpryCodeHints.xml 파일의 루트입니다.

속성

없음

내용

하나 이상의 menugroup 태그입니다.

컨테이너

없음

예제

```
<codehints>
```

<menugroup>

설명

각 menugroup 태그는 하나의 메뉴 유형에 해당합니다. [환경 설정] 대화 상자에서 [코드 힌트] 범주를 선택하면 Dreamweaver에 정의된 메뉴 유형을 볼 수 있습니다. [환경 설정] 대화 상자를 표시하려면 [편집] 메뉴에서 [환경 설정]을 선택합니다.

메뉴 그룹을 새로 만들거나 기존 그룹에 추가할 수 있습니다. 메뉴 그룹은 사용자가 [환경 설정] 대화 상자를 사용하여 활성화하거나 비활성화할 수 있는 메뉴의 논리적 모음입니다.

속성

name,enabled,id,version

- name 속성은 [환경 설정] 대화 상자의 [코드 힌트] 범주에 있는 메뉴 그룹 목록에 나타나는 지역화된 이름입니다.
- enabled 속성은 메뉴 그룹이 현재 체크되어 있는지, 즉 활성화되어 있는지 여부를 나타냅니다. 활성화된 메뉴 그룹은 [환경 설정] 대화 상자의 [코드 힌트] 범주에서 옆에 체크 표시가 있는 상태로 나타납니다. 메뉴 그룹을 활성화하려면 true 값을 지정하고 비활성화하려면 false 값을 지정합니다.
- id 속성은 메뉴 그룹을 나타내는 지역화되지 않은 식별자입니다.

내용

description, menu 및 function 태그입니다.

컨테이너

codehints 태그입니다.

예제

```
<menugroup name="Session Variables" enabled="true" id="Session_Code_Hints" version="1.4.2">
```

<description>

설명

description 태그에는 Dreamweaver의 [환경 설정] 대화 상자에서 메뉴 그룹을 선택할 때 표시되는 텍스트가 포함됩니다. 이 설명 텍스트는 메뉴 그룹 목록 아래에 표시됩니다. 경우에 따라 설명 텍스트에는 단일 a 태그가 포함되기도 합니다. 이때 href 속성은 사용자가 링크를 클릭할 때 Dreamweaver에서 실행되는 JavaScript URL이어야 합니다. 특수 문자나 허용되지 않는 문자를 문자열에 포함하여 Dreamweaver에서 텍스트로 처리되도록 하려면 XML CDATA 구조를 사용합니다.

속성

없음

내용

설명 텍스트입니다.

컨테이너

menugroup 태그입니다.

예제

```
<description>
<![CDATA[ To add or remove tags and attributes, use the
    <a href="javascript:dw.tagLibrary.showTagLibraryEditor()">Tag Library Editor</a>.]]>
</description>
```

<menu>

설명

이 태그는 단일 팝업 메뉴를 설명합니다. 사용자가 pattern 속성에 있는 문자열의 마지막 문자를 입력할 때마다 Dreamweaver에서 메뉴가 열립니다. 예를 들어, Session 변수의 내용을 보여 주는 메뉴에는 "Session."과 동일한 pattern 속성이 있을 수 있습니다.

속성

pattern, doctypes, casesensitive, classpattern, displayrestriction, alias

- pattern 속성은 Dreamweaver에서 코드 힌트 메뉴가 열리도록 하는 입력 문자의 패턴을 지정합니다. 패턴의 첫 글자가 문자, 숫자 또는 밑줄이면 문서에서 해당 패턴의 앞에 오는 글자가 문자, 숫자 또는 밑줄이 아닌 경우에만 해당 메뉴가 표시됩니다. 예를 들어 패턴이 "Session."일 경우 사용자가 "my_Session."을 입력하면 메뉴가 표시되지 않습니다.
- doctypes 속성은 지정된 문서 형식에 대해서만 메뉴가 활성화되도록 지정합니다. 이 속성을 사용하면 ASP-JS(ASP-JavaScript), JSP(Java Server Pages), Adobe ColdFusion 등에 대해 각기 다른 함수 이름 목록을 지정할 수 있습니다. doctypes 속성은 쉼표로 구분된 문서 형식 ID 목록으로 지정할 수 있습니다. Dreamweaver 문서 형식 목록은 Dreamweaver Configuration/Documenttypes/MMDocumentTypes.xml 파일을 참조하십시오.
- casesensitive 속성은 패턴에서 대/소문자를 구분할지 여부를 지정합니다. casesensitive 속성에는 true, false 또는 doctypes 속성에 지정한 쉼표로 구분된 목록의 하위 집합을 지정할 수 있습니다. 문서 형식 목록을 사용하면 특정 문서 형식의 경우에만 패턴의 대/소문자를 구분하도록 지정할 수 있습니다. 이 속성을 생략하는 경우 기본값은 false로 설정됩니다. casesensitive 속성의 값이 true인 경우 사용자가 입력한 텍스트가 pattern 속성에 지정된 패턴과 정확하게 일치할 때에만 코드 힌트 메뉴가 열립니다. casesensitive 속성의 값이 false인 경우에는 패턴이 소문자이고 텍스트는 대문자이더라도 메뉴가 표시됩니다.
- classpattern 속성은 클래스 멤버 목록과 클래스를 연결합니다.
- displayrestriction 속성은 CodeColoring.xml에 정의된 색상 코딩 체계에 따라 특정 프로그래밍 언어 구문 블록으로 코드 힌트 메뉴를 제한하는 데 사용됩니다. 예를 들어, displayrestriction= "JavaScript"인 경우 코드 힌트 메뉴는 JavaScript 구문 블록으로 제한됩니다.
- alias 인수는 pattern 또는 classpattern 인수에 나열된 패턴이 아닌 대체 패턴으로 코드 힌트를 호출하는 데 사용됩니다. 이 인수는 선택 사항입니다.

내용

menuitem 태그입니다.

컨테이너

menugroup 태그입니다.

예제

```
<menu pattern="CGI." doctypes="ColdFusion">
```

<menuitem>

설명

이 태그는 코드 힌트 팝업 메뉴의 항목에 텍스트를 지정합니다. menuitem 태그는 사용자가 항목을 선택할 때 텍스트에 삽입해야 하는 값도 지정합니다.

속성

label, value, {icon}, {texticon}, object, source

- label 속성은 Dreamweaver의 팝업 메뉴에 표시되는 문자열입니다.
- value 속성은 사용자가 명령을 선택할 때 문서에 삽입되는 문자열입니다. 사용자가 메뉴에서 항목을 선택하고 Enter 또는 Return 키를 누르면 해당 메뉴가 열린 이후 사용자가 입력한 모든 텍스트가 교체됩니다. 메뉴가 열리기 전에 사용자가 패턴 일치 문자를 입력했으므로 해당 문자는 다시 삽입되지 않습니다. 예를 들어 앰퍼샌드(&) 문자에 대한 HTML 엔터티인 &를 삽입하려는 경우 다음과 같이 menu 및 menuitem 태그를 정의합니다.

  ```
  <menu pattern="&amp;amp;">
  <menuitem label="&amp;amp;amp;" value="amp;" texticon="&amp;amp;"/>
  ```

  앰퍼샌드(&) 문자는 메뉴가 열리기 전에 사용자가 입력했기 때문에 value 속성에 포함되지 않습니다.
- icon 속성은 Dreamweaver에서 메뉴 텍스트 왼쪽에 아이콘으로 표시될 이미지 파일의 경로를 지정합니다. 이 속성은 선택 사항입니다. 이 위치는 Configuration 폴더에 대해 상대적인 URL로 표현됩니다.
- texticon 속성은 이미지 파일 대신 아이콘 영역에 표시될 텍스트 문자열을 지정합니다. 이 속성은 선택 사항입니다. 이 속성은 [HTML 엔터티] 메뉴에 사용됩니다.
- object 속성은 메뉴 항목이 속한 유형을 참조합니다. 예를 들어, 내장 데이터 유형: 문자열 또는 사용자 정의 데이터 유형인 사용자 정의 JavaScript 파일입니다.
- source 속성은 메뉴 항목이 정의되어 있거나 제공되는 위치를 참조합니다. 예를 들어, DOM/Javascript/ custom file.js입니다.

내용

없음

컨테이너

menu 태그입니다.

예제

```
<menuitem label="CONTENT_TYPE" value="&amp;quot;CONTENT_TYPE&amp;quot;)
    " icon="shared/mm/images/hintMisc.gif" />
```

<function>

설명

CodeHints.xml 파일에서 사용됩니다. 이 태그는 menu 태그 대신 사용되어 코드 힌트 팝업 메뉴의 함수 인수와 객체 메서드를 지정합니다. [코드] 뷰에 함수 또는 메서드 이름을 입력하면 Dreamweaver에 함수 원형의 메뉴가 나타나고 현재 인수는 굵게 표시됩니다. 쉼표를 입력할 때마다 메뉴가 업데이트되어 다음 인수가 굵게 표시됩니다. 예를 들어 ColdFusion 문서에서 함수 이름 ArrayAppend를 입력하면 코드 힌트 메뉴에 ArrayAppend(array, value)가 표시됩니다. array 다음에 쉼표를 입력하면 메뉴가 업데이트되어 ArrayAppend(array, value)가 표시됩니다.

객체 메서드의 경우 객체 이름을 입력하면 해당 객체에 대해 정의된 메서드의 메뉴가 열립니다.

인식된 함수는 Configuration/CodeHints 폴더의 XML 파일에 저장됩니다.

속성
pattern, doctypes, casesensitive

- pattern 속성은 함수의 이름과 해당 인수 목록을 지정합니다. 메서드의 경우 pattern 속성은 객체 이름, 메서드 이름 및 메서드 인수를 설명합니다. 함수 이름의 경우 코드 힌트 메뉴는 사용자가 functionname(을 입력할 때 표시됩니다. 이 메뉴에는 해당 함수의 인수 목록이 나타납니다. 객체 메서드의 경우 사용자가 objectname.(마침표 포함)을 입력하면 코드 힌드 메뉴가 나타납니다. 이 메뉴에는 해당 객체에 지정된 메서드가 표시됩니다. 그런 다음 함수의 경우와 같은 방식으로 해당 메서드에 대한 인수 목록이 코드 힌트 메뉴에 열립니다.
- doctypes 속성은 지정된 문서 형식에 대해서만 메뉴가 활성화되도록 지정합니다. 이 속성을 사용하면 ASP-JS(ASP-JavaScript), JSP(Java Server Pages), Adobe ColdFusion 등에 대해 각기 다른 함수 이름 목록을 지정할 수 있습니다. doctypes 속성은 쉼표로 구분된 문서 형식 ID 목록으로 지정할 수 있습니다. Dreamweaver 문서 형식 목록은 Dreamweaver Configuration/Documenttypes/MMDocumentTypes.xml 파일을 참조하십시오.
- casesensitive 속성은 패턴에서 대/소문자를 구분할지 여부를 지정합니다. casesensitive 속성에는 true, false 또는 doctypes 속성에 지정한 쉼표로 구분된 목록의 하위 집합을 지정할 수 있습니다. 문서 형식 목록을 사용하면 특정 문서 형식의 경우에만 패턴의 대/소문자를 구분하도록 지정할 수 있습니다. 이 속성을 생략하는 경우 기본값은 false로 설정됩니다. casesensitive 속성의 값이 true인 경우 사용자가 입력한 텍스트가 pattern 속성에 지정된 패턴과 정확하게 일치할 때에만 코드 힌트 메뉴가 표시됩니다. casesensitive 속성의 값이 false인 경우에는 패턴이 소문자이고 텍스트는 대문자이더라도 메뉴가 표시됩니다.

내용
없음

컨테이너
menugroup 태그입니다.

예제
```
// function example
<function pattern="CreateDate(year, month, day)" DOCTYPES="ColdFusion" />
// object method example
<function pattern="application.getAttribute(String name)" DOCTYPES="JSP" />
```

<method>

설명
Spry 프레임워크에 사용됩니다. 이 태그는 menu 태그 대신 사용되어 코드 힌트 팝업 메뉴의 메서드를 지정합니다. [코드] 뷰에 메서드 이름을 입력하면 메서드 원형 메뉴가 열려 해당 메서드의 매개 변수 목록이 제공되고 입력되는 매개 변수 순서가 추적됩니다. 매개 변수가 없는 메서드의 경우 괄호 "()" 문자를 추가하여 해당 메서드 호출이 닫힙니다.

현재 인수는 굵게 표시됩니다. 쉼표를 입력할 때마다 메뉴가 업데이트되어 다음 인수가 굵게 표시됩니다. 객체 메서드의 경우 객체 이름을 입력하면 해당 객체에 대해 정의된 메서드의 메뉴가 열립니다.

속성
pattern, icon, object, source, constructor, static, retType

- pattern 속성은 메서드의 이름과 해당 인수 목록을 지정합니다. 또한 객체와 메서드의 이름과 메서드의 인수를 설명합니다. 메뉴에는 해당 메서드의 인수 목록이 표시됩니다. 코드 힌트 메뉴는 사용자가 objectname.(마침표 포함)을 입력할 때 나타납니다. 이 메뉴에는 해당 객체에 지정된 메서드가 표시됩니다. 그런 다음 코드 힌트 메뉴에 함수의 경우와 같은 방법으로 해당 메서드에 대한 인수 목록이 열립니다.
- icon 속성은 사용할 아이콘을 지정합니다.
- object 속성은 메뉴 항목이 속한 유형을 참조합니다. 예를 들어, 내장 데이터 유형: 문자열 또는 사용자 정의 데이터 유형인 사용자 정의 JavaScript 파일입니다.

- source 속성은 메뉴 항목이 정의되어 있거나 제공되는 위치를 참조합니다. 예를 들어, DOM/Javascript/custom file.js입니다.
- constructor 속성은 부울 값입니다. constructor = true는 객체 인스턴스를 만들고 다른 객체 메서드와 비교하여 별도로 표시되는 메서드를 참조합니다.
- static 속성은 부울 값입니다. static = true는 메서드가 특정 객체 인스턴스에 적용되지 않고 객체 유형 자체에 적용됨을 나타냅니다. 예를 들면 다음과 같습니다.

  ```
  Date.parse(dateString)
  ```

- retType 속성은 계단식 코드 힌트를 지원하기 위한 객체 유형이 될 수 있는 메서드 반환 유형을 참조합니다.

내용
없음

컨테이너
menu 태그입니다.

<parammenu>

설명
모든 객체(JavaScript)에 사용되어 메서드 또는 함수에 사용되는 매개 변수의 매개 변수 힌트를 지정합니다.

속성
pattern, name, index, type

- pattern 속성은 코드 힌트 메뉴가 나타나게 하는 문자를 지정합니다. 이 인수는 필수입니다.
- name 속성은 매개 변수 이름을 지정합니다. 이 인수는 필수입니다.
- index 속성은 코드 힌트를 표시할 매개 변수의 인덱스 번호(0부터 시작)를 지정합니다. 이 인수는 필수입니다.
- type 속성은 데이터 유형을 지정합니다. 다음과 같은 데이터 유형이 지원됩니다.
  - enumerated(기본값): 표시할 중첩된 <optionparammenuitem>의 목록을 나타냅니다.
  - spryDataReferences: Spry 데이터 세트 열 목록을 나타냅니다.
  - cssStyle: 해당 페이지에 사용할 수 있는 CSS 클래스 목록을 나타냅니다.
  - cssId: 해당 페이지에 사용할 수 있는 CSS 선택기 ID 규칙 목록을 나타냅니다.
  - optionArray: 표시할 중첩된 <optionparammenu> 및 <parammenuitem>의 목록을 나타냅니다. 옵션 배열 매개 변수 유형을 지원하기 위해 사용됩니다.

내용
없음

컨테이너
method 또는 function 태그입니다.

<parammenuitem>

설명
모든 객체(JavaScript)에 사용되어 메서드 또는 함수에 사용되는 매개 변수의 매개 변수 힌트를 지정합니다.

속성
label, value, icon, {datatype}, object, source

- label 속성은 Dreamweaver에서 표시해야 하는 이름을 지정합니다. 이 인수는 필수입니다.
- value 속성은 해당 항목이 코드 힌트 메뉴에 선택되어 있을 때 Dreamweaver에서 드롭해야 하는 값을 지정합니다. 이 인수는 필수입니다.
- icon 속성은 Dreamwaever에서 코드 힌트 메뉴에 사용해야 하는 아이콘을 지정합니다. 이 인수는 필수입니다.
- datatype 속성을 사용하면 사용자가 코드 힌트 메뉴에서 값을 선택할 때 닫는 인용 부호가 추가되어야 함을 나타내는 string을 지정할 수 있습니다. 이 인수는 선택 사항입니다.
- object 속성은 메뉴 항목이 속한 유형을 참조합니다. 예를 들어, 내장 데이터 유형: 문자열 또는 사용자 정의 데이터 유형인 사용자 정의 JavaScript 파일입니다.
- source 속성은 메뉴 항목이 정의되어 있거나 제공되는 위치를 참조합니다. 예를 들어, DOM/Javascript/custom file.js입니다.

내용
없음

컨테이너
parammenu 태그입니다.

<optionparammenu>

설명
모든 객체(JavaScript)에 사용되어 메서드 또는 함수에 사용되는 인수의 옵션 배열 힌트를 지정합니다. 옵션 배열은 option:value 형식의 하위 인수가 포함될 수 있는 인수입니다. 대부분의 Spry 객체는 옵션 배열 인수를 사용하여 사용자가 데이터 세트, 위젯 또는 효과와 같은 객체의 비헤이비어를 구성할 수 있도록 합니다. 옵션 배열은 일반적으로 {option1: value1, option2: value2, option3: value3, ...}과 같이 나타납니다.

속성
pattern, label, value, icon, type

- pattern 속성은 코드 힌트 메뉴가 나타나게 하는 문자를 지정합니다. 이 인수는 필수입니다.
- label 속성은 매개 변수 이름을 지정합니다. 이 인수는 필수입니다.
- value 속성은 사용자가 코드 힌트를 선택할 때 삽입할 매개 변수 값을 지정합니다. 이 인수는 필수입니다.
- icon 속성은 사용할 아이콘을 지정합니다. 이 인수는 필수입니다.
- type 속성은 데이터 유형을 지정합니다. 다음과 같은 데이터 유형이 지원됩니다.
  - enumerated(기본값): 표시할 중첩된 optionparammenuitem의 목록을 나타냅니다.
  - spryDataReferences: Spry 데이터 세트 열 목록을 나타냅니다.
  - cssStyle: 해당 페이지에 사용할 수 있는 CSS 클래스 목록을 나타냅니다.
  - cssId: 해당 페이지에 사용할 수 있는 CSS 선택기 ID 규칙 목록을 나타냅니다.

내용
없음

컨테이너
유형이 optionArray인 parammenu 태그입니다.

<optionparammenuitem>

설명
모든 객체(JavaScript)에 사용되어 메서드 또는 함수에 사용되는 매개 변수의 매개 변수 힌트를 지정합니다.

속성
label, value, icon, {datatype}

- label 속성은 표시할 이름을 지정합니다. 이 인수는 필수입니다.
- value 속성은 해당 항목이 코드 힌트 메뉴에 선택되어 있을 때 드롭할 값을 지정합니다. 이 인수는 필수입니다.
- icon 속성은 코드 힌트 메뉴에 사용할 아이콘을 지정합니다. 이 인수는 필수입니다.
- datatype 속성을 사용하면 사용자가 코드 힌트 메뉴에서 값을 선택할 때 닫는 인용 부호가 추가됨을 나타내는 string을 지정할 수 있습니다. 이 인수는 선택 사항입니다.

내용
없음

컨테이너
<optionparammenu> 태그입니다.

<property>

설명
이 태그는 객체의 속성이나 필드를 설명하고 다음과 같은 표준 속성을 가집니다.

속성
label, value, icon, object, source, static, propType, item

- label 속성은 Dreamweaver의 팝업 메뉴에 표시되는 문자열입니다.
- value 속성은 사용자가 명령을 선택할 때 문서에 삽입되는 문자열입니다. 사용자가 메뉴에서 항목을 선택하고 Enter 또는 Return 키를 누르면 해당 메뉴가 열린 이후 사용자가 입력한 모든 텍스트가 교체됩니다. 메뉴가 열리기 전에 사용자가 패턴 일치 문자를 입력했으므로 해당 문자는 다시 삽입되지 않습니다.
- icon 속성은 Dreamweaver에서 메뉴 텍스트 왼쪽에 아이콘으로 표시될 이미지 파일의 경로를 지정합니다. 이 속성은 선택 사항입니다. 이 위치는 Configuration 폴더에 대해 상대적인 URL로 표현됩니다.
- object 속성은 메뉴 항목이 속한 유형을 참조합니다. 예를 들어, 내장 데이터 유형: 문자열 또는 사용자 정의 데이터 유형인 사용자 정의 JavaScript 파일입니다.
- source 속성은 메뉴 항목이 정의되어 있거나 제공되는 위치를 참조합니다. 예를 들어, DOM/Javascript/ custom file.js입니다.
- static 속성은 부울 값입니다. static = true는 메서드가 특정 객체 인스턴스에 적용되지 않고 객체 유형 자체에 적용됨을 나타냅니다. 예를 들면 다음과 같습니다.

  ```
  Number.MAX_VALUE
  ```

- propType 속성은 속성의 계단식 힌트를 지원하기 위한 객체 유형이 될 수 있는 속성 유형을 참조합니다. 예를 들면 다음과 같습니다.

```
domElement.innerHTML.<code hints for String type>
```

- item 속성은 propType 속성이 컬렉션 컨테이너 유형인 경우 요소의 유형을 참조합니다. item은 컨테이너에 있는 각 항목의 유형을 지정합니다. 이때 컨테이너는 동일한 유형의 요소로 구성된 동종 집합이라고 가정됩니다.

내용
없음

컨테이너
menu 태그입니다.

<event>

설명
이 태그는 객체의 이벤트를 설명하고 다음과 같은 표준 속성을 가집니다.

속성
label, icon, source, object

- label 속성은 이벤트의 이름입니다.
- icon 속성은 Dreamweaver에서 메뉴 텍스트 옆에 아이콘으로 표시하는 이미지 파일의 경로를 지정합니다.
- source 속성은 메뉴 항목이 정의되어 있거나 제공되는 위치를 참조합니다.
- object 속성은 메뉴 항목이 속한 유형을 참조합니다.

컨테이너
menu 태그입니다.

예제

```
<event label="onblur" source="DOM 1&amp;amp;2" icon="shared/mm/images/codeHintEvent.gif"/>
```

코드 색상 표시

Dreamweaver에서는 [코드] 뷰에 나타나는 코드 색상 표시 체계를 사용자 정의하거나 확장하여 특정 체계에 키워드를 새로 추가하거나 새로운 문서 형식에 대해 코드 색상 표시 체계를 추가할 수 있습니다. 예를 들어 클라이언트측 스크립트에 사용할 JavaScript 함수를 개발하는 경우 이 함수 이름을 keywords 섹션에 추가하여 [환경 설정] 대화 상자에 지정된 색상으로 표시되도록 할 수 있습니다. 마찬가지로, 응용 프로그램 서버에 사용할 새로운 프로그래밍 언어를 개발하고 Dreamweaver 사용자가 페이지를 작성할 때 활용할 수 있도록 새 문서 형식을 배포하려는 경우 해당 문서 형식에 대한 코드 색상 표시 체계를 추가할 수 있습니다.

Dreamweaver에는 사용자가 직접 편집할 수 있는 코드 색상 표시 XML 파일을 다시 로드할 수 있는 JavaScript 함수 dreamweaver.reloadCodeColoring()이 있습니다. 이 함수에 대한 자세한 내용은 **Dreamweaver API 참조** 설명서를 참조하십시오.

코드 색상 표시 체계를 업데이트하거나 새로운 체계를 추가하려면 코드 색상 표시 정의 파일을 수정해야 합니다.

코드 색상 표시 파일

Dreamweaver에서는 코드 색상 표시 스타일과 체계를 Configuration/CodeColoring 폴더에 저장되는 XML 파일에 정의합니다. 코드 색상 표시 스타일 파일은 구문 정의에 정의된 필드에 대한 스타일을 정의합니다. 이 파일의 루트 노드는 <codeColors>입니다. 코드 색상 표시 체계 파일은 코드 색상 표시 구문을 정의하며 이 파일의 루트 노드는 <codeColoring>입니다.

Dreamweaver에서 제공하는 코드 색상 표시 스타일 파일은 Colors.xml이고, 코드 색상 표시 구문 파일은 CodeColoring.xml, ASP JavaScript.xml, ASP VBScript.xml, ASP.NET CSharp.xml 및 ASP.NET VB.xml입니다.

다음은 코드 색상 표시 스타일 파일의 태그 계층 구조를 보여 주기 위해 Colors.xml 파일의 내용 중 일부를 발췌한 것입니다.

```
<codeColors>
    <colorGroup>
        <syntaxColor id="CodeColor_HTMLEntity" bold="true" />
        <syntaxColor id="CodeColor_JavascriptNative" text="#009999" />
        <syntaxColor id="CodeColor_JavascriptNumber" text="#FF0000" />
...
        <tagColor id="CodeColor_HTMLStyle" text="#990099" />
        <tagColor id="CodeColor_HTMLTable" text="#009999" />
        <syntaxColor id="CodeColor_SpryAttributes" text="#FF6208" />
...
    </colorGroup>
</codeColors>
```

색상은 RGB(Red-Green-Blue) 16진수 값으로 지정됩니다. 예를 들어 앞의 XML 코드에서 text="009999" 문은 ID "CodeColor_JavascriptNative"에 파랑-녹색(암녹색) 색상을 지정합니다.

다음은 코드 색상 표시 체계 파일의 태그 계층 구조를 보여 주기 위해 CodeColoring.xml 파일의 내용 중 일부를 발췌한 것입니다. 여기에서는 코드 색상 표시 스타일 파일과 코드 색상 표시 체계 파일 간의 관계도 보여 줍니다.

```
<codeColoring>
    <scheme name="Text" id="Text" doctypes="Text" priority="1">
        <ignoreTags>Yes</ignoreTags>
        <defaultText name="Text" id="CodeColor_TextText" />
        <sampleText doctypes="Text">
<![CDATA[Default file syntax highlighting.
The quick brown fox
jumped over the lazy dog.
]]>
        </sampleText>
    </scheme>
    <scheme name="HTML" id="HTML" doctypes=
"ASP.NET_VB,ASP.NET_CSharp,ASP-JS,ASP-VB,ColdFusion,CFC,HTML,JSP,PHP_MySQL,LibraryItem,
    WML,XSLT" priority="50">
        <ignoreCase>Yes</ignoreCase>
        <ignoreTags>No</ignoreTags>
        <defaultText name="Text" id="CodeColor_HTMLText" />
        <defaultTag name="Other Tags" id="CodeColor_HTMLTag" />
        <defaultAttribute />
        <commentStart name="Comment" id="CodeColor_HTMLComment"><![CDATA[<!--]]>
            </commentStart>
    ...
    <tagGroup name="HTML Anchor Tags" id="CodeColor_HTMLAnchor" taglibrary="DWTagLibrary_html"
         tags="a" />
    <tagGroup name="HTML Form Tags" id="CodeColor_HTMLForm" taglibrary="DWTagLibrary_html" tags
="select,form,input,option,textarea" />
```

Colors.xml 파일의 syntaxColor 및 tagColor 태그는 id 문자열 값에 색상 및 스타일 값을 지정합니다. id 값은 CodeColoring.xml 파일에 사용되어 scheme 태그에 스타일을 지정합니다. 예를 들어 앞의 CodeColoring.xml 내용에서 defaultTag 태그의 id는 "CodeColor_HTMLComment"입니다. Colors.xml 파일에서 id 값 "CodeColor_HTMLComment"에는 회색을 나타내는 text= 값 "#999999"가 지정되어 있습니다.

Dreamweaver에 포함된 코드 색상 표시 체계로는 Default, HTML, JavaScript, ASP_JavaScript, ASP_VBScript, JSP 및 ColdFusion이 있습니다. 기본 체계는 "Text"와 같은 id 값을 가지며, Dreamweaver에서는 정의된 코드 색상 체계가 없는 문서 유형일 경우에 이 기본 체계를 사용합니다.

코드 색상 표시 파일에는 아래에 설명된 다음 태그가 포함됩니다.

scheme, blockEnd, blockStart, brackets, charStart, charEnd, charEsc, commentStart, commentEnd, cssImport/, cssMedia/, cssProperty/, cssSelector/, cssValue/, defaultAttribute, defaultTag, defaultText/, endOfLineComment, entity/, functionKeyword, idChar1, idCharRest, ignoreCase, ignoreMMTParams, ignoreTags, isLocked, keyword, keywords, numbers/, operators, regexp, sampleText, searchPattern, stringStart, stringEnd, stringEsc, tagGroup

<scheme>

설명

scheme 태그는 코드 텍스트 블록에 대한 코드 색상 표시를 지정합니다. 한 파일에 여러 색상 표시 체계를 정의하여 스크립트 또는 태그 언어에 따라 다른 색상을 지정할 수 있습니다. 각 체계에는 우선 순위가 있어 서로 체계가 다른 텍스트 블록을 중첩시킬 수 있습니다.

Dreamweaver CS4부터 코드 색상 표시 파서에서는 ID가 동일한 <scheme> 태그를 병합합니다. 따라서 충돌하지 않는 모든 태그가 기존 <scheme>에 추가됩니다. 충돌이 발생할 경우에는 파일 날짜가 최신인 체계가 우선합니다.

속성

name, id, priority, {doctypes}

- ***name="scheme_name"*** 체계 이름을 지정하는 문자열입니다. 체계 이름은 [코드 색상 표시 체계 편집] 대화 상자에 나타납니다. 이때 HTML Comment와 같이 체계 이름과 필드 이름이 결합되어 표시됩니다. 이름을 지정하지 않으면 [코드 색상 표시 체계 편집] 대화 상자에 체계에 대한 필드가 표시되지 않습니다. [코드 색상 표시 체계 편집] 대화 상자에 대한 자세한 내용은 60페이지의 “체계 편집”을 참조하십시오.
- ***id="id_string"*** 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.
- ***priority="string"*** "1"부터 "99"까지의 값입니다. 가장 높은 우선 순위는 "1"입니다. 이 속성은 체계의 우선 순위를 지정합니다. 우선 순위가 더 높은 블록 내부에 존재하는 블록은 무시되고, 우선 순위가 같거나 낮은 블록 내부에 존재하는 블록이 우선적으로 처리됩니다. 우선 순위를 지정하지 않을 경우 기본값은 "50"입니다.
- ***doctypes="doc_list"*** 선택 사항이며, 현재 코드 색상 표시 체계를 지정할 문서 형식을 쉼표로 구분된 목록으로 지정합니다. 이 값은 서로 다른 시작 블록과 끝 블록에서 동일한 Extension을 사용할 경우 발생하는 충돌을 해결하는 데 필요합니다.

내용

```
blockEnd, blockStart, brackets, charStart, charEnd, charEsc, commentStart, commentEnd,
cssProperty/, cssSelector/, cssValue/, defaultAttribute, defaultText/, endOfLineComment,
entity/, functionKeyword, idChar1, idCharRest, ignoreCase, ignoreMMTParam, ignoreTags,
keywords, numbers/, operators, regexp, sampleText, searchPattern, stringStart, stringEnd,
stringEsc, urlProtocol, urlProtocols
```

컨테이너

codeColoring 태그입니다.

예제

```
<scheme name="Text" id="Text" doctypes="Text" priority="1">
```

<blockEnd>

설명

선택 사항이며, 현재 체계에 대한 텍스트 블록의 끝을 구분하는 텍스트 값입니다. blockEnd 및 blockStart 태그는 반드시 쌍을 이뤄야 하며 두 태그의 조합이 고유해야 합니다. 태그의 값은 대/소문자가 구분되지 않습니다. blockEnd 값은 한 문자도 가능합니다. 이 태그는 여러 번 사용할 수 있습니다. blockEnd 문자열에 대한 자세한 내용은 58페이지의 “와일드카드 문자”를 참조하십시오.

속성

없음

예제

```
<blockEnd><![CDATA[--->]]></blockEnd>
```

<blockStart>

설명

선택 사항이며, 현재 색상 표시 체계를 다른 색상 표시 체계에 포함할 수 있는 경우에만 지정합니다. blockStart 및 blockEnd 태그는 반드시 쌍을 이뤄야 하며 두 태그의 조합이 고유해야 합니다. 태그의 값은 대/소문자가 구분되지 않습니다. blockStart 값의 길이는 2자 이상이어야 합니다. 이 태그는 여러 번 사용할 수 있습니다. blockStart 문자열에 대한 자세한 내용은 58페이지의 “와일드카드 문자”를 참조하십시오. blockStartscheme 속성에 대한 자세한 내용은 55페이지의 “scheme 블록 구분 기호 색상 표시”를 참조하십시오.

속성

canNest, doctypes, id, name, scheme

- canNest 색상 표시 체계가 해당 체계 내에 중첩될 수 있는지 여부를 지정합니다. 이 태그의 값은 Yes 또는 No 중 하나입니다. 기본값은 No입니다.
- doctypes="***doc_type1, doc_type2,...*** " 필수 사항이며, 이 코드 색상 표시 체계를 중첩시킬 수 있는 문서 형식을 쉼표로 구분된 목록으로 지정합니다. 문서 형식은 Dreamweaver의 Configuration/Document Types/MMDocumentTypes.xml 파일에 정의됩니다.
- id="***id_string***"scheme="customText"인 경우 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.
- name="***display_name***"scheme="customText"인 경우 [코드 색상 표시 체계 편집] 대화 상자에 나타나는 문자열입니다.
- scheme 필수 사항이며, blockStart 및 blockEnd 문자열의 색상 표시 방법을 정의합니다. scheme 속성에 사용할 수 있는 값에 대한 자세한 내용은 55페이지의 “scheme 블록 구분 기호 색상 표시”를 참조하십시오.

예제

```
<blockStart doctypes="ColdFusion,CFC" scheme="innerText" canNest="Yes"><![CDATA[<!---]]>
</blockStart>
```

<brackets>

설명

괄호를 나타내는 문자 목록입니다.

속성

name, id

- name="***bracket_name***" 괄호 목록에 이름을 지정하는 문자열입니다.

• id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제

```
<brackets name="Bracket" id="CodeColor_JavaBracket"><![CDATA[{[()]}]]>
</brackets>
```

<charStart>

설명

문자 시작 구분 기호를 나타내는 텍스트 문자열을 포함합니다. charStart 및 charEnd 태그를 쌍으로 지정해야 합니다. charStart … charEnd 쌍은 여러 번 사용할 수 있습니다.

속성

없음

예제

```
<charStart><![CDATA[']]></charStart>
```

<charEnd>

설명

문자 끝 구분 기호를 나타내는 텍스트 문자열을 포함합니다. charStart 및 charEnd 태그를 쌍으로 지정해야 합니다. charStart … charEnd 쌍은 여러 번 사용할 수 있습니다.

속성

없음

예제

```
<charEnd><![CDATA[']]></charEnd>
```

<charEsc>

설명

이스케이프 문자를 나타내는 텍스트 문자열을 포함합니다. charEsc 태그는 여러 번 사용할 수 있습니다.

속성

없음

예제

```
<charEsc><![CDATA[\]]></charEsc>
```

<commentStart>

설명

주석 블록의 시작을 구분하는 텍스트 문자열입니다. commentStart 및 commentEnd 태그를 쌍으로 지정해야 합니다. commentStart … /commentEnd 쌍은 여러 번 사용할 수 있습니다.

속성
없음

예제
```
<commentStart><![CDATA[<%--]]></commentStart>
```

<commentEnd>

설명
주석 블록의 끝을 구분하는 텍스트 문자열입니다. commentStart 및 commentEnd 태그를 쌍으로 지정해야 합니다. commentStart … /commentEnd 쌍은 여러 번 사용할 수 있습니다.

속성
없음

예제
```
<commentEnd><![CDATA[--%>]]></commentEnd>
```

<cssImport/>

설명
CSS에서 style 요소의 @import 함수에 대한 코드 색상 표시 규칙을 나타내는 빈 태그입니다.

속성
name, id

- name="***cssImport_name***" CSS @import 함수에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제
```
<cssImport name="@import" id="CodeColor_CSSImport" />
```

<cssMedia/>

설명
CSS에서 style 요소 @media 함수에 대한 코드 색상 표시 규칙을 나타내는 빈 태그입니다.

속성
name, id

- name="***cssMedia_name***" CSS @media 함수에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제
```
<cssMedia name="@media" id="CodeColor_CSSMedia" />
```

<cssProperty/>

설명
CSS 규칙을 나타내며 코드 색상 표시 속성을 저장하는 빈 태그입니다.

속성
name, id

- name="***cssProperty_name***" CSS 속성에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

코드 색상 환경 설정
CSS 속성

예제

```
<cssProperty name="Property" id="CodeColor_CSSProperty" />
```

<cssSelector/>

설명
CSS 규칙을 나타내며 코드 색상 표시 속성을 저장하는 빈 태그입니다.

속성
name, id

- name="***cssSelector_name***" CSS 선택기에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제

```
<cssSelector name="Selector" id="CodeColor_CSSSelector" />
```

<cssValue/>

설명
CSS 규칙을 나타내며 코드 색상 표시 속성을 저장하는 빈 태그입니다.

속성
name, id

- name="***cssValue_name***" CSS 값에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제

```
<cssValue name="Value" id="CodeColor_CSSValue" />
```

<defaultAttribute>

설명
선택 사항이며, 이 태그는 ignoreTags="No"인 태그 기반 구문에만 적용됩니다. 이 태그가 존재하면 모든 태그 속성의 색상이 이 태그에 지정된 스타일을 따릅니다. 이 태그를 생략하면 속성은 태그와 같은 색상으로 표시됩니다.

속성
name •기본 속성에 이름을 지정하는 문자열입니다.

예제
```
<defaultAttribute name="Attribute"/>
```

<defaultTag>

설명
이 태그는 특정 체계의 태그에 대해 기본 색상과 스타일을 지정하는 데 사용됩니다.

속성
name, id

- name="***display_name***" Dreamweaver의 코드 색상 편집기에 표시되는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제
```
<defaultTag name="Other Tags" id="CodeColor_HTMLTag" />
```

<defaultText/>

설명
선택 사항이며, 이 태그가 존재하면 다른 태그에 의해 정의되지 않은 모든 텍스트의 색상이 이 태그에 지정된 스타일을 따릅니다. 이 태그를 생략하면 텍스트가 검은색으로 표시됩니다.

속성
name, id

- name="***cssSelector_name***" CSS 선택기에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제
```
<defaultText name="Text" id="CodeColor_TextText" />
```

<endOfLineComment>

설명
현재 행의 끝까지 연속되는 주석의 시작을 구분하는 텍스트 문자열입니다. endOfLineComment … /endOfLineComment 태그는 여러 번 사용할 수 있습니다.

속성
없음

예제
```
<endOfLineComment><![CDATA[//]]></endOfLineComment>
```

<entity/>

설명
HTML 특수 문자를 인식해야 한다는 것을 나타내며 색상 표시 속성을 저장하는 빈 태그입니다.

속성
name, id

- name="***entity_name***" 엔터티에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제
```
<entity name="Special Characters" id="CodeColor_HTMLEntity" />
```

<functionKeyword>

설명
함수를 정의하는 키워드를 나타냅니다. Dreamweaver에서는 이러한 키워드를 사용하여 코드를 탐색합니다. functionKeyword 태그는 여러 번 사용할 수 있습니다.

속성
name, id

- name="***functionKeyword_name***"functionKeyword 블록에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제
```
<functionKeyword name="Function Keyword" id="CodeColor_JavascriptFunction">function</functionKeyword>
```

<idChar1>

설명
Dreamweaver가 식별자에서 첫 번째 문자로 인식할 수 있는 문자의 목록입니다.

속성
includeAlpha, includeDecimal, includeHiAscii

모든 속성은 기본적으로 부울이며 값이 true 또는 false일 수 있습니다.

예제
```
<identifier name="Identifier" id="CodeColor_JavascriptIdentifier">
<idChar1>_$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ</idChar1> </identifier>
```

<idCharRest>

설명

식별자에서 나머지 문자열로 인식되는 문자의 목록입니다. idChar1이 지정되지 않으면 이 목록에서 식별자의 모든 문자를 검사합니다.

속성

includeAlpha, includeDecimal, includeHiAscii

모든 속성은 기본적으로 부울이며 값이 true 또는 false일 수 있습니다.

예제

```
<identifier name="Identifier" id="CodeColor_JavascriptIdentifier"> <idCharRest>
    _$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789</idCharRest> </identifier>
```

<identifier>

설명

<idChar1> 및 <idCharRest> 노드는 <identifier> 노드로 둘러싸야 합니다. name 및 id 속성은 <idCharRest>/<idChar1>에서 이들 속성을 둘러싸는 <identifier> 노드로 이동해야 합니다.

두 가지 종류 이상의 식별자를 정의하는 경우 이러한 모든 식별자의 idChar1에 고유 값을 지정해야 합니다. 이 규칙을 따르지 않으면 겹치는 두 스타일에 포함된 문자로 시작하는 식별자의 결과 색상 표시가 임의 색상이 됩니다. 예:

```
 <identifier name="N1" id="I1">
       <idchar1>a</idchar1>
       <idcharrest includeAlpha="true" includeDecimal="true" includeHiAscii="true">_</idCharRest>
 </identifier>
 <identifier name="N2" id="I2">
       <idchar1  includeAlpha="true" >$_</idchar1>  //  ERROR!!!  text that starts with "a" can be either
N1 or N2, the coloring will be random for it!
       <idcharrest includeAlpha="true" includeDecimal="true" includeHiAscii="true">_</idCharRest>
 </identifier>
```

Dreamweaver CS5 이전의 구문(<identifier> 노드 없음)도 계속 지원됩니다. 하지만 같은 파일에서 이전 구문과 새 구문을 함께 사용하지 마십시오. 스타일을 혼합하면 이전 구문 스타일은 무시됩니다. 즉, 구문 체계에 <identifier> 노드가 포함되면 <identifier> 노드로 묶이지 않은 idChar1/idCharRest 노드는 모두 무시됩니다.

속성

name, id

- name="***idCharRest_name***"stringStart 블록에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제

```
<identifier name="Identifier" id="CodeColor_JavascriptIdentifier"> <idCharRest>
    _$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789</idCharRest></identifier
```

<ignoreCase>

설명

키워드와 토큰을 비교할 때 대/소문자를 무시할 것인지 여부를 지정합니다. 이 태그의 값은 Yes 또는 No 중 하나입니다. 기본값은 Yes입니다.

속성
없음

예제
```
<ignoreCase>Yes</ignoreCase>
```

<ignoreMMTParams>

설명
MMTInstance:Param, <!-- InstanceParam 또는 <!-- #InstanceParam 태그에 특별히 색상을 표시해야 할지 여부를 지정합니다. 이 태그의 값은 Yes 또는 No 중 하나입니다. 기본값은 Yes입니다. 이 태그는 템플릿을 사용하는 페이지에 적절한 색상을 표시합니다.

속성
없음

예제
```
<ignoreMMTParams>No</ignoreMMTParams>
```

<ignoreTags>

설명
마크업 태그를 무시해야 할지 여부를 지정합니다. 이 태그의 값은 Yes 또는 No 중 하나입니다. 기본값은 Yes입니다. 구문이 < 및 >로 구분되는 태그 마크업 언어에 사용되는 경우에는 No로 설정합니다. 구문이 프로그래밍 언어에 사용되는 경우에는 Yes로 설정합니다.

속성
없음

예제
```
<ignoreTags>No</ignoreTags>
```

<isLocked>

설명
이 체계와 일치하는 텍스트를 [코드] 뷰에서 편집할 수 없도록 잠글지 여부를 지정합니다. 값은 Yes 또는 No이며 기본값은 No입니다.

속성
없음

예제
```
<isLocked>Yes</isLocked>
```

<keyword>

설명
키워드를 정의하는 텍스트 문자열입니다. keyword 태그는 여러 번 사용할 수 있습니다. 키워드는 어떠한 문자로도 시작할 수 있지만 그 이후의 문자는 a-z, A-Z, 0-9, _, $ 또는 @만 가능합니다.

코드 색상은 포함하는 keyword 태그에 따라 지정됩니다.

속성
없음

예제
```
<keyword>.getdate</keyword>
```

<keywords>

설명
범주 속성에 지정된 유형의 키워드의 목록입니다. keywords 태그는 여러 번 사용할 수 있습니다.

속성
name, id

- name="***keywords_name***" 키워드 목록에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

내용
<keyword></keyword>

예제
```
<keywords name="Reserved Keywords" id="CodeColor_JavascriptReserved">
    <keyword>break</keyword>
    <keyword>case</keyword>
</keywords>
```

<numbers/>

설명
인식되어야 하는 문자를 지정하고 색상 속성을 저장하는 빈 태그입니다.

속성
name, id

- name="***number_name***"numbers 태그에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제
```
<numbers name="Number" id="CodeColor_CFScriptNumber" />
```

<operators>

설명
연산자로 인식되는 문자의 목록입니다.

속성
name, id

- name="***operator_name***" 연산자 문자 목록에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.

예제

```
<operators name="Operator" id="CodeColor_JavaOperator"><![CDATA[+-*/%<>!?:=&|^~]]></operators>
```

<regexp>

설명
searchPattern 태그의 목록을 지정합니다.

속성
name, id, delimiter, escape

- name="***stringStart_name***" 검색 패턴 문자열 목록에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.
- delimiter 일반 표현식을 시작하고 끝내는 문자 또는 문자열입니다.
- escape 특수 문자로 처리할 것을 알리는 문자 또는 문자열입니다. 이를 이스케이프 문자 또는 문자열이라고 합니다.

내용
<searchPattern></searchPattern>

예제

```
<regexp name="RegExp" id="CodeColor_JavascriptRegexp" delimiter="/" escape="\\">
    <searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
    <searchPattern><![CDATA[=\s*/\e*\\/]]></searchPattern>
</regexp>
```

<sampleText>

설명
[코드 색상 표시 체계 편집] 대화 상자의 [미리 보기] 윈도우에 나타나는 대표 텍스트입니다. [코드 색상 표시 체계 편집] 대화 상자에 대한 자세한 내용은 60페이지의 “체계 편집”을 참조하십시오.

속성
doctypes

- doctypes="***doc_type1, doc_type2,...***" 이 샘플 텍스트가 표시될 문서 형식입니다.

예제

```
<sampleText doctypes="JavaScript"><![CDATA[/* JavaScript */
function displayWords(arrayWords) {
    for (i=0; i < arrayWords.length(); i++) {
        // inline comment
        alert("Word " + i + " is " + arrayWords[i]);
    }
}

var tokens = new Array("Hello", "world");
displayWords(tokens);
]]></sampleText>
```

<searchPattern>

설명

지원되는 와일드카드 문자를 사용하여 정규 검색 패턴을 정의하는 문자의 문자열입니다. searchPattern 태그는 여러 번 사용할 수 있습니다.

속성

없음

컨테이너

regexp 태그입니다.

예제

```
<searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
```

<stringStart>

설명

이러한 태그에는 문자열 시작 구분 기호를 나타내는 텍스트 문자열이 포함됩니다. stringStart 및 stringEnd 태그는 쌍으로 지정해야 합니다. stringStart … stringEnd 쌍은 여러 번 사용할 수 있습니다.

속성

name, id, wrap

- name="***stringStart_name***"stringStart 블록에 이름을 지정하는 문자열입니다.
- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.
- wrap="true" 또는 "false" 다음 행으로 줄이 바뀌는 텍스트 문자열을 인식하여 코드 색상을 표시할지 여부를 정의합니다. 기본값은 "true"입니다.

예제

```
<stringStart name="Attribute Value" id="CodeColor_HTMLString"><![CDATA["]]></stringStart>
```

<stringEnd>

설명

코드 문자열 끝의 구분 기호를 나타내는 텍스트 문자열을 포함합니다. stringStart 및 stringEnd 태그는 쌍으로 지정해야 합니다. stringStart … stringEnd 쌍은 여러 번 사용할 수 있습니다.

속성
없음

예제
```
<stringEnd><![CDATA["]]></stringEnd>
```

<stringEsc>

설명
문자열 이스케이프 문자의 구분 기호를 나타내는 텍스트 문자열을 포함합니다. stringEsc 태그는 여러 번 사용할 수 있습니다.

속성
없음

예제
```
<stringEsc><![CDATA[\]]></stringEsc>
```

<tagGroup>

설명
이 태그는 고유한 색상 및 스타일을 지정할 수 있는 하나 이상의 태그를 그룹화합니다.

속성
id, name, taglibrary, tags

- id="***id_string***" 필수 사항이며, 이 구문 항목에 색상 및 스타일을 매핑하는 식별자 문자열입니다.
- name="***display_name***" Dreamweaver의 코드 색상 편집기에 표시되는 문자열입니다.
- taglibrary="***tag_library_id***" 이 태그 그룹이 속한 태그 라이브러리의 식별자입니다.
- tags="**tag_list**" 태그 그룹을 구성하는 태그 또는 쉼표로 구분된 태그 목록입니다.

예제
```
<tagGroup name="HTML Table Tags" id="CodeColor_HTMLTable" taglibrary="DWTagLibrary_html"
    tags="table,tbody,td,tfoot,th,thead,tr,vspec,colw,hspec" />
```

scheme 블록 구분 기호 색상 표시

blockStart scheme 속성은 블록 열기 및 닫기 문자열 또는 블록 구분 기호의 색상 표시를 제어합니다. blockStart 속성에는 다음과 같은 값을 사용할 수 있습니다.

참고: blockStart.scheme 속성과 scheme 태그를 혼동하지 않도록 하십시오.

innerText

이 값은 블록 구분 기호를 블록 구분 기호 내부에 있는 해당 체계의 기본 텍스트와 동일한 색상으로 표시하도록 지정합니다.

템플릿 체계에 이 체계의 효과를 나타내는 예제가 제공되어 있습니다. 템플릿 체계는 편집할 수 없음을 나타내기 위하여 읽기 전용 코드 블록을 회색으로 표시합니다. 블록 구분 기호는 <!#EndEditable --> 및 <!-- #BeginEditable "..." --> 문자열로, 이 또한 편집할 수 없기 때문에 회색으로 표시됩니다.

샘플 코드

```
<!-- #EndEditable -->
    <p><b><font size="+2">header</font></b></p>
    <!-- #BeginEditable "test" -->
    <p>Here's some editable text </p>
    <p> </p>
    <!-- #EndEditable -->
```

예제

```
<blockStart doctypes="ASP-JS,ASP-VB, ASP.NET_CSharp, ASP.NET_VB, ColdFusion,CFC, HTML,
    JSP,LibraryItem,PHP_MySQL" scheme="innerText"><![CDATA[<!--\s*#BeginTemplate]]>
    </blockStart>
```

customText

이 값은 사용자 정의 색상을 사용하여 블록 구분 기호의 색상을 표시하도록 지정합니다.

샘플 코드

PHP 스크립트 블록의 구분 기호는 빨간색으로 표시되며 customText 값의 효과를 보여 줍니다.

```
<?php
    if ($loginMsg <> "")
    echo $loginMsg;
    ?>
```

예제

```
<blockStart name="Block Delimiter" id="CodeColor_JavaBlock" doctypes="JSP"
    scheme="customText"><![CDATA[<%]]></blockStart>
```

outerTag

outerTag 값은 blockStart 및 blockEnd 태그가 쌍을 이루며 이 태그를 둘러싸는 체계에서의 태그 색상 표시 방식대로 해당 태그의 색상을 표시해야 함을 지정합니다.

<script> 및 </script> 문자열이 blockStart 및 blockEnd 태그에 해당하는 JavaScript 체계에서 이 값의 예를 제공합니다. 이 체계는 태그를 인식하지 않는 JavaScript 코드 블록과 일치하므로 구분 기호를 둘러싸는 체계에 따라 구분 기호의 색상을 표시해야 합니다.

샘플 코드

```
<script language="JavaScript">
    // comment
    if (true)
    window.alert("Hello, World");
    </script>
```

예제

```
<blockStart doctypes="PHP_MySQL" scheme="outerTag">
    <![CDATA[<script\s+language="php">]]></blockStart>
```

innerTag

이 값은 태그 색상 표시가 구분 기호 내부의 체계에 의해 이루어지는 점을 제외하면 outerTag 값과 동일합니다. 이 값은 현재 html 태그에 사용됩니다.

nameTag

이 값은 blockStart 문자열이 열기 태그이고 blockEnd 문자열은 닫기 태그이며 이러한 구분 기호의 색상은 체계의 태그 설정에 따라 표시되도록 지정합니다.

이 형식의 체계는 cfoutput 태그와 같이 다른 태그 내에 포함될 수 있는 태그를 표시합니다.

샘플 코드

```
<input type="text" name="zip"
    <cfif newRecord IS "no">
    <cfoutput query="employee"> Value="#zip#" </cfoutput>
    </cfif>
    >
```

예제

```
<blockStart doctypes="ColdFusion,CFC" scheme="nameTag">
    <![CDATA[<cfoutput\n]]></blockStart>
```

nameTagScript

이 값은 nameTag 체계와 동일합니다. 그러나, 속성 name=value 쌍과 달리 할당 문이나 표현식과 같은 스크립트가 내용이 됩니다.

이 형식의 체계는 ColdFusion cfset, cfif 및 cfifelse 태그와 같이 태그 자체 내부에 스크립트를 포함하는 고유한 형식의 태그를 표시하며 다른 태그 내부에 포함될 수 있습니다.

샘플 코드

자세한 내용은 57페이지의 “nameTag”의 샘플 텍스트를 참조하십시오.

예제

```
<blockStart doctypes="ColdFusion,CFC" scheme="nameTagScript"><![CDATA[<cfset\n]]></blockStart>
```

체계 처리

Dreamweaver에는 CSS 모드, 스크립트 모드 및 태그 모드라는 세 가지 기본 코드 색상 표시 모드가 있습니다.

각 모드에서는 특정 필드에만 코드 색상 표시가 적용됩니다. 다음 표에서는 각 모드에서 코드 색상 표시가 적용되는 필드를 보여줍니다.

필드	CSS	태그	스크립트
defaultText		X	X
defaultTag		X	
defaultAttribute		X	
comment	X	X	X
string	X	X	X
cssProperty	X		
cssSelector	X		
cssValue	X		
character		X	X
function keyword			X

필드	CSS	태그	스크립트
identifier			X
number		X	X
operator			X
brackets		X	X
keywords		X	X

Dreamweaver에서는 체계 정의 과정을 보다 유연하게 하기 위해 와일드카드 및 이스케이프 문자를 지정할 수 있습니다.

와일드카드 문자

다음 목록에서는 Dreamweaver에서 지원하는 와일드카드 문자, 해당 문자를 지정하기 위한 문자열 및 사용 방법에 대해 설명합니다.

와일드카드	이스케이프 문자열	설명
와일드카드	*	이 와일드카드 다음의 문자가 발견될 때까지 규칙의 모든 문자를 건너뜁니다. 예를 들어 <MMTInstance:Editable name="*">는 name 속성이 지정된 이 유형의 모든 태그를 나타냅니다.
이스케이프 문자와 함께 사용되는 와일드카드	\e*x	여기에서 x는 이스케이프 문자입니다. 이는 이스케이프 문자를 지정할 수 있다는 것을 제외하면 와일드카드와 동일합니다. 이스케이프 문자 다음에 나오는 문자는 무시되므로 와일드카드 다음에 나오는 문자가 문자열에 나타나도록 할 수 있습니다. 즉, 와일드카드 처리를 종료시키는 기준이 적용되지 않습니다. 예를 들어 /\e*\\/는 슬래시(/)로 시작하고 끝나며 백슬래시(\) 다음에 슬래시가 포함될 수 있는 JavaScript 일반 표현식을 인식하는 데 사용됩니다. 백슬래시는 코드 색상 표시 이스케이프 문자이기 때문에 코드 색상 표시 XML에 백슬래시를 지정할 때에는 앞에 백슬래시를 추가해야 합니다.
선택적 공백 문자	\s*	0개 이상의 공백 문자나 개행 문자를 나타냅니다. 예를 들어 <!--\s*#include는 ASP의 include 지시문을 나타내는 데 사용됩니다. 이때 #include 토큰 앞의 공백 문자는 있든 없든 모두 유효하므로 두 경우가 모두 허용됩니다. 공백 문자 와일드카드는 공백 문자와 개행 문자의 모든 조합을 나타냅니다.
필수 공백 문자	\s+	1개 이상의 공백 문자나 개행 문자를 나타냅니다. 예를 들어 <!--#include\s+virtual은 #include와 virtual 사이에 임의의 공백 문자 조합이 있는 ASP include 지시문을 나타내는 데 사용됩니다. 공백 문자는 이 두 토큰 사이에 반드시 존재해야 합니다. 그러나 유효한 공백 문자라면 개수와 조합에 관계가 없습니다. 공백 문자 와일드카드는 공백 문자와 개행 문자의 모든 조합을 나타냅니다.

이스케이프 문자

다음 목록에서는 Dreamweaver에서 지원하는 이스케이프 문자, 해당 문자를 지정하기 위한 문자열 및 사용 방법에 대해 설명합니다.

이스케이프 문자	이스케이프 문자열	설명
백슬래시	\\	백슬래시 문자(\)는 코드 색상 표시 이스케이프 문자이므로 코드 색상 표시 규칙에서 백슬래시 문자를 지정하려면 이를 이스케이프 처리해야 합니다.
공백 문자	\s	이 이스케이프 문자는 개행 이스케이프 문자를 나타내는 문자를 제외하고 표시되지 않는 모든 문자(예: 공백 및 탭 문자)를 나타냅니다. 선택적 공백 문자 및 필수 공백 문자 와일드카드는 공백 문자와 개행 문자를 모두 나타냅니다.
개행 문자	\n	이 이스케이프 문자는 개행(줄 바꿈이라고도 함) 및 캐리지 리턴 문자를 나타냅니다.

최대 문자열 길이

데이터 문자열에 허용되는 최대 길이는 100자입니다. 예를 들어 다음 blockEnd 태그에는 와일드카드 문자 하나가 포함되어 있습니다.

```
<blockEnd><![CDATA[<!--\s*#BeginEditable\s*"\*"\s*-->]]></blockEnd>
```

선택적 공백 문자 와일드카드 문자열(\s*)이 Dreamweaver에서 자동으로 생성되는 단일 공백 문자라고 가정하면 데이터 문자열의 길이는 이름의 와일드카드 문자열(*)을 합쳐 26자가 됩니다.

```
<!-- #BeginEditable "\*" -->
```

이 경우 편집 가능한 영역 이름은 최대 길이인 100자에서 26을 뺀 74자가 남습니다.

체계 우선 순위

Dreamweaver에서는 다음과 같은 알고리즘을 사용하여 [코드] 뷰의 텍스트 구문에 색상을 표시합니다.

1. 초기 구문 체계는 현재 파일의 문서 형식에 따라 결정됩니다. 파일 문서 형식은 scheme.documentType 속성에 맞춰집니다. 일치하는 형식을 찾을 수 없는 경우에는 scheme.documentType = "Text"로 지정된 체계가 사용됩니다.
2. 체계에 blockStart… blockEnd 쌍이 지정되어 있으면 이 체계를 중첩시킬 수 있습니다. blockStart.doctypes 속성 중 하나에 나열된 현재 파일 확장명을 가진 중첩 가능한 모든 체계는 현재 파일에 사용할 수 있으며 나머지 체계는 사용할 수 없습니다.

   참고: 모든 blockStart/blockEnd 조합은 고유해야 합니다.

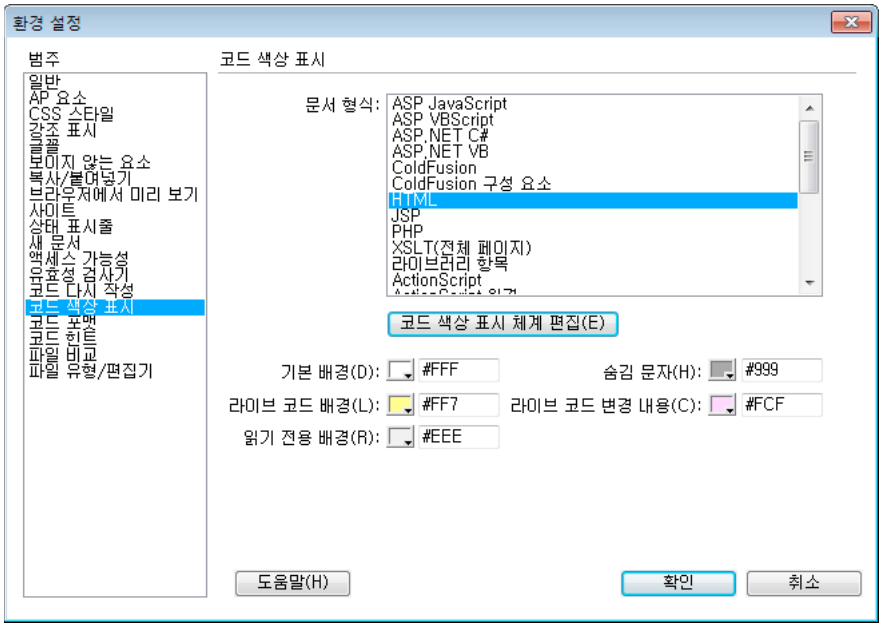
   scheme.priority가 외부 체계와 같거나 큰 체계는 다른 체계 내에 중첩될 수 있습니다. 우선 순위가 같을 경우 외부 체계의 본문 상태에만 체계를 중첩시킬 수 있습니다. 예를 들어 <script>...</script> 블록은 태그가 유효한 <html>...</html> 블록 내에만 중첩될 수 있고 태그, 속성, 문자열, 주석 등의 내부에는 중첩될 수 없습니다.

   외부 체계보다 우선 순위가 높은 체계는 외부 체계 내의 거의 모든 곳에서 중첩될 수 있습니다. 예를 들어 <%...%> 블록은 <html>...</html> 블록의 본문 상태에 중첩되는 것은 물론이고 태그, 속성, 문자열, 주석 등의 내부에도 중첩될 수 있습니다.

   최대 중첩 레벨은 4입니다.
3. blockStart 문자열이 일치하는 경우에는 항상 가장 길게 일치하는 문자열이 사용됩니다.
4. 현재 체계에 대한 blockEnd 문자열에 도달하면 구문 색상 표시는 blockStart 문자열이 발견된 상태로 돌아갑니다. 예를 들어 <%...%> 블록이 HTML 문자열 내에 있으면 색상 표시는 HTML 문자열 색상으로 다시 시작됩니다.

체계 편집

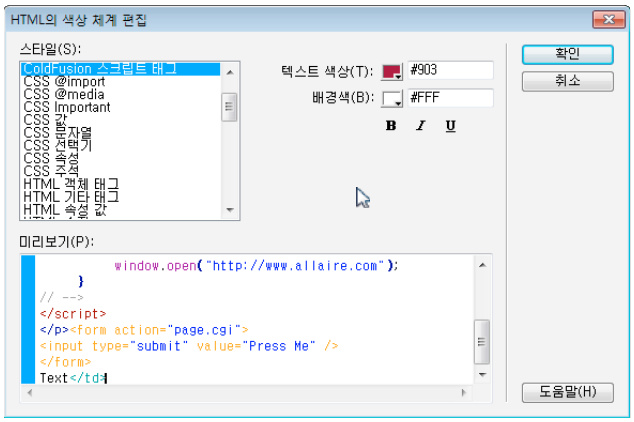
코드 색상 표시 파일을 편집하거나 다음 그림에 나타낸 것처럼 Dreamweaver [환경 설정] 대화 상자에서 [코드 색상 표시] 범주를 선택하여 코드 색상 표시 체계의 스타일을 편집할 수 있습니다.



stringStart와 같이 한 번 이상 지정할 수 있는 필드의 경우 첫 번째 태그에만 색상과 스타일을 설정합니다. 여러 태그에 개별적으로 색상과 스타일을 설정한 후 나중에 [환경 설정] 대화 상자에서 이 색상이나 스타일을 편집하면 데이터를 잃게 됩니다.

참고: 변경하기 전에 모든 XML 파일의 백업 사본을 만들어 두는 것이 좋습니다. [환경 설정] 대화 상자를 사용하여 색상 및 스타일 설정을 편집하기 전에 수동으로 변경한 모든 내용을 확인해야 합니다. [환경 설정] 대화 상자를 사용하여 잘못된 XML 파일을 편집하면 데이터를 잃게 됩니다.

[환경 설정] 대화 상자의 [코드 색상 표시] 범주를 사용하여 체계의 스타일을 편집하려면 문서 형식을 두 번 클릭하거나 [코드 색상 표시 체계 편집] 버튼을 클릭하여 [색상 표시 체계 편집] 대화 상자를 엽니다.



특정 요소의 스타일을 편집하려면 [스타일] 목록에서 해당 요소를 선택합니다. [스타일] 창에 나열된 항목에는 편집할 체계의 필드와 이 체계 내에 블록으로 나타날 수 있는 체계가 포함되어 있습니다. 예를 들어 HTML 체계를 편집하는 경우 CSS의 필드와 JavaScript 블록도 목록에 표시됩니다.

특정 체계에 대해 나열되는 필드는 해당 XML 파일에 정의된 필드입니다. scheme.name 속성의 값은 [스타일] 창에 나열되는 각 필드보다 먼저 나타납니다. 이름이 없는 필드는 목록에 표시되지 않습니다.

특정 요소의 스타일 또는 형식에는 코드 색상 표시 외에도 굵은 글꼴, 기울임체, 밑줄 및 배경색이 포함됩니다. [스타일] 창에서 요소를 선택한 후 이러한 스타일 특성을 변경할 수 있습니다.

[미리 보기] 영역에는 현재 설정 값을 사용하여 표현될 샘플 텍스트의 모양이 표시됩니다. 샘플 텍스트는 해당 체계의 sampleText 설정에서 가져온 것입니다.

[미리 보기] 영역에서 요소를 선택하면 [스타일] 목록의 선택 항목이 바뀝니다.

특정 체계의 요소에 대한 설정을 변경하면 Dreamweaver에서는 이 값을 코드 색상 표시 파일에 저장하고 원래 설정을 재정의합니다. [확인]을 클릭하면 변경된 모든 코드 색상 표시 설정이 자동으로 다시 로드됩니다.

코드 색상 표시 예제

다음 코드 색상 표시 예제에서는 CSS 문서와 JavaScript 문서에 대한 코드 색상 표시 체계를 보여 줍니다. JavaScript 예제의 키워드 목록은 예제를 짧게 만들기 위해 약자로 나타내었습니다.

CSS 코드 색상 표시

```
<scheme name="CSS" id="CSS" doctypes="CSS" priority="50">
    <ignoreCase>Yes</ignoreCase>
    <ignoreTags>Yes</ignoreTags>
    <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
        CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
        <![CDATA[<style>]]></blockStart>
    <blockEnd><![CDATA[</style>]]></blockEnd>
        <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
        CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
        <![CDATA[<style\s+\*>]]></blockStart>
    <blockEnd><![CDATA[</style>]]></blockEnd>
    <commentStart name="Comment" id="CodeColor_CSSComment"><![CDATA[ /*]]></commentStart>
    <commentEnd><![CDATA[*/]]></commentEnd>
    <endOfLineComment><![CDATA[<!--]]></endOfLineComment>
    <endOfLineComment><![CDATA[-->]]></endOfLineComment>
    <stringStart name="String" id="CodeColor_CSSString"><![CDATA["]]></stringStart>
    <stringEnd><![CDATA["]]></stringEnd>
    <stringStart><![CDATA[']]></stringStart>
    <stringEnd><![CDATA[']]></stringEnd>
    <stringEsc><![CDATA[\]]></stringEsc>
    <cssSelector name="Selector" id="CodeColor_CSSSelector" />
    <cssProperty name="Property" id="CodeColor_CSSProperty" />
    <cssValue name="Value" id="CodeColor_CSSValue" />
    <sampleText doctypes="CSS"><![CDATA[/* Comment */
H2, .head2     {
            font-family : 'Sans-Serif';
            font-weight : bold;
            color : #339999;
        }]]>
    </sampleText>
</scheme>
```

CSS 샘플 텍스트

CSS 체계에 대한 다음 샘플 텍스트에서는 CSS 코드 색상 표시 체계를 보여 줍니다.

```
/* Comment */
H2, .head2     {
    font-family : 'Sans-Serif';
    font-weight : bold;
    color : #339999;
    }
```

Colors.xml 파일의 다음 행은 코드 색상 표시 체계에 의해 지정되어 샘플 텍스트에 표시된 색상 및 스타일 값을 보여 줍니다.

```
<syntaxColor id="CodeColor_CSSSelector" text="#FF00FF" />
<syntaxColor id="CodeColor_CSSProperty" text="#000099" />
<syntaxColor id="CodeColor_CSSValue" text="#0000FF" />
```

JavaScript 코드 색상 표시

```
<scheme name="JavaScript" id="JavaScript" doctypes="JavaScript" priority="50">
        <ignoreCase>No</ignoreCase>
        <ignoreTags>Yes</ignoreTags>
        <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
            CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
            <![CDATA[<script>]]></blockStart>
        <blockEnd><![CDATA[</script>]]></blockEnd>
        <blockStart doctypes="ASP-JS,ASP-VB,ASP.NET_CSharp,ASP.NET_VB,ColdFusion,
            CFC,HTML,JSP,LibraryItem,DWTemplate,PHP_MySQL" scheme="outerTag">
            <![CDATA[<script\s+\*>]]></blockStart>
        <blockEnd><![CDATA[</script>]]></blockEnd>
        <commentStart name="Comment" id="CodeColor_JavascriptComment">
            <![CDATA[/*]]></commentStart>
        <commentEnd><![CDATA[*/]]></commentEnd>
        <endOfLineComment><![CDATA[//]]></endOfLineComment>
        <endOfLineComment><![CDATA[<!--]]></endOfLineComment>
        <endOfLineComment><![CDATA[-->]]></endOfLineComment>
        <stringStart name="String" id="CodeColor_JavascriptString">
            <![CDATA["]]></stringStart>
        <stringEnd><![CDATA["]]></stringEnd>
        <stringStart><![CDATA[']]></stringStart>
        <stringEnd><![CDATA[']]></stringEnd>
        <stringEsc><![CDATA[\]]></stringEsc>
        <brackets name="Bracket" id="CodeColor_JavascriptBracket">
            <![CDATA[{[()]}]]></brackets>
        <operators name="Operator" id="CodeColor_JavascriptOperator">
            <![CDATA[+-*/%<>!?:=&|^]]></operators>
        <numbers name="Number" id="CodeColor_JavascriptNumber" />
        <regexp name="RegExp" id="CodeColor_JavascriptRegexp" delimiter="/" escape="\\">
            <searchPattern><![CDATA[(\s*/\e*\\/]]></searchPattern>
            <searchPattern><![CDATA[=\s*/\e*\\/]]></searchPattern>
        </regexp>
        <idChar1>_$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ</idChar1>
        <idCharRest name="Identifier" id="CodeColor_JavascriptIdentifier">
            _$abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789</idCharRest>
        <functionKeyword name="Function Keyword" id="CodeColor_JavascriptFunction">
            function</functionKeyword>
        <keywords name="Reserved Keywords" id="CodeColor_JavascriptReserved">
            <keyword>break</keyword>
. . .
        </keywords>
        <keywords name="Native Keywords" id="CodeColor_JavascriptNative">
            <keyword>abs</keyword>
. . .
```

```
        </keywords>
        <keywords id="CodeColor_JavascriptNumber">
            <keyword>Infinity</keyword>
            <keyword>Nan</keyword>
        </keywords>
        <keywords name="Client Keywords" id="CodeColor_JavascriptClient">
            <keyword>alert</keyword>
. . .
        </keywords>
        <sampleText><![CDATA[/* JavaScript */
function displayWords(arrayWords) {
    for (i=0; i < arrayWords.length(); i++) {
        // inline comment
        alert("Word " + i + " is " + arrayWords[i]);
    }
}
var tokens = new Array("Hello", "world");
displayWords(tokens);
]]></sampleText>
    </scheme>
```

JavaScript 샘플 텍스트

JavaScript 체계에 대한 다음 샘플 텍스트에서는 JavaScript 코드 색상 표시 체계를 보여 줍니다.

```
/* JavaScript */ function displayWords(arrayWords) {
    for (i=0; i < arrayWords.length(); i++) {
        // inline comment
        alert("Word " + i + " is " + arrayWords[i]);
    }
}
var tokens = new Array("Hello", "world");
displayWords(tokens);
```

Colors.xml 파일의 다음 행은 코드 색상 표시 체계에 의해 지정되어 샘플 텍스트에 표시된 색상 및 스타일 값을 보여 줍니다.

```
<syntaxColor id="CodeColor_JavascriptComment" text="#999999" italic="true" />
<syntaxColor id="CodeColor_JavascriptFunction" text="#000000" bold="true" />
<syntaxColor id="CodeColor_JavascriptBracket" text="#000099" bold="true" />
<syntaxColor id="CodeColor_JavascriptNumber" text="#FF0000" />
<syntaxColor id="CodeColor_JavascriptClient" text="#990099" />
<syntaxColor id="CodeColor_JavascriptNative" text="#009999" />
```

코드 유효성 검사

[코드] 뷰에서 문서를 열면 사용자가 선택한 대상 브라우저에 사용할 수 없는 태그, 속성, CSS 속성 또는 CSS 값이 문서에 사용되고 있는지가 자동으로 검사됩니다. 오류가 있는 부분에는 빨간색의 물결 모양 밑줄이 표시됩니다.

또한 Dreamweaver에는 새로운 브라우저 호환성 확인 기능이 있어 브라우저 렌더링 문제를 일으키는 HTML과 CSS의 조합을 찾을 수도 있습니다.

브라우저 프로파일은 Dreamweaver Configuration 폴더 내의 Browser Profile 폴더에 저장됩니다. 각 브라우저 프로파일은 해당 브라우저를 나타내는 이름이 지정된 텍스트 파일로 정의됩니다. 예를 들어 Internet Explorer 버전 6.0에 대한 브라우저 프로파일은 Internet_Explorer_6.0.txt입니다. Dreamweaver에서는 대상 브라우저가 CSS를 지원하는지 검사하기 위해 브라우저에 대한 CSS 프로파일 정보를 XML 파일에 저장합니다. 이 XML 파일의 이름은 브라우저 프로파일 이름에 _CSS.xml이라는 접미어가 붙은 형태입니다. 예를 들어 Internet Explorer 6.0에 대한 CSS 프로파일은 Internet_Explorer_6.0_CSS.xml입니다. 원치 않는 오류가 보고되는 경우 CSS 프로파일 파일을 변경할 수 있습니다.

CSS 프로파일 파일은 css-support, property 및 value라는 세 가지 XML 태그로 구성됩니다. 다음 단원에서는 이 세 태그에 대해 설명합니다.

<css-support>

설명
이 태그는 특정 브라우저가 지원하는 property 및 value 태그 세트에 대한 루트 노드입니다.

속성
없음

내용
property 및 value 태그입니다.

컨테이너
없음

예제

```
<css-support>
. . .
</css-support>
```

<property>

설명
브라우저 프로파일에 대해 지원되는 CSS 속성을 정의합니다.

속성
name, names, supportlevel, message

- name="***property_name***" 지원할 속성의 이름입니다.
- names="***property_name***, **property_name,...**" 지원할 속성 이름의 쉼표로 구분된 목록입니다.

  names 속성은 일종의 약식 방법입니다. 예를 들어 다음 names 속성은 그 다음에 나오는 name 속성을 정의하는 간편한 방법입니다.

  ```
  <property names="foo,bar">
      <value type="named" name="top"/>
      <value type="named" name="bottom"/>
  </property>
  <property name="foo">
      <value type="named" name="top"/>
      <value type="named" name="bottom"/>
  </property>
  <property name="bar">
      <value type="named" name="top"/>
      <value type="named" name="bottom"/>
  </property>
  ```

- supportlevel="error", "warning", "info", 또는 "supported" 속성의 지원 레벨을 지정합니다. 이 속성을 지정하지 않으면 "supported"로 간주됩니다. "supported" 이외의 지원 레벨을 지정하고 message 속성을 생략하면 "CSS 속성 이름 **property_name**은(는) 지원되지 않습니다."라는 기본 메시지가 사용됩니다.

- message="***message_string***"message 속성은 문서에서 해당 속성이 발견될 때 표시되는 메시지 문자열을 정의합니다. 이 메시지 문자열은 속성 값에 대한 제한 사항이나 이를 해결하는 방법 등을 설명합니다.

내용
value

컨테이너
css-support

예제
```
<property name="background-color" supportLevel="supported">
```

<value>

설명
현재 속성에 지원되는 값의 목록을 정의합니다.

속성
type, name, names, supportlevel, message

- type="any", "named", "units", "color", "string" 또는 "function" 값의 유형을 지정합니다. "named", "units" 또는 "color"를 지정할 경우 name 또는 names 속성 중 하나에는 값 ID를 지정하여 이 항목과 일치시켜야 합니다. "units" 값은 숫자 값에 해당하며 그 뒤에는 names 속성에 지정된 단위 값 중 하나가 와야 합니다.
- name="***value_name***" CSS 값 식별자입니다. 하이픈(-) 이외의 공백 문자나 구두점은 사용할 수 없습니다. CSS 속성에 사용할 수 있는 값 중 하나의 이름은 부모 속성 노드에 지정되어 있습니다. 이 속성은 특정 값이나 단위 지정자를 나타낼 수 있습니다.
- names="***name1, name2, . . .***" 쉼표로 구분된 값 ID 목록을 지정합니다.
- supportlevel="error", "warning", "info"또는 "supported" 브라우저의 이 값에 대한 지원 레벨을 지정합니다. 이 속성을 지정하지 않으면 "supported"로 간주됩니다.
- message="***message_string***"message 속성은 문서에서 해당 속성 값이 발견될 때 표시되는 메시지 문자열을 정의합니다. message 속성을 생략하면 "**value_name**은(는) 지원되지 않습니다."라는 메시지 문자열이 표시됩니다.

내용
없음

컨테이너
property

예제
```
<property name="margin">
    <value type="units" name="ex" supportlevel="warning"
        message="The implementation of ex units is buggy in Safari 1.0."/>
    <value type="units" names="%,em,px,in,cm,mm,pt,pc"/>
    <value type="named" name="auto"/>
    <value type="named" name="inherit"/>
</property>
```

기본 HTML 서식 변경

일반적인 코드 서식 환경 설정을 변경하려면 [환경 설정] 대화 상자의 [코드 포맷] 범주를 사용합니다. 특정 태그 및 속성의 서식을 변경하려면 [태그 라이브러리 편집기]([편집] > [태그 라이브러리])를 사용합니다. 자세한 내용은 [Dreamweaver 도움말] 메뉴에서 **Dreamweaver 사용 설명서**를 참조하십시오.

Tag Libraries 구성 폴더의 하위 폴더에서 원하는 태그에 해당하는 VTM 파일을 편집하여 태그의 서식을 편집할 수도 있지만 Dreamweaver 내에서 서식을 변경하는 것이 훨씬 더 쉽습니다.

VTM 파일을 추가하거나 제거할 경우 TagLibraries.vtm 파일을 편집해야 합니다. TagLibraries.vtm에 나열되지 않은 VTM 파일은 모두 무시됩니다.

참고: 이 파일은 Dreamweaver가 아니라 텍스트 편집기에서 편집해야 합니다.

수직 분할 뷰

수직 분할 뷰 기능을 사용하여 코드 및 디자인이나 코드 및 코드 레이아웃 모드를 나란히 볼 수 있습니다. 이중 스크린 워크스테이션을 설정한 사용자는 이 기능을 사용하여 한 모니터에 코드를 표시하고 두 번째 모니터를 통해 [디자인] 뷰에서 작업할 수 있습니다.

수직 분할 뷰 기능을 사용하여 수행할 수 있는 작업은 다음과 같습니다.

- [코드] 및 [디자인] 뷰의 방향 선택(수평 또는 수직)
- [코드] 및 [디자인] 뷰와 분할 코드의 수평 및 수직 방향 전환

Dreamweaver를 다시 시작하고 문서를 열거나 만들면 [코드] 및 [디자인] 뷰에 마지막으로 사용된 크기 및 방향과 함께 문서가 표시됩니다. dreamweaver.setSplitViewOrientation() 함수는 방향을 설정하고 dreamweaver.setPrimaryView()는 기본 보기를 설정합니다. 이러한 함수를 사용하는 방법에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**의 "수직 분할 뷰 함수"를 참조하십시오.

관련 파일

관련 파일 기능을 사용하면 작업 중인 파일과 연관된 지원 파일과 관련 파일에 액세스할 수 있습니다. 관련 파일에는 CSS, 스크립트, SSI(서버측 포함) 또는 XML 파일 등이 있습니다.

예를 들어, CSS 파일이 주 파일과 연관되어 있는 경우 이 기능을 사용하여 이 CSS 파일을 쉽게 보고 편집할 수 있습니다. 관련 파일을 편집하는 동안 주 파일을 볼 수도 있습니다.

관련 파일의 작동 방식

관련 파일은 다음 작업을 쉽게 수행할 수 있도록 하여 사용자의 편집 환경을 개선합니다.

- 사용자는 주 파일을 보는 동안 관련 파일을 보고 액세스할 수 있습니다. CSS 파일 등의 관련 파일이 있는 페이지를 볼 때 다음과 같이 표시됩니다.
  - 한 쪽에 페이지의 디자인
  - 다른 쪽에 관련 파일
- 관련 파일 막대에는 상위 HTML의 생성에 영향을 미치는 문서가 포함되어 있습니다. 사용자는 소스 HTML, 생성된 HTML 및 첫 번째 수준 하위 문서를 볼 수 있습니다.

- 관련 파일 막대에 표시되는 관련 파일을 선택하면 다음 작업을 수행할 수 있습니다.
  - [코드] 뷰에서 관련 파일 보기 및 편집
  - [디자인] 뷰에서 상위 페이지 보기
- [디자인] 뷰에서 내용을 선택하고 관련 파일에서 변경하면 사용자가 [디자인] 뷰를 새로 고칠 때 선택 영역이 사라지지 않습니다.
- 관련 파일 코드를 변경하는 경우 변경 내용이 [디자인] 뷰에 반영됩니다.

파일을 찾을 수 없으면 파일을 찾을 수 없다는 메시지가 빈 윈도우 프레임 맨 위의 막대에 표시됩니다.

관련 파일 용어

다음은 관련 파일과 관련하여 일반적으로 사용되는 용어입니다.

용어	설명	예제
최상위 문서	사용자가 연 모든 문서	
상위 문서	[디자인] 보기에 렌더링된 모든 최상위 문서	• HTML — .lbi, .dwt 등 • CFML • PHP
첫 번째 수준 하위 문서	상위 문서에서 한 수준 아래에 있는 모든 문서. 이러한 문서는 CSS를 제외하고 HTML 코드의 생성에 영향을 미칩니다. CSS 파일에는 다른 CSS 파일이 포함될 수 있지만 이러한 파일이 모두 함께 페이지에 적용된 최종 스타일을 결정합니다.	• <SCRIPT src="file.js">로 지정된 스크립트 파일 • 서버측 포함 • 외부 CSS • Spry XML 및 HTML 데이터 세트 • 라이브러리 항목 • <iframe> - 원격 소스 • 객체 태그
더 낮은 수준 하위 문서	상위 문서에서 두 수준 이상 아래에 있는 모든 문서. 이러한 문서는 HTML 코드의 생성에 영향을 미칩니다.	• PHP 안의 PHP • DTD • 템플릿
비관련 파일	HTML 코드의 생성에 영향을 미치지 않는 모든 문서나 사용자가 활발하게 편집하지 않는 모든 파일	• 이미지 파일 • 미디어 파일 • <a> 태그로 외부에서 링크된 파일

다음과 같은 관련 파일이 지원됩니다.

유형	설명	중첩 수준
클라이언트측 스크립트	모든 언어	1(스크립트 중첩이 불가능함)
서버측 포함	다음 확장 가능한 조건 모두에 해당하는 경우: • 정의된 서버 모델 • 정의된 SSI 문(pattern) • 정의된 DW doctype 예외: HTML 문서에서 Apache 스타일 포함 파일 문(<!-- #include ... -->)은 인식됩니다.	1
Spry 데이터 세트		1(스크립트 중첩이 불가능함)
CSS	• 모든 미디어 유형에 대한 모든 외부 CSS • DTSS	무한

관련 파일 API

관련 파일 메뉴를 사용자 정의하여 다음을 표시할 수 있습니다.

- 관련 파일의 파일 이름
- 소스 HTML 및 생성된 소스 코드

dreamweaver.openRelatedFile() 함수는 [코드] 뷰에 관련 파일을 표시하고 dreamweaver.getActiveRelatedFilePath() 함수는 현재 열려 있는 관련 파일의 경로를 표시합니다. 이러한 API를 사용하는 방법에 대한 자세한 내용은 **Dreamweaver API** 참조 설명서의 "관련 파일 함수"를 참조하십시오.

라이브 뷰 정보

라이브 뷰 기능을 사용하면 Dreamweaver를 벗어나지 않고도 브라우저에 나타나는 대로 웹 페이지를 미리 볼 수 있습니다. 사용자는 계속해서 코드에 직접 액세스하고 코드를 편집할 수 있습니다. 코드 변경 내용은 즉시 반영되므로 사용자가 변경된 웹 페이지를 즉시 볼 수 있습니다. 사용자가 CSS 파일을 편집하는 경우 파일의 현재 상태가 유지되지만 CSS 변경 내용이 적용됩니다. 사용자는 Dreamweaver에서 웹 브라우저로 전환하지 않고도 페이지와 상호 작용하고 롤오버와 같은 JavaScript 효과를 볼 수도 있습니다.

라이브 뷰 기능은 시스템 Flash 플러그인(%SYSTEM%/Macromed/Flash, /Library/Internet Plug-Ins/)을 사용합니다. 경우에 따라 플러그인의 시스템 버전을 사용할 수 없으면 Firefox 버전으로 대체될 수 있습니다.

플러그인을 찾을 수 없는 경우 [정보] 막대에 알림이 표시됩니다. CSS 패널에는 항상 [라이브] 뷰에 표시되는 항목의 현재 관련 CSS가 표시됩니다. 이는 소스가 다른 위치에서 생성된 경우에도 해당됩니다. 사용자는 스타일 시트를 추가하거나 제거할 수 있습니다. 그러나 인라인 CSS 또는 <head>의 CSS에 대한 다른 편집 기능은 잠깁니다. CSS 패널에서 편집할 수 없는 규칙은 읽기 전용으로 표시됩니다.

Dreamweaver API를 사용하여 다음 작업을 수행할 수 있습니다.

- 디자인 뷰 모드 가져오기 및 설정
- 서버를 사용하여 라이브 뷰 모드 가져오기 및 설정
- 라이브 뷰 기본값 가져오기
- 라이브 뷰 종속 항목 가져오기 및 설정
- 라이브 뷰 매개 변수 보기

이러한 API에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**의 "라이브 뷰 함수"를 참조하십시오.

간단한 라이브 뷰 예제

이 예제에서는 Dreamweaver를 사용하여 사용자가 [명령] 메뉴에서 명령을 클릭할 때 간단한 브라우저를 만드는 명령을 만듭니다. 이 예제를 연습하기 전에 명령을 만드는 방법에 대한 자세한 내용은 142페이지의 124페이지의 "명령"을 참조하십시오. Dreamweaver에서 새 기본 HTML 파일(이 파일이 명령 정의 파일임)을 열고 liveviewexample.htm으로 파일을 저장합니다. 명령 정의 파일은 다음 예제와 유사합니다.

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0// dialog"> <html
xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Browser Test</title>
<script type="text/javascript">
var browserLoad = function(e)
{
    var uri = document.getElementById("uri");
    uri.value = e.currentBrowserLocation;
}
var promptToNavigate = function (e)
{
    if( ! confirm(" Is it ok to go from \n" + e.currentBrowserLocation + " to page \n " +
e.requestedBrowserLocation ) )
    {
        e.preventDefault();
    }
}
function initUI()
{
    var browser = document.getElementById("browser");
    browser.addEventListener("BrowserControlBeforeNavigation",
promptToNavigate, true);
    browser.addEventListener("BrowserControlLoad", browserLoad, true);
    browser.openURL("http://www.adobe.com");
}
function loadUri()
{
    var uri = document.getElementById("uri");
    var browser = document.getElementById("browser");
    browser.openURL(uri.value);
}
function showSource(){
    var browser = document.getElementById("browser");
    alert(browser.getWindowObj().document.documentElement.outerHTML);
}
function commandButtons() {
                              return new Array( "Close", "window.close()",
                              "Load URI", "loadUri()",
                              "Show Source", "showSource()"
);
}
</script>
</head>
<body onLoad="initUI()">
<form>
<p>
<label>
<input id="uri" type="text" style="width:500px">
</label>
</p> <mm:browsercontrol id="browser" style="width:500px; height:300px;"/> </form> </body> </html>
```

4장: Dreamweaver 확장

Dreamweaver에서는 기능을 추가하거나 사용자 정의하는 데 사용할 수 있는 광범위한 도구를 제공합니다.

Dreamweaver Extension을 만들려면 2페이지의 “Extension 만들기”에 설명된 단계를 따릅니다.

Extension을 만들 수 있는 Dreamweaver 기능은 다음과 같습니다.

- HTML 파서(렌더러라고도 함) - Extension의 UI(사용자 인터페이스)를 디자인할 수 있도록 합니다. 파서에서는 양식 필드, 절대 위치 요소, 이미지 및 기타 HTML 요소를 사용합니다. Dreamweaver에는 자체 HTML 파서가 포함되어 있습니다.
- 폴더 트리 - Dreamweaver 요소 및 Extension을 구현하고 구성하는 파일은 폴더 트리 내에 구성하고 저장합니다.
- 일련의 API(응용 프로그램 프로그래밍 인터페이스) - JavaScript를 통해 Dreamweaver 기능에 액세스할 수 있도록 합니다.
- JavaScript 인터프리터 - Extension 파일에서 JavaScript 코드를 실행합니다. Dreamweaver에서는 Netscape JavaScript 버전 1.5 인터프리터를 사용합니다. 이 버전의 인터프리터와 이전 버전의 차이점에 대한 자세한 내용은 73페이지의 “Dreamweaver에서 JavaScript가 처리되는 방식”을 참조하십시오.

Dreamweaver Extension의 유형

다음 목록에서는 Dreamweaver Extension의 유형에 대해 설명합니다.

삽입 막대 객체 Extension은 [삽입] 막대를 변경합니다. 일반적으로 객체는 문서에 코드를 삽입하는 과정을 자동화하는 데 사용됩니다. 사용자 입력을 받아들이는 양식과 입력을 처리하는 JavaScript를 포함할 수도 있습니다. 객체 파일은 Configuration/Objects 폴더에 저장됩니다.

명령 Extension은 사용자 입력 여부에 상관없이 대부분의 특정 작업을 수행할 수 있습니다. 일반적으로 명령 파일은 [명령] 메뉴에서 시작되지만 다른 Extension에서 호출할 수도 있습니다. 명령 파일은 Configuration/Commands 폴더에 저장됩니다.

메뉴 명령 Extension은 명령 API를 확장하여 메뉴에서 명령을 호출하는 데 관련된 작업을 수행합니다. 메뉴 명령 API를 사용하면 동적 하위 메뉴를 만들 수도 있습니다.

툴바 Extension은 기존 툴바에 요소를 추가하거나 새로운 툴바를 Dreamweaver 사용자 인터페이스에 만들 수 있습니다. 새로운 툴바는 기본 툴바 아래에 표시됩니다. 툴바 파일은 Configuration/Toolbars 폴더에 저장됩니다.

보고서 Extension은 사용자 정의 사이트 보고서를 추가하거나 Dreamweaver와 함께 제공되는 미리 작성된 보고서 세트를 수정할 수 있습니다. [결과] 윈도우 API를 사용하여 독립 실행형 보고서를 만들 수도 있습니다.

태그 라이브러리 및 편집기 Extension은 연관된 태그 라이브러리 파일에 대한 작업을 수행합니다. 태그 라이브러리 및 편집기 Extension은 기존 태그 대화 상자의 속성을 수정하고, 새 태그 대화 상자를 만들고, 태그 라이브러리에 태그를 추가할 수 있습니다. 태그 라이브러리 및 편집기 Extension 파일은 Configuration/TagLibraries 폴더에 저장됩니다.

속성 관리자 Extension은 속성 관리자 패널에 나타납니다. Dreamweaver에 있는 대부분의 관리자는 핵심 제품 코드의 일부로서 수정이 불가능하지만, 사용자 정의 속성 관리자 파일은 Dreamweaver의 내장 속성 관리자 인터페이스를 재정의하거나 사용자 정의 태그를 관리하기 위한 새 관리자를 만들 수 있습니다. 관리자는 Configuration/Inspectors 폴더에 저장됩니다.

부동 패널 Extension은 부동 패널을 Dreamweaver 사용자 인터페이스에 추가합니다. 부동 패널은 선택 영역, 문서 또는 작업 상태에 따라 다르게 동작합니다. 또한 부동 패널에는 유용한 정보가 표시됩니다. 부동 패널 파일은 Configuration/Floaters 폴더에 저장됩니다.

비헤이비어 Extension을 사용하면 문서에 JavaScript 코드를 추가할 수 있습니다. JavaScript 코드는 문서가 브라우저에 표시될 때 이벤트에 대한 응답으로 특정 작업을 수행합니다. 비헤이비어 Extension은 Dreamweaver [비헤이비어] 패널의 플러스(+) 메뉴에 나타납니다. 비헤이비어 파일은 Configuration/Behaviors/Actions 폴더에 저장됩니다.

서버 비헤이비어 Extension은 문서에 서버측 코드 블록(ASP 또는 ColdFusion)을 추가합니다. 서버측 코드는 문서가 브라우저에 표시될 때 서버에 대한 작업을 수행합니다. 서버 비헤이비어는 Dreamweaver [서버 비헤이비어] 패널의 플러스(+) 메뉴에 나타납니다. 서버 비헤이비어 파일은 Configuration/Server Behaviors 폴더에 저장됩니다.

데이터 소스 Extension을 사용하면 데이터베이스에 저장된 동적 데이터에 연결할 수 있습니다. 데이터 소스 Extension은 [바인딩] 패널의 플러스(+) 메뉴에 나타납니다. 데이터 소스 Extension 파일은 Configuration/Data Sources 폴더에 저장됩니다.

서버 형식 Extension을 사용하면 동적 데이터의 형식을 정의할 수 있습니다.

구성 요소 Extension을 사용하면 [구성 요소] 패널에 새로운 구성 요소 유형을 추가할 수 있습니다. **구성 요소**란 Dreamweaver에서 CFC(ColdFusion 구성 요소)와 같이 보다 대중적이고 현대적인 캡슐화 전략을 가리키는 데 사용하는 용어입니다.

서버 모델 Extension을 사용하면 새 서버 모델에 대한 지원을 추가할 수 있습니다. Dreamweaver에서는 가장 일반적인 서버 모델(ASP, JSP, ColdFusion, PHP 및 ASP.NET)을 지원합니다. 서버 모델 Extension은 사용자 정의 서버 솔루션, 서로 다른 언어 또는 사용자 정의 서버의 경우에만 필요합니다. 서버 모델 파일은 Configuration/ServerModels 폴더에 저장됩니다.

데이터 변환기 Extension은 문서 윈도우의 [디자인] 뷰에 나타나는 HTML이 아닌 코드를 HTML로 변환합니다. 또한 이 Extension은 HTML이 아닌 코드를 Dreamweaver에서 파싱하지 못하도록 잠급니다. 변환기 파일은 Configuration/Translators 폴더에 저장됩니다.

기타 Dreamweaver 확장 방법

또한 다음과 같은 Dreamweaver 요소의 기능을 필요에 맞게 확장할 수 있습니다.

문서 형식 - Dreamweaver에서 다양한 서버 모델을 처리하는 방법에 대해 정의합니다. 서버 모델의 문서 형식에 대한 정보는 Configuration/DocumentTypes 폴더에 저장됩니다. 자세한 내용은 12페이지의 “Dreamweaver의 확장 가능한 문서 형식”을 참조하십시오.

코드 단편 - Dreamweaver의 Configuration/Snippets 폴더에 CSN(코드 단편) 파일로 저장되는 재사용 가능한 코드 블록으로, Dreamweaver의 [코드 단편] 패널에서 액세스할 수 있습니다. 추가 코드 단편 파일을 만들고 Snippets 폴더에 설치하여 사용할 수도 있습니다.

코드 힌트 - 사용자가 입력하고 있는 문자열로 완성할 수 있는 문자열 목록을 표시하여 입력을 간단하게 해 주는 메뉴입니다. 메뉴에 표시된 문자열 중 하나가 입력하려는 문자열과 일치하는 경우 해당 항목을 선택하여 입력 중인 문자열 위치에 삽입할 수 있습니다. 코드 힌트 메뉴는 Configuration/CodeHints 폴더의 codehints.xml 파일에 정의됩니다. 새로 정의한 태그나 함수에 대한 새 코드 힌트 메뉴를 이 파일에 추가할 수도 있습니다.

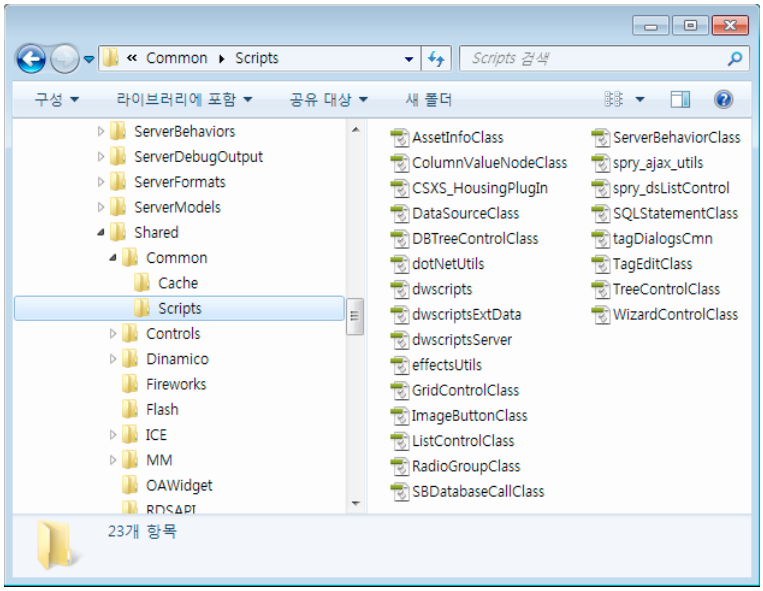
메뉴 - Configuration/Menus 폴더의 menus.xml 파일에 정의되어 있습니다. menus.xml 파일에 Extension에 대한 메뉴 태그를 추가하여 새로운 Dreamweaver 메뉴를 추가할 수 있습니다. 자세한 내용은 133페이지의 “메뉴 및 메뉴 명령”을 참조하십시오.

Configuration 폴더 및 Extension

Dreamweaver Configuration 폴더에 저장된 폴더 및 파일에는 Dreamweaver와 함께 제공되는 Extension이 들어 있습니다. Extension을 작성할 때는 올바른 폴더에 파일을 저장해야만 Dreamweaver에서 해당 파일이 인식됩니다. 예를 들어 속성 관리자 Extension을 만들 경우에는 해당 파일을 Configuration/Inspectors 폴더에 저장합니다. Adobe Exchange 웹 사이트(www.adobe.com/go/exchange_kr)에서 Extension을 다운로드하여 설치하는 경우에는 Extension Manager에서 자동으로 Extension 파일을 올바른 폴더에 저장합니다.

Dreamweaver Configuration 폴더의 파일을 예제로 사용할 수도 있지만 이러한 파일은 일반적으로 Adobe Exchange 웹 사이트에서 제공되는 Extension보다 복잡합니다. Configuration 폴더 내의 각 하위 폴더 내용에 대한 자세한 내용은 Configuration_ReadMe.htm 파일을 참조하십시오.

Configuration/Shared 폴더는 특정 Extension 유형에만 해당되지 않습니다. 이 폴더는 둘 이상의 Extension에서 사용되는 유틸리티 함수, 클래스 및 이미지를 모아 놓은 중앙 저장소입니다. Configuration/Shared/Common 폴더의 파일은 광범위한 Extension에 유용하도록 설계되어 있습니다. 이러한 파일은 JavaScript 기술 예제 및 유틸리티로 활용할 수 있습니다. 객체에 대한 유효한 DOM(문서 객체 모델) 참조를 만들거나, 현재 선택 항목이 특정 태그 내에 있는지 여부를 테스트하거나, 문자열 내의 특수 문자를 이스케이프 처리하는 등의 특정 작업을 수행하는 함수는 먼저 이 폴더에서 찾습니다. 공통 파일을 만드는 경우 다음 그림과 같이 Configuration/Shared/Common 폴더에 별도의 하위 폴더를 만들고 공통 파일을 이 폴더에 저장해야 합니다.



Configuration/Shared/Common/Scripts 폴더 구조

Shared 폴더에 대한 자세한 내용은 342페이지의 “Shared 폴더”를 참조하십시오.

다중 사용자 Configuration 폴더

Windows XP, Windows 2000 및 Macintosh OS X와 같은 다중 사용자 운영 체제의 경우, Dreamweaver에서는 Dreamweaver Configuration 폴더 외에 각 사용자에 대한 별도의 Configuration 폴더를 만듭니다. Dreamweaver 또는 JavaScript Extension에서 Configuration 폴더에 내용을 쓸 때마다 Dreamweaver에서는 해당 내용을 이 폴더 대신 사용자의 Configuration 폴더에 씁니다. 따라서 각 Dreamweaver 사용자는 다른 사용자의 구성 설정에 영향을 주지 않고 구성 설정을 사용자 정의할 수 있습니다. 자세한 내용은 10페이지의 “다중 사용자 환경에서 Dreamweaver 사용자 정의” 및 **Dreamweaver API 참조 설명서**의 "파일 액세스 및 다중 사용자 구성 API"를 참조하십시오.

시작 또는 종료 시 스크립트 실행

명령 파일을 Configuration/Startup 폴더에 넣으면 Dreamweaver가 시작될 때 해당 명령이 실행됩니다. 시작 명령은 menus.xml 파일이나 ThirdPartyTags 폴더의 파일보다 먼저 로드되며 다른 명령, 객체, 비헤이비어, 관리자, 부동 패널 또는 변환기보다도 먼저 로드됩니다. 따라서 시작 명령을 사용하여 menus.xml 파일이나 다른 Extension 파일을 수정할 수 있습니다. 또한 경고를 표시하거나 사용자에게 정보를 요구하거나 dreamweaver.runCommand() 함수를 호출할 수도 있습니다. 그러나 Startup 폴더 내에서 유효한 DOM(문서 객체 모델)이 필요한 명령을 호출할 수는 없습니다. Dreamweaver DOM에 대한 자세한 내용은 90페이지의 “Dreamweaver 문서 객체 모델”을 참조하십시오.

마찬가지로, Configuration/Shutdown 폴더에 명령 파일을 넣으면 해당 명령은 Dreamweaver가 종료될 때 실행됩니다. 종료 명령에서는 dreamweaver.runCommand() 함수를 호출하거나 경고를 표시하거나 사용자에게 정보를 요구할 수 있지만 종료 프로세스를 중단할 수는 없습니다.

명령에 대한 자세한 내용은 124페이지의 "명령"을 참조하십시오. dreamweaver.runCommand() 함수에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**를 참조하십시오.

Extension 다시 로드

Dreamweaver에서 작업하는 동안 Extension을 편집한 경우 Dreamweaver에서 변경 내용이 인식되도록 해당 Extension을 다시 로드할 수 있습니다.

Extension 다시 로드

1 [삽입] 패널의 제목 막대에서 [범주] 메뉴를 Ctrl 키를 누른 채 클릭(Windows)하거나 Option 키를 누른 채 클릭(Macintosh)합니다.

참고: 탭 모드에서는 이 옵션이 표시되지 않습니다. 탭 모드에서는 왼쪽 위에 있는 [패널] 메뉴를 마우스 오른쪽 버튼으로 클릭합니다. [메뉴로 표시]를 선택하고 Ctrl 키를 누른 채 [일반]을 클릭하여 [Extensions 새로 고침] 명령이 있는 메뉴를 표시합니다.

2 [Extensions 새로 고침]을 선택합니다.

참고: 다중 사용자 운영 체제에서는 마스터 구성 파일이 아니라 사용자의 Configuration 폴더에 있는 구성 파일 사본을 편집합니다. 자세한 내용은 71페이지의 "Configuration 폴더 및 Extension"을 참조하십시오.

Extension API

Dreamweaver는 Extension API의 함수를 호출하여 각 유형의 Extension을 구현합니다. 이러한 함수의 본문은 각 Extension 유형에 기술된 대로 작성해야 하며 Dreamweaver에서 필요로 하는 반환값을 지정해야 합니다.

C 프로그래밍 언어를 직접 사용하는 개발자는 DLL(동적 링크 라이브러리)을 만들 수 있게 해 주는 C 확장성 API를 사용할 수 있습니다. 이러한 API에 제공된 기능은 C DLL을 JavaScript로 래핑하므로 Extension이 Dreamweaver에서 문제 없이 작동할 수 있습니다.

Extension API의 설명서에는 각 함수의 기능, Dreamweaver에서 각 함수를 호출하는 시기 및 Dreamweaver에서 필요로 하는 함수 반환값에 대해 간략히 설명되어 있습니다.

Extension에서 특정 작업을 수행하는 데 사용할 수 있는 함수를 제공하는 유틸리티 API 및 JavaScript API에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**를 참조하십시오.

Dreamweaver에서 JavaScript가 처리되는 방식

Dreamweaver는 시작 시 Configuration/**extension_type** 폴더를 검사합니다. 이 폴더에 Extension 파일이 있으면 Dreamweaver에서는 다음 단계에 따라 JavaScript를 처리합니다.

- 시작 및 종료 SCRIPT 태그 사이의 모든 내용을 컴파일합니다.

- 함수 선언에 해당하지 않는 SCRIPT 태그 내의 모든 코드를 실행합니다.

  참고: 일부 Extension은 초기화하는 데 전역 변수가 필요하므로 시작할 때 반드시 이 과정을 거쳐야 합니다.

Dreamweaver에서는 SCRIPT 태그의 SRC 속성에 지정된 외부 JavaScript 파일에 대해 다음 작업을 수행합니다.

- 파일 내용을 읽습니다.
- 코드를 컴파일합니다.
- 프로시저를 실행합니다.

참고: Extension 파일의 JavaScript 코드에 문자열 "가 포함되어 있으면 JavaScript 인터프리터는 이 문자열을 종료 script 태그로 읽어 종료되지 않은 문자열 리터럴 오류를 보고합니다. 이 문제를 해결하려면 문자열을 여러 부분으로 나누고 "<' + '/SCRIPT>" 와 같이 결합합니다.

사용자가 명령 및 비헤이비어 액션 Extension 유형에 대한 메뉴에서 명령이나 액션을 선택하면 Dreamweaver에서는 onLoad 이벤트 핸들러(body 태그에 있는 경우)의 코드를 실행합니다.

문서 본문에 객체 Extension에 대한 양식이 포함되어 있으면 Dreamweaver에서는 body 태그의 onLoad 이벤트 핸들러에 있는 코드를 실행합니다.

다음 Extension의 경우 body 태그에 있는 onLoad 핸들러는 Dreamweaver에서 무시됩니다.

- 데이터 변환기
- 속성 관리자
- 부동 패널

모든 Extension에서 사용자가 다른 이벤트 핸들러(예: onBlur="alert('This is a required field.')")가 첨부된 양식 필드와 상호 작용하면 Dreamweaver에서는 해당 이벤트 핸들러의 코드를 실행합니다.

Dreamweaver에서는 링크 내에서 이벤트 핸들러를 사용할 수 있습니다. 링크의 이벤트 핸들러에는 다음 예제와 같은 구문을 사용해야 합니다.

```
<a href="#" onMouseDown=alert('hi')>link text</a>
```

항상 play로 설정되는 플러그인은 Extension의 BODY에서 지원됩니다. document.write() 문, Java 애플릿 및 Microsoft ActiveX 컨트롤은 Extension에서 지원되지 않습니다.

도움말 표시

일부 Extension API에는 displayHelp() 함수가 포함되며 Extension에 이 함수를 포함하면 Dreamweaver에서는 다음과 같은 두 가지 작업을 수행합니다.

- 인터페이스에 [도움말] 버튼을 추가합니다.
- 사용자가 [도움말] 버튼을 클릭하면 displayHelp()를 호출합니다.

도움말을 표시하려면 displayHelp() 함수의 본문을 작성해야 합니다. displayHelp() 함수의 코드를 작성하는 방법에 따라 Extension에서 도움말이 표시되는 방식이 달라집니다. dreamweaver.browseDocument() 함수를 호출하여 브라우저에서 파일을 열 수도 있고 경고 상자의 다른 절대 위치 요소에 메시지를 표시하는 것과 같은 독특한 방법으로 도움말을 표시할 수도 있습니다.

다음 예제에서는 displayHelp() 함수에서 dreamweaver.browseDocument()를 호출하여 도움말을 표시합니다.

```
// The following instance of displayHelp() opens a browser to display a file
// that explains how to use the extension.
function displayHelp() {

    var myHelpFile = dw.getConfigurationPath() + "ExtensionsHelp/myExtHelp.htm";
    dw.browseDocument(myHelpFile);
}
```

Extension 지역화

다음 기술을 사용하면 Extension을 해당 지역의 언어로 쉽게 번역할 수 있습니다.

- Extension을 HTML 파일과 JavaScript 파일로 분리합니다. HTML 파일은 복제하여 지역화할 수 있으나 JavaScript 파일은 지역화할 수 없습니다.
- 표시 문자열은 JavaScript 파일에 정의하지 않습니다. 경고 또는 UI 코드가 있는지 검사합니다. 지역화 가능한 모든 문자열을 Dreamweaver Configuration/Strings 폴더에 별도의 XML 파일로 추출합니다.
- 필수 이벤트 핸들러 외에는 JavaScript 코드를 HTML 코드에 작성하지 않습니다. 이렇게 하면 HTML 파일을 복제하고 다른 언어로 번역한 후에 각 번역에 대해 매번 버그를 수정하지 않아도 됩니다.

XML 문자열 파일

모든 문자열을 Dreamweaver Configuration/Strings 폴더의 XML 파일에 저장합니다. 이렇게 하면 서로 관련된 Extension 파일을 여러 개 설치하는 경우 단일 XML 파일에서 모든 문자열을 공유할 수 있습니다. 또한 가능한 경우 C++ 및 JavaScript Extension 모두에서 동일한 문자열을 참조할 수도 있습니다.

예를 들어 myExtensionStrings.xml이라는 파일을 만들 수 있습니다. 다음 예제에서는 이 파일의 형식을 보여 줍니다.

```
<strings>
    <!-- errors for feature X -->
    <string id="featureX/subProblemY" value="There was a with X when you did Y. Try not to
        do Y!"/>
    <string id="featureX/subProblemZ" value="There was another problem with X, regarding Z.
        Don't ever do Z!"/>
</strings>
```

이제 JavaScript 파일에서는 다음 예제와 같이 dw.loadString() 함수를 호출하여 이 번역 가능한 문자열을 참조할 수 있습니다.

```
function initializeUI()
{
    ...
    if (problemYhasOccured)
    {
        alert(dw.loadString("featureX/subProblemY");
    }
}
```

문자열 식별자에 슬래시(/) 문자를 사용할 수 있지만 공백 문자는 사용할 수 없습니다. 슬래시를 사용하여 필요에 맞게 계층 구조를 만들고 모든 문자열을 단일 XML 파일에 포함할 수 있습니다.

참고: Configuration/Strings 폴더에서 cc로 시작하는 파일은 Contribute 파일입니다. 예를 들어 ccSiteStrings.xml은 Contribute 파일입니다.

포함된 값이 있는 지역화 가능한 문자열

일부 표시 문자열에는 값이 포함됩니다. errMsg() 함수를 사용하여 이러한 문자열을 표시할 수 있습니다. C 언어의 printf() 함수와 유사한 errMsg() 함수는 Configuration/Shared/MM/Scripts/CMN 폴더의 string.js 파일에 정의되어 있습니다. 자리 표시자 문자인 퍼센트 기호(%)와 s를 사용하여 문자열 내에서 값이 나타날 위치를 나타낸 다음, 문자열과 변수 이름을 errMsg()에 인수로 전달합니다. 예를 들면 다음과 같습니다.

```
<string id="featureX/fileNotFoundInFolder" value="File %s could not be found in folder %s."/>
```

다음 예제에서는 문자열을 문자열에 포함된 변수와 함께 alert() 함수에 전달하는 방법을 보여 줍니다.

```
if (fileMissing)
{
    alert( errMsg(dw.loadString("featureX/fileNotFoundInFolder"),fileName,
        folderName) );
}
```

Extension Manager 작업

다른 사용자를 위한 Extension을 만들려면 [도움말] > [Extension을 만드는 방법] 범주에 있는 Adobe Exchange 웹 사이트(www.adobe.com/go/exchange_kr)의 지침에 따라 Extension을 패키지화해야 합니다. Extension Manager에서 Extension을 작성하고 테스트한 다음 [파일] > [확장 패키지]를 선택합니다. Extension 패키지를 만든 후에는 [파일] > [확장 전송]을 선택하여 Extension Manager에서 Exchange로 Extension 패키지를 전송할 수 있습니다.

Adobe Extension Manager는 Dreamweaver와 함께 제공됩니다. Extension Manager의 사용 방법에 대한 자세한 내용은 Extension Manager 도움말 파일과 Adobe Exchange 웹 사이트를 참조하십시오.

5장: Extension용 사용자 인터페이스

대부분의 Extension은 UI(사용자 인터페이스)를 통해 사용자로부터 정보를 받도록 구성됩니다. 예를 들어 marquee 태그에 대한 속성 관리자 Extension을 만드는 경우 사용자가 방향 및 높이와 같은 속성을 지정할 수 있도록 해야 합니다. Extension을 전송하여 Adobe 인증을 받으려면 Adobe Exchange 웹 사이트(www.adobe.com/go/exchange_kr)의 Extension Manager 파일에서 사용할 수 있는 지침을 따라야 합니다. 이러한 지침은 Extension 작성에 제한을 두기 위한 것이 아니라 인증된 Extension이 Adobe Dreamweaver UI(사용자 인터페이스) 내에서 효과적으로 작동하고 Extension UI의 디자인으로 인해 해당 기능이 저하되지 않도록 하기 위한 것입니다.

Extension 사용자 인터페이스 디자인 지침

일반적으로 Extension을 만드는 목적은 사용자가 빈번히 수행하는 작업을 쉽게 하는 것입니다. Extension을 만들어 반복적인 작업을 자동화할 수 있습니다. 작업의 일부 단계를 변경하거나 Extension에서 처리하는 코드의 특정 속성을 변경할 수 있습니다. 이러한 변수 값에 대한 사용자 입력을 받기 위해 UI를 만듭니다.

예를 들어 웹 카탈로그를 업데이트하기 위한 Extension을 만들 수 있습니다. 사용자는 이미지 소스, 항목 설명, 가격 등에 대한 값을 주기적으로 변경해야 합니다. 값을 변경하더라도 해당 값을 가져오고 웹 사이트에 표시할 정보의 서식을 지정하는 절차는 동일하게 유지됩니다. 간단한 Extension을 사용하면 서식 지정 작업만 자동화하고 이미지 소스, 항목 설명, 가격 등에 대해 새로 업데이트된 값은 사용자가 직접 입력하도록 할 수 있습니다. 보다 강력한 Extension에서는 이러한 값을 데이터베이스에서 주기적으로 검색할 수 있습니다.

Extension UI의 용도는 사용자가 입력하는 정보를 받기 위한 것입니다. 이 정보는 Extension에서 수행하는 반복 작업의 가변적인 부분을 처리합니다. Dreamweaver에서는 Extension UI 컨트롤을 만들기 위한 기본 구성 블록으로 HTML 및 JavaScript 양식 요소를 지원하며 자체 HTML 렌더러를 사용하여 UI를 표시합니다. 따라서 Extension UI는 텍스트 설명과 양식 입력 필드에 해당하는 두 개의 열로 구성된 테이블이 포함된 간단한 HTML 파일로 만들 수 있습니다.

Extension을 디자인할 때는 필요한 변수와 이러한 변수를 가장 잘 처리할 수 있는 양식 요소를 결정해야 합니다.

Extension UI를 디자인할 때 고려해야 할 기본 지침은 다음과 같습니다.

- Extension의 이름을 지정하려면 HTML 파일의 title 태그에 이름을 포함합니다. Dreamweaver에서 이 이름은 Extension 제목 막대에 표시됩니다.
- 텍스트 레이블은 UI의 왼쪽에 오른쪽 정렬로 표시하고 텍스트 상자는 UI의 오른쪽에 왼쪽 정렬로 표시합니다. 이 정렬 방법을 사용하면 사용자가 텍스트 상자의 처음 부분을 쉽게 찾을 수 있습니다. 텍스트 상자 다음에는 설명이나 측정 단위와 같은 짧은 텍스트를 표시할 수 있습니다.
- 체크 상자와 라디오 버튼의 레이블은 UI의 오른쪽에 왼쪽 정렬로 표시합니다.
- 코드를 쉽게 읽을 수 있도록 텍스트 상자에 논리적인 이름을 지정합니다. Dreamweaver를 사용하여 Extension UI를 만드는 경우, 속성 관리자나 [퀵 태그 편집기]를 사용하여 필드에 이름을 지정할 수 있습니다.

일반적으로 UI를 만든 다음에는 Extension 코드를 테스트하여 코드가 다음 UI 관련 작업을 올바르게 수행하는지 확인합니다.

- 텍스트 상자에서 값 가져오기
- 텍스트 상자의 기본값 설정 또는 선택 항목에서 값 가져오기
- 사용자 문서에 변경 사항 적용

Dreamweaver HTML 렌더링 컨트롤

Dreamweaver 4와 그 이전 버전의 Dreamweaver에서는 Microsoft Internet Explorer와 Netscape Navigator에서보다 더 많은 공간이 양식 컨트롤 주위에 렌더링되었습니다. Dreamweaver에서는 자체 HTML 렌더링 엔진을 사용하여 Extension UI를 표시하므로 Extension UI의 양식 컨트롤 주위에 여분의 공간이 나타납니다.

이후 버전의 Dreamweaver에서는 양식 컨트롤 렌더링을 개선하여 더욱 브라우저에 가깝게 표현하고 있습니다. 향상된 렌더링 기능을 활용하려면 다음 예제와 같이 세 가지 새로운 DOCTYPE 문 중 하나를 Extension 파일에 사용합니다.

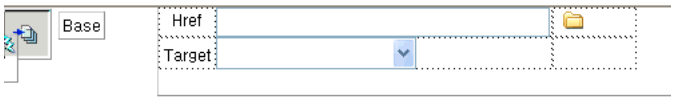
```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//floater">
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//pi">
```

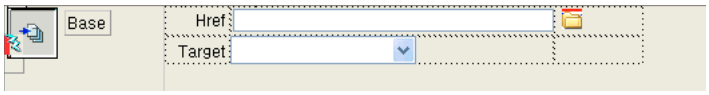
대부분의 경우 DOCTYPE 문은 문서의 첫 번째 행에 와야 합니다. 그러나 Extension 관련 지시문과의 충돌을 방지하기 위해 이제는 DOCTYPE 문 및 지시문의 순서에 상관이 없게 되었습니다. 단, 해당 문 및 지시문이 열기 HTML 태그 앞에 나타나야 합니다. 이전 버전에서는 Extension 관련 지시문이 파일의 첫 번째 행에 와야 했습니다. 예를 들어 속성 관리자 파일의 맨 위에 나타나는 주석이나 명령의 MENU-LOCATION=NONE 지시문이 이에 해당합니다.

새로운 DOCTYPE 문을 사용하면 Dreamweaver [디자인] 뷰에서 Extension을 보고 해당 Extension이 사용자에게 표시될 모양을 확인할 수 있습니다.

Configuration/Commands 폴더의 CFLoginWizard.htm, TeamAdminDlgDates.html 및 TeamAdminDlgDays.html 파일에서는 대화 상자가 사용된 세 가지 예제를 볼 수 있습니다.

다음 예제에서는 양식 컨트롤 렌더링을 개선하는 DOCTYPE 문이 없는 기본 속성 관리자와, DOCTYPE 문이 있는 기본 속성 관리자를 보여 줍니다.





Extension의 사용자 정의 UI 컨트롤

Dreamweaver에서는 표준 HTML 양식 요소 외에 사용자 정의 컨트롤도 지원하므로 전문적인 모양의 유연한 인터페이스를 만들 수 있습니다.

편집 가능한 선택 목록

편집 가능한 선택 목록(콤보 상자라고도 함)을 사용하면 선택 목록의 기능과 텍스트 상자의 기능을 조합할 수 있습니다.

Extension UI에는 select 태그를 사용하여 정의한 팝업 메뉴가 포함되는 경우가 많습니다. Dreamweaver에서는 select 태그에 editable="true"를 추가하여 Extension의 팝업 메뉴를 편집 가능하게 만들 수 있습니다. 기본값을 설정하려면 editText 속성을 설정하고 선택 목록에 표시할 값을 설정합니다.

다음 예제에서는 편집 가능한 선택 목록의 설정 방법을 보여 줍니다.

```
<select name="travelOptions" style="width:250px" editable="true" editText="other
    (please specify)">
<option value="plane">plane</option>
<option value="car">car</option>
<option value="bus">bus</option>
</select>
```

Extension에 선택 목록을 사용할 때는 편집 가능한 속성이 있는지 확인하고 해당 속성의 값을 확인합니다. 값이 없으면 선택 목록이 편집 가능하지 않음을 나타내는 기본값 false가 반환됩니다.

일반적인 편집 불가능한 선택 목록과 마찬가지로 편집 가능한 선택 목록에도 selectedIndex 속성이 있습니다. 자세한 내용은 91 페이지의 "Dreamweaver DOM의 객체, 속성 및 메서드"를 참조하십시오. 이 속성은 텍스트 상자가 선택된 경우 -1을 반환합니다.

편집 가능한 활성 텍스트 상자의 값을 Extension으로 읽어 들이려면 editText 속성의 값을 읽습니다. editText 속성은 사용자가 편집 가능한 텍스트 상자에 입력한 문자열이나 editText 속성의 값을 반환합니다. 텍스트가 입력되지 않고 editText에 지정된 값이 없는 경우에는 빈 문자열을 반환합니다.

Dreamweaver에서는 select 태그에 대해 다음과 같은 사용자 정의 속성을 추가하여 편집 가능한 팝업 메뉴를 제어합니다.

속성 이름	설명	허용되는 값
editable	팝업 메뉴에 편집 가능 텍스트 영역이 포함되도록 선언합니다.	부울 값(true 또는 false)
editText	편집 가능한 텍스트 영역 내의 텍스트를 포함하거나 설정합니다.	임의 값의 문자열

참고: 편집 가능한 선택 목록은 Dreamweaver에서 사용할 수 있습니다.

다음 예제에서는 일반적인 JavaScript 함수를 사용하여 편집 가능한 선택 목록이 포함된 명령 Extension을 만듭니다.

예제 만들기

1 텍스트 편집기에서 빈 파일을 새로 만듭니다.

2 다음 코드를 입력합니다.

```
<html>
<head>
    <title>Editable Dropdown Test</title>
    <script language="javascript">
    function getAlert()
    {

        var i=document.myForm.mySelect.selectedIndex;
        if (i>=0)
        {
            alert("Selected index: " + i + "\n" + "Selected text " +
                    document.myForm.mySelect.options[i].text);
        }
        else
        {
            alert("Nothing is selected" + "\n" + "or you entered a value");
        }
    }
    function commandButtons()
    {
        return new Array("OK", "getAlert()", "Cancel", "window.close()");
    }
    </script>
</head>

<body>
<div name="test">
```

```
<form name="myForm">
<table>
    <tr>
        <td colspan="2">
        <h4>Select your favorite</h4>
        </td>
    </tr>
    <tr>
        <td>Sport:</td>
        <td>
        <select name="mySelect" editable="true" style="width:150px"
            editText="Editable Text">
            <option> Baseball</option>
            <option> Football </option>
            <option> Soccer </option>
        </select>
        </td>
    </tr>
</table>
</form>
</div>
</body>
</html>
```

3 이 파일을 Dreamweaver의 Configuration/Commands 폴더에 EditableSelectTest.htm이라는 이름으로 저장합니다.

예제 테스트

1 Dreamweaver를 다시 시작합니다.

2 [명령] > [EditableSelectTest]를 선택합니다.

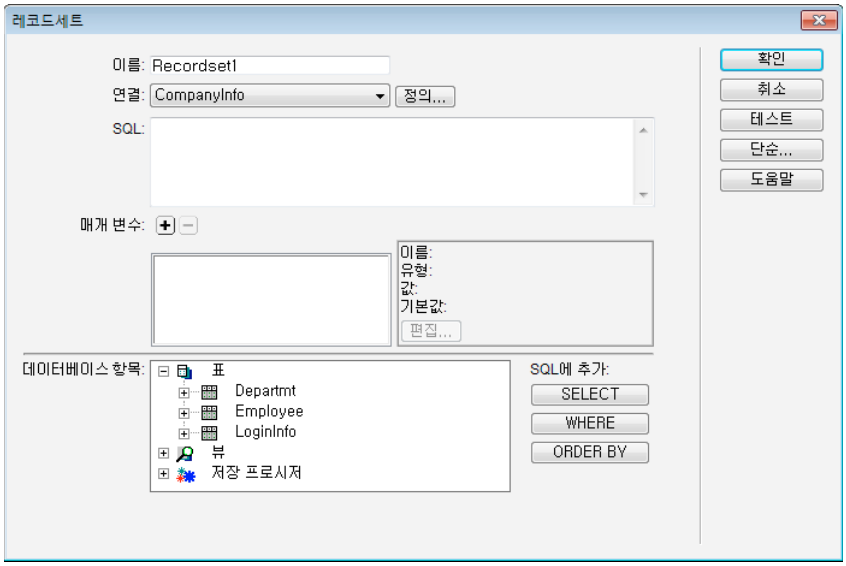
목록에서 값을 선택하면 해당 값의 인덱스와 텍스트가 경고 메시지에 표시됩니다. 값을 입력하면 선택된 항목이 없다는 경고 메시지가 나타납니다.

데이터베이스 컨트롤

데이터베이스 컨트롤을 사용하면 데이터 계층 구조 및 필드를 쉽게 표시할 수 있습니다.

Dreamweaver에서 HTML select 태그를 확장하여 데이터베이스 트리 컨트롤을 만들 수 있습니다. 또한 변수 표 컨트롤을 추가할 수도 있습니다. 데이터베이스 트리 컨트롤은 데이터베이스 스키마를 표시하고 변수 표 컨트롤은 표 형식 정보를 표시합니다.

다음 그림에서는 데이터베이스 컨트롤과 변수 표 컨트롤을 사용하는 고급 [레코드세트] 대화 상자를 보여 줍니다.



데이터베이스 트리 컨트롤 추가

데이터베이스 트리 컨트롤에는 다음과 같은 속성이 있습니다.

속성 이름	설명
name	데이터베이스 트리 컨트롤의 이름
control.style	폭 및 높이(픽셀)
type	컨트롤 유형
connection	[연결 관리자]에서 정의한 데이터베이스 연결의 이름. 이 속성이 비어 있으면 해당 컨트롤이 비어 있게 됩니다.
noexpandbuttons	이 속성을 지정하면 트리 컨트롤에 확장 플러스(+) 또는 축소 마이너스(-) 표시기나 연관된 화살표(Macintosh)가 표시되지 않습니다. 이 속성은 여러 개의 열로 구성된 목록 컨트롤을 그리는 데 유용합니다.
showheaders	이 속성을 지정하면 트리 컨트롤의 맨 위에 각 열의 이름을 나타내는 머리글이 표시됩니다.

select 태그 안에 있는 option 태그는 무시됩니다.

대화 상자에 데이터베이스 트리 컨트롤을 추가하려면 다음 샘플 코드에서 큰따옴표로 묶인 변수를 적절히 변경하여 사용하면 됩니다.

```
<select name="DBTree" style="width:400px;height:110px" ¬
type="mmdatabasetree" connection="connectionName" noexpandbuttons showHeaders></select>
```

connection 속성을 변경하면 선택한 데이터를 검색하고 트리에 표시할 수 있습니다. 새 태그의 JavaScript 래퍼 객체로 DBTreeControl 속성을 사용할 수 있습니다. 자세한 예제는 Configuration/Shared/Common/Scripts 폴더의 DBTreeControlClass.js 파일을 참조하십시오.

변수 표 컨트롤 추가

변수 표 컨트롤에는 다음과 같은 속성이 있습니다.

속성 이름	설명
name	변수 표 컨트롤의 이름
style	컨트롤의 폭(픽셀)
type	컨트롤 유형
columns	각 열에는 이름이 있어야 하며 각 이름은 쉼표로 구분해야 합니다.
columnWidth	각 열의 폭. 각 폭은 쉼표로 구분해야 합니다. 열의 폭을 지정하지 않으면 모든 열의 폭이 동일하게 표시됩니다.

다음 예제에서는 대화 상자에 간단한 변수 표 컨트롤을 추가합니다.

```
<select name="ParamList" style="width:515px;" ¬
type="mmparameterlist columns"="Name,SQL Data ¬
Type,Direction,Default Value,Run-time Value" size=6></select>
```

다음 예제에서는 전체 폭이 500픽셀이며 각기 다른 폭의 열이 다섯 개 포함된 변수 표 컨트롤을 만듭니다.

```
<select
    name="ParamList"
    style="width:500px;"
    type="mmparameterlist"
    columns="Name,SQL Data Type,Direction, Default Value,Run-time Value"
    columnWidth="100,25,11,"
    size=6>
```

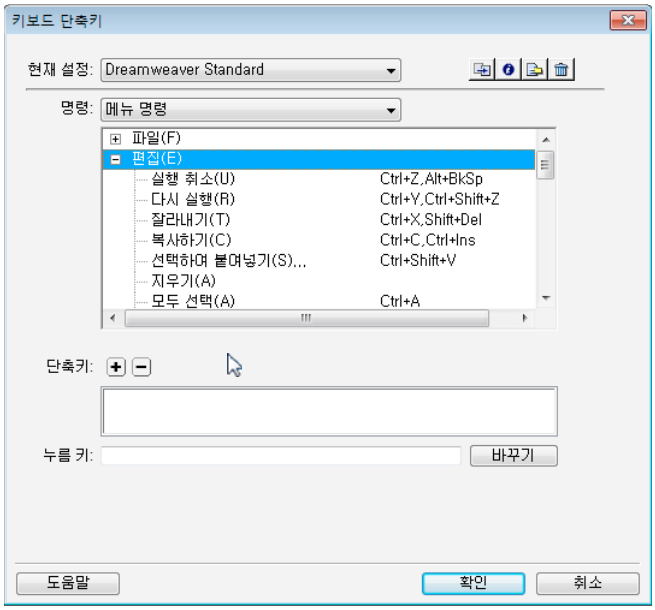
이 예제에서는 폭이 182픽셀인 빈 열을 두 개 만듭니다. 지정된 열의 폭은 총 136픽셀이고 변수 표 컨트롤의 전체 폭은 500픽셀이므로 처음 세 열 다음의 남은 공간은 364픽셀이 됩니다. 따라서 남은 두 열의 폭은 364를 2로 나눈 182픽셀이 됩니다.

이 변수 표 컨트롤에는 변수 표 컨트롤의 데이터에 액세스하고 이를 조작하는 데 사용할 JavaScript 래퍼 객체도 있습니다. Configuration/Shared/MM/Scripts/Class 폴더의 GridControlClass.js 파일에서 이 컨트롤의 구현을 볼 수 있습니다.

트리 컨트롤

트리 컨트롤은 정보를 확장/축소 가능한 노드로 구성합니다.

트리 컨트롤은 데이터를 계층 구조로 표시하며 사용자는 트리의 노드를 확장하거나 축소할 수 있습니다. MM:TREECONTROL 태그를 사용하면 모든 유형의 정보에 대한 트리 컨트롤을 만들 수 있습니다. 81페이지의 "데이터베이스 트리 컨트롤 추가"에서 설명한 데이터베이스 트리 컨트롤과 달리 트리 컨트롤은 데이터베이스와 연결할 필요가 없습니다. Dreamweaver 키보드 단축키 편집기에서는 다음 그림과 같이 트리 컨트롤을 사용합니다.



트리 컨트롤 만들기

MM:TREECONTROL 태그는 트리 컨트롤을 만들며, 하나 이상의 태그를 사용하여 다음 목록에서 설명하는 것과 같은 구조를 추가할 수 있습니다.

- MM:TREECOLUMN은 트리 컨트롤의 열을 정의하는 빈 선택적 태그입니다.
- MM:TREENODE는 트리의 노드를 정의하는 선택적 태그입니다. 이 태그는 비어 있지 않은 태그로, 다른 MM:TREENODE 태그만 포함할 수 있습니다.

MM:TREECONTROL 태그에는 다음과 같은 속성이 있습니다.

속성 이름	설명
name	트리 컨트롤의 이름
size	선택 사항으로, 컨트롤에 표시되는 행의 수를 나타내며 기본값은 5입니다.
theControl	선택 사항으로, theControl 속성에 있는 노드의 수가 size 속성의 값을 초과하면 스크롤 막대가 표시됩니다.
multiple	선택 사항으로, 복수 선택이 가능하게 하며 기본값은 단일 선택입니다.
style	선택 사항으로, 트리 컨트롤의 높이 및 폭에 대한 스타일 정의입니다. 이 속성이 지정되면 size 속성보다 우선합니다.
noheaders	선택 사항으로, 열 머리글이 표시되지 않도록 지정합니다.

MM:TREECOLUMN 태그에는 다음과 같은 속성이 있습니다.

속성 이름	설명
name	열의 이름
value	열 머리글에 나타나는 문자열
width	열의 픽셀 단위 폭(백분율은 지원되지 않음)으로서, 기본값은 100입니다.
align	선택 사항으로, 열에 표시되는 텍스트의 정렬 방식(왼쪽, 오른쪽 또는 가운데)을 지정합니다. 기본값은 왼쪽입니다.
state	열의 표시/숨김 여부를 지정합니다.

보다 쉽게 읽을 수 있도록, TREECOLUMN 태그는 다음 예제와 같이 MM:TREECONTROL 태그 바로 다음에 옵니다.

```
<MM:TREECONTROL name="tree1">
<MM:TREECOLUMN name="Column1" width="100" state="visible">
<MM:TREECOLUMN name="Column2" width="80" state="visible">
...
</MM:TREECONTROL>
```

다음 표에서는 MM:TREENODE 속성에 대해 설명합니다.

속성 이름	설명
name	노드의 이름
value	지정된 노드의 내용이 포함됩니다. 열이 둘 이상인 경우에는 각 열의 문자열을 파이프 문자로 구분하여 지정합니다. 빈 열을 지정하려면 파이프 문자(|) 앞에 공백 문자를 하나 입력합니다.
state	문자열 "expanded" 또는 "collapsed"를 사용하여 노드가 확장되는지 또는 축소되는지를 지정합니다.
selected	트리에 MULTIPLE 속성이 있으면 둘 이상의 트리 노드에 이 속성을 설정하여 여러 개의 노드를 선택할 수 있습니다.
icon	선택 사항으로, 사용할 내장 아이콘의 인덱스는 0부터 시작합니다. 0 = 아이콘 없음, 1 = Dreamweaver 문서 아이콘, 2 = 다중 문서 아이콘

예를 들어 다음 트리 컨트롤의 모든 노드는 확장됩니다.

```
<mm:treecontrol name="test" style="height:300px;width:300px">

<mm:treenode value="rootnode1" state="expanded">
<mm:treenode value="node1" state="expanded"></mm:treenode>
<mm:treenode value="node3" state="expanded"></mm:treenode>
</mm:treenode>

<mm:treenode value="rootnode2"0state="expanded">
<mm:treenode value="node2" state="expanded"></mm:treenode>
<mm:treenode value="node4" state="expanded"></mm:treenode>
</mm:treenode>

</mm:treecontrol>
```

트리 컨트롤을 사용하여 내용 조작

트리 컨트롤과 트리 컨트롤 내의 노드는 HTML 태그로 구현되며, Dreamweaver에서 파싱되어 문서 트리에 저장됩니다. 이러한 태그는 다른 문서 노드와 동일한 방법으로 조작할 수 있습니다. DOM 함수 및 메서드에 대한 자세한 내용은 90페이지의 “Dreamweaver 문서 객체 모델”을 참조하십시오.

노드 추가 프로그래밍 방식으로 기존 트리 컨트롤에 노드를 추가하려면 MM:TREECONTROL 태그나 기존 MM:TREENODE 태그 중 하나의 innerHTML 속성을 설정합니다. 트리 노드의 inner HTML 속성을 설정하면 중첩 노드가 만들어집니다.

다음 예제에서는 트리의 최상위 수준에 노드를 추가합니다.

```
var tree = document.myTreeControl;
//add a top-level node to the bottom of the tree
tree.innerHTML = tree.innerHTML + '<mm:treenode name="node3" value="node3">';
```

자식 노드 추가 현재 선택한 노드에 자식 노드를 추가하려면 선택한 노드의 innerHTML 속성을 설정합니다.

다음 예제에서는 현재 선택된 노드에 자식 노드를 추가합니다.

```
var tree = document.myTreeControl;
var selNode = tree.selectedNodes[0];
//deselect the node, so we can select the new one
selnode.removeAttribute("selected");
//add the new node to the top of the selected node's children
selNode.innerHTML = '<mm:treenode name="item10" value="New item11" expanded selected>' + ¬
selNode.innerHTML;
```

노드 삭제 문서 구조에서 현재 선택한 노드를 삭제하려면 innerHTML 또는 outerHTML 속성을 사용합니다.

다음 예제에서는 선택된 노드 전체와 해당 노드의 모든 자식 노드를 삭제합니다.

```
var tree = document.myTreeControl;
var selNode = tree.selectedNodes[0];
selNode.outerHTML = "";
```

색상 버튼 컨트롤 추가

색상 버튼 컨트롤을 사용하면 Extension에 색상 선택기 인터페이스를 추가할 수 있습니다.

Dreamweaver에서는 텍스트, 체크 상자 및 버튼과 같은 표준 입력 유형 외에 Extension의 추가 입력 유형인 mmcolorbutton을 지원합니다.

코드에 <input type="mmcolorbutton">이라고 입력하면 사용자 인터페이스에 색상 선택기가 나타납니다. 입력 태그에 값 속성을 설정하여 색상 선택기의 기본 색상을 설정할 수 있습니다. 값을 설정하지 않으면 색상 선택기는 기본적으로 회색으로 표시되며, 입력 객체의 값 속성은 빈 문자열을 반환합니다.

다음 예제에서는 유효한 mmcolorbutton 태그를 보여 줍니다.

```
<input type="mmcolorbutton" name="colorbutton" value="#FF0000">
<input type="mmcolorbutton" name="colorbutton" value="teal">
```

색상 버튼에는 색상이 변경될 때 발생하는 하나의 이벤트 onChange가 있습니다.

텍스트 상자와 색상 선택기의 동기화 상태를 유지합니다. 다음 예제에서는 텍스트 상자의 색상과 색상 선택기의 색상을 동기화하는 텍스트 상자를 만듭니다.

```
<input type = "mmcolorbutton" name="fgcolorPicker" onChange="document.fgcolorText.value=this.value">
<input type = "test" name="fgcolorText" onBlur="document.fgColorPicker.value=this.value">
```

이 예제에서 사용자가 텍스트 상자의 값을 변경한 다음 탭으로 이동하거나 다른 위치를 클릭하면 색상 선택기가 업데이트되어 텍스트 상자에서 지정한 색상이 표시됩니다. 사용자가 색상 선택기를 사용하여 새 색상을 선택할 때마다 텍스트 상자가 업데이트되어 해당 색상의 16진수 값을 표시합니다.

Dreamweaver에 Flash 내용 추가

Flash 내용(SWF 파일)은 Dreamweaver 인터페이스에 객체나 명령의 일부로 표시될 수 있습니다. 이 Flash 지원 기능은 Flash 양식, 애니메이션, ActionScript 또는 그 밖의 Flash 내용을 사용하는 Extension을 만드는 경우에 특히 유용합니다.

기본적으로는 Dreamweaver 객체 및 명령의 대화 상자 표시 기능을 활용합니다. 즉, form 태그와 object 태그를 사용하여 Flash 내용을 Dreamweaver 대화 상자에 포함할 수 있습니다. 객체를 만드는 방법에 대한 자세한 내용은 99페이지의 “삽입 막대 객체”를 참조하고, 명령에 대한 자세한 내용은 124페이지의 “명령”을 참조하십시오.

간단한 Flash 대화 상자 예제

이 예제에서는 Dreamweaver를 사용하여 명령을 만듭니다. 만들어진 명령은 사용자가 [명령] 메뉴에서 해당 명령을 클릭하면 myFlash.swf라는 SWF 파일을 표시합니다. 이 예제를 연습하기 전에 명령을 만드는 방법을 자세히 알아보려면 124페이지의 "명령"을 참조하십시오.

참고: 이 예제에서는 Dreamweaver 응용 프로그램 설치 폴더의 Configuration/Commands 폴더에 myFlash.swf라는 SWF 파일이 이미 있다고 가정합니다. 다른 SWF 파일로 이 예제를 테스트하려면 원하는 SWF 파일을 응용 프로그램의 Commands 폴더에 저장하고 코드에 나타나는 myFlash.swf를 모두 해당 SWF의 파일 이름으로 대체합니다.

Dreamweaver에서 새 기본 HTML 파일을 엽니다. 이 파일이 사용자의 명령 정의 파일입니다. 열기 및 닫기 title 태그 사이에 **My Flash Movie**라고 입력합니다. 그러면 페이지의 헤드에 다음과 같이 표시됩니다.

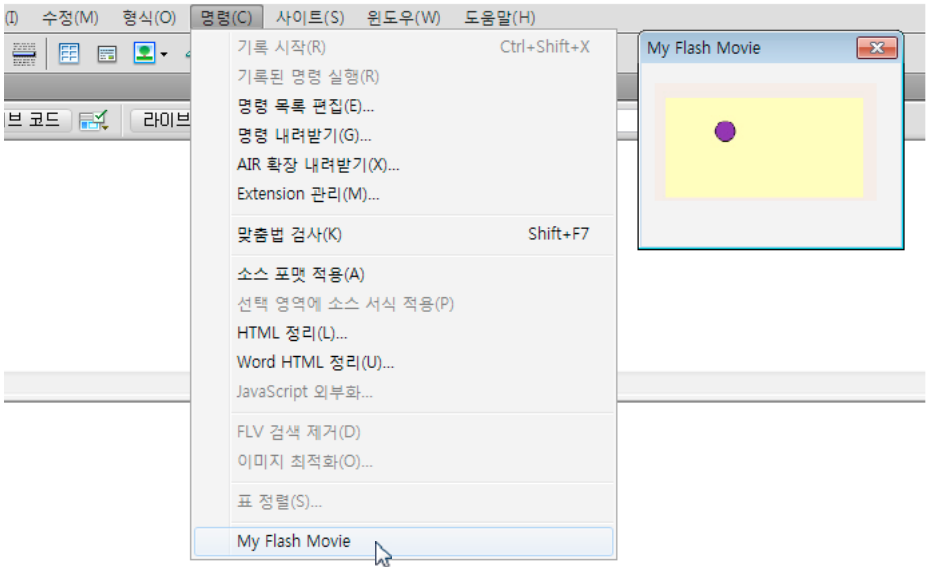
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>My Flash Movie</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
```

이제, 이 파일을 My Flash Movie.htm이라는 이름으로 응용 프로그램의 Configuration/Commands 폴더에 저장합니다. 아직 파일을 닫지는 마십시오. 상대 경로로 SWF 파일을 포함할 수 있도록 이 시점에서 파일을 저장합니다. 그렇지 않으면 Dreamweaver에서 절대 경로가 사용됩니다.

HTML 문서로 돌아가서 열기 및 닫기 body 태그 사이에 열기 및 닫기 form 태그를 추가합니다. 그런 다음 form 태그 내에서 [삽입] > [미디어] > [Flash] 옵션을 사용하여 SWF 파일을 이 명령 정의 파일에 추가합니다. 대화 상자가 나타나면 Commands 폴더에 있는 SWF 파일을 선택하고 [확인]을 클릭합니다. 이제 이 명령 정의 파일은 다음 예제와 유사합니다. 단, width 및 height 속성은 지정한 SWF 파일의 속성에 따라 달라질 수 있습니다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>My Flash Movie</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body>
<body>
<form>
    <object id="FlashID" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000" width="700" height="150">
    <param name="movie" value="myFlash.swf">
    <!--[if !IE]>-->
    <object type="application/x-shockwave-flash" data="myFlash.swf" width="700"  height="150">
    <!--<![endif]-->
    <param name="quality" value="high"/>
    <param name="wmode" value="opaque" />
    <param name="swfversion" value="8.0.35.0" />
    <!-- This param tag prompts users with Flash Player 6.0 r65 and higher to download the latest version
of Flash Player. Delete it if you don't want users to see the prompt. -->
    <param name="expressinstall" value="../../../../../ColdFusion8/wwwroot/lr/Scripts/expressInstall.swf"
/>
    <!-- The browser displays the following alternative content for users with Flash Player 6.0 and older.
-->
    <div>
    <h4>Content on this page requires a newer version of Adobe Flash Player.</h4>
    <p><a href="http://www.adobe.com/go/getflashplayer"><img
src="http://www.adobe.com/images/shared/download_buttons/get_flash_player.gif" alt="Get Adobe Flash
player" width="112" height="33" /></a></p>
    </div>
    <!--[if !IE]>-->
</object>
<!--<![endif]-->
</object>
</form>
</body>
</html>
```

파일을 다시 저장한 다음 Dreamweaver를 종료했다가 다시 시작합니다. [명령] > [My Flash Movie] 옵션을 선택하면 다음 그림과 같이 SWF 파일이 Dreamweaver 대화 상자에 나타납니다.



이 예제에서는 Dreamweaver의 SWF 파일 지원 기능을 간단하게 구현하는 방법을 보여 줍니다. 객체와 명령을 익숙하게 만들 수 있고 보다 정교한 양식을 설계할 수 있게 되면 SWF 파일을 Dreamweaver Extension에 통합하여 더욱 동적인 사용자 환경을 제공할 수 있습니다. SWF 파일을 표시하는 대화 상자에 버튼을 추가하는 commandButtons() 함수를 작성하는 방법에 대한 자세한 내용은 124페이지의 “명령”을 참조하십시오.

Photoshop 통합 및 스마트 오브젝트

Adobe® Dreamweaver CS5®에서는 Photoshop 파일을 스마트 오브젝트로 가져오고 처리합니다. Photoshop을 사용하여 원본 이미지를 수정하면 Dreamweaver에서 즉시 반영됩니다. Photoshop을 Dreamweaver에 통합하는 API에 대한 자세한 내용은 **Dreamweaver API** 참조 설명서의 "Photoshop 통합"을 참조하십시오.

스마트 오브젝트 예제

이 예제에서는 Dreamweaver를 사용하여 사용자가 [명령] 메뉴에서 명령을 클릭할 때 Photoshop 파일(PSD)을 업데이트하는 명령을 만듭니다. 이 명령이 작동할 수 있으려면 HTML 페이지에 스마트 오브젝트 웹 이미지가 있어야 합니다. 이 예제를 연습하기 전에 명령을 만드는 방법을 자세히 알아보려면 124페이지의 “명령”을 참조하십시오.

Dreamweaver에서 새 기본 HTML 파일을 엽니다. 이 파일이 사용자의 명령 정의 파일입니다. 명령 정의 파일은 다음 예제와 유사합니다.

```
<html xmlns:MMStr ing="http://www.adobe.com /schemes/data/   string/">
<head>
    <title> Smart Objects API</ title >
    <SC RIPT SRC="../Shared/Common/Scripts /dwscripts.js"></SC RIPT>
    <SCRIPT LANGUAGE="Javascript">
function invokeSmartObjectJavaScriptAPICall() {
    var selection = dw.getSelection();
    if (!selection) {
    alert("Err: No selection!");
    return;
    }
    var node = dw.offsetsToNode(selection[0], selection[1]);
    if (!node) {
                alert("Err: No Node!");
                return;
                }
    var imageSrc = node.getAttribute("src");
    if (!imageSrc) {
                    alert("Err: No src attribute!");
                    return;
                    }
    var fullPath = getFullPath(imageSrc);
    if (!fullPath) {
                    alert("Err: No path!");
                    return;
                    }
    //alert(fullPath);
    alert("updateSmartObjectFromOriginal");
    dw.updateSmartObjectFromOriginal(fullPath);
}
    <script>
</head>
<body onload="invokeSmartObjectJavaScriptAPICall()">
</body>
</html>
```

이제 이 파일을 Configuration/Commands 폴더에 smartobjects.htm으로 저장합니다. 대화 상자가 나타나면 Commands 폴더에 있는 SWF 파일을 선택하고 [확인]을 클릭합니다.

6장: Dreamweaver 문서 객체 모델

Adobe Dreamweaver 에서 DOM(문서 객체 모델)은 Extension 개발자에게 있어서 매우 중요한 구조입니다.

DOM은 마크업 언어를 사용하여 만드는 문서의 구조를 정의합니다. 태그와 속성을 객체 및 속성으로 표현하는 DOM을 사용하면 프로그래밍 언어로 문서와 문서 구성 요소에 액세스하고 이를 조작할 수 있습니다.

Dreamweaver DOM

HTML 문서의 구조는 문서 트리로 볼 수 있습니다. 루트는 HTML 태그이며 두 개의 가장 큰 줄기는 head 태그와 body 태그입니다. head 태그의 가지로는 title, style, script, isindex, base, meta, link 태그가 있습니다. body 태그의 가지에는 다음이 포함됩니다.

- 머리글(h1, h2 등)
- 블록 수준 요소(p, div, form 등)
- 인라인 요소(br, img 등)
- 기타 요소 유형

이러한 가지의 잎에는 width, height, alt 등의 속성이 있습니다.

DOM에서 트리 구조는 부모 노드와 자식 노드의 계층 구조로 유지되고 표시됩니다. 루트 노드에는 부모 노드가 없으며 최하위 노드에는 자식 노드가 없습니다. HTML 구조 내의 각 수준에 있는 HTML 요소는 JavaScript에 노드로 노출될 수 있습니다. 이 구조를 사용하여 문서나 문서 내의 요소에 액세스할 수 있습니다.

JavaScript에서는 문서 내의 모든 객체를 이름이나 인덱스로 참조할 수 있습니다. 예를 들어 이름 또는 ID가 "myButton"인 전송 버튼이 문서의 첫 번째 양식에 있는 두 번째 요소일 경우, 이 버튼에 대한 다음 두 가지 참조가 모두 유효합니다.

- 이름으로 참조하는 경우 - document.myForm.myButton
- 인덱스로 참조하는 경우 - document.forms[0].elements[1]

옵션 그룹과 같이 이름이 동일한 여러 객체는 배열로 축소됩니다. 인덱스를 0부터 시작하여 증가시키면 배열의 특정 객체에 액세스할 수 있습니다. 예를 들어, "myForm" 양식에 있는 "myRadioGroup"이라는 이름의 첫 번째 옵션은 document.myForm.myRadioGroup[0]으로 참조됩니다.

사용자 문서 DOM과 Extension DOM 구별

사용자 문서의 DOM과 Extension의 DOM을 구별하는 것은 매우 중요합니다. 이 항목의 내용은 두 유형의 Dreamweaver 문서 모두에 적용되지만 각 DOM의 참조 방식은 서로 다릅니다.

브라우저에서 JavaScript를 사용하는 데 익숙하다면 document를 입력하여 활성 문서에 있는 객체를 참조할 수 있습니다. 예를 들어 document.forms[0]와 같이 입력합니다. Dreamweaver에서 document는 Extension 파일을 참조하고 document.forms[0]는 Extension UI의 첫 번째 양식을 참조합니다. 사용자 문서의 객체를 참조하려면 dw.getDocumentDOM(), dw.createDocument() 또는 사용자 문서 객체를 반환하는 다른 함수를 호출해야 합니다.

예를 들어 활성 문서의 첫 번째 이미지를 참조하려면 dw.getDocumentDOM().images[0]를 사용합니다. 또한 다음 예제에서처럼 변수에 문서 객체를 저장하고 나중에 참조할 때 이 변수를 사용할 수도 있습니다.

```
var dom = dw.getDocumentDOM();  //get the dom of the current document
var firstImg = dom.images[0];
firstImg.src = "myImages.gif";
```

이러한 유형의 표기법은 Configuration 폴더의 모든 파일, 특히 명령 파일에 공통적입니다. dw.getDocumentDOM() 메서드에 대한 자세한 내용은 **Dreamweaver API** 참조 설명서에서 dreamweaver.getDocumentDOM() 함수를 참조하십시오.

Dreamweaver DOM

Dreamweaver DOM에는 W3C(World Wide Web Consortium) DOM 레벨 1 사양의 객체, 속성 및 메서드 중 일부가 포함되어 있습니다. 이러한 객체, 속성 및 메서드는 Microsoft Internet Explorer 4.0 DOM의 일부 속성과 결합됩니다.

Dreamweaver DOM의 객체, 속성 및 메서드

다음 표에서는 Dreamweaver DOM에서 지원하는 객체, 속성, 메서드 및 이벤트를 보여 줍니다. 일부 속성은 특정 객체의 속성으로 액세스될 때 읽기 전용입니다. 불릿(•)은 목록 컨텍스트에 사용될 때 읽기 전용인 속성을 나타냅니다.

객체	속성	메서드	이벤트
window	navigator • document • innerWidth • innerHeight • screenX • screenY •	alert() confirm() escape() unescape() close() setTimeout() clearTimeout() setInterval() clearInterval() resizeTo()	onResize
navigator	platform •	없음	없음
document	forms •(form 객체의 배열) images •(image 객체의 배열) layers •(LAYER, ILAYER 및 절대 위치 요소의 배열) 이름으로 참조하는child 객체 • nodeType • parentNode • childNodes • previousSibling • nextSibling • documentElement • body • URL • parentWindow •	getElementsByTagName() getElementsByAttributeName() getElementById() hasChildNodes()	onLoad

객체	속성	메서드	이벤트
모든 태그/ 요소	nodeType • parentNode • childNodes • tagName • previousSibling • nextSibling • 이름으로 참조하는 속성 innerHTML outerHTML	getAttribute() setAttribute() removeAttribute() getElementsByTagName() getElementsByAttributeName() hasChildNodes()	
form	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: tags:elements • (button, checkbox, password, radio, reset, select, submit, text, file, hidden, image 및 textarea 객체의 배열) mmcolorbutton 이름으로 참조하는 child 객체 •	모든 태그에 사용 가능한 메서드만	없음
layer	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: visibility left top width height zIndex	모든 태그에 사용 가능한 메서드만	없음
image	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: src	모든 태그에 사용 가능한 메서드만	onMouseOver onMouseOut onMouseDown onMouseUp
button reset submit	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: form •	모든 태그에 사용 가능한 메서드 외에 다음 메서드 포함: blur() focus()	onClick
checkbox radio	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: checked • form •	모든 태그에 사용 가능한 메서드 외에 다음 메서드 포함: blur() focus()	onClick

객체	속성	메서드	이벤트
password text file hidden image (field) textarea	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: form • value	모든 태그에 사용 가능한 메서드 외에 다음 메서드 포함: blur() focus() select()	onBlur onFocus
select	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: form • options •(option 객체의 배열) selectedIndex	모든 태그에 사용 가능한 메서드 외에 다음 메서드 포함: blur()(Windows 전용) focus()(Windows 전용)	onBlur(Windows 전용) onChange onFocus(Windows 전용)
option	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: text	모든 태그에 사용 가능한 메서드만	없음
mmcolorbutton	모든 태그에 사용 가능한 속성 외에 다음 속성 포함: name value	없음	onChange
array boolean date function number object string regexp	Netscape Navigator 4.0과 일치	Netscape Navigator 4.0과 일치	없음
text	nodeType • parentNode • childNodes • previousSibling • nextSibling • data	hasChildNodes()	없음

객체	속성	메서드	이벤트
comment	nodeType • parentNode • childNodes • previousSibling • nextSibling • data	hasChildNodes()	없음
NodeList	length •	item()	없음
NamedNodeMap	length •	item()	없음

document 객체의 속성 및 메서드

다음 표에서는 Dreamweaver에서 지원되는 document 객체의 속성과 메서드를 자세히 설명합니다. 불릿(•)은 읽기 전용 속성을 표시합니다.

속성 또는 메서드	반환값
nodeType •	Node.DOCUMENT_NODE
parentNode •	null
parentWindow •	문서의 부모 윈도우에 해당하는 JavaScript 객체. 이 속성은 Microsoft Internet Explorer 4.0 DOM에 정의되어 있지만 DOM 레벨 1이나 2의 일부는 아닙니다.
childNodes •	document 객체의 모든 직계 자식이 포함된 NodeList. 일반적으로 문서에는 HTML 객체라는 단일 자식이 있습니다.
previousSibling •	null
nextSibling •	null
documentElement •	HTML 태그에 해당하는 JavaScript 객체. 이 속성은 document.childNodes의 값을 가져오고 NodeList에서 HTML 태그를 추출하는 대표 속성입니다.
body •	BODY 태그에 해당하는 JavaScript 객체. 이 속성은 document.documentElement.childNodes를 호출하고 NodeList에서 body 태그를 추출하는 대표 속성입니다. 프레임세트 문서의 경우 이 속성은 가장 바깥쪽에 있는 프레임세트의 노드를 반환합니다.
URL •	문서에 대한 *file://*URL을 반환하거나, 파일이 저장되지 않은 경우에는 빈 문자열을 반환합니다.
getElementsByTagName(tagName)	*tagName* 유형의 태그(예: img, div 등)를 단계별로 실행하는 데 사용할 수 있는 NodeList. *tagName* 인수가 "LAYER"이면 이 함수는 모든 LAYER 및 ILAYER 태그와 모든 절대 위치 요소를 반환합니다. *tagName* 인수가 "INPUT"이면 이 함수는 모든 양식 요소를 반환합니다. 하나 이상의 *tagName* 객체에 하나의 name 속성이 지정된 경우에는 이름이 HTML 4.01 사양에서 요구하는 문자로 시작해야 합니다. 그렇지 않으면 이 함수가 반환하는 배열의 길이가 올바르지 않게 됩니다.

속성 또는 메서드	반환값
getElementsById(Id)	지정된 *id*를 가진 요소 노드를 가져옵니다. 여기서 *id*는 가져올 요소의 ID가 포함된 문자열입니다. var dom = dw.getDocumentDOM(); var contObj = dom.getElementById('content'); alert("The element with the id 'content' is a " + contObj.tagName);
getElementsByAttributeName(attrName)	*attrName* 속성이 있는 요소(예: 속성 "for"가 있는 모든 요소)를 단계별로 실행하는 데 사용할 수 있는 NodeList. DOM 레벨 1이나 2의 일부가 아닙니다.
getElementById(id)	지정된 ID를 가진 HTML 요소
hasChildNodes()	true

HTML의 속성 및 메서드

다음 표에서는 Dreamweaver에서 사용하는 HTML 요소의 속성 및 메서드와 각 반환값 또는 설명을 보여 줍니다. 불릿(•)은 읽기 전용 속성을 표시합니다.

속성 또는 메서드	반환값
nodeType •	Node.ELEMENT_NODE
parentNode •	부모 태그. 이 태그가 HTML 태그이면 document 객체가 반환됩니다.
childNodes •	태그의 모든 직계 자식이 포함된 NodeList
previousSibling •	이 값 바로 전의 형제 노드. 예를 들어 HTML 문서의 경우 body 요소의 previousSibling은 head 요소입니다.
nextSibling •	이 값 바로 다음의 형제 노드. 예를 들어 HTML 문서의 경우 head 요소의 nextSibling은 body 요소입니다. 헤드의 모든 script, style 또는 meta 태그는 head 요소의 자식 노드입니다.
tagName •	요소에 대한 HTML tagName(예: IMG, A, DIV). 이 값은 항상 대문자로 반환됩니다.
attrName	지정된 태그 속성의 값이 포함된 문자열입니다. tag.*attrName*은 *attrName* 속성이 JavaScript 언어의 예약어(예: class)이면 사용할 수 없습니다. 이러한 경우 getAttribute() 및 setAttribute()를 사용합니다.
innerHTML	열기 태그와 닫기 태그 사이에 포함된 소스 코드. 예를 들어 Hello, World! 코드에서 p.innerHTML은 Hello, World!를 반환합니다. 이 속성을 쓰면 문서의 새 구조에 맞게 DOM 트리가 즉시 업데이트됩니다. 이 속성은 Microsoft Internet Explorer 4.0 DOM에 정의되어 있지만 DOM 레벨 1이나 2의 일부는 아닙니다.
outerHTML	이 태그의 소스 코드(태그 포함). 앞에 나온 예제 코드의 경우 p.outerHTML은 Hello, World! 를 반환합니다. 이 속성을 쓰면 문서의 새 구조에 맞게 DOM 트리가 즉시 업데이트됩니다. 이 속성은 Microsoft Internet Explorer 4.0 DOM에 정의되어 있지만 DOM 레벨 1이나 2의 일부는 아닙니다.
getAttribute(*attrName*)	속성이 명시적으로 지정되었으면 지정된 속성의 값을 반환하고, 그렇지 않으면, null을 반환합니다.
getTranslatedAttribute(*attrName*)	지정된 속성의 변환된 값을 반환하거나, 속성 값이 변환되지 않은 경우 getAttribute()가 반환하는 것과 동일한 값을 반환합니다. 이 속성은 DOM 레벨 1에 포함되지 않지만 속성 변환을 지원하기 위해 Dreamweaver 3에 추가되었습니다.

속성 또는 메서드	반환값
setAttribute(*attrName*, *attrValue*)	값을 반환하지는 않고, 지정된 속성을 지정된 값에 설정합니다(예: img.setAttribute("src", "image/roses.gif").
removeAttribute(*attrName*)	값을 반환하지는 않고, 이 태그에 대한 HTML에서 지정된 속성과 해당 값을 제거합니다.
getElementsByTagName(*tagName*)	*tagName* 유형의 자식 태그(예: IMG, DIV)를 단계별로 실행하는 데 사용할 수 있는 NodeList. *tagName* 인수가 "layer"이면 이 함수는 모든 LAYER 및 ILAYER 태그와 모든 절대 위치 요소를 반환합니다. *tagName* 인수가 "input"이면 함수는 모든 양식 요소를 반환합니다. 하나 이상의 tagName 객체에 하나의 name 속성이 지정된 경우에는 이름이 HTML 4.01 사양에서 요구하는 문자로 시작해야 합니다. 그렇지 않으면 이 함수가 반환하는 배열의 길이가 올바르지 않게 됩니다.
getElementsByAttributeName(*attrName*)	*attrName* 속성이 있는 요소(예: 속성 "for"가 있는 모든 요소)를 단계별로 실행하는 데 사용할 수 있는 NodeList. DOM 레벨 1이나 2의 일부가 아닙니다.
hasChildNodes()	태그에 자식이 있는지 여부를 나타내는 부울 값
hasTranslatedAttributes()	태그에 변환된 속성이 있는지 여부를 나타내는 부울 값. 이 속성은 DOM 레벨 1에 포함되지 않지만 속성 변환을 지원하기 위해 Dreamweaver 3에 추가되었습니다.

text 객체의 속성 및 메서드

HTML 문서에서 인접한 각 텍스트 블록(예: p 태그 내의 텍스트)은 JavaScript 객체로 표현됩니다. text 객체에는 자식이 없습니다. 다음 표에서는 Dreamweaver에서 사용하는 DOM 레벨 1의 text 객체에 대한 속성 및 메서드를 자세히 설명합니다. 불릿(•)은 읽기 전용 속성을 표시합니다.

속성 또는 메서드	반환값
nodeType •	Node.TEXT_NODE
parentNode •	부모 태그
child Nodes •	비어 있음
previousSibling •	이 값 바로 전의 형제 노드. 예를 들어 blah blah 코드에서 태그에는 세 개의 자식 노드(텍스트 노드, 요소 노드, 텍스트 노드)가 있습니다. 세 번째 자식에서 previousSibling은 태그이고, 첫 번째 자식에서 previousSibling은 null입니다.
nextSibling •	이 값 바로 다음의 형제 노드. 예를 들어 blah blah 코드에서 p 태그에 있는 첫 번째 자식의 nextSibling은 태그이고 세 번째 자식의 nextSibling은 null입니다.
data	실제 텍스트 문자열. 텍스트의 엔터티는 단일 문자로 표시됩니다. 예를 들어 텍스트 Joseph & I는 Joseph & I로 반환됩니다.
hasChildNodes()	false

comment 객체의 속성 및 메서드

JavaScript 객체는 각 HTML 주석을 나타냅니다. 다음 표에서는 Dreamweaver에서 사용하는 DOM 레벨 1의 comment 객체에 대한 속성 및 메서드를 자세히 설명합니다. 불릿(•)은 읽기 전용 속성을 표시합니다.

속성 또는 메서드	반환값
nodeType •	Node.COMMENT_NODE
parentNode •	부모 태그
childNodes •	빈 NodeList 배열
previousSibling •	이 값 바로 전의 형제 노드.
nextSibling •	이 값 바로 다음의 형제 노드.
data	주석 표시 기호(<!-- 및 -->)
hasChildNodes()	false

dreamweaver 및 site 객체

Dreamweaver에서는 DOM을 통해 액세스할 수 있는 표준 객체를 구현하고 두 개의 사용자 정의 객체 dreamweaver 및 site를 추가합니다. 이러한 사용자 정의 객체는 API 내에서 또는 Extension을 작성할 때 광범위하게 사용됩니다. dreamweaver 및 site 객체의 메서드에 대한 자세한 내용은 **Dreamweaver API 참조** 설명서를 참조하십시오.

dreamweaver 객체의 속성

dreamweaver 객체에는 다음과 같은 두 개의 읽기 전용 속성이 있습니다.

- appName 속성은 "Dreamweaver" 값을 가집니다.
- appVersion 속성은 "***versionNumber.releaseNumber.buildNumber***[***languageCode***](***platform***)" 형식의 값을 가집니다.

  예를 들어 Dreamweaver 의 스웨덴어 Windows 버전에 대한 appVersion 속성의 값은 "8.0.XXXX [se] (Win32)"이고, 영어 Macintosh 버전에 대한 값은 "8.0.XXXX [en] (MacPPC)"입니다.

  참고: [도움말] > [정보] 메뉴 항목을 선택하면 버전 및 빌드 번호를 확인할 수 있습니다.

appName 및 appVersion 속성은 Dreamweaver 3에서 구현되었으며 그 이전 버전의 Dreamweaver에서는 사용할 수 없습니다.

Dreamweaver의 버전을 확인하려면 다음 예제와 같이 appVersion이 있는지 확인한 다음 해당 버전 번호를 확인합니다.

```
if (dreamweaver.appVersion && dreamweaver.appVersion.indexOf('3.01') != -1){¬
    // execute code
}
```

dreamweaver 객체에는 사용자 운영 체제의 언어를 쿼리하는 데 사용할 수 있는 systemScript라는 속성이 있습니다. 지역화된 운영 체제를 고려하여 Extension 코드에 특별한 경우를 포함해야 하는 경우 다음 예제에서처럼 이 속성을 사용합니다.

```
if (dreamweaver,systemScript && (dreamweaver.systemScript.indexOf('ja')!=-1){¬
    SpecialCase }
```

systemScript 속성은 지역화된 각 운영 체제에 대해 다음 값을 반환합니다.

언어	값
일본어	ja
한국어	ko
중국어 번체	zh_tw
중국어 간체	zh_cn

모든 유럽 언어의 운영 체제는 'en'을 반환합니다.

site 객체

site 객체에는 속성이 없습니다. site 객체의 메서드에 대한 자세한 내용은 **Dreamweaver API** 참조 설명서를 참조하십시오.

7장: 삽입 막대 객체

삽입 막대에 항목을 추가하여 사용자를 위해 반복 작업을 자동화하거나 사용자가 특정 속성을 설정할 수 있는 대화 상자를 만들 수도 있습니다.

객체는 Dreamweaver 응용 프로그램 폴더 내의 Configuration/Objects 폴더에 있습니다. Objects 하위 폴더는 삽입 막대에서의 위치에 따라 그룹화되며 이러한 파일을 열어 현재 객체의 구성을 볼 수 있습니다. 예를 들어 Configuration/Objects/Common/Hyperlink.htm 파일을 열어 삽입 막대의 하이퍼텍스트 링크 객체 버튼에 해당하는 코드를 볼 수 있습니다.

다음 표에서는 객체를 만들 때 사용하는 파일을 보여 줍니다.

경로	파일	설명
Configuration/Objects/**objecttype**/	**objectname**.htm	문서에 삽입할 항목을 지정합니다.
Configuration/Objects/**objecttype**/	**objectname**.js	실행할 함수가 들어 있습니다.
Configuration/Objects/**objecttype**/	**objectname**.gif	삽입 막대에 나타나는 이미지를 포함합니다.
Configuration/Objects	insertbar.xml	삽입 막대에 나타나는 객체와 그 순서를 지정합니다.

객체 파일의 작동 방식

객체는 코드의 특정 문자열을 사용자 문서에 삽입합니다. 객체를 사용하면 사용자가 메뉴의 옵션이나 아이콘을 클릭하여 이미지, AP(절대 위치) 요소 및 표와 같은 내용을 추가할 수 있습니다.

객체의 구성 요소는 다음과 같습니다.

- 문서에 삽입되는 내용을 정의하는 HTML 파일

  객체 파일의 head 섹션은 body 섹션에서 입력된 양식을 처리하고 사용자 문서에 추가된 내용을 제어하는 JavaScript 함수를 포함하거나 외부 JavaScript 파일을 참조합니다. 객체 파일의 본문에는 객체의 매개 변수(예: 표에 삽입할 행과 열의 수)를 사용하고 사용자가 속성을 입력할 수 있는 대화 상자를 활성화하는 HTML 양식이 포함될 수 있습니다.

  참고: 가장 간단한 객체는 삽입할 HTML만 포함하며 body 및 head 태그는 포함하지 않습니다. 자세한 내용은 Adobe 지원 센터에서 "Dreamweaver 사용자 정의"를 참조하십시오.

- 삽입 막대에 나타나는 18x18픽셀 이미지
- insertbar.xml 파일에 추가되는 내용. insertbar.xml 파일은 객체가 삽입 막대에 나타나는 위치를 정의합니다.

  삽입 막대에 있는 아이콘을 클릭하거나 [삽입] 메뉴에서 항목을 선택하여 객체를 선택할 수 있습니다. 사용자가 객체를 선택하면 다음 이벤트가 발생합니다.

  1 Adobe Dreamweaver에서 canInsertObject() 함수를 호출하여 대화 상자 표시 여부를 결정합니다.

  객체 파일에서 form 태그가 검색됩니다. 특정 양식이 존재하고 [일반] 환경 설정 대화 상자에서 [객체 삽입시 대화 상자 보기] 옵션을 선택하면 windowDimensions() 함수가 정의되어 있는 경우 Dreamweaver에서 이 함수를 호출하여 이 함수는 해당 양식을 표시할 대화 상자의 크기를 결정하기 위해 호출됩니다. 객체 파일에 양식이 없으면 Dreamweaver에서는 대화 상자를 표시하지 않고 2단계를 건너뜁니다.

  2 1단계에서 Dreamweaver에 대화 상자가 표시된 경우, 사용자는 이 대화 상자의 텍스트 필드에 표의 행 및 열 수와 같은 객체 매개 변수를 입력하고 [확인]을 클릭합니다.

3 Dreamweaver에서는 objectTag() 함수를 호출하고 해당 반환값을 문서의 현재 선택 영역 뒤에 삽입합니다. 현재 선택 영역을 대체하지는 않습니다.

4 Dreamweaver에서는 objectTag() 함수를 찾지 못하면 이 함수 대신 insertObject() 함수를 찾아 호출합니다.

삽입 막대 정의 파일

Configuration/Objects/insertbar.xml 파일은 삽입 막대 속성을 정의합니다. 이 XML 파일에는 각 개별 객체에 대한 정의가 객체 표시 순서대로 포함되어 있습니다.

사용자가 처음 Dreamweaver를 시작하면 삽입 막대가 문서에서 수평으로 나타납니다. 이후에는 삽입 막대의 표시/숨김 여부 및 위치가 레지스트리에 저장됩니다.

Insertbar.xml 태그 계층 구조

다음 예제에서는 insertbar.xml 파일에 있는 중첩된 태그의 형식 및 계층 구조를 보여 줍니다.

```
<?xml version="1.0" ?>
<!DOCTYPE insertbarset SYSTEM "-//Adobe//DWExtension insertbar 10.0">
<insertbar xmlns:MMString="http://www.adobe.com/schemes/data/string/">
<category id="DW_Insertbar_Common" MMString:name="insertbar/categorycommon" folder="Common">
        <button id="DW_Hyperlink" image="Common\Hyperlink.png"
        MMString:name="insertbar/hyperlink" file="Common\Hyperlink.htm" />
        <button id="DW_Email" image="Common\E-Mail Link.png"
        MMString:name="insertbar/email" file="Common\E-Mail Link.htm" />
        <separator />
    <menubutton id="DW_Images" MMString:name="insertbar/images"
    image="Common\Image.png">
        <button id="DW_Image" image="Common\Image.png"
        MMString:name="insertbar/image" file="Common\Image.htm" />
    ...
        </menubutton>
        <separator />
            <button id="DW_TagChooser" MMString:name="insertbar/tagChooser"
            image="Common\Tag Chooser.gif" command="dw.showTagChooser()"
            codeOnly="TRUE"/>
    </category>
        ...
    </insertbar>
```

참고: insertbar 및 category 태그는 </insertbar> 및 </category> 닫기 태그를 사용하여 내용의 끝을 표시합니다. button, checkbutton 및 separator 태그는 닫기 괄호 앞에 슬래시(/)를 사용하여 해당 속성 및 내용의 끝을 표시합니다.

삽입 막대 정의 태그

insertbar.xml 파일에는 다음 태그 및 속성이 포함되어 있습니다.

<insertbar>

설명

이 태그는 삽입 막대 정의 파일의 내용이 시작됨을 알려 주며 </insertbar> 닫기 태그는 내용의 끝을 나타냅니다.

속성

없음

예제

```
<insertbar>
    <category id="DW_Insertbar_Common" folder="Common">
        <button id="DW_Hyperlink" image="Common\Hyperlink.gif"
        file="Common\Hyperlink.htm"/>0
...
</insertbar>
```

<category>

설명

이 태그는 삽입 막대에 나타나는 범주(예: 일반, 양식, HTML)를 정의하며 </category> 닫기 태그는 범주 내용의 끝을 나타냅니다.

참고: 기본적으로 삽입 막대는 사용 범주(예: 일반, 양식, HTML)로 구성됩니다. 이전 버전의 Dreamweaver에서 삽입 막대는 탭을 기반으로 유사하게 구성되었습니다. 사용자는 범주나 탭을 기반으로 삽입 막대 객체의 구성 방식에 대해 고유한 환경 설정을 만들 수 있습니다. 사용자가 탭 구성을 선택한 경우 category 태그는 각 탭을 정의합니다.

속성

id, {folder}, {showIf}

예제

```
<category id="DW_Insertbar_Common" folder="Common">
    <button id="DW_Hyperlink" image="Common\Hyperlink.gif"
    file="Common\Hyperlink.htm"/>
</category>
```

<menubutton>

설명

이 태그는 삽입 막대의 팝업 메뉴를 정의합니다.

속성

id, image, {showIf}, {name}, {folder}

예제

```
<menubutton
    id="DW_ImageMenu"
    name="Images"
    image="Common\imagemenu.gif"
    folder="Images">
    <button id="DW_Image"
        image="Common\Image.gif"
        enabled=""
        showIf=""
        file="Common\Image.htm" />
</menubutton>
```

<button />

설명

이 태그는 삽입 막대의 버튼을 정의하며 사용자는 이를 클릭하여 command 또는 file 속성이 지정하는 코드를 실행할 수 있습니다.

속성
id, image, name, {canDrag}, {showIf}, {enabled}, {command}, {file}, {tag}, {codeOnly}

예제
```
<button id="DW_Object"
image="Common\Object.gif"
name="Object"
enabled="true"
showIf=""
file="Common\Obect.htm"
    />
```

<checkbutton />

설명
checkbutton은 체크 표시되거나 체크 표시되지 않은 상태로 나타나는 버튼입니다. checkbutton은 클릭 시 눌려진 상태로 나타나며 강조 표시됩니다. 체크 표시되지 않은 checkbutton은 평평하게 나타납니다. Dreamweaver의 버튼 상태로는 Mouse-over, Pressed, Mouse-over-while-pressed 및 Disabled-while-pressed가 있습니다. 명령은 체크 버튼을 클릭할 때 버튼 상태가 변경되도록 해야 합니다.

속성
id, image, checked, {showIf}, {enabled}, {command}, {file}, {tag}, {name}, {codeOnly}

예제
```
<checkbutton id="DW_StandardView"
name = "Standard View"
image="Tools\Standard View.gif"
checked="_View_Standard"
command="dw.getDocumentDOM().setShowLayoutView(false)"/>
```

<separator />

설명
이 태그는 삽입 막대에 수직선을 표시합니다.

속성
{showIf}

예제
```
<separator showIf="_VIEW_CODE"/>
```

삽입 막대 정의 태그 속성
삽입 막대 정의 태그의 속성에는 다음과 같은 의미가 있습니다.

id="unique id"

설명
id 속성은 삽입 막대에 나타나는 버튼의 식별자입니다. 이 id 속성은 파일 내의 요소에 대해 고유해야 합니다.

예제

```
id="DW_Anchor"
```

image="image_path”

설명

이 속성은 삽입 막대에 나타나는 아이콘 파일의 경로를 Dreamweaver Configuration 폴더에 대한 상대 경로로 지정합니다. 아이콘 형식은 Dreamweaver에서 렌더링할 수 있는 어느 형식이나 가능하지만 대개는 18x18픽셀 크기의 GIF 또는 JPEG 파일 형식입니다.

예제

```
image="Common/table.gif"
```

canDrag="Boolean”

설명

이 속성은 사용자가 객체를 문서에 삽입하기 위해 해당 아이콘을 코드 또는 작업 영역으로 드래그할 수 있는지 여부를 지정합니다. 이 속성을 생략할 경우 기본값은 true입니다.

예제

```
canDrag="false"
```

showIf="enabler"

설명

이 속성은 지정된 Dreamweaver 활성자가 true 값인 경우에만 해당 버튼이 삽입 막대에 표시되도록 지정합니다. showIf를 지정하지 않으면 버튼은 항상 표시됩니다. 사용 가능한 활성자로는 _SERVERMODEL_ASP, _SERVERMODEL_ASPNET, _SERVERMODEL_JSP, _SERVERMODEL_CFML (모든 버전의 ColdFusion에 해당), _SERVERMODEL_CFML_UD4 (UltraDev 버전 4 ColdFusion에 해당), _SERVERMODEL_PHP, _FILE_TEMPLATE, _VIEW_CODE, _VIEW_DESIGN, _VIEW_LAYOUT, _VIEW_EXPANDED_TABLES, 및 _VIEW_STANDARD가 있습니다.

여러 개의 활성자를 지정하려면 각 활성자 사이에 AND를 의미하는 쉼표를 사용합니다. NOT을 지정하려면 느낌표(!)를 사용합니다.

예제

버튼이 ASP 페이지의 [코드] 뷰에만 표시되도록 하려면 활성자를 다음과 같이 지정합니다.

```
showIf="_VIEW_CODE, _SERVERMODEL_ASP"
```

enabled="enabler"

설명

이 속성은 **DW_enabler** 값이 true인 경우 사용자가 해당 항목을 사용할 수 있도록 지정합니다. enabled 함수를 지정하지 않으면 항목은 기본적으로 항상 활성화됩니다. 사용 가능한 활성자로는 _SERVERMODEL_ASP, _SERVERMODEL_ASPNET, _SERVERMODEL_JSP, _SERVERMODEL_CFML(모든 버전의 ColdFusion에 해당), _SERVERMODEL_CFML_UD4(UltraDev 버전 4 ColdFusion에만 해당), _SERVERMODEL_PHP, _FILE_TEMPLATE, _VIEW_CODE, _VIEW_DESIGN, _VIEW_LAYOUT, _VIEW_EXPANDED_TABLES, 및 _VIEW_STANDARD가 있습니다.

여러 개의 활성자를 지정하려면 각 활성자 사이에 AND를 의미하는 쉼표를 사용합니다. NOT을 지정하려면 느낌표(!)를 사용합니다.

예제

버튼을 [코드] 뷰에서만 사용할 수 있도록 하려면 다음을 지정합니다.

```
enabled="_VIEW_CODE"
```

그러면 다른 뷰에서는 해당 버튼이 희미하게 표시됩니다.

checked="enabler"

설명

checkbutton 태그를 사용하는 경우 checked 속성은 필수입니다.

항목은 **DW_enabler** 값이 true인 경우에 체크 표시됩니다. 사용 가능한 활성자로는 _SERVERMODEL_ASP, _SERVERMODEL_ASPNET, _SERVERMODEL_JSP, _SERVERMODEL_CFML(모든 버전의 ColdFusion에 해당), _SERVERMODEL_CFML_UD4(UltraDev 버전 4 ColdFusion에만 해당), _SERVERMODEL_PHP, _FILE_TEMPLATE, _VIEW_CODE, _VIEW_DESIGN, _VIEW_LAYOUT, _VIEW_EXPANDED_TABLES, 및 _VIEW_STANDARD가 있습니다.

여러 개의 활성자를 지정하려면 각 활성자 사이에 AND를 의미하는 쉼표를 사용합니다. NOT을 지정하려면 느낌표(!)를 사용합니다.

예제

```
checked="_View_Layout"
```

command="API_function"

설명

Dreamweaver에서 삽입할 코드가 포함된 HTML 파일을 참조하도록 하는 대신, 이 태그를 사용하여 버튼을 클릭할 때 Dreamweaver에서 수행될 명령을 지정합니다.

예제

```
command="dw.showTagChooser()"
```

file="file_path"

설명

file 속성은 객체 파일의 경로를 Dreamweaver Configuration 폴더에 대한 상대 경로로 지정합니다. name 속성을 지정하지 않으면 Dreamweaver에서는 객체 파일의 제목을 사용하여 객체 아이콘에 대한 도구 설명을 지정합니다.

예제

```
file="Templates/Editable.htm"
```

tag="editor"

설명

이 속성은 Dreamweaver에서 태그 편집기를 호출하도록 지정합니다. [코드] 뷰에서는 tag 속성이 정의되어 있을 때 사용자가 객체를 클릭하면 Dreamweaver에서 [태그] 대화 상자가 호출됩니다. [코드] 뷰에서 tag 및 command 속성을 지정하면 Dreamweaver에서 태그 편집기가 호출됩니다. [디자인] 뷰에서는 codeOnly="TRUE"일 때 file 속성을 지정하지 않으면 Dreamweaver MX에서 [코드 및 디자인] 뷰가 호출되고 코드에 포커스가 놓이며 태그 편집기가 호출됩니다.

예제

```
tag = "form"
```

name="tooltip_text"

설명

name 속성은 마우스 포인터가 객체 위에 있을 때 표시되는 도구 설명을 지정합니다. 객체 파일만 지정하고 name 속성은 지정하지 않으면 객체 파일의 이름이 도구 설명에 사용됩니다.

참고: name 속성을 지정하지 않으면 삽입 막대 UI의 [즐겨찾기] 범주에서 객체 그룹을 만들 수 없습니다.

일부 삽입 막대 객체에서는 접두어 MMString으로 name 속성 형식을 변형하여 사용합니다. MMString은 지역화된 문자열을 나타냅니다. 이러한 문자열 값에 대해서는 75페이지의 “Extension 지역화”에서 설명합니다.

예제

```
name = "cfoutput"
```

삽입 막대 수정

한 범주에서 다른 범주로 객체를 이동하고, 범주의 이름을 다시 지정하고, 패널에서 객체를 완전히 제거할 수 있습니다. 삽입 막대에 변경 사항을 반영하려면 Dreamweaver를 다시 시작하거나 Extension을 다시 로드해야 합니다. Extension을 다시 로드하는 방법에 대한 자세한 내용은 73페이지의 “Extension 다시 로드”를 참조하십시오.

삽입 막대의 한 범주에서 다른 범주 또는 해당 범주 내의 새 위치로 객체 이동 또는 복사

1. insertbar.xml의 백업 사본을 insertbar.backup.xml과 같은 이름으로 저장합니다.
2. 원본 insertbar.xml 파일을 엽니다.
3. 이동하거나 복사할 객체를 나타내는 button 태그를 찾습니다. 예를 들어 이미지 객체를 [일반] 범주의 해당 위치에서 이동하려면 id 속성이 "DW_Image"인 button 태그를 찾습니다.
4. 전체 button 태그를 잘라내거나 복사합니다.
5. 객체를 이동하거나 복사할 범주를 나타내는 category 태그를 찾습니다.
6. 범주에서 해당 객체를 표시할 위치를 찾습니다.
7. 복사한 button 태그를 붙여넣습니다.
8. insertbar.xml 파일을 저장합니다.
9. Extension을 다시 로드합니다.

삽입 막대에서 객체 제거

1. insertbar.xml의 백업 사본을 insertbar.backup.xml과 같은 이름으로 저장합니다.
2. 원본 insertbar.xml 파일을 엽니다.
3. 제거할 객체를 나타내는 button 태그를 찾습니다.
4. 전체 button 태그를 삭제합니다.
5. insertbar.xml 파일을 저장합니다.
6. 디스크에서 객체의 HTML, GIF 및 JavaScript 파일을 현재 폴더 외에 insertbar.xml 파일에 나타나지 않은 다른 폴더로 이동합니다. 예를 들어 Configuration/Objects 폴더에 Unused라는 이름의 새 폴더를 만들고 객체의 파일을 이 폴더로 이동할

수 있습니다. 객체를 제거하려면 이러한 파일을 완전히 삭제합니다. 그러나 객체를 나중에 다시 복원해야 할 경우도 있을 수 있으므로 파일의 백업 사본을 만들어 두는 것이 좋습니다.

7 Extension을 다시 로드합니다.

삽입 막대의 범주 순서 변경

1 insertbar.xml의 백업 사본을 insertbar.backup.xml과 같은 이름으로 저장합니다.

2 원본 insertbar.xml 파일을 엽니다.

3 이동할 범주에 해당하는 category 태그를 찾고 해당 태그와 해당 태그에 포함된 모든 태그를 선택합니다.

4 해당 태그를 잘라냅니다.

5 태그를 새 위치에 붙여넣습니다. 다른 category 태그의 내부가 아닌 위치에 태그를 붙여넣어야 합니다.

6 insertbar.xml 파일을 저장합니다.

7 Extension을 다시 로드합니다.

새 범주 만들기

1 insertbar.xml의 백업 사본을 "insertbar.backup.xml"과 같은 이름으로 저장합니다.

2 원본 insertbar.xml 파일을 엽니다.

3 새 category 태그를 만들고 해당 범주와 해당 범주에 표시할 일련의 객체를 저장할 기본 폴더를 지정합니다.

4 insertbar.xml의 태그 구문에 대한 자세한 내용은 100페이지의 “삽입 막대 정의 태그”를 참조하십시오.

5 insertbar.xml 파일을 저장합니다.

6 Extension을 다시 로드합니다.

삽입 막대에 새 객체 추가

삽입 막대에 객체를 추가할 수 있습니다. 삽입 막대에 변경 사항을 반영하려면 Dreamweaver를 다시 시작하거나 Extension을 다시 로드해야 합니다. Extension을 다시 로드하는 방법에 대한 자세한 내용은 73페이지의 “Extension 다시 로드”를 참조하십시오.

1 HTML 또는 JavaScript(선택 사항)를 사용하여 사용자의 문서에 사용할 코드의 특정 문자열을 정의합니다.

2 Dreamweaver 인터페이스의 버튼에 사용할 이미지(18x18픽셀)를 지정하거나 만듭니다.

더 큰 이미지를 만든 경우에는 이미지 크기가 자동으로 18x18픽셀로 조절됩니다. 객체의 이미지를 만들지 않으면 물음표(?)가 있는 기본 객체 아이콘이 삽입 막대에 표시됩니다.

3 Configuration/Objects 폴더에 새 파일을 추가합니다.

4 insertbar.xml 파일을 편집하여 새 파일의 위치를 지정하고 버튼의 모양에 대한 속성을 설정합니다. 100페이지의 “삽입 막대 정의 파일”을 참조하십시오.

5 Dreamweaver를 다시 시작하거나 Extension을 다시 로드합니다.

새 객체가 삽입 막대의 지정된 위치에 나타납니다.

참고: 객체 파일을 별도의 폴더에 저장할 수도 있지만 이때 각 파일 이름은 고유해야 합니다. 예를 들어 dom.insertObject() 함수는 Objects 폴더의 하위 폴더를 포함한 모든 위치에서 파일을 찾습니다. dom.insertObject() 함수에 대한 자세한 내용은 Dreamweaver API 참조 설명서를 참조하십시오. Forms 폴더에 Button.htm이라는 파일이 있고 MyObjects 폴더에도 Button.htm이라는 다른 객체 파일이 있으면 Dreamweaver에서는 이 두 파일을 구별할 수 없습니다. Button.htm의 두 인스턴스가 있으면 dom.insertObject()는 Button이라는 두 객체를 표시하며 사용자는 두 객체의 차이점을 알지 못할 수 있습니다.

삽입 메뉴에 객체 추가

[삽입] 메뉴 또는 다른 메뉴에서 표시되는 객체의 위치를 추가하거나 제어하려면 menus.xml 파일을 수정합니다. 이 파일은 Dreamweaver의 전체 메뉴 구조를 제어합니다. menus.xml 파일을 수정하는 방법에 대한 자세한 내용은 133페이지의 "메뉴 및 메뉴 명령"을 참조하십시오.

다른 Dreamweaver 사용자에게 Extension을 배포하려는 경우 Extension 패키지화에 대한 자세한 내용은 76페이지의 "Extension Manager 작업"을 참조하십시오.

간단한 객체 삽입 예제

이 예제에서는 사용자가 버튼을 클릭하면 선택한 텍스트를 관통하는 선(취소선)을 추가할 수 있도록 하는 객체를 삽입 막대에 추가합니다. 이 객체는 문서에 주석을 추가하려는 경우에 유용합니다.

이 예제는 텍스트 조작 작업을 수행하므로 삽입 막대의 [HTML] 범주에 있는 [텍스트] 팝업 메뉴에서 일부 객체를 검색하는 작업을 모델로 참고할 수 있습니다. 예를 들어 Bold, Emphasis 및 Heading 객체 파일에서 유사한 기능을 찾아보십시오. Dreamweaver에서는 이 파일 내의 선택된 텍스트를 태그로 묶습니다.

취소선 삽입 객체를 만들려면 다음 단계를 수행합니다.

HTML 파일 만들기

객체의 제목은 열기 및 닫기 title 태그 사이에서 지정합니다. 또한 스크립팅 언어를 JavaScript로 지정합니다.

1 비어 있는 새 파일을 만듭니다.

2 다음 코드를 추가합니다.

```
<html>
<head>
<title>Strikethrough</title>
<script language="javascript">
</script>
</head>
<body>
</body>
</html>
```

3 파일을 Configuration/Objects/Text 폴더에 Strikethrough.htm이라는 이름으로 저장합니다.

JavaScript 함수 추가

이 예제에서 JavaScript 함수는 비헤이비어를 정의하고 취소선 객체의 코드를 삽입합니다. 모든 API 함수는 파일의 head 섹션에 배치해야 합니다. Configuration/Objects/Text/Em.htm과 같은 기존 객체 파일은 서로 비슷한 함수 및 주석 패턴을 따릅니다.

객체 정의 파일에서 사용하는 첫 번째 함수는 isDOMRequired()이며, 이 함수는 실행을 계속하기 전에 [디자인] 뷰를 기존 [코드] 뷰와 동기화해야 하는지 여부를 알려 줍니다. 그러나 위첨자 객체는 [코드] 뷰의 다른 여러 객체와 함께 사용될 수 있으므로 동기화를 수행할 필요가 없습니다.

isDOMRequired() 함수 추가

1 Strikethrough.htm 파일의 head 섹션에서 열기 및 닫기 script 태그 사이에 다음 함수를 추가합니다.

```
<script language="javascript">
    function isDOMRequired() {
        // Return false, indicating that this object is available in Code view.
        return false;
}
</script>
```

2 파일을 저장합니다.

이 단계를 수행한 후에는 다음 함수에 objectTag() 또는 insertObject() 함수를 사용할지 여부를 결정합니다. 취소선 객체는 단순히 선택된 텍스트를 s 태그로 묶으므로 insertObject() 함수를 사용하기 위한 기준을 만족시키지 않습니다. 자세한 내용은 115페이지의 “insertObject()” 함수를 참조하십시오.

objectTag() 함수 내에서 dw.getFocus()를 사용하여 [코드] 뷰가 현재 뷰인지 확인합니다. [코드] 뷰에 입력 포커스가 있으면 이 함수는 선택된 텍스트를 적절한(대문자 또는 소문자) 태그로 묶습니다. [디자인] 뷰에 입력 포커스가 있으면 이 함수는 dom.applyCharacterMarkup()을 사용하여 선택된 텍스트에 서식을 지정할 수 있습니다. 이 함수는 지원되는 태그에 대해서만 작동합니다. 자세한 내용은 **Dreamweaver API 참조 설명서**에서 dom.applyCharacterMarkup()을 참조하십시오. 다른 태그 또는 작업에는 다른 API 함수를 사용해야 합니다. Dreamweaver에서는 서식을 적용한 후 메시지나 대화 상자를 표시하지 않고 삽입점(커서)을 문서에 둡니다. 다음 절차에서는 objectTag() 함수를 추가하는 방법을 보여 줍니다.

objectTag() 함수 추가

1 Strikethrough.htm 파일의 head 섹션에서 isDOMRequired() 함수 뒤에 다음 함수를 추가합니다.

```
function objectTag() {
    // Determine if the user is in Code view.
    var dom = dw.getDocumentDOM();
    if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){
        var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper Case", "") ==
        'TRUE');
        // Manually wrap tags around selection.
        if (upCaseTag){
            dom.source.wrapSelection('<S>','</S>');
        }else{
            dom.source.wrapSelection('<s>','</s>');
        }
    // If the user is not in Code view, apply the formatting in Design view.
    }else if (dw.getFocus() == 'document'){
        dom.applyCharacterMarkup("s");
    }
    // Just return--don't do anything else.
    return;
}
```

2 파일을 Configuration/Objects/Text 폴더에 Strikethrough.htm이라는 이름으로 저장합니다.

HTML 파일의 head 섹션에 JavaScript 함수를 포함하는 대신 별도의 JavaScript 파일을 만들 수 있습니다. 이러한 별도의 구조는 객체에 여러 함수 또는 다른 객체에서 공유할 수 있는 함수가 포함된 경우에 매우 유용합니다.

HTML 객체 정의 파일과 지원 JavaScript 함수 분리

1 비어 있는 새 파일을 만듭니다.

2 모든 JavaScript 함수를 파일에 붙여넣습니다.

3 Strikethrough.htm에서 함수를 제거하고 다음 예제와 같이 해당 스크립트 태그의 src 속성에 JavaScript 파일 이름을 추가합니다.

```
<html>
<head>
<title>Strikethrough</title>
<script language="javascript" src="Strikethrough.js">
</script>
</head>
<body>
</body>
</html>
```

4 Strikethrough.htm 파일을 저장합니다.

5 JavaScript 함수가 들어 있는 파일을 Configuration/Objects/Text 폴더에 Strikethrough.js라는 이름으로 저장합니다.

삽입 막대의 이미지 만들기

1 다음 그림과 같이 GIF 이미지(18x18픽셀)를 만듭니다.

2 파일을 Configuration/Objects/Text 폴더에 Strikethrough.gif라는 이름으로 저장합니다.

insertbar.xml 파일 편집

그런 다음 Dreamweaver에서 이 두 항목을 삽입 막대 인터페이스와 연관시킬 수 있도록 insertbar.xml 파일을 편집해야 합니다.

참고: insertbar.xml 파일을 편집하기 전에 원본 파일을 insertbar.xml.bak로 복사하여 백업을 만들어 두는 것이 좋습니다.

insertbar.xml 파일 내의 코드는 삽입 막대에 있는 기존의 모든 객체를 식별합니다.

- XML 파일의 각 category 태그는 인터페이스에 하나의 범주를 만듭니다.
- 각 menubutton 태그는 삽입 막대에 하나의 팝업 메뉴를 만듭니다.
- XML 파일의 각 button 태그는 삽입 막대에 아이콘을 배치하여 적절한 HTML 파일이나 함수와 연결합니다.

삽입 막대에 새 객체 추가

1 insertbar.xml 파일의 시작 부분 근처에서 다음 행을 찾습니다.

```
<category id="DW_Insertbar_Common" MMString:name="insertbar/category/common" folder="Common">
```

이 행은 삽입 막대에 있는 [일반] 범주의 시작을 나타냅니다.

2 이 category 태그 뒤에서 새 행을 시작한 다음, button 태그를 삽입하고 취소선 객체의 id, image 및 file 속성을 이 태그에 지정합니다.

ID는 표준 명명 규칙을 따르는 고유한 버튼 이름(이 객체의 경우 DW_Text_Strikethrough 사용)이어야 합니다. image 및 file 속성은 다음과 같이 지원 파일의 위치를 Dreamweaver에 알려 줍니다.

```
<button id="DW_Text_Strikethrough"
image="Text\Strikethrough.gif"
file="Text\Strikethrough.htm"/>
```

3 insertbar.xml 파일을 저장합니다.

4 Extension을 다시 로드합니다. 73페이지의 “Extension 다시 로드”를 참조하십시오.

새 객체는 삽입 막대에 있는 [일반] 범주의 시작 부분에 나타납니다.

대화 상자 추가

Dreamweaver에서 지정된 코드가 삽입되기 전에 사용자가 매개 변수를 입력할 수 있도록 객체에 양식을 추가할 수 있습니다. 예를 들어 하이퍼링크 객체의 경우 사용자가 텍스트, 링크, 대상, 범주 인덱스, 제목 및 액세스 키 값을 지정해야 합니다. 이 예제에서는 이전 예제의 취소선 객체에 양식을 추가합니다. 이 양식은 텍스트 색상을 빨강으로 변경하고 취소선 태그를 추가하는 옵션을 사용자에게 제공하는 대화 상자를 엽니다.

이 예제에서는 Strikethrough.js라는 별도의 JavaScript 파일을 이미 만들었다고 가정합니다.

먼저 Strikethrough.js에서 사용자가 텍스트 색상을 변경할 경우 이 양식이 호출할 함수를 추가합니다. 이 함수는 Strikethrough 객체의 objectTag() 함수와 유사하지만 선택적 함수입니다.

함수 만들기

1 Strikethrough.js의 objectTag() 함수 뒤에 다음 코드를 입력하여 fontColorRed()라는 함수를 만듭니다.

```
function fontColorRed(){
    var dom = dw.getDocumentDOM();
        if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){
            var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper Case", "")
            == 'TRUE');
            // Manually wrap tags around selection.
            if (upCaseTag){
                 dom.source.wrapSelection('<FONT COLOR="#FF0000">','</FONT>');
            }else{
                dom.source.wrapSelection('<font color="#FF0000">','</font>');
                 }
         }else if (dw.getFocus() == 'document'){
                dom.applyFontMarkup("color", "#FF0000");
                }
    // Just return -- don't do anything else.
    return;
}
```

참고: dom.applyCharacterMarkup()은 글꼴 색상 변경을 지원하지 않으므로 글꼴 색상 변경에 적절한 API 함수를 찾아야 합니다. 자세한 내용은 Dreamweaver API 참조 설명서의 dom.applyFontMarkup()을 참조하십시오.

2 파일을 Strikethrough.js라는 이름으로 저장합니다.

그런 다음 Strikethrough.htm 파일에 이 양식을 추가합니다. 이 예제의 양식은 사용자가 클릭할 때 fontColorRed() 함수를 호출하는 간단한 체크 상자입니다. form 태그를 사용하여 양식을 정의하고 레이아웃 제어를 위한 table 태그를 정의합니다. 그렇지 않으면 대화 상자의 단어 줄바꿈이나 크기가 부자연스럽게 될 수도 있습니다.

양식 추가

1 body 태그 뒤에 다음 코드를 추가합니다.

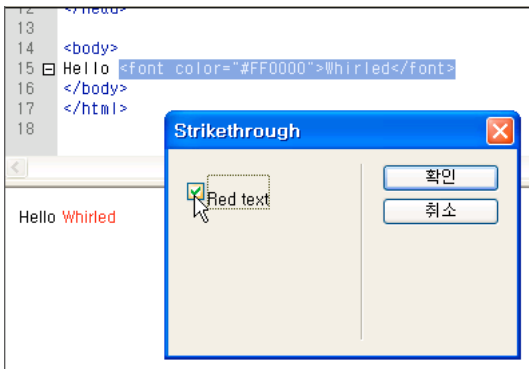
```
<form>
<table border="0" height="100" width="100">
    <tr valign="baseline">
    <td align="left" nowrap>
    <input type="checkbox" name="red" onClick=fontColorRed()>Red text</input>
    </td>
    </tr>
</table>
</form>
```

2 파일을 Strikethrough.htm이라는 이름으로 저장합니다.

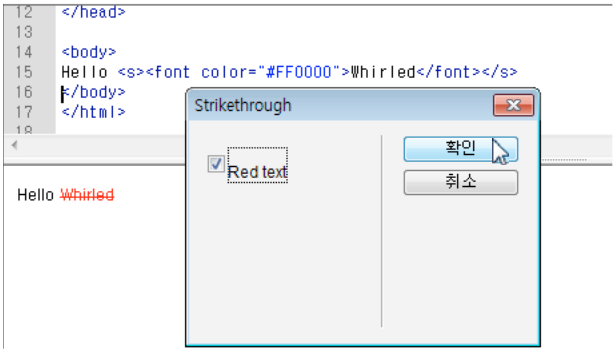
3 Extension을 다시 로드합니다. 73페이지의 “Extension 다시 로드”를 참조하십시오.

대화 상자 테스트

1 [Red Text] 체크 상자를 선택합니다.

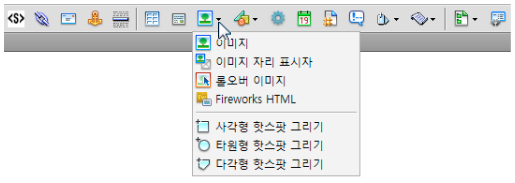


2 [확인] 버튼을 클릭하면 다음과 같이 objectTag() 함수가 실행되어 취소선이 추가됩니다.



삽입 막대 팝업 메뉴 만들기

Dreamweaver의 삽입 막대에는 객체에 대한 새로운 구조가 도입되었습니다. 이제 삽입 막대에서는 다음 그림에서와 같이 객체를 소규모의 그룹으로 구성하도록 도와주는 팝업 메뉴를 지원합니다.



다음 예제에서는 삽입 막대에 Editorial이라는 새 범주를 만들고 해당 범주에 팝업 메뉴를 추가합니다. 이 팝업 메뉴는 이미 만든 취소선 객체를 포함하며 작성하는 Blue Text 객체와 함께 취소선 객체의 그룹을 만듭니다. [Editorial] 범주에 속한 객체를 사용하면 다음을 수행할 수 있습니다.

- 파일에 주석 만들기
- 제거할 내용에 취소선을 긋거나 새로운 내용은 파랑으로 색상 지정

파일 구성

1 Dreamweaver 설치 폴더에 새 Configuration/Objects/Editorial 폴더를 만듭니다.

2 앞에서 만든 취소선 객체 예제의 파일(Strikethrough.htm, Strikethrough.js 및 Strikethrough.gif)을 Editorial 폴더에 복사합니다.

Blue Text 객체 만들기

1 HTML 파일을 만듭니다.

2 다음 코드를 추가합니다.

```
<html>
<head>
<title>Blue Text</title>
<script language="javascript">
//--------------- API FUNCTIONS---------------
function isDOMRequired() {
    // Return false, indicating that this object is available in Code view.
    return false;
}
function objectTag() {
    // Manually wrap tags around selection.
    var dom = dw.getDocumentDOM();
    if (dw.getFocus() == 'textView' || dw.getFocus(true) == 'html'){
        var upCaseTag = (dw.getPreferenceString("Source Format", "Tags Upper Case", "") ==
        'TRUE');
         // Manually wrap tags around selection.
         if (upCaseTag){
              dom.source.wrapSelection('<FONT COLOR="#0000FF">','</FONT>');
        }else{
              dom.source.wrapSelection('<font color="#0000FF">','</font>');
              }
     }else if (dw.getFocus() == 'document'){
            dom.applyFontMarkup("color", "#0000FF");
            }
    // Just return -- don't do anything else.
    return;
}
</script>
</head>
<body>
</body>
</html>
```

3 이 파일을 Editorial 폴더에 AddBlue.htm이라는 이름으로 저장합니다.

이제 Blue Text 객체에 대한 이미지를 만들 수 있습니다.

이미지 만들기

1 다음 그림과 같은 GIF 파일(18x18픽셀)을 만듭니다.

2 이 이미지를 Editorial 폴더에 AddBlue.gif라는 이름으로 저장합니다.

그런 다음 insertbar.xml 파일을 편집합니다. 이 파일은 삽입 막대의 객체와 객체 위치를 정의합니다. category 태그 내에는 다양한 menubutton 태그와 해당 속성이 있습니다. 이러한 menubutton 태그는 HTML 범주에 속하는 각 팝업 메뉴를 정의합니다. menubutton 태그 내에서 각 button 태그는 팝업 메뉴의 항목을 정의합니다.

insertbar.xml 편집

1 파일의 시작 부분에서 다음 코드 행을 찾습니다.

```
<insertbar xmlns:MMString="http://www.adobe.com/schemes/data/string/">
```

insertbar 태그는 모든 삽입 막대 내용의 시작을 정의합니다.

2 그 줄 뒤에, 만들 Editorial 범주에 대한 새 category 태그를 추가합니다. 다음 예제와 같이 고유 ID, 이름 및 폴더 속성을 지정한 다음 닫기 category 태그를 추가합니다.

```
<category id="DW_Insertbar_Editorial" name="Editorial" folder="Editorial">
</category>
```

3 Extension을 다시 로드합니다. Extension을 다시 로드하는 방법에 대한 자세한 내용은 73페이지의 “Extension 다시 로드”를 참조하십시오.

[Editorial] 범주가 삽입 막대에 나타납니다.

4 열기 및 닫기 category 태그 사이에 menubutton 태그와 고유 ID를 비롯한 다음 속성을 사용하여 팝업 메뉴를 추가합니다.

```
<menubutton id="DW_Insertbar_Markup" name="markup" image="Editorial\Strikethrough.gif"
folder="Editorial">
```

속성에 대한 자세한 내용은 102페이지의 “삽입 막대 정의 태그 속성”을 참조하십시오.

5 다음과 같이 button 태그를 사용하여 새 팝업 메뉴의 객체를 추가합니다.

```
<button id="DW_Editorial_Strikethrough" image="Editorial\Strikethrough.gif"
file="Editorial\Strikethrough.htm"/>
```

6 취소선 객체의 button 태그 뒤에 다음과 같이 하이퍼텍스트 객체를 추가합니다.

```
<button id="DW_Blue_Text" image="Editorial\AddBlue.gif name="Blue Text" file="Editorial\AddBlue.htm"/>
```

참고: button 태그는 별도의 닫기 태그가 없으므로 "/>"로 끝납니다.

7 </menubutton> 닫기 태그를 사용하여 팝업 메뉴를 종료합니다.

다음 코드에서는 팝업 메뉴와 두 개의 객체가 포함된 전체 범주를 보여 줍니다.

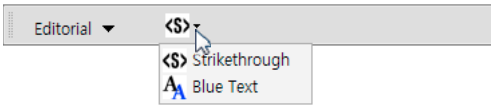
```
<category id="DW_Insertbar_Editorial" name="Editorial" folder="Editorial">
    <menubutton id="DW_Insertbar_Markup" name="markup"
    image="Editorial\Strikethrough.gif" folder="Editorial">
        <button id="DW_Editorial_Strikethrough"
        image="Editorial\Strikethrough.gif" file="Editorial\Strikethrough.htm"/>
        <button id="DW_Blue_Text" image="Editorial\AddBlue.gif" name="Blue Text"
        file="Editorial\AddBlue.htm"/>
    </menubutton>
</category>
```

새 팝업 메뉴 테스트

1 Extension을 다시 로드합니다. Extension을 다시 로드하는 방법에 대한 자세한 내용은 73페이지의 “Extension 다시 로드”를 참조하십시오.

2 [Editorial] 메뉴를 클릭합니다.

다음 팝업 메뉴가 나타납니다.



객체 API 함수

이 단원에서는 객체 API의 함수에 대해 설명합니다. insertObject() 또는 objectTag() 함수 중 하나를 정의해야 합니다. 이러한 함수에 대한 자세한 내용은 115페이지의 “insertObject()”를 참조하십시오. 나머지 함수는 선택 사항입니다.

canInsertObject()

지원 버전
Dreamweaver MX

설명
이 함수는 [객체] 대화 상자를 표시할지 여부를 결정합니다.

인수
없음

반환값
부울 값을 반환합니다.

예제
다음 코드에서는 사용자가 선택한 객체를 삽입하도록 허용하기 전에 Dreamweaver에서 문서에 특정 문자열이 있는지 확인하도록 지정합니다.

```
function canInsertObject(){
    var docStr = dw.getDocumentDOM().documentElement.outerHTML;
    var patt = /hava/;
    var found = ( docStr.search(patt) != -1 );
    var insertionIsValid = true;
    if (!found){
        insertionIsValid = false;
        alert("the document must contain a 'hava' string to use this object.");}
    return insertionIsValid;}
```

displayHelp()

설명
이 함수를 정의하면 [매개 변수] 대화 상자의 [확인] 및 [취소] 버튼 아래에 [도움말] 버튼이 표시됩니다. 사용자가 [도움말] 버튼을 클릭하면 이 함수가 호출됩니다.

인수
없음

반환값
없음

예제
다음 예제에서는 myObjectHelp.htm 파일을 브라우저에서 엽니다. 이 파일은 Extension의 사용 방법을 설명합니다.

```
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/myObjectHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

isDOMRequired()

설명
이 함수는 객체를 실행하는 데 유효한 DOM이 필요한지 여부를 확인합니다. 이 함수가 true 값을 반환하거나 이 함수가 정의되어 있지 않은 경우, Dreamweaver에서는 명령에 유효한 DOM이 필요하다고 가정하고 명령을 실행하기 전에 문서의 [코드] 뷰와 [디자인] 뷰를 동기화합니다. 동기화하면 [코드] 뷰의 모든 편집 내용이 [디자인] 뷰에 업데이트됩니다.

인수
없음

반환값
명령을 실행하는 데 유효한 DOM이 필요한 경우 true 값을 반환하고, 그렇지 않으면 false를 반환합니다.

insertObject()

지원 버전
Dreamweaver MX

설명
이 함수는 objectTag() 함수가 정의되어 있지 않은 경우에 필요하며, 사용자가 [확인]을 클릭할 때 호출됩니다. 이 함수는 사용자 문서에 코드를 삽입하고 대화 상자를 닫거나, 오류 메시지를 표시하고 대화 상자를 열어 둡니다. 이 함수는 objectTag() 함수 대신 객체에 사용하는 대체 함수로 동작합니다. 이 함수는 사용자가 현재 삽입점에 텍스트를 삽입한다고 가정하지 않으며 사용자가 [확인]을 클릭할 때 데이터의 유효성을 검사할 수 있도록 합니다. 다음 중 하나에 해당하는 경우에 insertObject() 함수를 사용해야 합니다.

- 코드를 둘 이상의 위치에 삽입해야 하는 경우
- 삽입점 이외의 위치에 코드를 삽입해야 하는 경우
- 코드를 삽입하기 전에 입력 내용의 유효성을 검사해야 하는 경우

위와 같은 경우가 아니면 objectTag() 함수를 사용합니다.

인수
없음

반환값
오류 메시지가 포함된 문자열이나 빈 문자열을 반환합니다. 빈 문자열이 반환되면 사용자가 [확인]을 클릭할 때 [객체] 대화 상자가 닫힙니다. 문자열이 비어 있지 않으면 Dreamweaver에 오류 메시지가 표시되며 대화 상자는 그대로 유지됩니다.

활성자
canInsertObject()

예제
다음 예제에서는 코드를 삽입하기 전에 입력 내용의 유효성을 검사해야 하므로 insertObject() 함수를 사용합니다.

```
function insertObject() {
     var theForm = document.forms[0];
     var nameVal = theForm.firstField.value;
     var passwordVal = theForm.secondField.value;
     var errMsg = "",
     var isValid = true;
     // ensure that field values are complete and valid
     if (nameVal == "" || passwordVal == "") {
         errMsg = "Complete all values or click Cancel."
     } else if (nameVal.length < 4 || passwordVal.length < 6) {
            errMsg = "Your name must be at least four characters, and your password at
            least six";
    }
    if (!errMsg) {
         // do some document manipulation here. Exercise left to the reader
        }
    return errMsg;
}
```

objectTag()

설명
objectTag() 함수와 insertObject() 함수는 함께 사용할 수 없습니다. 문서에 두 함수가 모두 정의되어 있으면 objectTag() 함수가 사용됩니다. 자세한 내용은 115페이지의 “insertObject()”를 참조하십시오.

이 함수는 사용자의 문서에 코드 문자열을 삽입합니다. Dreamweaver MX에서 빈 문자열 또는 null 값("return"이라고도 함)이 반환되면 Dreamweaver에서 아무 작업도 수행하지 않습니다.

참고: 일반적으로 return 문 앞에서 수동 편집 작업이 수행되었다고 가정하므로 이 경우에 아무 작업도 수행하지 않는 것은 [취소]를 클릭하는 것과는 같지 않습니다.

Dreamweaver 에서 포커스가 [코드] 뷰에 있고 선택 영역이 삽입점이 아니라 특정 범위이면 이 범위는 objectTag() 함수가 반환하는 문자열로 대체됩니다. objectTag() 함수가 빈 문자열을 반환하거나 아무 것도 반환하지 않은 경우에도 해당 값은 true입니다. objectTag() 함수는 이미 수동으로 문서를 편집했으므로 빈 문자열이나 null 값을 반환합니다. 그렇지 않으면, 이중 인용 부호("")로 선택 영역이 대체되어 편집된 내용이 삭제되는 경우가 있습니다.

인수
없음

반환값
사용자의 문서에 삽입할 문자열을 반환합니다.

예제
objectTag() 함수의 다음 예제에서는 특정 ActiveX 컨트롤과 플러그인의 OBJECT/EMBED 조합을 삽입합니다.

```
function objectTag() {
return '\n' +
'<OBJECT CLASSID="clsid:166F1OOB-3A9R-11FB-8075444553540000" \n'¬
+ 'CODEBASE="http://www.mysite.com/product/cabs/¬
myproduct.cab#version=1,0,0,0" \n' + 'NAME="MyProductName"> \n' ¬
+ '<PARAM NAME="SRC" VALUE=""> \n' + '<EMBED SRC="" HEIGHT="" ¬
WIDTH="" NAME="MyProductName"> \n' + '</OBJECT>'
}
```

windowDimensions()

설명

이 함수는 [옵션] 대화 상자의 구체적 크기를 설정합니다. 이 함수를 정의하지 않으면 윈도우 크기가 자동으로 계산됩니다.

참고: [옵션] 대화 상자의 크기를 640x480픽셀보다 크게 하려는 경우에만 이 함수를 정의하십시오.

인수

platform

- **platform** 인수의 값은 사용자의 플랫폼에 따라 "macintosh" 또는 "windows"입니다.

반환값

"widthInPixels,heightInPixels" 형식의 문자열을 반환합니다.

반환된 크기는 [확인] 및 [취소] 버튼의 영역을 포함하지 않으므로 전체 대화 상자의 크기보다 작습니다. 반환된 크기에 옵션이 모두 들어가지 않으면 스크롤 막대가 나타납니다.

예제

windowDimensions() 함수의 다음 예제에서는 [매개 변수] 대화 상자의 크기를 648x520픽셀(Windows) 또는 660x580픽셀(Macintosh)로 설정합니다.

```
function windowDimensions(platform){
    var retval = ""
    if (platform = = "windows"){
    retval = "648, 520";
    }else{
    retval = "660, 580";
    }
    return retval;
}
```

8장: 브라우저 호환성 확인 문제 API

Adobe Dreamweaver의 BCC(브라우저 호환성 확인) 기능을 사용하면 브라우저 렌더링 버그를 발생시키는 HTML과 CSS의 조합을 찾아 여러 브라우저에서 제대로 작동하는, 즉 모양과 기능이 모두 동일한 페이지 레이아웃을 만들 수 있습니다. 이 기능은 JavaScript를 사용하여 사용자의 문서에서 문제가 있는 HTML과 CSS의 조합을 검색합니다. JavaScript 코드는 문제 검색 파일이라는 HTML 파일에 저장되며, 이러한 파일은 Configuration/BrowserProfiles/Issues/ 폴더에 저장되어 있어야 제대로 작동합니다.

검색 작동 방식

사용자가 브라우저 호환성 확인을 처음 실행할 때나 [대상 브라우저] 대화 상자에서 [확인]을 클릭할 때마다 Dreamweaver에서 다음 이벤트가 발생합니다.

1. Configuration/BrowserProfiles/ 폴더에서 선택된 브라우저에 대한 프로파일을 읽습니다.
2. Configuration/BrowserProfiles/Issues/ 폴더의 각 문제 파일에서 getIssueID() 함수를 호출하여 각 문제의 고유 ID를 가져옵니다.
3. 각 문제에 대해 getAffectedBrowserDisplayNames() 함수(정의되어 있는 경우)를 호출합니다.
4. 각 문제에 대해 getAffectedBrowserProfiles() 함수를 호출하여 해당 문제가 현재 선택된 브라우저 중 하나 이상에 영향을 주는지 여부를 확인합니다.
5. 문제가 검색된 경우 각 문제에 대해 getIssueName() 함수를 호출하여 [결과] 패널에 표시할 이름을 결정합니다.
6. 문제가 검색된 경우 각 문제에 대해 getIssueDescription 함수를 호출하여 [결과] 패널의 오른쪽 영역에 표시할 텍스트와 사용자가 [코드] 뷰에서 해당 문제 위로 마우스를 가져갈 때 도구 설명에 표시할 텍스트를 결정합니다.

위 절차의 6단계 이후에는 후속 브라우저 호환성 확인이 수행될 때마다 [BCC 설정] 대화 상자에서 선택한 각 브라우저에 대해 다음 이벤트가 발생합니다.

이벤트 순서

1. 현재 문서에 적용된 스타일을 해당 브라우저에서 읽는 방식대로 파싱합니다. 이때 스타일이 인라인으로 정의되어 있는지, 헤드에 정의되어 있는지, 외부 스타일 시트에 정의되어 있는지는 관계가 없습니다.
2. 각 문제 파일에서 해당 브라우저에 적용되는 findIssue() 함수를 호출합니다.

문제 예제

다음 예제는 Configuration/BrowserProfiles/Issues/ 폴더에 있는 ColAndColgroupCapturedByCaption.htm 및 ColAndColgroupCapturedByCaption.js 파일입니다.

ColAndColgroupCapturedByCaption.htm

```
<!DOCTYPE HTML SYSTEM "-//
//DWExtension layout-engine 5.0//dialog">
<html>
<head>
<title>Col and Colgroup Captured by Caption</title>

<script src="../../Shared/Common/Scripts/dwscripts.js"></script>
<script src="issue_utils.js"></script>
<script src="ColAndColgroupCapturedByCaption.js"></script>
<script>
//--------------- LOCALIZEABLE GLOBALS------------------
var ISSUE_NAME = "Col and Colgroup/Caption Conflict";
var ISSUE_DESC = "If the caption tag is placed directly after the opening table tag as required by the HTML
4.01 specification, any styles applied to col and colgroup tags in the same table are ignored.";

//--------------- END LOCALIZEABLE---------------
</script>
</head>

<body>
</body>
</html>
```

ColAndColgroupCapturedByCaption.js

```
function findIssue(){
  var DOM = dw.getDocumentDOM();
  var issueNodes = new Array();

  if (DOM){
    // first see if there are any caption tags in the doc.
    var captions = DOM.getElementsByTagName('caption');

    // declare a mess of variables that we'll need in the
    // for loop below.
    var currCap = null, props = null, parentTable = null;
    var colgroups = null, cols = null, allcol = null;
    var property = "", definedStyles = new Array();

    // ok, now loop through all the captions, if any.
    for (var i=0; i < captions.length; i++){
      currCap = captions[i];
      parentTable = currCap.parentNode;

      // the caption is only a problem if it's in the valid
      // spot (i.e., the first child of the table)
      if (currCap == parentTable.childNodes[0]){

        // find all colgroup and col tags that are in the
        // same table as the caption.
        colgroups = parentTable.getElementsByTagName('colgroup');
        cols = parentTable.getElementsByTagName('col');
        allcol = colgroups.concat(cols);

        for (var x=0; x < allcol.length; x++){
          // if styles are declared for any colgroup or col
          // tag in this table, we have a problem node. don't
          // bother looking further.
          props = window.getDeclaredStyle(allcol[x]);
          property = "";
          definedStyles.length = 0;
```

```
          for (property in props) {
            definedStyles.push(property);
          }
          if (definedStyles.length > 0){
            issueNodes.push(currCap);
            break;
          }
        }
      }
    }
  }
  return issueNodes;
}
function getAffectedBrowserDisplayNames(){
  return new Array("Safari 2.0");
}
function getAffectedBrowserProfiles(){
  return new Array("Safari 2.0");
}
function getIssueID(){
  return "COL_AND_COLGROUP_CAPTURED_BY_CAPTION";
}
function getIssueName(){
  return ISSUE_NAME;
}
function getIssueDescription(){
  return ISSUE_DESC;
}
function getConfidenceLevel(){
  //DETCON 4
  return issueUtils.CONFIDENCE_HIGH;
}
```

문제 API 함수

문제 API에서 getAffectedBrowserDisplayNames()만 제외하고는 모든 함수가 필수입니다. 모든 Extension API를 사용할 때와 마찬가지로 각 함수의 본문을 작성하고 적절한 값을 Dreamweaver에 반환해야 합니다. 브라우저 호환성 확인 함수에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**에서 "페이지 내용" 항목을 참조하십시오.

findIssue()

지원 버전
Dreamweaver CS3

설명
문서에서 특정 브라우저 렌더링 문제를 발생시키는 CSS와 HTML의 조합을 검색합니다.

인수
없음

반환값
문제가 있거나 문제를 나타내는 요소 노드의 배열입니다. Dreamweaver에서는 사용자가 여러 브라우저 호환성 문제 사이를 이동할 때 이러한 노드가 선택됩니다.

예제

다음 findIssue() 함수는 <button> 태그 중 float: left 또는 float: right가 적용된 태그의 배열을 반환합니다.

```
function findIssue(){
    var DOM = dw.getDocumentDOM();
    var issueNodes = new Array();
    var buttons = DOM.getElementsByTagName('button');
    var props = null;
    for (var i=0; i < buttons.length; i++){
        props = window.getDeclaredStyle(buttons[i]);
        if (props.cssFloat == "left" || props.cssFloat == "right"){
            issueNodes.push(buttons[i]);
        }
    }
    return issueNodes;
}
```

getAffectedBrowserProfiles()

지원 버전

Dreamweaver CS3

설명

Dreamweaver에 해당 문제와 관련된 브라우저 목록을 제공합니다.

인수

없음

반환값

브라우저 이름의 배열로, 각 브라우저 이름은 유효한 브라우저 프로파일의 첫 번째 행과 정확히 일치해야 합니다. 자세한 내용은 Configuration/BrowserProfiles 폴더의 TXT 파일을 참조하십시오.

예제

```
function getAffectedBrowsers(){
    return new Array("Microsoft Internet Explorer 5.0",
        "Microsoft Internet Explorer 5.5",
        "Microsoft Internet Explorer 6.0");
}
```

getAffectedBrowserDisplayNames()

지원 버전

Dreamweaver CS3

설명

Dreamweaver에 해당 문제와 관련된 브라우저의 표시 이름 목록을 제공합니다. 이 함수는 선택 사항이므로 제공되지 않는 경우에는 getAffectedBrowserProfiles()에서 제공되는 프로파일 이름이 대신 사용됩니다.

인수

없음

반환값
브라우저 이름의 배열입니다. 이 배열은 getAffectedBrowserProfiles()에서 반환되는 배열과 병렬이어야 합니다.

예제
```
function getAffectedBrowsers(){
    return new Array("IE/Win 5.0",
        "IE/Win 5.5",
        "IE/Win 6.0");

}
```

getIssueID()

지원 버전
Dreamweaver CS3

설명
Dreamweaver에 문제에 대한 고유 ID를 제공합니다.

인수
없음

반환값
고유한 문제 식별자가 포함된 문자열입니다.

예제
```
function getIssueID() {
    return "EXPANDING_BOX_PROBLEM";
}
```

getIssueName()

지원 버전
Dreamweaver CS3

설명
Dreamweaver에 문제의 이름 또는 문제에 대한 간단한 설명을 제공합니다.

인수
없음

반환값
문제의 이름 또는 문제에 대한 간단한 설명이 포함된 문자열입니다.

예제
```
function getIssueName(){
    return "The Expanding Box Problem";
}
```

getIssueDescription()

지원 버전
Dreamweaver CS3

설명
Dreamweaver에 문제에 대한 상세한 설명을 제공합니다.

인수
없음

반환값
문제의 이름 또는 문제에 대한 간단한 설명이 포함된 문자열입니다.

예제

```
function getIssueDescription(){
    return "Fixed-dimension boxes will incorrectly expand to fit their
            content instead of clipping content at the specified width
            or height.";
}
```

9장: 명령

Adobe Dreamweaver 명령을 사용하여 사용자의 현재 문서, 열려 있는 다른 문서 또는 로컬 드라이브의 모든 HTML 문서에 대해 거의 모든 종류의 편집 작업을 수행할 수 있습니다. 명령을 사용하면 HTML 태그 및 속성, 주석, 텍스트를 삽입, 제거 또는 재배열할 수 있습니다.

명령은 HTML 파일입니다. 명령 파일의 body 섹션에는 해당 명령의 옵션(예: 테이블 정렬 방식 및 기준 열)을 지정할 수 있는 HTML 양식이 포함될 수 있습니다. 명령 파일의 head 섹션에는 body 섹션에서 입력된 양식을 처리하고 사용자 문서에 대해 수행할 편집 작업을 제어하는 JavaScript 함수가 포함됩니다.

다음 표에서는 명령을 만드는 데 사용하는 파일을 보여 줍니다.

경로	파일	설명
Configuration/Commands/	**commandname**.htm	사용자 인터페이스를 지정합니다.
Configuration/Commands/	**commandname**.js	실행할 함수가 들어 있습니다.

명령의 작동 방식

사용자가 명령이 포함된 메뉴를 클릭하면 다음과 같은 이벤트가 발생합니다.

1. Dreamweaver가 canAcceptCommand() 함수를 호출하여 해당 메뉴 항목을 비활성화할지 여부를 결정합니다. canAcceptCommand() 함수가 false 값을 반환하면 해당 명령은 메뉴에서 희미하게 표시되며 프로시저가 중단됩니다. canAcceptCommand() 함수가 true 값을 반환하면 프로시저가 계속됩니다.
2. 사용자가 메뉴에서 명령을 선택합니다.
3. receiveArguments() 함수가 정의되어 있는 경우 Dreamweaver가 선택된 명령 파일에서 이 함수를 호출하여 해당 명령이 해당 메뉴 항목이나 dreamweaver.runCommand() 함수에서 전달되는 인수를 처리하도록 합니다. dreamweaver.runCommand() 함수에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**를 참조하십시오.
4. commandButtons() 함수가 정의되어 있는 경우 Dreamweaver가 이 함수를 호출하여 [옵션] 대화 상자의 오른쪽에 표시할 버튼과 사용자가 이 버튼을 클릭할 때 실행할 코드를 확인합니다.
5. Dreamweaver가 명령 파일에서 form 태그를 검색합니다. 양식이 존재하면 Dreamweaver는 windowDimensions() 함수를 호출하여 파일의 body 요소가 포함된 [옵션] 대화 상자의 크기를 조절합니다. windowDimensions() 함수가 정의되어 있지 않으면 Dreamweaver는 자동으로 이 대화 상자의 크기를 결정합니다.
6. 명령 파일의 body 태그에 onLoad 핸들러가 포함되어 있는 경우 Dreamweaver가 대화 상자의 표시 여부에 관계없이 이 핸들러를 실행합니다. 대화 상자가 나타나지 않으면 나머지 단계는 수행되지 않습니다.
7. 사용자가 이 명령에 대한 옵션을 선택합니다. 사용자가 필드를 선택하면 Dreamweaver에서는 이 필드와 관련된 이벤트 핸들러를 실행합니다.
8. 사용자가 commandButtons() 함수에 의해 정의된 버튼 중 하나를 클릭합니다.
9. Dreamweaver가 관련된 코드를 실행합니다. 대화 상자는 명령에 있는 스크립트 중 하나가 window.close() 함수를 호출할 때까지 계속 표시됩니다.

명령 메뉴에 명령 추가

Dreamweaver에서는 Configuration/Commands 폴더 내에 있는 모든 파일이 [명령] 메뉴 아래에 자동으로 추가됩니다. 특정 명령을 [명령] 메뉴에 표시하지 않으려면 해당 명령 파일의 첫 행에 다음 주석을 삽입합니다.

```
<!-- MENU-LOCATION=NONE -->
```

이 행이 존재하면 Dreamweaver에서는 해당 파일에 대한 메뉴 항목을 만들지 않으므로 개발자가 dw.runCommand()를 호출하여 명령을 실행해야 합니다.

간단한 명령 예제

이 간단한 Extension은 [명령] 메뉴에 항목을 추가하고 개발자가 문서에 있는 선택된 텍스트를 대문자나 소문자로 변환할 수 있게 해 줍니다. 이 메뉴 항목을 클릭하면 세 개의 버튼으로 구성된 인터페이스가 활성화됩니다. 이 인터페이스는 사용자의 선택 사항을 전송할 수 있도록 해 줍니다.

이 Extension을 만들려면 UI를 만들고 JavaScript 코드를 작성한 다음 해당 Extension을 테스트합니다.

이 예제에서는 Commands 폴더에 두 개의 파일, 즉 UI가 포함된 Change Case.htm과 JavaScript 코드가 포함된 Change Case.js를 만듭니다. 필요한 경우 Change Case.htm 파일만 만들고 JavaScript 코드는 head 섹션에 삽입할 수도 있습니다.

사용자 인터페이스 만들기

사용자 인터페이스는 사용자가 대문자 또는 소문자를 선택하는 데 사용할 수 있는 두 개의 옵션이 포함된 양식입니다.

1 비어 있는 새 파일을 만듭니다.

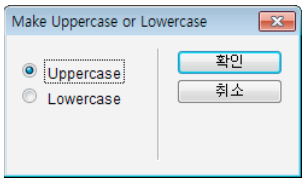
2 파일에 다음 코드를 추가하여 양식을 만듭니다.

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<HTML>
<HEAD>
<Title>Make Uppercase or Lowercase</Title>
<SCRIPT SRC="Change Selection Case.js"></SCRIPT>
</HEAD>
<BODY>
<form name="uorl">
    <table border="0">
        <!--DWLayoutTable-->
        <tr>
            <td valign="top" nowrap> <p>
                <label>
                <input type="radio" name="RadioGroup1" value="uppercase" checked>
                Uppercase</label>
                <br>
                <label>
                <input type="radio" name="RadioGroup1" value="lowercase">
                Lowercase</label>
            </p></td>
        </tr>
    </table>
</form>
</BODY>
</HTML>
```

3 이 파일을 Configuration/Commands 폴더에 Change Case.htm이라는 이름으로 저장합니다.

Title 태그의 내용인 Make Uppercase or Lowercase는 해당 대화 상자의 위쪽 막대에 나타납니다. 양식 내에서 두 개의 셀이 포함된 표는 관련 요소의 레이아웃을 제어합니다. 이 표의 셀에는 두 개의 옵션인 [대문자] 및 [소문자]가 있습니다. [대문자] 옵션은 checked 속성이 있으므로 기본 선택 사항이 됩니다. 이 속성은 사용자가 두 옵션 중 하나를 선택하거나 명령을 취소하도록 합니다.

이 양식은 다음 그림과 같습니다.



commandButtons() 함수는 사용자가 선택 사항을 전송하거나 작업을 취소하는 데 사용할 수 있는 [OK] 및 [Cancel] 버튼을 제공합니다. 자세한 내용은 130페이지의 “commandButtons()”를 참조하십시오.

JavaScript 코드 작성

다음 예제는 Dreamweaver가 호출하는 두 개의 Extension API 함수인 canAcceptCommand() 및 commandButtons()와 사용자가 정의하는 함수이며 commandButtons() 함수에서 호출되는 changeCase()로 구성되어 있습니다.

이 예제에서는 다음 작업을 수행하는 JavaScript를 작성합니다.

명령을 활성화할지 희미하게 표시할지 여부 결정

명령을 만들기 위한 첫 번째 작업은 항목이 활성화될 때와 희미하게 표시될 때를 결정하는 것입니다. 사용자가 [명령] 메뉴를 클릭하면 Dreamweaver에서는 각 메뉴 항목에 대해 canAcceptCommand() 함수를 호출하여 해당 항목을 활성화할지 여부를 결정합니다. canAcceptCommand()가 true 값을 반환하면 해당 메뉴 항목 텍스트는 활성화된 상태로 표시됩니다. canAcceptCommand()가 false 값을 반환하면 해당 메뉴 항목은 희미하게 표시됩니다. 이 예제에서는 사용자가 문서에서 텍스트를 선택할 때 해당 메뉴 항목이 활성화됩니다.

1 비어 있는 새 파일을 만듭니다.

2 다음 코드를 추가합니다.

```
function canAcceptCommand(){
    var theDOM = dw.getDocumentDOM(); // Get the DOM of the current document
    var theSel = theDOM.getSelection(); // Get start and end of selection
    var theSelNode = theDOM.getSelectedNode(); // Get the selected node
    var theChildren = theSelNode.childNodes; // Get children of selected node
    return (theSel[0] != theSel[1] && (theSelNode.nodeType == Node.TEXT_NODE¬
        || theSelNode.hasChildNodes() && (theChildren[0].nodeType == ¬
        Node.TEXT_NODE)));
}
```

3 이 파일을 Configuration/Commands 폴더에 Change Case.js라는 이름으로 저장합니다.

canAcceptCommand() 함수의 첫 번째 행은 사용자 문서에 대한 DOM을 검색하고 해당 문서 객체에 대해 getSelection() 함수를 호출하여 선택된 텍스트를 검색합니다. 그런 다음에는 이후의 코드에서처럼 선택된 텍스트가 포함된 노드와 이 노드의 자식 노드를 검색합니다. 그런 다음 마지막 행에서는 선택 항목이나 선택 항목의 첫 번째 자식이 텍스트인지 확인하고 그 결과를 true 또는 false로 반환합니다.

return 문의 첫 번째 부분(theSel[0] != theSel[1])은 사용자가 문서에서 선택한 내용이 있는지 확인합니다. theSel 변수는 두 개의 슬롯이 있는 배열로, 문서 내 선택 항목의 시작 및 끝 오프셋을 저장합니다. 두 값이 동일하지 않으면 내용이 선택된 것입니다. 두 슬롯의 값이 동일하면 삽입점만 있고 내용은 선택되지 않은 것입니다.

return 문의 다음 부분(&& (theSelNode.nodeType == Node.TEXT_NODE)은 선택된 노드 유형이 텍스트인지 확인합니다. 노드 유형이 텍스트이면 canAcceptCommand() 함수는 true 값을 반환합니다. 노드 유형이 텍스트가 아니면 해당 노드가 자식인지 여부|| theSelNode.hasChildNodes()와 첫 번째 자식 노드의 유형이 텍스트인지 여부(&&(theChildren[0].nodeType == Node.TEXT_NODE)))를 계속해서 확인합니다. 두 조건이 모두 true이면 canAcceptCommand()는 true 값을 반환하고 Dreamweaver에서는 다음 그림에서와 같이 [명령] 메뉴의 아래쪽에 있는 해당 메뉴 항목을 활성화합니다.

그렇지 않으면, canAcceptCommand()는 false 값을 반환하고 Dreamweaver에서는 다음 그림에서와 같이 해당 항목을 희미하게 표시합니다.

OK 및 Cancel 버튼에 함수 연결

사용자가 [OK] 또는 [Cancel]을 클릭할 때 Extension은 적절한 액션을 수행해야 합니다. 버튼을 클릭할 때 수행될 JavaScript 함수를 지정하여 적절한 액션을 결정할 수 있습니다.

1 Configuration/Commands 폴더에 있는 Change Case.js 파일을 엽니다.

2 파일 끝에 다음 코드를 추가합니다.

```
function commandButtons() {
    return new Array("OK", "changeCase()", "Cancel", "window.close()");
}
```

3 파일을 저장합니다.

commandButtons() 함수는 Dreamweaver에서 [OK] 및 [Cancel] 버튼이 제공되도록 하고 사용자가 이 버튼을 클릭할 때 수행될 동작을 Dreamweaver에 알려 줍니다. commandButtons() 함수는 사용자가 [OK]를 클릭하면 changeCase()를 호출하고 사용자가 [Cancel]을 클릭하면 window.close()를 호출하라고 Dreamweaver에 알려 줍니다.

사용자가 대/소문자를 지정할 수 있도록 구현

사용자가 메뉴 항목을 클릭할 때 Extension에서는 사용자가 대문자나 소문자를 선택할 수 있도록 하는 메커니즘을 필요로 합니다. 이 UI에서는 사용자가 이와 같이 선택할 수 있도록 두 개의 라디오 버튼을 정의하여 이 메커니즘을 구현합니다.

1 Change Case.js 파일을 엽니다.

2 파일 끝에 다음 코드를 추가합니다.

```
function changeCase() {
    var uorl;
    // Check whether user requested uppercase or lowercase.
    if (document.forms[0].elements[0].checked)
        uorl = 'u';
    else
        uorl = 'l';
    // Get the DOM.
    var theDOM = dw.getDocumentDOM();
    // Get the outerHTML of the HTML tag (the
    // entire contents of the document).
    var theDocEl = theDOM.documentElement;
    var theWholeDoc = theDocEl.outerHTML;
    // Get the node that contains the selection.
    var theSelNode = theDOM.getSelectedNode();
    // Get the children of the selected node.
    var theChildren = theSelNode.childNodes;
    var i = 0;
    if (theSelNode.hasChildNodes()){
        while (i < theChildren.length){
            if (theChildren[i].nodeType == Node.TEXT_NODE){
                var selText = theChildren[i].data;
                var theSel = theDOM.nodeToOffsets(theChildren[0]);
                break;
            }
            ++i;
        }
    }
    else {
    // Get the offsets of the selection
    var theSel = theDOM.getSelection();
    // Extract the selection
    var selText = theWholeDoc.substring(theSel[0],theSel[1]);
}
if (uorl == 'u'){
        theDocEl.outerHTML = theWholeDoc.substring(0,theSel[0]) +
        selText.toUpperCase() + theWholeDoc.substring(theSel[1]);
    }
    else {
        theDocEl.outerHTML = theWholeDoc.substring(0,theSel[0]) +
        selText.toLowerCase() + theWholeDoc.substring(theSel[1]);
    }
// Set the selection back to where it was when you started
    theDOM.setSelection(theSel[0],theSel[1]);
    window.close(); // close extension UI
}
```

3 이 파일을 Configuration/Commands 폴더에 Change Case.js라는 이름으로 저장합니다.

changeCase() 함수는 사용자가 [OK]를 클릭할 때 commandButtons() 함수에 의해 호출되는 사용자 정의 함수입니다. 이 함수는 선택된 텍스트를 대문자나 소문자로 변경합니다. 이 UI는 라디오 버튼을 사용하므로 선택 사항에 따라 코드가 달라집니다. 또한 사용자가 어느 항목도 선택하지 않을 가능성은 테스트하지 않아도 됩니다.

changeCase() 함수는 먼저 document.forms[0].elements[0].checked 속성을 테스트합니다. document.forms[0].elements[0] 속성은 현재 문서 객체의 첫 번째 양식에 있는 첫 번째 요소, 즉 해당 Extension의 UI를 참조합니다. checked 속성은 해당 요소가 선택되어 있으면(활성 상태) true 값을 반환하고, 그렇지 않으면 false를 반환합니다. 인터페이스에서 elements[0]은 첫 번째 라디오 버튼인 [Uppercase] 버튼을 참조합니다. 사용자가 [OK]를 클릭할 때는 항상 두 라디오 버튼 중 하나가 선택되어 있으므로 코드에서는 선택 사항이 대문자가 아니면 반드시 소문자가 된다고 가정합니다. 이 함수는 uorl 변수를 u 또는 l로 설정하여 사용자의 응답을 저장합니다.

함수의 나머지 코드는 선택된 텍스트를 검색하고 이 텍스트를 지정된 대/소문자, 즉 대문자나 소문자로 변환한 다음 문서의 원래 위치에 다시 복사해 넣습니다.

선택된 텍스트를 사용자의 문서에서 검색하기 위해 이 함수는 DOM을 가져옵니다. 그런 다음 문서의 루트 요소(html 태그)를 가져옵니다. 마지막으로 전체 문서를 theWholeDoc 변수에 추출합니다.

그런 다음 changeCase()는 getSelectedNode()를 호출하여 선택된 텍스트가 포함된 노드를 검색합니다. 또한 이 함수는 선택 항목이 text와 같이 텍스트를 포함하는 태그인 경우 자식 노드(theSelNode.childNodes)를 검색합니다.

자식 노드가 있으면(hasChildNodes()가 true 값을 반환하는 경우) 해당 명령은 자식 노드를 반복하여 텍스트 노드를 찾습니다. 텍스트 노드가 검색되면 해당 텍스트(theChildren[i].data)는 selText에 저장되고 텍스트 노드의 오프셋은 theSel에 저장됩니다.

자식 노드가 없으면 해당 명령은 getSelection()을 호출하여 선택 항목의 시작 및 끝 오프셋을 theSel에 저장합니다. 그런 다음 두 오프셋 사이의 문자열을 추출하여 selText에 저장합니다.

이 함수는 사용자가 대문자를 선택했는지 여부를 확인하기 위해 uorl 변수를 확인합니다. 사용자가 대문자를 선택했으면 이 함수는 해당 HTML 코드를 문서의 섹션에 다시 작성합니다. 먼저 문서의 시작 부분부터 선택 항목의 시작 부분까지 작성하고, 선택된 텍스트를 대문자로 변환(selText.toUppercase())하여 작성한 다음, 마지막으로 선택된 텍스트의 끝 부분부터 문서의 끝 부분까지 작성합니다.

사용자가 소문자를 선택한 경우에도 이 함수는 selText.toLowerCase()를 호출하여 선택된 텍스트를 소문자로 변환한다는 점을 제외하고는 동일한 작업을 수행합니다.

마지막으로, changeCase()는 선택 항목을 다시 설정하고 window.close()를 호출하여 해당 UI를 닫습니다.

Extension 테스트

파일을 Commands 폴더에 저장한 후 Extension을 테스트할 수 있습니다.

1 Dreamweaver를 다시 시작하거나 Extension을 다시 로드합니다. Extension을 다시 로드하는 방법에 대한 자세한 내용은 73페이지의 “Extension 다시 로드”를 참조하십시오.

이제 [명령] 메뉴에 [Change Case] 항목이 나타납니다.

2 문서에 몇 개의 텍스트를 입력합니다.

3 텍스트를 선택합니다.

참고: 사용자가 문서의 텍스트를 선택하기 전까지는 [Change Case]가 흐리게 표시됩니다.

4 [명령] 메뉴에서 [Change Case]를 선택합니다.

텍스트의 대/소문자가 바뀝니다.

명령 API 함수

명령 API의 사용자 정의 함수는 필수 사항이 아닙니다.

canAcceptCommand()

설명

이 함수는 현재 선택 영역에 해당 명령이 적절한지 여부를 확인합니다.

참고: 적어도 한 번 이상 false가 반환되는 경우에만 canAcceptCommand()를 정의하십시오. 이 함수가 정의되어 있지 않으면 해당 명령은 적절한 것으로 간주됩니다. 따라서 시간이 절약되고 성능이 향상됩니다.

인수

없음

반환값

해당 명령이 허용되면 true 값을 반환합니다. 값이 false이면 해당 명령은 메뉴에서 희미하게 표시됩니다.

예제

canAcceptCommand() 함수의 다음 예제에서는 표를 선택한 경우에만 해당 명령을 사용할 수 있도록 합니다.

```
function canAcceptCommand(){
    var retval=false;
    var selObj=dw.getDocumentDOM.getSelectedNode();
    return (selObj.nodeType == Node.ELEMENT_NODE && ¬
    selObj.tagName=="TABLE");{
        retval=true;
}
return retval;
}
```

commandButtons()

설명

이 함수는 [옵션] 대화 상자에 나타나는 버튼을 정의하고 이 버튼을 클릭할 때 수행되는 비헤이비어를 정의합니다. 이 함수가 정의되어 있지 않으면 버튼이 나타나지 않고 명령 파일의 body 섹션이 확장되어 전체 대화 상자를 채웁니다.

기본적으로 이러한 버튼은 대화 상자의 위쪽에 표시됩니다. commandButtons() 함수에 두 개의 추가 값을 지정하여 이러한 버튼을 대화 상자의 아래쪽에 표시할 수 있습니다.

기본적으로 버튼은 오른쪽에 정렬됩니다. PutButtonOnLeft 값을 지정하면 이후의 버튼은 같은 행의 왼쪽에 정렬됩니다.

인수

없음

반환값

짝수 개의 요소가 포함된 배열을 반환합니다. 첫 번째 요소는 맨 위 버튼의 레이블이 포함된 문자열입니다. 두 번째 요소는 맨 위 버튼을 클릭할 때 발생하는 비헤이비어를 정의하는 JavaScript 코드의 문자열입니다. 나머지 요소도 동일한 방법으로 추가 버튼을 정의합니다.

예제

commandButtons()의 다음 인스턴스는 대화 상자의 오른쪽 위 모서리(기본 위치)에 나타나는 OK, Cancel 및 Help 버튼을 정의합니다.

```
function commandButtons(){
return new Array("OK" , "doCommand()" ,
                 "Cancel" , "window.close()" ,
                 "Help" , "showHelp()");
}
```

버튼의 배치 및 정렬 방식은 사용자 정의할 수 있습니다.

예제

commandButtons()의 다음 인스턴스는 대화 상자의 아래쪽에 버튼을 표시합니다. 반환 배열의 첫 번째 값이 PutButtonsOnBottom이면 두 번째 값은 버튼 이름 중 하나와 함께 defaultButton으로 지정할 수 있습니다. 이 버튼은 기본적으로 선택되어 있으며 Enter 키를 누를 때 사용됩니다. 이 예제에서 OKbutton은 기본값으로 정의되어 있습니다.

```
function commandButtons(){
return new Array("PutButtonsOnBottom" , "OkButton defaultButton" ,
                 "OK" , "doCommand()" ,
                 "Cancel" , "window.close()" ,
                 "Help" , "showHelp()");
}
```

예제

다음 예제에서는 PutButtonOnLeft를 사용하여 [도움말] 버튼을 왼쪽에 정렬합니다.

```
function commandButtons(){
return new Array("PutButtonsOnBottom", "OkButton defaultButton",
                  "OK", "doCommand()",
                  "Cancel", "window.close()",
                  "PutButtonOnLeft",
                  "Help" , "showHelp()");}
```

isDOMRequired()

설명

이 함수는 명령을 실행하는 데 유효한 DOM이 필요한지 여부를 확인합니다. 이 함수가 true 값을 반환하거나 이 함수가 정의되어 있지 않으면 Dreamweaver에서는 유효한 DOM이 명령에 필요하다고 간주하고 실행 전에 문서의 [디자인] 뷰와 [코드] 뷰를 동기화합니다. 동기화하면 [코드] 뷰의 모든 편집 내용이 [디자인] 뷰에 업데이트됩니다.

인수

없음

반환값

명령을 실행하는 데 유효한 DOM이 필요한 경우 true 값을 반환하고, 그렇지 않으면 false를 반환합니다.

receiveArguments()

설명

이 함수는 메뉴 항목이나 dw.runCommand() 함수에서 전달되는 인수를 처리합니다.

인수

{arg1}, {arg2},...{argN}

- menuitem 태그에 arguments 속성이 정의되어 있으면 해당 속성 값이 receiveArguments() 함수에 하나 이상의 인수로 전달됩니다. 인수는 dw.runCommand() 함수에 의해 명령에 전달될 수도 있습니다.

반환값
없음

windowDimensions()

설명
이 함수는 [매개 변수] 대화 상자의 구체적 크기를 설정합니다. 이 함수를 정의하지 않으면 윈도우 크기가 자동으로 계산됩니다.

참고: [옵션] 대화 상자의 크기를 640x480픽셀보다 크게 하려는 경우에만 이 함수를 정의하십시오.

인수
platform

- **platform** 인수의 값은 사용자의 플랫폼에 따라 "macintosh" 또는 "windows"입니다.

반환값
"widthInPixels,heightInPixels" 형식의 문자열을 반환합니다.

반환된 크기는 [확인] 및 [취소] 버튼의 영역을 포함하지 않으므로 전체 대화 상자의 크기보다 작습니다. 반환된 크기에 옵션이 모두 들어가지 않으면 스크롤 막대가 나타납니다.

예제
windowDimensions() 함수의 다음 예제에서는 [매개 변수] 대화 상자의 크기를 648x520픽셀로 설정합니다.

```
function windowDimensions(){
    return "648,520";
}
```

10장: 메뉴 및 메뉴 명령

Adobe Dreamweaver에서는 Dreamweaver Configuration/Menus 폴더의 menus.xml 파일에 정의된 구조에서 모든 메뉴를 만듭니다. menus.xml 파일을 편집하여 메뉴 명령을 다시 정렬하고 이름을 바꾸고 제거할 수 있습니다. 또한 메뉴 명령에 사용할 키보드 단축키를 추가, 변경 및 제거할 수 있습니다. 그러나 대부분의 경우 키보드 단축키 편집기(Dreamweaver 도움말 참조)를 사용하여 키보드 단축키를 쉽게 편집할 수 있습니다. Dreamweaver 메뉴를 변경하면 다음에 Dreamweaver를 시작하거나 Extension을 다시 로드할 때 변경 사항이 적용됩니다.

menus.xml 파일

menus.xml 파일에는 메뉴 막대, 메뉴, 메뉴 명령, 분리 기호, 단축키 목록 및 키보드 단축키에 대한 구조화된 목록이 포함되어 있습니다. XML 태그는 이러한 항목을 설명하며 텍스트 편집기에서 XML 태그를 편집할 수 있습니다.

참고: 메뉴를 변경할 때는 주의해야 합니다. Dreamweaver에서는 XML 구문 오류가 있는 메뉴나 메뉴 명령이 무시됩니다.

다중 사용자 운영 체제에서는 Dreamweaver 내에서 변경하여 menus.xml을 변경할 수 있습니다. 예를 들어, 키보드 단축키 편집기를 사용하여 키보드 단축키를 변경할 수 있습니다. 이러한 경우에 사용자 Configuration 폴더에 menus.xml 파일이 만들어집니다. 다중 사용자 운영 체제에서 menus.xml을 사용자 정의하려면 사용자의 Configuration 폴더에 있는 이 파일 사본을 편집합니다. 사용자의 Configuration 폴더에 menus.xml 파일이 아직 만들어지지 않은 경우에는 마스터 menus.xml 파일을 사용자의 Configuration 폴더에 복사합니다. 자세한 내용은 72페이지의 “다중 사용자 Configuration 폴더”를 참조하십시오.

XML 편집기에서 menus.xml을 열면 menus.xml 파일 내의 앰퍼샌드(&)와 관련된 오류 메시지가 나타날 수 있습니다. 따라서 텍스트 편집기에서 menus.xml을 편집하는 것이 가장 좋습니다. Dreamweaver에서는 menus.xml을 편집하지 마십시오. XML에 대한 기본 정보는 Dreamweaver 도움말을 참조하십시오.

참고: 현재 menus.xml 파일 또는 Dreamweaver의 다른 구성 파일을 수정하기 전에 해당 파일의 백업 사본을 만들어 두는 것이 좋습니다. 미리 백업해 놓지 못한 경우에는 Configuration 폴더에서 menus.xml이 menus.bak의 사본으로 대체됩니다.

열기 및 닫기 menu bar 태그로 정의하는 메뉴 막대는 개별 메뉴이거나 일련의 메뉴입니다. 예를 들어 기본 메뉴 막대, 별도의 [사이트] 패널 메뉴 막대(Windows에서만 표시되고 Macintosh에서는 표시되지 않음) 및 각 컨텍스트 메뉴에 대한 메뉴 막대가 있습니다. 각 메뉴 막대에는 메뉴가 하나 이상 포함되어 있으며 메뉴는 메뉴 태그 안에 들어 있습니다. 각 메뉴에는 메뉴 명령이 하나 이상 포함되어 있으며 각 항목은 menuitem 태그 및 속성에 의해 표시됩니다. 또한 메뉴에는 분리 기호 태그에 의해 표시되는 분리 기호 및 하위 메뉴가 포함될 수도 있습니다.

Dreamweaver에서는 메뉴 명령과 연관된 기본 키보드 단축키뿐만 아니라 대체 단축키 및 특정 컨텍스트에서만 사용 가능한 단축키를 제공합니다. 예를 들어 Ctrl+Y(Windows) 또는 Command+Y(Macintosh)는 [다시 실행] 명령의 단축키이고, Ctrl+Shift+Z 또는 Command+Shift+Z는 [다시 실행] 명령의 대체 단축키입니다. 메뉴 명령의 태그 안에 표현될 수 없는 이러한 대체 단축키와 특정 컨텍스트에서만 사용 가능한 단축키는 menus.xml 파일의 단축키 목록에 정의됩니다. shortcutlist 태그로 기술되는 각 단축키 목록에는 단축키가 하나 이상 포함되며 각 단축키는 단축키 태그로 기술됩니다.

다음 단원에서는 menus.xml 파일의 태그 구문에 대해 설명합니다. 선택적 속성은 속성 목록에서 중괄호({})로 표시됩니다. 중괄호로 표시되지 않은 모든 속성은 필수 속성입니다.

<menubar>

설명

Dreamweaver 메뉴 구조에 포함된 메뉴 막대에 대한 정보를 제공합니다.

속성
name, {app}, id, {platform}

- name 메뉴 막대의 이름입니다. name은 필수 속성이지만 값 ""을 지정할 수 있습니다.
- app 메뉴 막대를 사용할 수 있는 응용 프로그램의 이름입니다. 이 속성은 현재 사용되지 않습니다.
- id 메뉴 막대의 메뉴 ID입니다. menus.xml 파일에 있는 각 메뉴 ID는 고유해야 합니다.
- platform 메뉴 막대가 지정된 플랫폼에서만 나타남을 나타냅니다. 유효한 값은 "win" 및 "mac"입니다.

내용
이 태그에는 menu 태그가 하나 이상 포함되어야 합니다.

컨테이너
없음

예제
기본([문서] 윈도우) 메뉴 막대는 다음 menubar 태그를 사용합니다.

```
<menubar name="Main Window" id="DWMainWindow">
<!-- menu tags here -->
</menubar>
```

<menu>

설명
Dreamweaver 메뉴 구조에 나타나는 메뉴 또는 하위 메뉴에 대한 정보를 제공합니다.

속성
name, {app}, id, {platform}, {showIf}

- name 메뉴 막대에 나타나는 메뉴의 이름입니다. Windows에서 메뉴의 선택 키(니모닉)를 설정하려면 액세스 문자 앞에 밑줄(_)을 사용합니다. Macintosh에서는 밑줄이 자동으로 제거됩니다.
- app 메뉴를 사용할 수 있는 응용 프로그램의 이름입니다. 이 속성은 현재 사용되지 않습니다.
- id 메뉴의 메뉴 ID입니다. 파일의 모든 ID는 고유해야 합니다.
- platform 메뉴가 지정된 플랫폼에서만 나타남을 알려 줍니다. 유효한 값은 "win" 및 "mac"입니다.
- showIf 지정된 Dreamweaver 활성자의 값이 true인 경우에만 메뉴가 나타나도록 지정합니다. 사용 가능한 활성자로는 _SERVERMODEL_ASP, _SERVERMODEL_ASPNET, _SERVERMODEL_JSP, _SERVERMODEL_CFML (모든 버전의 ColdFusion에 해당), _SERVERMODEL_CFML_UD4 (UltraDev 버전 4 ColdFusion에 해당), _SERVERMODEL_PHP, _FILE_TEMPLATE, _VIEW_CODE, _VIEW_DESIGN, _VIEW_LAYOUT, _VIEW_EXPANDED_TABLES, 및 _VIEW_STANDARD가 있습니다. 각 활성자 사이에 AND를 의미하는 쉼표를 사용하여 여러 개의 활성자를 지정할 수 있습니다. "!"를 사용하여 NOT을 지정할 수도 있습니다. 예를 들어 메뉴가 ASP 페이지의 [코드] 뷰에만 나타나도록 하려면 속성을 showIf="_VIEW_CODE, _SERVERMODEL_ASP"로 지정하십시오.

내용
이 태그에는 menuitem 태그 및 separator 태그가 하나 이상 포함될 수 있습니다. 또한 이 태그에는 하위 메뉴를 만들 때 사용하는 다른 menu 태그 및 표준 HTML 주석 태그도 포함될 수 있습니다.

컨테이너
이 태그는 menubar 태그에 포함되어야 합니다.

예제

```
<menu name="_File" id="DWMenu_File">
    <!-- menuitem, separator, menu, and comment tags here -->
</menu>
```

<menuitem>

설명

Dreamweaver 메뉴에 사용할 메뉴 명령을 정의합니다.

속성

name, id, {app}, {key}, {platform}, {enabled}, {arguments}, {command}, {file}, {checked}, {dynamic}, {isdomrequired}, {showIf}

- name - 메뉴에 나타나는 메뉴 명령 이름입니다. 밑줄은 밑줄 다음의 문자가 해당 명령의 선택 키(니모닉)라는 것을 알려 줍니다(Windows에만 해당).
- id - 메뉴 항목을 식별하는 데 사용됩니다. 이 ID는 메뉴 구조에서 고유해야 합니다. menus.xml에 새 메뉴 명령을 추가하는 경우 회사 이름이나 다른 고유한 문자열을 각 메뉴 명령 ID의 접두어로 사용하여 고유성을 유지하십시오.
- app - 해당 메뉴 명령을 사용할 수 있는 응용 프로그램의 이름입니다. 이 속성은 현재 사용되지 않습니다.
- key - 명령에 대한 키보드 단축키입니다(있는 경우). 다음 문자열을 사용하여 수정자 키를 지정합니다.
- Cmd - Ctrl 키(Windows) 또는 Command 키(Macintosh)를 지정합니다.
- Alt 및 Opt - Alt 키(Windows) 또는 Option 키(Macintosh)와 같은 의미로 사용됩니다.
- Shift - 두 플랫폼 모두에서 Shift 키를 지정합니다.
- Ctrl - 두 플랫폼 모두에서 Ctrl 키를 지정합니다.
- 플러스(+) 기호 - 지정된 단축키가 수정자를 두 개 이상 사용하는 경우 수정자 키를 구분합니다. 예를 들어 key 속성의 Cmd+Opt+5는 사용자가 Ctrl+Alt+5(Windows) 또는 Command+Option+5(Macintosh)를 누를 때 해당 메뉴 명령이 실행된다는 것을 의미합니다.
- 특수 키 - 이름으로 지정되며, F1-F12, PgDn, PgUp, Home, End, Ins, Del, Tab, Esc, BkSp 및 Space가 있습니다. 이 특수 키에 수정자 키를 적용할 수도 있습니다.
- platform - 해당 메뉴 항목이 나타나는 플랫폼을 나타냅니다. 유효한 값은 Windows를 의미하는 "win" 또는 Macintosh를 의미하는 "mac"입니다. platform 속성을 지정하지 않으면 해당 메뉴 명령이 두 플랫폼 모두에 나타납니다. 메뉴 명령이 두 플랫폼에서 서로 다르게 동작하도록 하려면 두 메뉴 명령의 이름은 같게 하고 ID는 다르게 하십시오(예: platform="win" 및 platform="mac").
- enabled - 해당 메뉴 명령이 현재 활성 상태인지 여부를 결정하는 JavaScript 코드(보통 JavaScript 함수 호출)를 제공합니다. 함수가 false를 반환하면 해당 메뉴 명령이 희미하게 표시됩니다. 기본값은 "true"입니다. 하지만 값이 "true"인 경우에도 항상 명확하게 값을 지정하는 것이 좋습니다.
- arguments - Dreamweaver에서 file 속성에 지정된 JavaScript 파일의 코드에 전달할 인수를 제공합니다. 속성 값을 구분하는 데 사용되는 이중 인용 부호(") 내에서 인수를 단일 인용 부호(')로 묶습니다.
- command - 사용자가 메뉴에서 이 항목을 선택할 때 실행되는 JavaScript 표현식을 지정합니다. 복잡한 JavaScript 코드의 경우 file 속성에 지정된 JavaScript 파일을 대신 사용합니다. 각 메뉴 명령에 대해 file이나 command를 지정해야 합니다.
- file - 해당 메뉴 명령을 제어하는 JavaScript가 포함된 HTML 파일의 이름입니다. 파일 경로는 Configuration 폴더에 대한 상대 경로로 지정합니다. 예를 들어 [도움말] > [환영합니다] 메뉴 명령의 경우 file="Commands/Welcome.htm"을 지정합니다. file 속성은 command, enabled 및 checked 속성보다 우선합니다. 각 메뉴 명령에 대해 file이나 command를 지정해야 합니다. [작업 내역] 패널을 사용한 명령 만들기에 대한 자세한 내용은 Dreamweaver 도움말을 참조하십시오. JavaScript 명령 작성에 대한 자세한 내용은 124페이지의 “명령”을 참조하십시오.

- checked - 메뉴에서 해당 메뉴 명령 옆에 체크 표시가 나타날지 여부를 나타내는 JavaScript 표현식입니다. 표현식이 true이면 메뉴 명령 옆에 체크 표시가 나타납니다.
- dynamic - 이 속성이 존재하면 메뉴 명령이 동적으로 HTML 파일에 의해 결정된다는 것을 나타냅니다. 이 HTML 파일에는 해당 메뉴 명령의 텍스트 및 상태를 설정하는 JavaScript 코드가 포함되어 있습니다. 태그를 dynamic으로 지정하려면 file 속성도 지정해야 합니다.
- isdomrequired - 해당 메뉴 명령의 코드를 실행하기 전에 [디자인] 뷰와 [코드] 뷰를 동기화할지 여부를 나타냅니다. 유효한 값은 "true"(기본값) 및 "false"입니다. 이 속성을 "false"로 설정하면 해당 메뉴 명령으로 만들어진 파일의 변경 사항이 Dreamweaver DOM을 사용하지 않는다는 것을 의미합니다. DOM에 대한 자세한 내용은 90페이지의 “Dreamweaver 문서 객체 모델”을 참조하십시오.
- showIf - 지정된 Dreamweaver 활성자가 true 값을 갖는 경우에만 해당 메뉴 항목이 나타나도록 지정합니다. 사용 가능한 활성자로는 _SERVERMODEL_ASP, _SERVERMODEL_ASPNET, _SERVERMODEL_JSP, _SERVERMODEL_CFML (모든 버전의 ColdFusion에 해당), _SERVERMODEL_CFML_UD4 (UltraDev 버전 4 ColdFusion에 해당), _SERVERMODEL_PHP, _FILE_TEMPLATE, _VIEW_CODE, _VIEW_DESIGN, _VIEW_LAYOUT, _VIEW_EXPANDED_TABLES 및 _VIEW_STANDARD가 있습니다. 각 활성자 사이에 AND를 의미하는 쉼표를 사용하여 여러 개의 활성자를 지정할 수 있습니다. "!"를 사용하여 NOT을 지정할 수도 있습니다. 예를 들어 메뉴 명령이 [코드] 뷰에는 나타나지만 ColdFusion 페이지에는 나타나지 않도록 하려면 이 속성을 showIf="_VIEW_CODE, !_SERVERMODEL_CFML"로 지정하십시오.

내용
없음(빈 태그)

컨테이너
이 태그는 menu 태그에 포함되어야 합니다.

예제
```
<menuitem name="_New" key="Cmd+N" enabled="true" command="dw.createDocument()" id="DWMenu_File_New" />
```

<separator>

설명
메뉴의 해당 위치에 분리 기호가 나타나야 함을 나타냅니다.

속성
{app}

app - 분리 기호가 나타나는 응용 프로그램의 이름입니다. 이 속성은 현재 사용되지 않습니다.

내용
없음(빈 태그)

컨테이너
이 태그는 menu 태그에 포함되어야 합니다.

예제
```
<separator />
```

<shortcutlist>

설명
menus.xml 파일의 단축키 목록을 지정합니다.

속성
{app}, id, {platform}

- app - 단축키 목록을 사용할 수 있는 응용 프로그램의 이름입니다. 이 속성은 현재 사용되지 않습니다.
- id - 단축키 목록의 ID입니다. 이 속성 값은 Dreamweaver에서 해당 단축키가 연관되는 메뉴 막대나 컨텍스트 메뉴의 메뉴 ID와 같아야 합니다. 유효한 값은 "DWMainWindow", "DWMainSite", "DWTimelineContext" 및 "DWHTMLContext"입니다.
- platform - 단축키 목록이 지정된 플랫폼에서만 나타나야 함을 나타냅니다. 유효한 값은 "win" 및 "mac"입니다.

내용
이 태그에는 shortcut 태그가 한 개 이상 포함될 수 있습니다. 또한 이 태그에는 HTML 주석 태그와 동일한 구문을 사용하는 주석 태그가 한 개 이상 포함될 수 있습니다.

컨테이너
없음

예제
```
<shortcutlist id="DWMainWindow">
<!-- shortcut and comment tags here -->
</shortcutlist>
```

<shortcut>

설명
menus.xml 파일의 키보드 단축키를 지정합니다.

속성
key, {app}, {platform}, {file}, {arguments}, {command}, id, {name}

- key - 키보드 단축키를 활성화하는 키 조합입니다. 구문에 대한 자세한 내용은 135페이지의 “<menuitem>”을 참조하십시오.
- app - 단축키를 사용할 수 있는 응용 프로그램의 이름입니다. 이 속성은 현재 사용되지 않습니다.
- platform - 단축키가 지정된 플랫폼에서만 동작하도록 지정합니다. 유효한 값은 "win" 및 "mac"입니다. 이 속성을 지정하지 않으면 이 단축키가 두 플랫폼 모두에서 작동합니다.
- file - 키보드 단축키를 사용할 때 Dreamweaver에서 실행되는 JavaScript 코드가 포함된 파일의 경로입니다. file 속성은 command 속성보다 우선합니다. 각 단축키에 대해 file이나 command를 지정해야 합니다.
- arguments - Dreamweaver에서 file 속성에 지정된 JavaScript 파일의 코드에 전달할 인수를 제공합니다. 속성 값을 구분하는 데 사용되는 이중 인용 부호(") 내에서 인수를 단일 인용 부호(')로 묶습니다.
- command - 키보드 단축키를 사용할 때 Dreamweaver에서 실행되는 JavaScript 코드입니다. 각 단축키에 대해 file이나 command를 지정합니다.
- id - 단축키에 대한 고유 식별자입니다.

- name - 키보드 단축키로 실행되는 명령의 이름을 메뉴 명령 이름의 스타일로 나타낸 것입니다. 예를 들어 F12 단축키에 대한 name 속성은 "기본 브라우저에서 미리보기"입니다.

내용
없음(빈 태그)

컨테이너
이 태그는 shortcutlist 태그에 포함되어야 합니다.

예제
```
<shortcut key="Cmd+Shift+Z" file="Menus/MM/Edit_Clipboard.htm"
arguments="'redo'" id="DWShortcuts_Edit_Redo" />
```

<tool>

설명
한 개의 도구를 나타냅니다. menus.xml 파일에 도구에 대한 모든 단축키가 하위 태그로 포함됩니다.

속성
{name}, id

- name - 지역화된 버전의 도구 이름입니다.
- id - 단축키가 적용되는 도구를 식별하는 내부 도구 식별자입니다.

내용
이 태그에는 activate, override 또는 action 태그가 한 개 이상 포함될 수 있습니다.

컨테이너
이 태그는 menu 태그에 포함되어야 합니다.

예제
```
<tool name="Hand tool" id="com.Macromedia.dreamweaver.tools.hand">
    <!-- tool tags here -->
</tool>
```

<action>

설명
도구가 활성화된 상태에서 키 조합을 눌렀을 때 실행되는 키 조합과 JavaScript가 포함되어 있습니다.

속성
{name}, key, command, id

- name - 지역화된 버전의 액션입니다.
- key - 액션을 실행하는 데 사용되는 키 조합입니다. 구문에 대한 자세한 내용은 135페이지의 “<menuitem>”을 참조하십시오.
- command - 실행될 JavaScript 명령문입니다. 이 속성의 형식은 137페이지의 “<shortcut>”에서 command 속성과 동일합니다.

- id - 액션을 참조하는 데 사용되는 고유 ID입니다.

내용
없음(빈 태그)

컨테이너
이 태그는 tool 태그에 포함되어야 합니다.

예제
```
<action name="Set magnification to 50%" key="5" command="dw.activeViewScale = 0.50" id ="DWTools_Zoom_50" />
```

<activate>

설명
도구를 활성화하는 키 조합이 포함되어 있습니다.

속성
{name}, key, id

- name - 지역화된 버전의 액션입니다.
- key - 도구를 활성화하는 데 사용되는 키 조합입니다. 구문에 대한 자세한 내용은 135페이지의 “<menuitem>”을 참조하십시오.
- id - 액션을 참조하는 데 사용되는 고유 ID입니다.

내용
없음(빈 태그)

컨테이너
이 태그는 tool 태그에 포함되어야 합니다.

예제
```
<activate name="Switch to Hand tool" key="H" id="DWTools_Hand_Active1" />
```

<override>

설명
도구를 일시적으로 활성화하는 키 조합이 포함되어 있습니다. 다른 모달 도구를 사용하고 있는 경우에 이 키를 계속 누르고 있으면 이 도구로 전환할 수 있습니다.

속성
{name}, key, id

- name - 지역화된 버전의 액션입니다.
- key - 도구를 빠르게 활성화하는 데 사용되는 키 조합입니다. 구문에 대한 자세한 내용은 135페이지의 “<menuitem>”을 참조하십시오.
- id - 액션을 참조하는 데 사용되는 고유 ID입니다.

내용
없음(빈 태그)

컨테이너
이 태그는 tool 태그에 포함되어야 합니다.

예제
```
<override name="Quick switch to Hand tool" key="Space" id="DWTools_Hand_Override" />
```

메뉴 및 메뉴 명령 변경

menus.xml 파일을 편집하면 특정 메뉴 내에서 또는 한 메뉴에서 다른 메뉴로 메뉴 명령을 이동할 수 있고, 메뉴에 분리 기호를 추가하거나 메뉴에서 분리 기호를 제거할 수 있으며, 특정 메뉴 막대 내에서 또는 한 메뉴 막대에서 다른 메뉴 막대로 메뉴를 이동할 수 있습니다.

다른 메뉴와 동일한 절차를 사용하여 컨텍스트 메뉴의 내부나 외부로 항목을 이동할 수 있습니다.

자세한 내용은 133페이지의 “menus.xml 파일”을 참조하십시오.

메뉴 명령 이동
1. Dreamweaver를 종료합니다.
2. menus.xml 파일의 백업 사본을 만듭니다.
3. BBEdit, Macromedia® HomeSite® 또는 Wordpad와 같은 텍스트 편집기에서 menus.xml을 엽니다. 이 파일을 Dreamweaver에서 열지 마십시오.
4. 시작 부분의 <menuitem부터 끝 부분의 />까지 전체 menuitem 태그를 잘라냅니다.
5. 메뉴 명령의 새 위치에 삽입 포인터를 놓습니다. 이때 삽입 포인터는 menu 태그와 해당하는 /menu 태그 사이에 두어야 합니다.
6. 메뉴 명령을 새 위치에 붙여넣습니다.

메뉴 명령을 이동하는 동안 하위 메뉴 만들기
1. 메뉴 내(menu 태그와 해당하는 /menu 태그 사이)에 삽입 포인터를 놓습니다.
2. 새 menu 태그 및 /menu 태그 쌍을 해당 메뉴 내에 삽입합니다.
3. 새 메뉴 명령을 새 하위 메뉴에 추가합니다.

두 메뉴 명령 사이에 분리 기호 삽입
- 두 menuitem 태그 사이에 separator/ 태그를 배치합니다.

기존 분리 기호 제거
- 해당 separator/ 행을 삭제합니다.

메뉴 이동
1. Dreamweaver를 종료합니다.
2. menus.xml 파일의 백업 사본을 만듭니다.
3. BBEdit, HomeSite 또는 Wordpad와 같은 텍스트 편집기에서 menus.xml을 엽니다. 이 파일을 Dreamweaver에서 열지 마십시오.

4 열기 menu 태그부터 닫기 /menu 태그까지의 전체 메뉴 및 해당 내용을 잘라냅니다.

5 메뉴의 새 위치에 삽입 포인터를 배치합니다. 이때 삽입 포인터는 menubar 태그와 해당하는 /menubar 태그 사이에 두어야 합니다.

6 메뉴를 새 위치에 붙여넣습니다.

메뉴 명령 또는 메뉴 이름 변경

menus.xml 파일을 편집하여 메뉴 명령 또는 메뉴의 이름을 손쉽게 변경할 수 있습니다.

1 Dreamweaver를 종료합니다.

2 menus.xml 파일의 백업 사본을 만듭니다.

3 HomeSite, BBEdit 또는 Wordpad와 같은 텍스트 편집기에서 menus.xml을 엽니다. 이 파일을 Dreamweaver에서 열지 마십시오.

4 메뉴 명령을 변경하려면 적절한 menuitem 태그를 찾고 해당 name 속성의 값을 변경합니다. 메뉴를 변경하려면 적절한 menu 태그를 찾고 해당 name 속성의 값을 변경합니다. 두 경우 모두 id 속성을 변경하면 안 됩니다.

5 menus.xml을 저장하고 닫은 후 Dreamweaver를 다시 시작하여 변경 내용을 확인합니다.

키보드 단축키 변경

기본 키보드 단축키가 사용하는 데 불편한 경우 기존 단축키를 변경하거나 제거하고 새 단축키를 추가할 수 있습니다. 이 작업을 수행하는 가장 쉬운 방법은 [키보드 단축키 편집기]를 사용하는 것입니다. 자세한 내용은 Dreamweaver 도움말을 참조하십시오. 또는 menus.xml에서 키보드 단축키를 직접 수정할 수도 있습니다. 하지만 menus.xml에 직접 단축키를 입력하게 되면 [키보드 단축키 편집기]를 사용하는 경우보다 실수할 확률이 더 높습니다.

1 Dreamweaver를 종료합니다.

2 menus.xml 파일의 백업 사본을 만듭니다.

3 BBEdit, HomeSite 또는 Wordpad와 같은 텍스트 편집기에서 menus.xml을 엽니다. 이 파일을 Dreamweaver에서 열지 마십시오.

4 Dreamweaver 기술 지원 센터(http://www.adobe.com/go/dreamweaver_support_kr)에서 제공되는 Keyboard Shortcut Matrix에서 현재 사용되지 않거나 다시 지정하려는 단축키를 찾습니다.

   키보드 단축키를 다시 지정할 경우 나중에 참조할 수 있도록 키보드 매트릭스의 인쇄본에서 해당 키보드 단축키를 변경합니다.

5 키보드 단축키를 다시 지정하려면 해당 단축키가 지정된 메뉴 명령을 찾고 해당 메뉴 명령에서 key="shortcut" 속성을 제거합니다.

6 키보드 단축키를 새로 지정하거나 다시 지정할 메뉴 명령을 찾습니다.

7 해당 메뉴 명령에 이미 키보드 단축키가 있으면 단축키가 있는 행에서 key 속성을 찾습니다. 메뉴 명령에 단축키가 없으면 menuitem 태그 내의 속성 사이에 key=""를 추가합니다.

8 key 속성의 이중 인용 부호(") 사이에 새 키보드 단축키를 입력합니다.

   키 조합의 각 키 사이에는 플러스(+) 기호를 사용합니다. 수정자에 대한 자세한 내용은 135페이지의 “<menuitem>”에서 menuitem 태그에 대한 설명을 참조하십시오.

   키보드 단축키가 다른 메뉴 명령에도 사용되고 있고 이를 제거할 수 없으면 해당 단축키는 menus.xml에서 이 단축키를 사용하는 첫 번째 메뉴 명령에만 적용됩니다.

   참고: Windows 전용 메뉴 명령 및 Macintosh 전용 메뉴 명령에 대해서도 동일한 키보드 단축키를 사용할 수 있습니다.

9 Keyboard Shortcut Matrix에서 적절한 위치에 새 단축키를 작성합니다.

팝업 메뉴 및 컨텍스트 메뉴

Dreamweaver의 여러 패널과 대화 상자에서는 다양한 팝업 메뉴와 컨텍스트 메뉴가 제공됩니다. 일부 컨텍스트 메뉴는 menus.xml 파일에 정의되어 있고 일부 컨텍스트 메뉴는 다른 XML 파일에 정의되어 있습니다. 이러한 메뉴에서 항목을 추가, 제거 또는 수정할 수 있습니다. 하지만 대부분의 경우 Extension을 작성하여 항목을 변경하는 것이 더 좋습니다.

Dreamweaver의 다음 팝업 메뉴 및 컨텍스트 메뉴는 XML 파일에 지정되어 있으며 menus.xml의 태그와 동일한 태그를 사용합니다.

- [바인딩] 패널의 플러스(+) 팝업 메뉴에 있는 데이터 소스는 DataSources 폴더의 하위 폴더에 있는 DataSources.xml 파일에 지정되어 있습니다.
- [서버 비헤이비어] 패널의 플러스(+) 팝업 메뉴에 있는 서버 비헤이비어는 ServerBehaviors 폴더의 하위 폴더에 있는 ServerBehaviors.xml 파일에 지정되어 있습니다.
- [서식 목록 편집] 대화 상자의 플러스(+) 팝업 메뉴에 있는 서버 서식은 ServerFormats 폴더의 하위 폴더에 있는 ServerFormats.xml 파일에 지정되어 있습니다.
- [바인딩] 패널의 바인딩에 대한 서식 팝업 메뉴의 항목은 ServerFormats 폴더의 하위 폴더에 있는 Formats.xml 파일에 지정되어 있습니다. Dreamweaver 내에서 [서식 추가] 대화 상자를 사용하여 이 메뉴에 항목을 추가할 수 있습니다.
- [태그 라이브러리 편집기] 대화 상자의 메뉴 명령은 TagLibraries/TagImporters/TagImporters.xml 파일에 지정되어 있습니다.
- [서버 비헤이비어 구성 관리자]의 일부인 [생성 비헤이비어 대화 상자]에 있는 매개 변수에 대한 메뉴 명령은 Shared/Controls/String Menu/Controls.xml에 지정되어 있습니다.
- ColdFusion 구성 요소와 연관된 컨텍스트 메뉴에 대한 항목은 Components/ColdFusion/CFCs/CFCsMenus.xml에 지정되어 있습니다.
- ColdFusion 데이터 소스와 연관된 컨텍스트 메뉴에 대한 항목은 Components/ColdFusion/DataSources/DataSourcesMenus.xml에 지정되어 있습니다.
- 다양한 서버 구성 요소와 연관된 컨텍스트 메뉴에 대한 항목은 Components 폴더의 하위 폴더에 있는 XML 파일에 지정되어 있습니다.

메뉴 명령

메뉴 명령을 사용하면 더욱 유연하고 동적인 메뉴를 만들 수 있습니다. 메뉴 명령을 사용하면 현재 문서, 열려 있는 다른 문서 또는 로컬 하드 드라이브의 모든 HTML 문서에 대해 대부분의 편집 작업을 수행할 수 있습니다.

메뉴 명령은 menus.xml 파일에 있는 menuitem 태그의 file 속성에서 참조하는 HTML 파일입니다. 메뉴 명령 파일의 body 섹션에는 명령의 옵션(예: 표 정렬 방법 및 기준 열)을 받아들이는 HTML 양식이 포함될 수 있습니다. 메뉴 명령 파일의 head 섹션에는 JavaScript 함수가 포함되어 있습니다. 이 함수는 body 섹션에서 입력된 양식을 처리하고 사용자 문서에 대한 편집을 제어합니다.

메뉴 명령은 Dreamweaver 응용 프로그램 폴더 내의 Configuration/Menus 폴더에 저장됩니다.

참고: Mac OS X 10.3에서의 명령은 youUser/Library/Application Support/Adobe/Dreamweaver 9/Commands/yourcommandname.html에 저장됩니다.

다음 표에서는 메뉴 명령을 만들 때 사용하는 파일에 대해 설명합니다.

경로	파일	설명
Configuration/Menus/	menus.xml	메뉴 막대, 메뉴, 메뉴 명령, 분리 기호, 단축키 목록 및 키보드 단축키에 대한 구조화된 목록이 포함되어 있습니다. 새 메뉴와 메뉴 명령을 추가하려면 이 파일을 수정하십시오.
Configuration/Menus/	**commandname**.htm	메뉴 명령에 필요한 함수가 들어 있습니다.

참고: Dreamweaver에 사용자 정의 메뉴 명령을 추가할 때는 Menus 폴더의 최상위 수준에 추가하거나 하위 폴더를 만드십시오. MM 폴더는 Dreamweaver와 함께 제공되는 메뉴 명령에 사용하도록 예약되어 있습니다.

명령 메뉴 수정

menus.xml 파일을 편집하지 않고 [명령] 메뉴에 특정 종류의 명령을 추가하거나 해당 이름을 변경할 수 있습니다. menus.xml에 대한 자세한 내용은 140페이지의 "메뉴 및 메뉴 명령 변경"을 참조하십시오.

참고: Dreamweaver에서 명령이라는 용어에는 두 가지 의미가 있습니다. 엄밀하게 말하면 명령이란 특별한 종류의 Extension입니다. 하지만 일부 경우에 명령은 메뉴 항목과 같은 의미로 Dreamweaver 메뉴에 나타나는 임의의 항목을 나타내기도 합니다. 이때 해당 항목이 수행하는 작업이나 해당 항목의 구현 방식은 상관이 없습니다.

[명령] 메뉴에 자동으로 배치되는 새 명령을 만들려면 [작업 내역] 패널을 사용합니다. 또는 Extension Manager를 사용하여 명령을 포함한 새 Extension을 설치할 수도 있습니다. 자세한 내용은 Dreamweaver 도움말을 참조하십시오.

[명령] 메뉴에 있는 항목의 순서를 다시 정렬하거나 메뉴 사이에서 항목을 이동하려면 menus.xml 파일을 편집해야 합니다.

만든 명령의 이름 바꾸기

1 [명령] > [명령 목록 편집]을 선택합니다.

이름을 변경할 수 있는 모든 명령이 나열된 대화 상자가 나타납니다. 기본 [명령] 메뉴에 있는 명령은 이 목록에 나타나지 않으며 이 방법으로 편집할 수 없습니다.

2 이름을 바꿀 명령을 선택합니다.

3 새 이름을 입력합니다.

4 [닫기]를 클릭합니다.

[명령] 메뉴에서 해당 명령의 이름이 바뀝니다.

만든 명령 삭제

1 [명령] > [명령 목록 편집]을 선택합니다.

삭제할 수 있는 모든 명령이 나열된 대화 상자가 나타납니다. 기본 [명령] 메뉴에 있는 명령은 이 목록에 나타나지 않으며 이 방법으로 삭제할 수 없습니다.

2 삭제할 명령을 선택합니다.

3 [삭제]를 클릭하고 해당 명령을 삭제할 것인지 확인합니다.

해당 명령이 삭제됩니다. 이 경우 해당 명령의 코드가 포함된 파일도 삭제됩니다. 특정 명령을 삭제하면 메뉴에서 해당 메뉴 명령만 단순히 제거하는 것이 아니라 해당 메뉴 명령에 대한 코드도 함께 삭제됩니다. 따라서 이 방법을 사용하기 전에 명령을 삭제할 것인지 반드시 확인하십시오. 해당 명령의 코드가 포함된 파일은 삭제하지 않고 명령 메뉴에서 해당 명령만 제거하려면 Configuration/Commands에서 해당 파일을 찾은 후 이 파일을 다른 폴더로 이동하면 됩니다.

4 [닫기]를 클릭합니다.

메뉴 명령 작동 방식

사용자가 메뉴 명령이 포함된 메뉴 항목이 있는 메뉴를 클릭하면 다음과 같은 이벤트가 발생합니다.

1 메뉴의 menuitem 태그에 dynamic 속성이 포함된 경우, Dreamweaver가 관련된 메뉴 명령 파일에서 getDynamicContent() 함수를 호출하여 메뉴를 채웁니다.

2 Dreamweaver가 메뉴에서 참조되는 각 메뉴 명령 파일에서 canAcceptCommand() 함수를 호출하여 해당 명령이 선택 영역에 적합한지 확인합니다.

- canAcceptCommand() 함수가 false 값을 반환하면 해당 메뉴 항목이 희미하게 표시됩니다.
- canAcceptCommand() 함수가 true 값을 반환하거나 정의되어 있지 않으면 Dreamweaver는 isCommandChecked() 함수를 호출하여 해당 메뉴 항목 옆에 체크 표시를 표시할지 여부를 결정합니다. isCommandChecked() 함수가 정의되어 있지 않으면 체크 표시가 나타나지 않습니다.

3 Dreamweaver가 setMenuText() 함수를 호출하여 메뉴에 표시할 텍스트를 결정합니다.

setMenuText() 함수가 정의되어 있지 않으면 Dreamweaver에서는 menuitem 태그에 지정되어 있는 텍스트를 사용합니다.

4 사용자가 메뉴에서 항목을 선택합니다.

5 receiveArguments() 함수가 정의된 경우 Dreamweaver가 선택된 메뉴 명령 파일에서 이 함수를 호출하여 해당 명령이 메뉴 항목으로부터 전달된 인수를 처리하도록 합니다.

참고: 동적 메뉴 항목인 경우에는 메뉴 항목의 ID가 유일한 인수로 전달됩니다.

6 commandButtons() 함수가 정의되어 있는 경우 Dreamweaver가 이 함수를 호출하여 [옵션] 대화 상자의 오른쪽에 표시할 버튼과 사용자가 이 버튼을 클릭할 때 실행할 코드를 확인합니다.

7 Dreamweaver가 메뉴 명령 파일에서 form 태그를 검색합니다.

- form 태그가 있으면 Dreamweaver에서는 windowDimensions() 함수를 호출하여 파일의 BODY 요소가 포함되는 [옵션] 대화 상자의 크기를 결정합니다.
- windowDimensions() 함수가 정의되어 있지 않으면 Dreamweaver는 자동으로 이 대화 상자의 크기를 결정합니다.

8 메뉴 명령 파일의 body 태그에 onLoad 핸들러가 있으면 Dreamweaver가 대화 상자의 표시 여부에 관계없이 관련 코드를 실행합니다. 대화 상자가 나타나지 않으면 나머지 단계는 수행되지 않습니다.

9 사용자가 대화 상자에서 옵션을 선택합니다. 사용자가 필드를 선택하면 Dreamweaver에서는 이 필드와 관련된 이벤트 핸들러를 실행합니다.

10 사용자가 commandButtons() 함수로 정의된 버튼 중 하나를 클릭합니다.

11 Dreamweaver가 클릭한 버튼과 연관된 코드를 실행합니다.

12 메뉴 명령 파일의 스크립트 중 하나가 window.close() 함수를 호출할 때까지 대화 상자가 표시됩니다.

간단한 메뉴 명령 예제

이 간단한 메뉴 명령 예제에서는 [실행 취소] 및 [다시 실행] 메뉴 명령이 동작하는 방식을 보여 줍니다. [실행 취소] 메뉴 명령은 사용자의 편집 작업 결과를 되돌립니다. [다시 실행] 항목은 [실행 취소] 작업을 되돌리고 사용자의 마지막 편집 작업 결과를 복원합니다.

이 예제를 구현하려면 메뉴 명령을 만들고, JavaScript 코드를 작성하고, Menu 폴더에 명령 파일을 배치합니다.

메뉴 명령 만들기

[실행 취소] 및 [다시 실행] 메뉴 명령이 포함된 MyMenu라는 메뉴를 만들려면 다음 HTML 메뉴 태그를 추가합니다. menus.xml의 마지막 닫기 </menu> 뒤에 태그를 추가합니다.

```
<menu name="MyMenu" id="MyMenu_Edit">
<menuitem name="MyUndo" key="Cmd+Z" file="Menus/MyMenu.htm" arguments="'undo'" id="MyMenu_Edit_Undo" />
<menuitem name="MyRedo" key="Cmd+Y" file="Menus/MyMenu.htm" arguments="'redo'" id="MyMenu_Edit_Redo" />
</menu>
```

key 속성은 사용자가 메뉴 명령을 로드하기 위해 입력할 수 있는 키보드 단축키를 정의합니다. file 속성은 명령이 로드될 때 실행되는 메뉴 명령 파일의 이름을 지정합니다. arguments 속성의 값은 receiveArguments() 함수 호출 시 Dreamweaver에서 전달하는 인수를 정의합니다.

다음 그림에서는 이러한 메뉴 명령을 보여 줍니다.

메뉴의 JavaScript 코드

사용자가 MyMenu 메뉴에서 [실행 취소] 또는 [다시 실행]을 선택하면 menuitem 태그의 file 속성에 지정된 MyMenu.htm 명령 파일이 호출됩니다. Dreamweaver Configuration/Menus 폴더에서 MyMenu.htm 명령 파일을 만들고 세 개의 메뉴 명령 API 함수, 즉 canAcceptCommand(), receiveArguments() 및 setMenuText()를 추가하여 [실행 취소] 및 [다시 실행] 메뉴 항목과 연관된 논리를 구현합니다. 다음 단원에서는 이러한 함수에 대해 설명합니다.

canAcceptCommand()

Dreamweaver에서는 MyMenu 메뉴에 있는 각 메뉴 항목에 대해 canAcceptCommand() 함수를 호출하여 각 메뉴 항목을 활성화할지 비활성화할지를 결정합니다. MyMenu.htm 파일에서 canAcceptCommand() 함수는 arguments[0]의 값을 확인하여 [다시 실행] 메뉴 항목을 처리할지 [실행 취소] 메뉴 항목을 처리할지를 결정합니다. 이 인수가 "undo"이면 canAcceptCommand() 함수는 활성자 함수 dw.canUndo()를 호출하여 반환된 값을 반환합니다. 이때 반환된 값은 true 또는 false입니다. 이와 같이 인수가 "redo"이면 canAcceptCommand() 함수는 활성자 함수 dw.canRedo()를 호출하고 그 값을 Dreamweaver에 반환합니다. canAcceptCommand() 함수가 false를 반환하면 이 함수를 호출한 메뉴 항목이 Dreamweaver에서 희미하게 표시됩니다. 다음 예제에서는 canAcceptCommand() 함수에 대한 코드를 보여 줍니다.

```
function canAcceptCommand()
{
    var selarray;
    if (arguments.length != 1) return false;
    var bResult = false;

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
    {
        bResult = dw.canUndo();
    }
    else if (whatToDo == "redo")
    {
        bResult = dw.canRedo();
    }
    return bResult;
}
```

receiveArguments()

Dreamweaver에서는 menuitem 태그에 대해 정의된 인수를 처리하기 위해 receiveArguments() 함수를 호출합니다. [실행 취소] 및 [다시 실행] 메뉴 항목의 경우, receiveArguments() 함수는 arguments[0] 인수의 값이 "undo"인지 "redo"인지에 따라 dw.undo() 함수나 dw.redo() 함수를 호출합니다. dw.undo() 함수는 이전 단계에서 사용자가 문서 윈도우, 대화 상자 또는 포커스를 가진 패널에서 수행한 작업을 실행 취소합니다. dw.redo() 함수는 실행 취소되었던 마지막 작업을 다시 실행합니다.

다음 예제 코드에서는 receiveArguments() 함수의 구문을 보여 줍니다.

```
function receiveArguments()
{
    if (arguments.length != 1) return;

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
    {
        dw.undo();
    }
    else if (whatToDo == "redo")
    {
        dw.redo();
    }
}
```

이 명령에서 receiveArguments() 함수는 인수를 처리하고 명령을 실행합니다. 더 복잡한 메뉴 명령은 다른 함수를 호출하여 명령을 실행할 수 있습니다. 예를 들어 다음 코드에서는 첫 번째 인수가 "foo"인지 확인하고, 그런 경우 doOperationX() 함수를 호출한 다음 이 함수에 두 번째 인수를 전달합니다. 첫 번째 인수가 "bar"이면 doOperationY() 함수를 호출하고 이 함수에 두 번째 인수를 전달합니다. doOperationX() 또는 doOperationY() 함수는 이 명령의 실행을 담당합니다.

```
function receiveArguments(){
    if (arguments.length != 2) return;

    var whatToDo = arguments[0];

    if (whatToDo == "foo"){
        doOperationX(arguments[1]);
    }else if (whatToDo == "bar"){
        doOperationX(arguments[1]);
    }
}
```

setMenuText()

Dreamweaver에서는 setMenuText() 함수를 호출하여 메뉴 항목에 표시할 텍스트를 결정합니다. setMenuText() 함수를 정의하지 않으면 menuitem 태그의 name 속성에 지정한 텍스트가 사용됩니다.

setMenuText() 함수는 Dreamweaver에서 전달하는 인수, 즉 arguments[0]의 값을 확인합니다. 이 인수의 값이 "undo"이면 dw.getUndoText() 함수가 호출되고 인수의 값이 "redo"이면 dw.getRedoText() 함수가 호출됩니다. dw.getUndoText() 함수는 실행 취소될 작업을 지정하는 텍스트를 반환합니다. 예를 들어 사용자가 다시 실행 작업을 여러 번 실행할 경우 dw.getUndoText()는 "실행 취소 : 소스 편집"이라는 메뉴 텍스트를 반환할 수 있습니다. 마찬가지로 dw.getRedoText() 함수도 다시 실행될 작업을 지정하는 텍스트를 반환합니다. 사용자가 실행 취소 작업을 여러 번 실행할 경우 dw.RedoText() 함수는 "다시 실행 : 소스 편집"이라는 메뉴 텍스트를 반환할 수 있습니다.

다음 예제 코드에서는 setMenuText() 함수의 구문을 보여 줍니다.

```
function setMenuText()
{
    if (arguments.length != 1) return "";

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
        return dw.getUndoText();
    else if (whatToDo == "redo")
        return dw.getRedoText();
    else return "";
}
```

메뉴 폴더에 명령 파일 저장

메뉴에 [실행 취소] 및 [다시 실행] 메뉴 항목을 구현하려면 Dreamweaver Configuration/Menus 폴더나 직접 만든 하위 폴더에 MyMenu.htm 명령 파일을 저장해야 합니다. 이 MyMenu.htm 명령 파일의 위치는 menuitem 태그에 지정한 위치와 일치해야 합니다. Dreamweaver에서 이 파일에 액세스할 수 있도록 하려면 Dreamweaver를 다시 시작하거나 Extension을 다시 로드하십시오. Extension을 다시 로드하는 방법에 대한 자세한 내용은 73페이지의 “Extension 다시 로드”를 참조하십시오.

메뉴 명령을 실행하려면 해당 메뉴 항목이 활성화되어 있을 때 이 항목을 선택합니다. Dreamweaver에서는 144페이지의 “메뉴 명령 작동 방식”에 설명된 것처럼 명령 파일에 있는 함수를 호출합니다.

동적 메뉴 예제

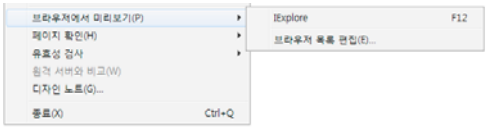
이 예제에서는 사용 가능한 브라우저 목록을 표시하는 [Preview In Browser]라는 하위 메뉴를 Dreamweaver에 구현합니다. 또한 이 예제는 [사이트] 패널의 현재 파일이나 선택된 파일을 사용자가 지정한 브라우저에서 엽니다. 이 동적 메뉴를 구현하려면 동적 메뉴 항목을 만들고 JavaScript 코드를 작성합니다.

동적 메뉴 항목 만들기

menus.xml 파일에 있는 다음 menu 태그는 [파일] 메뉴의 [브라우저에서 미리 보기] 하위 메뉴를 정의합니다.

```
<menu name="_Preview in Browser" id="DWMenu_File_PIB">
    <menuitem dynamic name="No Browsers Selected"
        file="Menus/MM/PIB_Dynamic.htm" arguments="'No Browsers'"
        id="DWMenu_File_PIB_Default" />
    <separator />
    <menuitem name="_Edit Browser List..." enabled="true"
        command="dw.editBrowserList()" id="DWMenu_File_PIB_EditBrowserList" />
    </menu>
```

첫 번째 menuitem 태그는 기본 메뉴 항목인 [선택된 브라우저 없음]을 정의합니다. [환경 설정] 대화 상자에서 [브라우저에서 미리 보기] 항목에 사용할 브라우저를 지정하지 않은 경우 이 기본 메뉴 항목이 하위 메뉴에 나타납니다. Microsoft Internet Explorer 브라우저를 지정한 경우에는 하위 메뉴가 다음 그림과 같이 나타납니다.



첫 번째 메뉴 항목의 name 속성은 명령 파일 PIB_Dynamic.htm을 지정합니다. 이 파일에는 다음과 같은 행이 포함되어 있습니다.

```
<SCRIPT LANGUAGE="javascript" SRC="PIB_Dynamic.js"></SCRIPT>
```

script 태그에는 PIB_Dynamic.js 파일의 JavaScript 코드가 포함되어 있습니다. 이 PIB_Dynamic.js 파일은 [브라우저에서 미리 보기] 하위 메뉴와 상호 작용하는 JavaScript 코드를 제공합니다. PIB_Dynamic.htm 파일에 이 코드를 직접 저장할 수도 있습니다. 하지만, 이 코드를 별도의 파일에 저장해 두면 한 코드를 여러 명령에 포함할 수 있습니다.

동적 메뉴 항목의 JavaScript 코드

첫 번째 menuitem 태그에 dynamic 속성이 포함되어 있으므로 Dreamweaver에서는 다음 예제와 같이 PIB_Dynamic.js 파일의 getDynamicContent() 함수를 호출합니다.

```
function getDynamicContent(itemID)
{
    var browsers = null;
    var PIB = null;
    var i;
    var j=0;
    browsers = new Array();
    PIB = dw.getBrowserList();

    for (i=0; i<PIB.length; i=i+2)
    {
        browsers[j] = new String(PIB[i]);

        if (dw.getPrimaryBrowser() == PIB[i+1])
            browsers[j] += "\tF12";
        else if (dw.getSecondaryBrowser() == PIB[i+1])
            browsers[j] += "\tCmd+F12";

        browsers[j] += ";id='"+escQuotes(PIB[i])+"'";

        if (itemID == "DWPopup_PIB_Default")
            browsers[j] = MENU_strPreviewIn + browsers[j];

        j = j+1;
    }
    return browsers;
}
```

getDynamicContent() 함수는 dw.getBrowserList() 함수를 호출하여 Dreamweaver [환경 설정] 대화 상자의 [브라우저에서 미리 보기] 섹션에 지정되어 있는 브라우저 이름의 배열을 가져옵니다. 이 배열에는 각 브라우저의 이름과 실행 파일 경로가 포함되어 있습니다. 다음으로, getDynamicContents() 함수는 배열 (i=0; i<PIB.length; i=i+2)의 각 항목에 대해 해당 브라우저의 이름(PIB[i])을 browsers(browsers[j] = new String(PIB[i]);)라는 두 번째 배열로 이동합니다. 해당 브라우저가 기본 또는 보조 브라우저로 지정되어 있으면 이 함수는 키보드 단축키를 호출하는 키보드 단축키 이름을 추가합니다. 그런 다음 이 함수는 ";id="라는 문자열과 단일 인용 부호로 묶은 브라우저 이름(예: ;id='iexplore')을 차례로 추가합니다. itemID 인수가 "DWPopup_PIB_Default"이면 이 함수는 배열 항목의 앞에 Preview in이라는 문자열을 접두어로 사용합니다. getDynamicContent() 함수는 [환경 설정] 대화 상자에 있는 각 브라우저에 대한 항목을 만든 후, browsers 배열을 Dreamweaver에 반환합니다. 브라우저를 선택하지 않으면 이 함수는 null을 반환하고 해당 메뉴에는 [선택된 브라우저 없음]이 표시됩니다.

canAcceptCommand()

그런 다음 Dreamweaver에서는 file 속성을 사용하여 명령 파일을 참조하는 각 menuitem 태그에 대해 canAcceptCommand() 함수를 호출합니다. canAcceptCommand() 함수가 false를 반환하면 해당 메뉴 항목은 희미하게 표시됩니다. canAcceptCommand() 함수가 true를 반환하면 Dreamweaver에서 해당 메뉴 항목이 활성화됩니다. 이 함수가 true를 반환하거나 정의되어 있지 않으면 해당 메뉴 항목 옆에 체크 표시를 표시할지 여부를 결정하기 위해 isCommandChecked() 함수가 호출됩니다.
isCommandChecked() 함수가 정의되어 있지 않으면 체크 표시가 나타나지 않습니다.

```
function canAcceptCommand()
{
    var PIB = dw.getBrowserList();

    if (arguments[0] == 'primary' || arguments[0] == 'secondary')
        return havePreviewTarget();

    return havePreviewTarget() && (PIB.length > 0);
}
```

PIB_Dynamic.js 파일에 있는 canAcceptCommand() 함수는 [환경 설정] 대화 상자에서 만든 브라우저 목록을 다시 검색합니다. 그런 다음 첫 번째 인수(arguments[0])가 기본 또는 보조인지 확인합니다. 첫 번째 인수가 기본이거나 보조이면 havePreviewTarget() 함수에서 반환된 값을 반환합니다. 첫 번째 인수가 기본 또는 보조가 아니면 havePreviewTarget() 함수 호출을 테스트하고 브라우저가 지정되었는지 테스트합니다(PIB.length > 0). 두 테스트 모두 true이면 이 함수는 true를 반환합니다. 두 테스트 중 하나 또는 둘 모두가 false이면 이 함수는 false 값을 반환합니다.

havePreviewTarget()

havePreviewTarget() 함수는 브라우저에 표시할 유효한 대상이 Dreamweaver에 있는 경우 true 값을 반환하는 사용자 정의 함수입니다. 유효한 대상은 [사이트] 패널에 있는 문서 또는 선택된 파일 그룹입니다. 다음 예제에서는 havePreviewTarget() 함수의 구문을 보여 줍니다.

```
function havePreviewTarget()
{
    var bHavePreviewTarget = false;

    if (dw.getFocus(true) == 'site')
    {
        if (site.getFocus() == 'remote')
        {
            bHavePreviewTarget = site.getRemoteSelection().length > 0 &&
                                        site.canBrowseDocument();
        }
        else if (site.getFocus() != 'none')
        {
            var selFiles = site.getSelection();

            if (selFiles.length > 0)
            {
                var i;

                bHavePreviewTarget = true;

                for (i = 0; i < selFiles.length; i++)
                {
                    var selFile = selFiles[i];

                    // For server connections, the files will
                    // already be remote URLs.

                    if (selFile.indexOf("://") == (-1))
                    {
                        var urlPrefix = "file:///";
                        var strTemp = selFile.substr(urlPrefix.length);

                        if (selFile.indexOf(urlPrefix) == -1)
                            bHavePreviewTarget = false;
                        else if (strTemp.indexOf("/") == -1)
                            bHavePreviewTarget = false;
                        else if (!DWfile.exists(selFile))
                            bHavePreviewTarget = false;
                        else if (DWfile.getAttributes(selFile).indexOf("D") != -1)
                            bHavePreviewTarget = false;
                    }
                    else
                    {
```

```
                        bHavePreviewTarget = true;
                    }
                }
            }
        }
    }
    else if (dw.getFocus() == 'document' ||
        dw.getFocus() == 'textView' || dw.getFocus("true") == 'html' )
    {
        var dom = dw.getDocumentDOM('document');
        if (dom != null)
        {
            var parseMode = dom.getParseMode();
            if (parseMode == 'html' || parseMode == 'xml')
                bHavePreviewTarget = true;
        }
    }

    return bHavePreviewTarget;
}
```

havePreviewTarget() 함수는 값 bHavePreviewTarget을 false로 설정하며 이 값은 기본 반환값이 됩니다. 이 함수는 응용 프로그램의 어떤 부분이 현재 입력 포커스를 가지고 있는지 확인하기 위해 dw.getFocus() 함수를 호출하여 두 가지의 기본 테스트를 수행합니다. 첫 번째 테스트에서는 [사이트] 패널에 포커스가 있는지 확인합니다(if (dw.getFocus(true) == 'site')). [사이트] 패널에 포커스가 없으면 두 번째 테스트에서는 문서(dw.getFocus() == 'document'), [텍스트] 뷰(dw.getFocus() == 'textView') 또는 코드 관리자(dw.getFocus("true") == 'html')에 포커스가 있는지 확인합니다. 두 테스트 모두 true가 아니면 반환값은 false입니다.

[사이트] 패널에 포커스가 있으면 이 함수는 뷰 설정이 [원격] 뷰인지 확인합니다. 뷰 설정이 [원격] 뷰이면 이 함수는 원격 파일이 있는 경우(site.getRemoteSelection().length > 0) bHavePreviewTarget을 true로 설정합니다. 이 원격 파일은 브라우저에서 열 수 있습니다(site.canBrowseDocument()). 뷰 설정이 [원격] 뷰가 아니고 [없음]도 아니면 이 함수는 file:/// URL의 형식으로 선택된 파일 목록을 가져 옵니다(var selFiles = site.getSelection();).

선택된 목록의 각 항목에 대해 이 함수는 "://"라는 문자열이 존재하는지 테스트합니다. 이 문자열이 검색되지 않으면 코드는 해당 목록 항목에 대해 일련의 테스트를 수행합니다. 항목의 형식이 file:/// URL이 아니면(if (selFile.indexOf(urlPrefix) == -1)) 반환값을 false로 설정합니다. file:/// 접두어 뒤의 나머지 문자열에 슬래시(/)가 포함되어 있지 않으면(if (strTemp.indexOf("/") == -1)) 이 함수는 반환값을 false로 설정합니다. 해당 파일이 존재하지 않으면(else if (!DWfile.exists(selFile))) 이 함수는 반환값을 false로 설정합니다. 마지막으로, 이 함수는 지정된 파일이 폴더인지 확인합니다(else if (DWfile.getAttributes(selFile).indexOf("D") != -1)). selfile이 폴더이면 이 함수는 false를 반환합니다. 그렇지 않고 대상이 파일이면 이 함수는 bHavePreviewTarget를 true 값으로 설정합니다.

문서, [텍스트] 뷰 또는 코드 관리자에 입력 포커스가 있으면(else if (dw.getFocus() == 'document' || dw.getFocus() == 'textView' || dw.getFocus("true") == 'html')) 이 함수는 DOM을 가져와서 이 문서가 HTML인지 또는 XML 문서인지 확인합니다. 만약 문서가 HTML이거나 XML 문서이면 이 함수는 bHavePreviewTarget을 true로 설정합니다. 마지막으로, 이 함수는 bHavePreviewTarget에 저장된 값을 반환합니다.

receiveArguments()

Dreamweaver에서는 명령이 메뉴 항목에서 전달되는 인수를 처리하도록 하기 위해 receiveArguments() 함수를 호출합니다. [브라우저에서 미리 보기] 메뉴의 경우, receiveArguments() 함수는 사용자가 선택하는 브라우저를 호출합니다. 다음 예제에서는 receiveArguments() 함수의 구문을 보여 줍니다.

```
function receiveArguments()
{
    var whichBrowser = arguments[0];
    var theBrowser = null;
    var i=0;
    var browserList = null;
    var result = false;

    if (havePreviewTarget())
    {
        // Code to check if we were called from a shortcut key
        if (whichBrowser == 'primary' || whichBrowser == 'secondary')
        {
            // get the path of the selected browser
            if (whichBrowser == 'primary')
            {
                theBrowser = dw.getPrimaryBrowser();
            }
            else if (whichBrowser == 'secondary')
            {
                theBrowser = dw.getSecondaryBrowser();
            }

            // Match the path with the name of the corresponding browser
            // that appears in the menu.
            browserList = dw.getBrowserList();
            while(i < browserList.length)
            {
                if (browserList[i+1] == theBrowser)
                    theBrowser = browserList[i];
                i+=2;
            }
        }
        else
            theBrowser = whichBrowser;
        // Only launch the browser if we have a valid browser selected.
        if (theBrowser != "file:///" && typeof(theBrowser) != "undefined" &&
        theBrowser.length > 0)
        {
            if (dw.getFocus(true) == 'site')
            {
                // Only get the first item of the selection because
                // browseDocument() can't take an array.
                //dw.browseDocument(site.getSelection()[0],theBrowser);
                site.browseDocument(theBrowser);
            }
            else
                dw.browseDocument(dw.getDocumentPath('document'),theBrowser);
```

```
        }
        else
        {
            // Otherwise, F12 or Ctrl+F12 was pressed, ask if the user wants
            // to specify a primary or secondary browser now.
            if (whichBrowser == 'primary')
            {
                result = window.confirm(MSG_NoPrimaryBrowserDefined);
            }
            else if (whichBrowser == 'secondary')
            {
                result = window.confirm(MSG_NoSecondaryBrowserDefined);
            }

            // If the user clicked OK, show the prefs dialog with the browser panel.
            if (result)
                dw.showPreferencesDialog('browsers');
        }
    }
}
```

먼저, 이 함수는 변수 whichBrowser를 Dreamweaver에서 전달하는 값, 즉 arguments[0]로 설정합니다. 다른 기본값을 설정하는 것뿐만 아니라, 이 함수는 result를 기본값인 false로 설정합니다.

변수가 초기화된 후 receiveArguments() 함수는 사용자 정의된 함수 havePreviewTarget()을 호출하고 그 결과를 테스트합니다. 이 테스트의 결과가 true이면 이 함수는 사용자가 기본 또는 보조 브라우저를 선택했는지 확인합니다. 사용자가 기본 또는 보조 브라우저를 선택했으면 이 함수는 브라우저를 시작하는 실행 파일의 경로로 theBrowser 변수를 설정합니다(dw.getPrimaryBrowser() 또는 dw.getSecondaryBrowser()). 그런 다음 이 함수는 dw.getBrowsersList()에서 반환된 브라우저 목록을 검사하는 루프를 수행합니다. 목록에 있는 특정 브라우저의 경로가 기본 또는 보조 브라우저의 경로와 일치하면 이 함수는 theBrowser 변수를 browserList의 일치하는 값으로 설정합니다. 이 값에는 브라우저 이름과 브라우저를 시작하는 실행 파일의 경로가 포함되어 있습니다. havePreviewTarget()이 false를 반환하면 이 함수는 theBrowser 변수를 whichBrowser 변수의 값으로 설정합니다.

그런 다음 receiveArguments() 함수는 theBrowser 변수를 테스트하여 이 변수가 경로로 시작하지 않는지, 이 변수가 "undefined"가 아닌지, 변수 길이가 0보다 큰지 등을 확인합니다. 이러한 조건이 모두 충족되고 [사이트] 패널에 포커스가 있으면 receiveArguments() 함수는 site.browseDocument() 함수를 호출하여 선택된 브라우저를 실행하고 [사이트] 패널의 선택된 파일을 표시합니다. [사이트] 패널에 포커스가 없으면 receiveArguments() 함수는 dw.browseDocument() 함수를 호출하여 현재 문서의 경로와 해당 문서가 열릴 브라우저의 이름을 지정하는 theBrowser 변수의 값을 전달합니다.

사용자가 단축키(F12 또는 Ctrl+F12)를 눌렀지만 기본 또는 보조 브라우저가 지정되지 않았으면 사용자에게 알림 대화 상자가 표시됩니다. 사용자가 [확인]을 클릭하면 이 함수는 browsers 인수로 dw.showPreferencesDialog() 함수를 호출하여 사용자가 브라우저를 지정할 수 있도록 합니다.

메뉴 명령 API 함수

메뉴 명령 API의 사용자 정의 함수는 선택 사항입니다.

canAcceptCommand()

설명
메뉴 항목이 활성화될지 또는 희미하게 표시될지 결정합니다.

인수
{arg1}, {arg2},...{argN}}

동적 메뉴 항목인 경우, getDynamicContents() 함수가 지정하는 고유한 ID가 유일한 인수가 됩니다. 동적 메뉴 항목이 아닌 경우, arguments 속성이 menuitem 태그에 정의되어 있으면 이 속성의 값이 isCommandChecked(), 154페이지의 "isCommandChecked()", 155페이지의 "receiveArguments()" 및 155페이지의 "setMenuText()" 함수에 하나 이상의 인수로 전달됩니다. arguments 속성은 동일한 메뉴 명령을 호출하는 두 메뉴 항목을 구별하는 데 유용합니다.

참고: 동적 메뉴 항목의 경우 arguments 인수는 무시됩니다.

반환값
부울 값을 반환합니다. 항목이 활성화되면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

commandButtons()

설명
[옵션] 대화 상자의 오른쪽에 나타나는 버튼과 이 버튼을 클릭할 때의 비헤이비어를 정의합니다. 이 함수가 정의되어 있지 않으면 버튼이 나타나지 않고 메뉴 명령 파일의 body 섹션이 확장되어 전체 대화 상자를 채웁니다.

인수
없음

반환값
짝수 개의 요소가 포함된 배열을 반환합니다. 첫 번째 요소는 맨 위 버튼의 레이블이 포함된 문자열입니다. 두 번째 요소는 맨 위 버튼을 클릭할 때 발생하는 비헤이비어를 정의하는 JavaScript 코드의 문자열입니다. 나머지 요소는 동일한 방법으로 추가 버튼을 정의합니다.

예제
commandButtons() 함수의 다음 예제에서는 [확인], [취소] 및 [도움말] 버튼을 정의합니다.

```
function commandButtons(){
    return new Array("OK" , "doCommand()" , "Cancel" , ¬
    "window.close()" , "Help" , "showHelp()");
}
```

getDynamicContent()

설명
메뉴의 동적 부분에 대한 내용을 가져옵니다.

인수
menuID

menuID 인수는 해당 항목과 연관된 menuitem 태그에 있는 id 속성의 값입니다.

반환값
메뉴 항목의 이름과 고유 ID를 포함하는 문자열이 세미콜론으로 구분된 문자열 배열을 반환합니다. 이 함수가 null 값을 반환하면 메뉴는 변경되지 않습니다.

예제
getDynamicContent() 함수의 다음 예제에서는 My Menu Item 1, My Menu Item 2, My Menu Item 3 및 My Menu Item 4라는 네 개의 메뉴 항목이 들어 있는 배열을 반환합니다.

```
function getDynamicContent(){
    var stringArray= new Array();
    var i=0;
    var numItems = 4;

    for (i=0; i<numItems;i++)
        stringArray[i] = new String("My Menu Item " + i + ";¬
        id='My-MenuItem" + i + "'");

    return stringArray;
}
```

isCommandChecked()

설명

메뉴 항목 옆에 체크 표시를 표시할지 여부를 결정합니다.

인수

{arg1}, {arg2},...{argN}

동적 메뉴 항목인 경우 getDynamicContents() 함수가 지정하는 고유한 ID가 유일한 인수가 됩니다. 동적 메뉴 항목이 아닌 경우, arguments 속성이 menuitem 태그에 정의되어 있으면 이 속성의 값이 isCommandChecked(), 152페이지의 “canAcceptCommand()”, 155페이지의 “receiveArguments()” 및 155페이지의 “setMenuText()” 함수에 하나 이상의 인수로 전달됩니다. arguments 속성은 동일한 메뉴 명령을 호출하는 두 메뉴 항목을 구별하는 데 유용합니다.

참고: 동적 메뉴 항목의 경우 arguments 인수는 무시됩니다.

반환값

부울 값을 반환합니다. 체크 표시가 해당 메뉴 항목 옆에 나타나면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

예제

```
function isCommandChecked()
{
    var bChecked = false;
    var cssStyle = arguments[0];

    if (dw.getDocumentDOM() == null)
        return false;

    if (cssStyle == "(None)")
    {
        return dw.cssStylePalette.getSelectedStyle() == '';
    }
    else
    {
        return dw.cssStylePalette.getSelectedStyle() == cssStyle;
    }
        return bChecked;
}
```

receiveArguments()

설명

메뉴 항목이나 dw.runCommand() 함수로부터 전달된 인수를 처리합니다. 동적 메뉴 항목인 경우에는 동적 메뉴 항목 ID를 처리합니다.

인수

{arg1}, {arg2},...{argN}

동적 메뉴 항목인 경우 getDynamicContents() 함수가 지정하는 고유한 ID가 유일한 인수가 됩니다. 동적 메뉴 항목이 아닌 경우, arguments 속성이 menuitem 태그에 정의되어 있으면 이 속성의 값이 receiveArguments(), 152페이지의 "canAcceptCommand()", 154페이지의 "isCommandChecked()" 및 155페이지의 "setMenuText()" 함수에 하나 이상의 인수로 전달됩니다. arguments 속성은 동일한 메뉴 명령을 호출하는 두 메뉴 항목을 구별하는 데 유용합니다.

참고: 동적 메뉴 항목의 경우 arguments 인수는 무시됩니다.

반환값

없음

예제

```
function receiveArguments()
{
    var styleName = arguments[0];
    if (styleName == "(None)")
        dw.getDocumentDOM('document').applyCSSStyle('','');
    else
        dw.getDocumentDOM('document').applyCSSStyle('',styleName);
}
```

setMenuText()

설명

메뉴에 나타낼 텍스트를 지정합니다.

참고: 153페이지의 "getDynamicContent()"를 사용하는 경우에는 이 함수를 사용하지 마십시오.

인수

{arg1}, {arg2},...{argN}

arguments 속성이 menuitem 태그에 정의되어 있으면 이 속성의 값이 setMenuText(), 152페이지의 "canAcceptCommand()", 154페이지의 "isCommandChecked()" 및 155페이지의 "receiveArguments()" 함수에 하나 이상의 인수로 전달됩니다. arguments 속성은 동일한 메뉴 명령을 호출하는 두 메뉴 항목을 구별하는 데 유용합니다.

반환값

메뉴에 나타낼 문자열을 반환합니다.

예제

```
function setMenuText()
{
    if (arguments.length != 1) return "";

    var whatToDo = arguments[0];
    if (whatToDo == "undo")
        return dw.getUndoText();
    else if (whatToDo == "redo")
        return dw.getRedoText();
    else return "";
}
```

windowDimensions()

설명

이 함수는 [매개 변수] 대화 상자의 구체적 크기를 설정합니다. 이 함수를 정의하지 않으면 윈도우 크기가 자동으로 계산됩니다.

참고: 대화 상자의 크기를 640x480픽셀보다 크게 하려는 경우에만 이 함수를 정의하십시오.

인수

platform

platform 인수의 값은 사용자의 플랫폼에 따라 "macintosh" 또는 "windows"입니다.

반환값

"widthInPixels,heightInPixels" 형식의 문자열을 반환합니다.

반환된 크기는 [확인] 및 [취소] 버튼의 영역을 포함하지 않으므로 전체 대화 상자의 크기보다 작습니다. 반환된 크기에 옵션이 모두 들어가지 않으면 스크롤 막대가 나타납니다.

예제

다음 windowDimensions() 예제에서는 [매개 변수] 대화 상자의 크기를 648x520픽셀로 설정합니다.

```
function windowDimensions(){
    return "648,520";
}
```

11장: 툴바

Adobe Dreamweaver 툴바를 만들려면 툴바를 정의하는 파일을 만들고 이 파일을 Configuration/Toolbars 폴더에 저장하면 됩니다.

다음 표에서는 툴바를 만들 때 사용하는 파일을 보여 줍니다.

경로	파일	설명
Configuration/Toolbars/	toolbars.xml	이 파일을 편집하여 툴바의 내용을 변경합니다.
Configuration/Toolbars/	**newtoolbar**.xml	이 파일을 만들어 툴바를 만듭니다.
Configuration/Toolbars/	**imagefile**.gif	툴바 컨트롤에 대한 아이콘 이미지입니다.
Configuration/Commands/	MyCommand.htm	툴바 항목과 연관된 명령 파일입니다.

툴바 동작 방식

툴바는 Dreamweaver 주 Configuration 폴더의 Toolbars 폴더에 저장된 XML 및 이미지 파일을 사용하여 정의됩니다. 기본 Dreamweaver 툴바는 Configuration/Toolbars 폴더의 toolbars.xml 파일에 저장되어 있습니다. Dreamweaver를 시작하면 Toolbars 폴더에 있는 툴바 파일이 모두 로드됩니다. 새 툴바를 추가하려면 기본 toolbars.xml 파일을 수정하는 대신, 추가할 파일을 Toolbars 폴더에 복사하면 됩니다.

툴바 XML 파일은 하나 이상의 툴바와 해당 툴바 항목을 정의합니다. 툴바는 버튼, 텍스트 상자, 팝업 메뉴 등과 같은 항목의 목록입니다. 툴바의 각 항목은 사용자가 툴바에서 액세스할 수 있는 각 컨트롤을 나타냅니다.

누름 버튼 및 팝업 메뉴와 같은 일부 종류의 툴바 컨트롤에는 컨트롤과 연관된 아이콘 이미지가 있습니다. 아이콘 이미지는 Toolbars 폴더의 images 폴더에 저장됩니다. 이미지 형식은 Dreamweaver에서 렌더링할 수 있는 어느 형식이나 가능하지만 대개는 GIF 또는 JPEG 파일 형식입니다. Adobe에서 작성한 툴바의 이미지는 Toolbars/images/MM 폴더에 저장됩니다.

메뉴의 경우와 마찬가지로 개별 툴바 항목의 기능을 항목 속성 또는 명령 파일을 통해 지정할 수 있습니다. Adobe에서 작성한 툴바 명령 파일은 Toolbars/MM 폴더에 저장됩니다.


툴바 API는 메뉴 명령 API와 호환되므로, 툴바 컨트롤은 메뉴 명령 파일을 다시 사용할 수 있습니다.

메뉴와 달리 툴바 항목은 해당 항목을 사용하는 툴바와 독립적으로 정의할 수 있습니다. 이러한 유연성이 있으므로 itemref 태그를 사용하여 툴바 항목을 여러 툴바에서 사용할 수 있습니다.

Dreamweaver에서 툴바를 처음 로드할 때 툴바의 표시 여부와 위치는 툴바 정의에 따라 설정됩니다. 그 후에는 툴바의 표시 여부 및 위치가 레지스트리(Windows) 또는 Dreamweaver 환경 설정 파일(Macintosh)에 저장되고 해당 위치에서 복원됩니다.

툴바 동작 방식

Windows의 경우, Dreamweaver MX 툴바는 일반적으로 표준 Windows 도구 모음과 동일하게 동작합니다. Dreamweaver MX 툴바의 특성은 다음과 같습니다.

- 툴바를 드래그 앤 드롭하여 툴바를 결합하거나, 결합을 해제하거나, 다른 툴바와의 상대 위치를 변경할 수 있습니다.
- 툴바를 프레임 윈도우의 위 또는 아래에 가로로 결합할 수 있습니다.
- 툴바의 크기는 고정적입니다. 즉, 툴바는 해당 컨테이너의 크기가 줄어들거나 다른 툴바가 옆에 놓여도 크기가 줄어들지 않습니다.

- [보기] > [도구 막대] 메뉴에서 툴바를 표시하거나 숨길 수 있습니다.
- 툴바는 겹쳐 놓을 수 없습니다.
- 툴바를 드래그하는 동안에는 툴바의 윤곽선만 나타납니다.

Macintosh의 경우, 툴바는 항상 문서 윈도우에 첨부됩니다. 툴바를 메뉴에 표시하거나 메뉴에서 숨길 수는 있지만 드래그 앤 드롭하거나, 다시 정렬하거나, 결합을 해제할 수는 없습니다.

모든 Dreamweaver 문서 윈도우를 단일 부모 프레임 내에 통합하는 Dreamweaver 작업 영역에서 툴바를 작업 영역 프레임에 결합할지 문서 윈도우에 결합할지를 지정할 수 있습니다.

Dreamweaver 작업 영역 프레임에 결합하는 툴바의 경우, 각 툴바마다 인스턴스가 하나씩 존재합니다. 이 경우 툴바는 맨 앞에 있는 문서에 대해 작동합니다. Dreamweaver 작업 영역에서 위쪽, 아래쪽 또는 [삽입] 툴바의 왼쪽이나 오른쪽에 툴바를 결합할 수 있습니다. Dreamweaver 작업 영역 프레임에 첨부된 툴바는 문서 윈도우가 없어도 자동으로 비활성화되지 않습니다. 열려 있는 문서가 없을 때 툴바의 활성화 여부는 툴바 항목에 따라 결정됩니다.

툴바가 문서 윈도우에 결합된 채로 있으면 각 윈도우의 인스턴스는 하나만 존재하게 됩니다. 한 문서 윈도우에 첨부된 툴바는 해당 윈도우가 맨 앞에 있는 문서가 아니면 완전히 비활성화되고, 해당 윈도우가 맨 앞에 나타날 때 툴바 업데이트 핸들러를 모두 다시 실행합니다.

문서 윈도우와 Dreamweaver 작업 영역 프레임 간에는 툴바를 드래그 앤 드롭할 수 없습니다.

툴바 명령의 작동 방식

Dreamweaver에서 툴바를 그릴 때는 다음 이벤트가 발생합니다.

1. Dreamweaver가 각 툴바 컨트롤 항목에 대해 file 속성이 있는지 여부를 확인합니다.
2. file 속성이 있으면 Dreamweaver가 canAcceptCommand() 함수를 호출하여 문서의 현재 컨텍스트에서 해당 컨트롤을 활성화할지 여부를 결정합니다.

   예를 들어 Dreamweaver 툴바에 있는 [문서 제목] 텍스트 상자의 경우, canAcceptCommand() 함수는 현재 DOM이 있는지와 현재 문서가 HTML 파일인지를 확인합니다. 이 두 조건이 모두 해당되는 경우 이 함수는 true를 반환하며 툴바에서 해당 텍스트 상자가 활성화됩니다.
3. file 속성이 있는 경우 checked, command, DOMRequired, enabled, script, showif, update 및 value 속성이 지정되어 있으면 이러한 속성은 무시됩니다.
4. file 속성이 없는 경우 툴바 컨트롤 항목에 대해 설정된 checked, command, DomRequired 등의 속성이 처리됩니다.

   특정 항목 태그 속성에 대한 자세한 내용은 169페이지의 “항목 태그 속성”을 참조하십시오.
5. Dreamweaver가 update 속성으로 지정된 업데이트 주기에 따라 getCurrentValue() 함수를 호출하여 해당 컨트롤에 대해 표시할 값을 결정합니다.
6. 사용자가 툴바에서 항목을 선택합니다.
7. Dreamweaver가 receiveArguments() 함수를 호출하여 툴바 항목의 arguments 속성이 지정하는 인수를 처리합니다.

툴바 명령 API에 있는 특정 함수의 용도에 대한 자세한 내용은 173페이지의 “툴바 명령 API 함수”를 참조하십시오.

간단한 툴바 명령 파일

이 간단한 예제에서는 Dreamweaver 문서 툴바에 표시되는 [제목] 텍스트 상자 항목을 구현합니다. 사용자는 이 텍스트 상자 항목을 사용하여 현재 Dreamweaver 문서의 이름을 입력할 수 있습니다. 다음 단계에 따라 이 툴바 예제를 구현할 수 있습니다.

텍스트 상자 만들기

Dreamweaver에 툴바를 추가하려면 해당 툴바 정의가 포함된 XML 파일을 Dreamweaver Configuration 폴더 내의 Toolbars 폴더에 배치합니다.

다음 그림에서는 [제목] 텍스트 상자를 보여 줍니다.

다음 editcontrol 툴바 항목은 [Title:]이라는 텍스트 상자를 정의합니다.

```
<EDITCONTROL ID="DW_SetTitle"
    label="Title: "
    tooltip="Document Title"
    width="150"
    file="Toolbars/MM/EditTitle.htm"/>
```

tooltip 속성은 사용자가 해당 텍스트 상자 위에 마우스 포인터를 놓으면 도구 설명 상자에 'Document Title'이 표시되도록 합니다. width 속성은 해당 필드의 크기(픽셀)를 지정합니다. file 속성은 해당 텍스트 상자에 대해 동작하는 JavaScript 함수가 EditTitle.htm 파일에 포함되도록 지정합니다. Dreamweaver 문서 툴바의 전체 정의를 보려면 toolbars.xml 파일에서 기본 툴바(id="DW_Toolbar_Main")를 참조하십시오.

텍스트 상자에 대한 JavaScript 코드

사용자가 텍스트 상자와 상호 작용하면 Toolbars/MM 폴더의 EditTitle.htm 명령 파일이 호출됩니다. 이 파일에는 [Title] 텍스트 상자에 대해 동작하는 세 개의 JavaScript 함수가 포함되어 있습니다. 이 세 함수는 canAcceptCommand(), receiveArguments() 및 getCurrentValue()입니다.

canAcceptCommand(): 툴바 항목 활성화

canAcceptCommand() 함수는 현재 DOM(문서 객체 모델)이 있는지 여부와 문서가 HTML로 파싱되는지 여부를 확인하는 한 줄의 코드로 구성되어 있습니다. 이 함수는 그러한 테스트의 결과를 반환합니다. 조건이 true이면 Dreamweaver 툴바에서 해당 텍스트 상자 항목이 활성화됩니다. 이 함수가 false를 반환하면 해당 항목은 비활성화됩니다.

이 함수는 다음과 같습니다.

```
function canAcceptCommand()
{
    return (dw.getDocumentDOM() != null && dw.getDocumentDOM().getParseMode() == 'html');
}
```

receiveArguments(): 제목 설정

사용자가 [제목] 텍스트 상자에 값을 입력하고 Enter 키를 누르거나 해당 컨트롤에서 다른 곳으로 포커스를 옮기면 Dreamweaver에서는 다음 예제와 같은 receiveArguments() 함수를 호출합니다.

이 함수는 다음과 같습니다.

```
function receiveArguments(newTitle)
{
    var dom = dw.getDocumentDOM();
    if (dom)
        dom.setTitle(newTitle);
}
```

Dreamweaver에서는 receiveArguments() 함수에 사용자가 입력한 값인 newTitle을 전달합니다. receiveArguments() 함수는 먼저 현재 DOM이 있는지 확인합니다. 현재 DOM이 있으면 receiveArguments() 함수는 newTitle을 dom.setTitle() 함수에 전달하여 새 문서 제목을 설정합니다.

getCurrentValue(): 제목 가져오기

onEdit 이벤트 핸들러의 기본 업데이트 빈도에 따라 결정된 업데이트 주기가 발생할 때마다 Dreamweaver에서는 getCurrentValue() 함수를 호출하여 해당 컨트롤에 표시할 값을 결정합니다. [Title] 텍스트 편집 컨트롤은 update 속성이 없으므로 onEdit 핸들러의 기본 업데이트 빈도에 따라 업데이트 빈도가 결정됩니다.

다음 getCurrentValue() 함수는 [Title] 텍스트 상자에 대해 현재 제목을 가져와 반환하는 JavaScript API(응용 프로그램 프로그래밍 인터페이스) 함수 dom.getTitle()을 호출합니다.

이 함수는 다음과 같습니다.

```
function getCurrentValue()
{
    var title = "";
    var dom = dw.getDocumentDOM();
    if (dom)
        title = dom.getTitle();
    return title;
}
```

사용자가 문서의 제목을 입력하기 전까지 getTitle() 함수는 '무제 문서'라는 텍스트를 반환하며 이 텍스트가 텍스트 상자에 나타납니다. 사용자가 제목을 입력한 후 getTitle() 함수는 해당 값을 반환하며 Dreamweaver에서는 이 값을 새 문서 제목으로 표시합니다.

[Title] 텍스트 상자에 대한 JavaScript 함수가 포함된 전체 HTML 파일을 보려면 Toolbars/MM 폴더의 EditTitle.htm 파일을 참조하십시오.

MM 폴더는 Adobe 파일용으로 예약되어 있습니다. Toolbars 폴더에 다른 폴더를 하나 만들고 이 폴더에 JavaScript 코드를 저장합니다.

툴바 정의 파일

툴바는 단순히 라디오 버튼, 체크 버튼, 편집 상자 및 다른 툴바 항목을 나열한 것으로, 경우에 따라separator 태그로 구분될 수 있습니다. 각 툴바 항목은 예를 들어 체크 상자나 편집 상자에 대한 itemref 태그를 사용한 항목 참조, separator 태그를 사용한 분리 기호 또는 전체 툴바 항목 정의가 될 수 있습니다. 자세한 내용은 164페이지의 “툴바 항목 태그”를 참조하십시오.

각 툴바 정의 파일은 다음 선언으로 시작합니다.

```
<?xml version="1.0" encoding="optional_encoding"?>
<!DOCTYPE toolbarset SYSTEM "-//Macromedia//DWExtension toolbar 5.0">
```

인코딩을 생략하면 운영 체제의 기본 인코딩이 사용됩니다.

이 선언 다음의 파일 내용은 단일 toolbarset 태그로 구성됩니다. 이 태그에는 toolbar, itemref, separator, include 및 **itemtype** 태그가 개수에 제한 없이 포함됩니다. 여기서 **itemtype**은 button, checkbutton, radiobutton, menubutton, dropdown, combobox, editcontrol 또는 colorpicker입니다. 다음 예제는 toolbars.xml 파일에서 일부 발췌한 것으로 툴바 파일의 태그 계층 구조를 보여줍니다. 이 예제에서는 툴바 항목 속성에 대한 설명을 줄임표(. . .)로 대체합니다.

```
<?xml version="1.0"?>
<!DOCTYPE toolbarset SYSTEM "-//Adobe//DWExtension toolbar 10.0">
<toolbarset>

<!-- main toolbar -->
    <toolbar id="DW_Toolbar_Main" label="Document">
        <radiobutton id="DW_CodeView" . . ./>
        <radiobutton id="DW_SplitView" . . ./>
        <radiobutton id="DW_DesignView" . . ./>
        <separator/>
        <checkbutton id="DW_LiveDebug" . . ./>
        <checkbutton id="DW_LiveDataView" . . ./>
        <separator/>
        <editcontrol id="DW_SetTitle" . . ./>
        <menubutton id="DW_FileTransfer" . . ./>
        <menubutton id="DW_Preview" . . ./>
        <separator/>
        <button id="DW_DocRefresh" . . ./>
        <button id="DW_Reference" . . ./>
        <menubutton id="DW_CodeNav" . . ./>
        <menubutton id="DW_ViewOptions" . . ./>
    </toolbar>
</toolbarset>
```

각 툴바 태그에 대한 설명이 아래에 나와 있습니다.

<toolbar>

설명

툴바를 정의합니다. Dreamweaver에서는 항목과 분리 기호를 왼쪽에서 오른쪽으로 지정된 순서대로 표시하고 자동으로 배치합니다. 툴바 파일에서는 항목 사이의 간격을 지정하지 않지만 특정 항목 종류의 폭은 지정할 수 있습니다.

속성

id, label, {container}, {initiallyVisible}, {initialPosition}, {relativeTo}

- id="***unique_id***" - 필수 사항입니다. 식별자 문자열은 특정 파일 내에서 또는 해당 파일에 포함된 모든 파일에서 고유해야 합니다. 툴바를 조작하는 JavaScript API 함수는 ID를 통해 툴바를 참조합니다. JavaScript API 함수에 대한 자세한 내용은 **Dreamweaver API 참조** 설명서를 참조하십시오. 동일한 파일에 포함된 두 툴바의 ID가 동일하면 Dreamweaver에서 오류가 표시됩니다.
- label="string" - 필수 사항입니다. label 속성은 Dreamweaver에서 사용자에게 표시되는 문자열인 레이블을 지정합니다. 레이블은 [보기] > [도구 막대] 메뉴에 나타나며 부동 툴바의 경우에는 툴바의 제목 막대에 나타납니다.
- container="mainframe" 또는 "document" - 기본값은 "mainframe"입니다. Windows의 Dreamweaver 작업 영역에서 툴바가 결합될 위치를 지정합니다. container가 "mainframe"으로 설정되어 있으면 툴바는 바깥쪽 작업 영역 프레임에 나타나며 맨 앞의 문서에 대해 동작합니다. container가 "document"로 설정되어 있으면 툴바는 각 문서 윈도우에 나타납니다. Macintosh에서는 모든 툴바가 각 문서 윈도우에 나타납니다.
- initiallyVisible="true" 또는 "false" - 이 태그는 Dreamweaver에서 Toolbars 폴더의 툴바를 처음 로드할 때 툴바가 화면에 표시될지 여부를 지정합니다. 처음 로드한 후부터는 사용자가 화면 표시 여부를 제어합니다. Dreamweaver를 종료할 때는 현재 상태가 시스템 레지스트리(Windows) 또는 Dreamweaver 환경 설정 파일(Macintosh)에 저장됩니다. Dreamweaver를 다시 시작하면 레지스트리 또는 환경 설정 파일에서 설정이 복원됩니다. dom.getToolbarVisibility() 및 dom.setToolbarVisibility() 함수를 사용하여 툴바의 화면 표시 여부를 조작할 수 있습니다. 자세한 내용은 **Dreamweaver API 참조** 설명서를 참조하십시오. initiallyVisible 속성을 설정하지 않을 경우 기본값은 true입니다.
- initialPosition="top", "below" 또는 "floating" - Dreamweaver에서 처음으로 툴바를 로드할 때의 초기 툴바 위치를 다른 툴바에 대한 상대 위치로 지정합니다. intialPosition의 사용 가능한 값은 다음 목록에 설명되어 있습니다.

- top - 이 값이 기본 위치이므로 해당 툴바가 문서 윈도우의 위쪽에 나타납니다. 특정 윈도우 유형에 대해 이 속성이 top으로 지정된 툴바가 여러 개이면 툴바는 Dreamweaver에 로드되는 순서대로 표시됩니다. 하지만 툴바가 서로 다른 파일에 있는 경우에는 툴바가 로드되는 순서를 예측할 수 없습니다.
- below - 툴바가 relativeTo 속성에 지정된 툴바의 바로 아래 행 처음 부분에 표시됩니다. relativeTo 툴바를 찾지 못하면 Dreamweaver에서 오류가 보고됩니다. 이 속성이 동일한 툴바에 대해 상대적인 below로 지정된 툴바가 여러 개이면 툴바는 Dreamweaver에 로드되는 순서대로 표시됩니다. 하지만 툴바가 서로 다른 파일에 있는 경우에는 툴바가 로드되는 순서를 예측할 수 없습니다.
- floating - 툴바가 처음에는 윈도우에 결합되지 않고 문서 위에 부동 툴바로 표시됩니다. 이때 다른 부동 툴바와 겹치지 않도록 자동으로 툴바 위치가 조절됩니다. Macintosh에서 floating은 top과 같은 방식으로 처리됩니다.

  initiallyVisible 속성과 마찬가지로, initialPosition 속성도 Dreamweaver에서 처음으로 툴바를 로드할 때에만 적용됩니다. 그 후에는 툴바의 위치가 레지스트리 또는 Dreamweaver 환경 설정 파일에 저장됩니다. dom.setToolbarPosition() 함수를 사용하여 툴바의 위치를 다시 설정할 수 있습니다. dom.setToolbarPosition() 함수에 대한 자세한 내용은 **Dreamweaver API 참조** 설명서를 참조하십시오.

  initialPosition 속성을 지정하지 않으면 툴바는 Dreamweaver에 로드되는 순서대로 표시됩니다.
- relativeTo="***toolbar_id***" - 이 속성은 initialPosition 속성이 below로 지정된 경우에 필수 사항입니다. 그 외의 경우에는 무시됩니다. 이 툴바가 배치될 지점 위에 있는 툴바의 ID를 지정합니다.

내용

toolbar 태그에는 button, combobox, dropdown 등과 같은 개별 항목 정의뿐만 아니라, include, itemref 및 separator 태그도 포함됩니다. 사용자가 지정할 수 있는 항목 정의에 대한 자세한 내용은 164페이지의 “툴바 항목 태그”를 참조하십시오.

컨테이너

toolbarset 태그

예제

```
<toolbar id="MyDWedit_toolbar" label="Edit">
```

<include/>

설명

현재 파일의 로드를 계속하기 전에 지정된 파일에서 툴바 항목을 로드합니다. 포함된 파일에 정의되어 있는 툴바 항목을 현재 파일에서 참조할 수 있습니다. 파일에서 다른 파일을 재귀적으로 포함하려고 하면 오류 메시지가 표시되고 재귀적 포함이 무시됩니다. 포함 파일의 툴바 항목을 현재 파일에서 참조로 사용할 수는 있지만 포함 파일의 모든 toolbar 태그는 건너뜁니다.

속성

- 포함할 툴바 XML 파일의 file 경로(Toolbars 폴더에 대한 상대 경로)

내용

없음

컨테이너

toolbar 태그 또는 toolbarset 태그

예제

```
<include file="mine/editbar.xml"/>
```

<itemtype/>

설명
한 개의 툴바 항목을 정의합니다. 툴바 항목에는 버튼, 라디오 버튼, 체크 버튼, 콤보 상자, 팝업 메뉴 등이 포함됩니다. 정의할 수 있는 툴바 항목 종류의 목록은 164페이지의 “툴바 항목 태그”를 참조하십시오.

속성
속성은 정의하는 항목에 따라 달라집니다. 툴바 항목에 지정할 수 있는 속성의 전체 목록은 169페이지의 “항목 태그 속성”을 참조하십시오.

내용
없음

컨테이너
toolbar 태그 또는 toolbarset 태그

예제
```
<button id="strikeout_button" .../>
```

<itemref/>

설명
이전 툴바의 내부 또는 모든 툴바의 외부에 정의되었던 툴바 항목을 참조하여 현재 툴바에 포함합니다.

속성
id, {showIf}

- id="***id_reference***" - 필수 사항입니다. 이전에 정의되었거나 파일에 포함된 항목의 ID여야 합니다. Dreamweaver에서는 전방 참조를 허용하지 않습니다. 툴바 항목 태그가 정의되지 않은 ID를 참조하면 오류가 보고되고 해당 참조는 무시됩니다.
- showIf="***script***" - 지정된 스크립트가 true 값을 반환하는 경우에만 이 항목이 툴바에 나타나도록 지정합니다. 예를 들어 showIf를 사용하면 일부 버튼을 특정 응용 프로그램에서만 표시하거나 페이지가 ColdFusion, ASP 또는 JSP와 같은 서버측 언어로 작성된 경우에만 표시할 수 있습니다. showIf를 지정하지 않으면 항목이 항상 표시됩니다. Dreamweaver에서는 항목의 활성자가 실행될 때마다, 즉 update 속성의 값에 따라 이 속성을 확인합니다. 이 속성은 신중하게 사용해야 합니다. 이 속성은 항목을 정의할 때 또는 툴바에서 항목을 참조할 때 사용할 수 있습니다. 정의와 참조 모두에서 showIf 속성을 지정하면 해당 항목은 두 조건이 모두 충족되는 경우에만 표시됩니다. showIf 속성은 명령 파일의 showIf() 함수와 동일한 기능을 수행합니다.

내용
없음

컨테이너
toolbar 태그 또는 toolbarset 태그

예제
```
<itemref id="strikeout_button">
```

<separator/>

설명
툴바의 현재 위치에 구분 기호를 삽입합니다.

속성
{showIf}

- showif 속성은 지정된 스크립트가 true를 반환하는 경우에만 분리 기호가 툴바에 나타나도록 지정합니다. 예를 들어 showIf를 사용하면 분리 기호를 지정된 응용 프로그램에서만 표시하거나 페이지가 특정 문서 형식인 경우에만 표시할 수 있습니다. showIf 속성을 지정하지 않으면 분리 기호가 항상 표시됩니다.

내용
없음

컨테이너
toolbar 태그

예제
```
<separator/>
```

툴바 항목 태그

각 유형의 툴바 항목에는 고유한 태그와 일련의 필수 및 선택 속성이 있습니다. toolbar 항목은 툴바 내부 또는 외부에서 정의할 수 있습니다. 일반적으로 툴바 외부에서 툴바 항목을 정의하고 툴바 내부에서는 itemref 태그를 사용하여 해당 항목을 참조하는 것이 좋습니다.

툴바에는 다음과 같은 유형의 항목을 정의할 수 있습니다.

<button>

설명
이 누름 버튼을 클릭하면 특정 명령이 실행됩니다. 버튼의 모양 및 액션은 Dreamweaver 툴바의 [참조] 버튼과 동일합니다.

속성
id, image, tooltip, command, {showIf}, {disabledImage}, {overImage}, {label}, {file}, {domRequired}, {enabled}, {update}, {arguments}

각 속성에 대한 자세한 내용은 169페이지의 “항목 태그 속성”을 참조하십시오.

내용
없음

컨테이너
toolbar 태그 또는 toolbarset 태그

예제

```
<BUTTON ID="DW_DocRefresh"
    image="Toolbars/images/MM/refresh.gif"
    disabledImage="Toolbars/images/MM/refresh_dis.gif"
    tooltip="Refresh Design View (F5)"
    enabled="((dw.getDocumentDOM() != null) && (dw.getDocumentDOM().getView() != 'browse')
        && (!dw.getDocumentDOM().isDesignViewUpdated()))"
    command="dw.getDocumentDOM().synchronizeDocument()"
    update="onViewChange,onCodeViewSyncChange"/>
```

<checkbutton>

설명

체크 버튼은 체크 표시된 상태 또는 체크 표시되지 않은 상태로 나타나는 버튼으로, 이 버튼을 클릭하면 특정 명령이 실행됩니다. 체크 버튼은 표시된 상태에서는 누른 모양으로 강조 표시되고, 체크 표시되지 않은 상태에서는 평평하게 표시됩니다. Dreamweaver에서는 체크 버튼에 대해 Mouse-over, Pressed, Mouse-over-while-pressed 및 Disabled-while-pressed 상태를 구현합니다. checked 속성 또는 isCommandChecked() 함수로 지정된 핸들러는 체크 버튼이 클릭될 때 버튼의 상태가 전환되도록 해야 합니다.

속성

id, {showIf}, image, {disabledImage}, {overImage}, tooltip, {label}, {file}, {domRequired}, {enabled}, checked, {update}, command, {arguments}

각 속성에 대한 자세한 내용은 169페이지의 “항목 태그 속성”을 참조하십시오.

내용

없음

컨테이너

toolbar 태그 또는 toolbarset 태그

예제

```
<CHECKBUTTON ID="DW_LiveDebug"
    image="Toolbars/images/MM/debugview.gif"
    disabledImage="Toolbars/images/MM/globe_dis.gif"
    tooltip="Live Debug"
    enabled="dw.canLiveDebug()"
    checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() == 'browse'"
    command="dw.toggleLiveDebug()"
    showIf="dw.canLiveDebug()"
    update="onViewChange"/>
```

<radiobutton>

설명

라디오 버튼은 선택되지 않은 경우 누르지 않은 상태의 버튼으로 표시된다는 점만 제외하면 체크 버튼과 동일합니다. Dreamweaver에서는 라디오 버튼에 대해 Mouse-over, Pressed, Mouse-over-while-pressed 및 Disabled-while-pressed 상태를 구현합니다. Dreamweaver에서는 두 개 이상의 라디오 버튼을 선택할 수도 있습니다. checked 속성 또는 isCommandChecked() 함수로 지정된 핸들러는 라디오 버튼의 체크 표시된 상태와 체크 표시되지 않은 상태가 서로 일관성 있도록 해야 합니다.

라디오 버튼은 Dreamweaver 문서 툴바의 [코드] 뷰, [디자인] 뷰 및 [코드 및 디자인] 뷰 버튼과 동일하게 동작합니다.

속성
id, image, tooltip, checked, command, {showIf}, {disabledImage}, {overImage}, {label}, {file}, {domRequired}, {enabled}, {update}, {arguments}

각 속성에 대한 자세한 내용은 169페이지의 "항목 태그 속성"을 참조하십시오.

내용
없음

컨테이너
toolbar 태그 또는 toolbarset 태그

예제
```
<RADIOBUTTON ID="DW_CodeView"
    image="Toolbars/images/MM/codeView.gif"
    disabledImage="Toolbars/images/MM/codeView_dis.gif"
    tooltip="Show Code View"
    domRequired="false"
    enabled="dw.getDocumentDOM() != null"
    checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() == 'code'"
    command="dw.getDocumentDOM().setView('code')"
    update="onViewChange"/>
```

<menubutton>

설명
메뉴 버튼은 menuid 속성으로 지정된 컨텍스트 메뉴를 호출하는 버튼입니다. Dreamweaver에서는 메뉴 버튼에 대해 Mouse-over 및 Pressed 상태를 구현합니다. Dreamweaver에서는 버튼에 메뉴 항목이 연결되었음을 나타내는 아래쪽 화살표인 메뉴 화살표를 그리지 않으므로 아이콘에 이 메뉴 화살표를 포함해야 합니다. Dreamweaver 문서 툴바의 [파일 관리] 및 [코드 내비게이션] 버튼이 메뉴 버튼에 해당됩니다.

속성
id, image, tooltip, menuID, domRequired, enabled, {showIf}, {disabledImage}, {overImage}, {label}, {file}, {update}

각 속성에 대한 자세한 내용은 169페이지의 "항목 태그 속성"을 참조하십시오.

내용
없음

컨테이너
toolbar 태그 또는 toolbarset 태그

예제
```
<MENUBUTTON ID="DW_CodeNav"
    image="Toolbars/images/MM/codenav.gif"
    disabledImage="Toolbars/images/MM/codenav_dis.gif"
    tooltip="Code Navigation"
    enabled="dw.getFocus() == 'textView' || dw.getFocus() == 'html'"
    menuID="DWCodeNavPopup"
    update="onViewChange"/>
```

<dropdown>

설명

드롭 다운 메뉴(또는 팝업 메뉴)는 편집 불가능한 메뉴로, 이 메뉴에서 항목을 선택하면 특정 명령이 실행됩니다. 이 메뉴는 첨부된 JavaScript 함수를 기반으로 자동으로 업데이트됩니다. 드롭 다운 메뉴는 속성 관리자의 크기가 작지 않고 표준 크기라는 점만 제외하면, 텍스트 속성 관리자의 [포맷] 컨트롤과 모양 및 역할이 동일합니다.

속성

`id, tooltip, file, enabled, checked, value, command, {showIf}, {label}, {width}, {domRequired}, {update}, {arguments}`

각 속성에 대한 자세한 내용은 169페이지의 “항목 태그 속성”을 참조하십시오.

내용

없음

컨테이너

toolbar 태그 또는 toolbarset 태그

예제

```
<dropdown id="Font_Example"
    width="115"
    tooltip="Font"
    domRequired="false"
    file="Toolbars/mine/fontExample.htm"
    update="onSelChange"/>
```

<combobox>

설명

콤보 상자는 편집 가능한 팝업 메뉴로, 사용자가 메뉴 항목을 선택하거나 텍스트 상자의 내용을 편집한 후 포커스를 전환하면 해당 명령을 실행합니다. 이 메뉴는 속성 관리자의 크기가 작지 않고 표준 크기라는 점만 제외하면 텍스트 속성 관리자의 [글꼴] 컨트롤과 모양 및 역할이 동일합니다.

속성

id, file, tooltip, enabled, value, command, {showiI}, {label}, {width}, {domRequired}, {update}, {arguments}

각 속성에 대한 자세한 내용은 169페이지의 “항목 태그 속성”을 참조하십시오.

내용

없음

컨테이너

toolbar 태그 또는 toolbarset 태그

예제

```
<COMBOBOX ID="Address_URL"
    width="300"
    tooltip="Address"
    label="Address: "
    file="Toolbars/MM/AddressURL.htm"
    update="onBrowserPageBusyChange"/>
```

<editcontrol>

설명

편집 컨트롤 상자는 텍스트 편집 상자로, 사용자가 이 상자에서 텍스트를 변경하고 포커스를 전환하면 해당 명령을 실행합니다.

속성

id, tooltip, file, value, command, {showIf}, {label}, {width}, {domRequired}, {enabled}, {update}, {arguments}

각 속성에 대한 자세한 내용은 169페이지의 “항목 태그 속성”을 참조하십시오.

내용

없음

컨테이너

toolbar 태그 또는 toolbarset 태그

예제

```
<EDITCONTROL ID="DW_SetTitle"
    label="Title: "
    tooltip="Document Title"
    width="150"
    file="Toolbars/MM/EditTitle.htm"/>
```

<colorpicker>

설명

색상 선택기는 연관된 텍스트 상자가 없는 색상 패널로, 사용자가 새 색상을 선택할 때 해당 명령을 실행합니다. 이 패널은 Dreamweaver 속성 관리자의 색상 선택기와 모양 및 역할이 동일합니다. 다른 아이콘을 지정하여 기본 아이콘을 대체할 수 있습니다.

속성

id, tooltip, value, command, {showIf}, {image}, {disabledImage}, {overImage}, {label}, {colorRect}, {file}, {domRequired}, {enabled}, {update}, {arguments}

각 속성에 대한 자세한 내용은 169페이지의 “항목 태그 속성”을 참조하십시오.

내용

없음

컨테이너

toolbar 태그 또는 toolbarset 태그

예제

```
<colorpicker id="Color_Example"
    image="Toolbars/images/colorpickerIcon.gif"
    disabledImage="Toolbars/images/colorpickerIconD.gif"
    colorRect="0 12 16 16"
    tooltip="Text Color"
    domRequired="false"
    file="Toolbars/mine/colorExample.htm"
    update="onSelChange"/>
```

항목 태그 속성

툴바 항목에 속성 및 명령을 할당하여 툴바의 모양과 동작 방식을 지정할 수 있습니다. 다른 툴바 파일을 포함하고, 다른 툴바에 정의된 툴바 항목을 참조할 수도 있습니다. 툴바 항목 태그의 속성에는 다음과 같은 의미가 있습니다.

id="unique_id"

필수. id 속성은 툴바 항목의 식별자입니다. id 속성은 현재 파일과 현재 파일에 포함된 모든 파일에서 고유해야 합니다. itemref 태그는 항목 id를 사용하여 항목을 식별하고 툴바에 포함합니다.

예제

```
<button id="DW_DocRerefresh" . . . >
```

showIf="script"

선택 사항입니다. 이 속성은 스크립트가 true 값을 반환하는 경우에만 해당 항목이 툴바에 나타나도록 지정합니다. 예를 들어 showIf를 사용하면 페이지가 ColdFusion, ASP 또는 JSP와 같은 서버측 언어로 작성된 경우에만 특정 버튼을 표시할 수 있습니다. showIf를 지정하지 않으면 항목이 항상 표시됩니다.

showIf 속성은 항목 활성자가 실행될 때마다, 즉 update 속성의 값에 따라 확인됩니다. showIf 속성은 신중하게 사용해야 합니다.

showIf 속성은 항목을 정의할 때와 itemref 태그에서 항목을 참조할 때 지정할 수 있습니다. 정의와 참조 모두에서 showIf 속성을 지정하면 해당 항목은 두 조건이 모두 충족되는 경우에만 표시됩니다. showIf 속성은 툴바 명령 파일의 showIf() 함수와 동일합니다. showIf 속성과 showif() 함수를 모두 지정하면 함수가 속성보다 우선합니다.

예제

```
showIf="dw.canLiveDebug()"
```

image="image_path"

이 속성은 버튼, 체크 버튼, 라디오 버튼, 메뉴 버튼 및 콤보 버튼에 필수 사항입니다. image 속성은 색상 선택기에는 선택 사항이지만 다른 종류의 항목에는 무시됩니다. image 속성은 버튼에 표시되는 아이콘 파일의 경로를 Configuration 폴더에 대한 상대 경로로 지정합니다. 아이콘은 Dreamweaver에서 렌더링할 수 있는 어느 형식이나 가능하지만 대개는 GIF 또는 JPEG 파일 형식입니다.

색상 선택기에 아이콘이 지정된 경우 아이콘이 색상 선택기를 완전히 대체합니다. colorRect 속성이 함께 설정된 경우에는 지정된 사각형 안에서 현재 색상이 아이콘의 위에 나타납니다.

예제

```
image="Toolbars/images/MM/codenav.gif"
```

disabledImage="image_path"

선택 사항입니다. 버튼, 체크 버튼, 라디오 버튼, 메뉴 버튼, 색상 선택기 및 콤보 버튼 이외의 항목에 대해서는 disabledImage 속성이 무시됩니다. 이 속성은 버튼이 비활성화될 경우 표시되는 아이콘 파일의 경로를 Configuration 폴더에 대한 상대 경로로 지정합니다. disabledImage 속성을 지정하지 않으면 버튼이 비활성화되어 있을 때 image 속성에 지정된 이미지가 표시됩니다.

예제

```
disabledImage="Toolbars/images/MM/codenav_dis.gif"
```

overImage="image_path"

선택 사항입니다. 버튼, 체크 버튼, 라디오 버튼, 메뉴 버튼, 색상 선택기 및 콤보 버튼 이외의 항목에 대해서는 overImage 속성이 무시됩니다. 이 속성은 사용자가 마우스를 버튼 위로 이동할 경우에 표시되는 아이콘 파일의 경로를 Configuration 폴더에 대한 상대 경로로 지정합니다. overImage 속성을 지정하지 않으면 사용자가 마우스를 버튼 위로 이동해도 버튼이 변경되지 않으며 버튼 주위에 원 모양만 표시됩니다.

예제

```
overImage="Toolbars/images/MM/codenav_ovr.gif"
```

tooltip="tooltip string"

필수. 이 속성은 마우스 포인터가 툴바 항목을 가리킬 때 나타나는 텍스트 또는 도구 설명을 지정합니다.

예제

```
tooltip="Code Navigation"
```

label="label string"

선택 사항입니다. 이 속성은 항목 옆에 표시되는 레이블을 지정합니다. Dreamweaver에서는 레이블에 콜론이 자동으로 추가되지 않습니다. 버튼이 아닌 항목에 대한 레이블은 항상 항목의 왼쪽에 배치됩니다. 버튼, 체크 버튼, 라디오 버튼, 메뉴 버튼 및 콤보 버튼의 레이블은 버튼 내부 및 아이콘의 오른쪽에 배치됩니다.

예제

```
label="Title: "
```

width="number"

선택 사항입니다. 이 속성은 해당 항목의 폭(픽셀)을 지정하며 텍스트 상자, 팝업 메뉴 및 콤보 상자 항목에만 적용됩니다. width 속성을 지정하지 않으면 적절한 기본 폭이 사용됩니다.

예제

```
width="150"
```

menuID="menu_id"

연관된 명령 파일에 getMenuID() 함수를 지정하지 않은 경우 이 속성은 메뉴 버튼 및 콤보 버튼에 필수 사항입니다. 다른 유형의 항목에 대해서는 menuID 속성이 무시됩니다. 이 속성은 사용자가 버튼, 메뉴 버튼 또는 콤보 버튼을 클릭할 때 표시되는 컨텍스트 메뉴가 포함된 메뉴 막대의 ID를 지정합니다. 이 ID는 menus.xml에 있는 menubar 태그의 ID 속성에서 가져옵니다.

예제

```
menuID="DWCodeNavPopup"
```

colorRect="left top right bottom"

이 속성은 image 속성이 있는 색상 선택기에 대한 선택 사항입니다. 다른 유형의 항목과 이미지를 지정하지 않은 색상 선택기에 대해서는 colorRect 속성이 무시됩니다. colorRect 속성을 지정하면 색상 선택기에서 현재 선택된 색상이 아이콘의 왼쪽 또는 위쪽을 기준으로 한 사각형 영역에 표시됩니다. colorRect 속성을 지정하지 않으면 이미지에 현재 색상이 표시되지 않습니다.

예제

```
colorRect="0 12 16 16"
```

file="command_file_path"

팝업 메뉴와 콤보 상자에 대해 필수 사항입니다. 다른 유형의 항목에는 file 속성이 선택 사항입니다. file 속성은 항목을 채우고, 업데이트하고, 실행하는 JavaScript 함수가 포함된 명령 파일의 경로를 Configuration 폴더에 대한 상대 경로로 지정합니다. file 속성은 enabled, checked, value, update, domRequired, menuID, showIf 및 command 속성보다 우선합니다. 일반적으로 file 속성을 사용하여 명령 파일을 지정하면 태그에 지정된 같은 기능의 속성은 모두 무시됩니다. 명령 파일에 대한 자세한 내용은 173페이지의 “툴바 명령 API 함수”를 참조하십시오.

예제

```
file="Toolbars/MM/EditTitle.htm"
```

domRequired="true" or "false"

선택 사항입니다. 메뉴의 경우와 마찬가지로, domRequired 속성은 Dreamweaver에서 연관된 명령을 실행하기 전에 [디자인] 뷰와 [코드] 뷰를 동기화할지 여부를 지정합니다. 이 속성을 지정하지 않을 경우 기본값은 true입니다. 이 속성은 툴바 명령 파일의 isDOMRequired() 함수와 동일한 기능을 수행합니다.

예제

```
domRequired="false"
```

enabled="script"

선택 사항입니다. 메뉴의 경우와 마찬가지로 이 스크립트는 항목의 활성화 여부를 나타내는 값을 반환합니다. 이 속성을 지정하지 않을 경우 해당 항목은 기본적으로 활성화됩니다. enabled 속성은 툴바 명령 파일의 canAcceptCommand() 함수와 동일한 기능을 수행합니다.

예제

```
enabled="dw.getFocus() =='textView' || dw.getFocus() == 'html'"
```

checked="script"

이 속성은 체크 버튼 및 라디오 버튼에 필수 사항입니다. 다른 유형의 항목에 대해서는 checked 속성이 무시됩니다. 메뉴의 경우와 마찬가지로 이 스크립트는 항목의 체크 여부를 나타내는 값을 반환합니다. checked 속성은 툴바 명령 파일의 isCommandChecked()와 동일한 기능을 수행합니다. 이 속성을 지정하지 않을 경우 해당 버튼은 기본적으로 체크되지 않습니다.

예제

```
checked="dw.getDocumentDOM() != null && dw.getDocumentDOM().getView() == 'code'"
```

value="script"

이 속성은 팝업 메뉴, 콤보 상자, 텍스트 상자 및 색상 선택기에 필수 사항입니다. 다른 유형의 항목에 대해서는 value 속성이 무시됩니다.

팝업 메뉴와 콤보 상자에 표시할 값을 결정하기 위해 Dreamweaver에서는 먼저 메뉴의 각 항목에 대해 isCommandchecked()를 호출합니다. isCommandchecked() 함수가 true 값을 반환하는 항목이 있으면 Dreamweaver에서는 이 중 첫 번째 항목에 대해 해당 값을 표시합니다. true 값을 반환하는 항목이 없거나 isCommandChecked() 함수가 정의되어 있지 않으면 Dreamweaver에서는 getCurrentValue() 함수를 호출하거나 value 속성이 지정하는 스크립트를 실행합니다. 컨트롤이 콤보 상자인 경우에는 반환된 값이 표시됩니다. 컨트롤이 팝업 메뉴인 경우에는 반환된 값이 임시로 목록에 추가되고 이 목록이 표시됩니다.

다른 모든 경우에 스크립트는 표시할 현재 값을 반환합니다. 팝업 메뉴 또는 콤보 상자의 경우 이 값은 메뉴 목록에 있는 항목 중 하나여야 합니다. 콤보 상자 및 텍스트 상자의 경우 이 값은 스크립트가 반환하는 임의의 문자열일 수 있습니다. 색상 선택기의 경우 이 값은 유효한 색상이어야 하지만 다른 값이 반환될 수도 있습니다.

value 속성은 툴바 명령 파일의 getCurrentValue() 함수와 동일한 기능을 수행합니다.

update="update_frequency_list"

선택 사항입니다. 이 속성은 해당 항목의 화면 표시 상태를 업데이트하기 위해 enabled, checked, showif 및 value 핸들러가 실행되는 빈도를 지정합니다. update 속성은 툴바 명령 파일의 getUpdateFrequency() 함수와 동일한 기능을 수행합니다.

툴바 항목은 메뉴 항목과는 달리 항상 표시되므로 업데이트 빈도를 지정해야 합니다. 따라서 항상 가능한 가장 낮은 빈도를 선택하여 enabled, checked 및 value 핸들러를 가능한 한 단순하게 하는 것이 좋습니다.

다음 목록에서는 **update_frequency_list**에 사용 가능한 핸들러를 낮은 빈도에서 높은 빈도순으로 보여 줍니다. update 속성을 지정하지 않을 경우 업데이트 빈도의 기본값은 onEdit 빈도가 됩니다. 업데이트 빈도를 여러 개 지정하려면 쉼표로 구분하면 됩니다. 핸들러는 다음의 지정된 이벤트에서 실행됩니다.

- onServerModelChange는 현재 페이지의 서버 모델이 변경될 때 실행됩니다.
- onCodeViewSyncChange는 [코드] 뷰와 [디자인] 뷰의 동기화 상태가 변경될 때 실행됩니다.
- onViewChange는 사용자가 [코드] 뷰와 [디자인] 뷰 사이에서 포커스를 전환하거나 [코드] 뷰, [디자인] 뷰 또는 [코드 및 디자인] 뷰를 전환할 때마다 실행됩니다.
- onEdit는 [디자인] 뷰에서 문서를 편집할 때마다 실행됩니다. [코드] 뷰에서 변경할 때는 이 이벤트가 트리거되지 않습니다.
- onSelChange는 [디자인] 뷰에서 선택 사항을 변경할 때마다 실행됩니다. [코드] 뷰에서 변경할 때는 이 이벤트가 트리거되지 않습니다.
- onEveryIdle은 응용 프로그램이 유휴 상태일 때 정기적으로 실행됩니다. enabler/checked/showif/value 핸들러는 자주 실행되므로 이 작업에는 시간이 걸릴 수 있습니다. 그러므로 이 핸들러는 특정 시간에 활성 상태를 변경할 필요가 있는 버튼에 대해서만 사용되어 빠른 시간에 작업을 수행해야 합니다.

  참고: 이러한 모든 경우에 응용 프로그램이 정지 상태이면 지정된 이벤트가 발생한 후 핸들러가 실제로 실행됩니다. 이 핸들러는 편집 또는 선택 사항이 변경될 때마다 실행되는 것이 아니라, 편집 또는 선택 사항 변경이 여러 번 발생한 뒤 "얼마 후에" 실행됩니다. 사용자가 툴바 항목을 클릭할 때는 핸들러가 반드시 실행됩니다.

예제

```
update="onViewChange"
```

command="script"

이 속성은 메뉴 버튼을 제외한 모든 항목에 필수 사항입니다. 메뉴 버튼에 대해서는 command 속성이 무시됩니다. 이 속성은 사용자가 다음 중 한 가지 작업을 수행할 때 실행할 JavaScript 함수를 지정합니다.

- 버튼 클릭

- 팝업 메뉴 또는 콤보 상자에서 항목 선택
- 텍스트 상자 또는 콤보 상자에서 Tab 키를 누르거나, Return 키를 누르거나, 다른 부분을 클릭
- 색상 선택기에서 색상 선택

command 속성은 툴바 명령 파일의 receiveArguments() 함수와 동일한 기능을 수행합니다.

예제

```
command="dw.toggleLiveDebug()"
```

arguments="argument_list"

선택 사항입니다. 이 속성은 툴바 명령 파일의receiveArguments() 함수에 전달할 인수 목록을 쉼표로 구분하여 지정합니다. arguments 속성을 지정하지 않으면 Dreamweaver에서는 툴바 항목의 ID를 전달합니다. 또한 팝업 메뉴, 콤보 상자, 텍스트 상자 및 색상 선택기는 해당 컨트롤의 현재 값을 arguments 속성으로 지정된 인수보다 우선하여 첫 번째 인수로 전달하며 인수가 지정되지 않은 경우에는 항목 ID보다도 우선하여 전달합니다.

예제

[실행 취소] 및 [다시 실행] 버튼이 있는 툴바에서 각 버튼은 다음 예제에서와 같이 메뉴 명령 파일인 Edit_Clipboard.htm을 호출하고 액션을 지정하는 인수를 전달합니다.

```
<button id="DW_Undo"
 image="Toolbars/images/MM/undo.gif"
 disabledImage="Toolbars/images/MM/undo_dis.gif"
 tooltip="Undo"
 file="Menus/MM/Edit_Clipboard.htm"
 arguments="'undo'"
 update="onEveryIdle"/>
<button id="DW_Redo"
 image="Toolbars/images/MM/redo.gif"
 disabledImage="Toolbars/images/MM/redo_dis.gif"
 tooltip="Redo"
 file="Menus/MM/Edit_Clipboard.htm"
 arguments="'redo'"
 update="onEveryIdle"/>
```

툴바 명령 API 함수

속성에 스크립트를 지정하는 대부분의 경우 명령 파일의 JavaScript 함수를 통해서도 속성을 구현할 수 있습니다. 이 액션은 텍스트 상자에 대한 명령 핸들러의 경우처럼 함수가 인수를 받아야 하는 경우에 필요합니다. 팝업 메뉴와 콤보 상자의 경우에는 이 방법을 필수적으로 사용해야 합니다.

툴바 항목의 명령 파일 API는 메뉴 명령 파일 API의 Extension이므로 메뉴 명령 파일을 직접 툴바 명령 파일처럼 다시 사용할 수 있으며, 툴바와 관련된 추가 함수도 일부 사용할 수 있습니다.

canAcceptCommand()

지원 버전

Dreamweaver MX

설명
툴바 항목의 활성화 여부를 확인합니다. 활성 상태가 항목의 기본 상태이므로 false 값을 반환하는 경우가 한 번이라도 있는 경우가 아니면 이 함수를 정의하지 않습니다.

인수
팝업 메뉴, 콤보 상자, 텍스트 상자 및 색상 선택기의 경우 첫 번째 인수는 컨트롤 내의 현재 값입니다. getDynamicContent() 함수는 선택적으로 팝업 메뉴 내의 항목에 개별 ID를 첨부할 수 있습니다. 팝업 메뉴 내의 선택된 항목에 ID가 첨부되어 있으면 Dreamweaver에서는 값 대신 해당 ID를 canAcceptCommand()에 전달합니다. 콤보 상자의 경우 텍스트 상자의 현재 내용이 팝업 메뉴의 항목과 일치하지 않으면 Dreamweaver에서는 텍스트 상자의 내용을 전달합니다. Dreamweaver에서는 이 내용을 팝업 메뉴와 대/소문자를 구분하지 않고 비교하여 텍스트 상자의 내용이 목록의 항목과 일치하는지 여부를 확인합니다.

toolbars.xml 파일에 있는 이 툴바 항목에 대해 arguments 속성을 지정하면 해당 인수가 그 다음으로 전달됩니다. arguments 속성을 지정하지 않은 경우 Dreamweaver에서는 항목의 ID를 전달합니다.

반환값
부울 값을 반환하며, 항목이 활성화되어 있으면 true이고 그렇지 않으면 false입니다.

예제
```
function canAcceptCommand()
{
    return (dw.getDocumentDOM() != null);
}
```

getCurrentValue()

지원 버전
Dreamweaver MX

설명
항목에 표시할 현재 값을 반환합니다. Dreamweaver에서는 팝업 메뉴, 콤보 상자, 텍스트 상자 및 색상 선택기에 대해 getCurrentValue() 함수를 호출합니다. 팝업 메뉴의 경우 현재 값은 메뉴 내의 항목 중 하나여야 합니다. 값이 팝업 메뉴에 없으면 첫 번째 항목이 선택됩니다. 콤보 상자 및 텍스트 상자의 경우 이 값은 함수가 반환하는 임의의 문자열이 될 수 있습니다. 색상 선택기의 경우 이 값은 유효한 색상이어야 하지만 다른 값이 반환될 수도 있습니다. 이 함수는 value 속성과 동일한 기능을 수행합니다.

인수
없음

반환값
표시할 현재 값을 포함하는 문자열을 반환합니다. 색상 선택기의 경우 이 문자열에는 선택된 색상이 RGB 형식으로 포함됩니다. 예를 들어 흰색은 "#FFFFFF"입니다.

예제

```
function getCurrentValue()
{
    var title = "";
    var dom = dw.getDocumentDOM();
    if (dom)
        title = dom.getTitle();
    return title;
}
```

getDynamicContent()

지원 버전

Dreamweaver MX

설명

이 함수는 팝업 메뉴와 콤보 상자에 필수 사항입니다. 메뉴의 경우와 마찬가지로 이 함수는 팝업 메뉴 항목으로 사용할 문자열의 배열을 반환합니다. 각 문자열은 선택적으로 ";id=id"로 끝날 수 있습니다. ID가 지정된 경우 메뉴에 나타날 실제 문자열 대신 해당 ID가 receiveArguments() 함수에 전달됩니다.

메뉴의 항목 목록이 고정된 경우에도 이 함수를 사용해야 하므로 getDynamicContent()라는 이름은 잘못된 이름입니다. 예를 들어 Configuration/Menus/MM 폴더의 Text_Size.htm 파일은 동적 메뉴가 아니며, 일련의 정적 메뉴 항목 중 한 항목으로부터 호출되도록 디자인되었습니다. 그러나 사용 가능한 글꼴 크기 목록을 반환하는 getDynamicContent() 함수를 추가하면 툴바 팝업 메뉴에도 동일한 명령 파일을 사용할 수 있습니다. 툴바 항목은 반환된 배열의 문자열에서 밑줄을 무시하므로 메뉴 명령 파일을 다시 사용할 수 있습니다. 메뉴 명령 파일에서는 메뉴 항목이 동적으로 표시되어 있지 않으므로 getDynamicContent() 함수가 무시됩니다.

인수

없음

반환값

메뉴 항목으로 사용될 문자열의 배열을 반환합니다.

예제

```
function getDynamicContent()
{
    var items = new Array;
    var filename = dw.getConfigurationPath() + "/Toolbars/MM/AddressList.xml";
    var location = MMNotes.localURLToFilePath(filename);
    if (DWfile.exists(location))
    {
        var addressData = DWfile.read(location);
        var addressDOM = dw.getDocumentDOM(dw.getConfigurationPath() +
            '/Shared/MM/Cache/empty.htm');
        addressDOM.documentElement.outerHTML = addressData;
        var addressNodes = addressDOM.getElementsByTagName("url");
        if (addressNodes.length)
        {
            for (var i=0; i < addressNodes.length ; i++ )
            {
                items[i] = addressNodes[i].address + ";id='" +
                    addressNodes[i].address + "'";
            }
        }
    }
    return items;
```

getMenuID()

지원 버전

Dreamweaver MX

설명

메뉴 버튼에 대해서만 유효합니다. Dreamweaver에서는 getMenuID() 함수를 호출하여 사용자가 이 버튼을 클릭할 때 나타나는 메뉴의 ID를 가져옵니다.

인수

없음

반환값

menus.xml 파일에 정의된 메뉴 ID가 포함된 문자열을 반환합니다.

예제

```
function getMenuID()
{
    var dom = dw.getDocumentDOM();
    var menuID = '';
    if (dom)
    {
        var view = dom.getView();
        var focus = dw.getFocus();
        if (view == 'design')
        {
            menuID = 'DWDesignOnlyOptionsPopup';
        }
        else if (view == 'split')
        {
            if (focus == 'textView')
            {
                menuID = 'DWSplitCodeOptionsPopup';
            }
            else
            {
                menuID = 'DWSplitDesignOptionsPopup';
            }
        }
        else if (view == 'code')
        {
            menuID = 'DWCodeOnlyOptionsPopup';
        }
        else
        {
            menuID = 'DWBrowseOptionsPopup';
        }
    }
    return menuID;
}
```

getUpdateFrequency()

지원 버전

Dreamweaver MX

설명

항목의 화면 표시 상태를 업데이트하기 위해 enabled, checked, showIf 및 value 속성에 대한 핸들러가 실행되는 빈도를 지정합니다.

툴바 항목은 메뉴와는 달리 항상 표시되므로 업데이트 빈도를 지정해야 합니다. 따라서 항상 가능한 가장 낮은 빈도를 선택하여 enabled, checked 및 value 핸들러를 가능한 한 단순하게 하는 것이 좋습니다.

이 함수는 툴바 항목의 update 속성과 동일한 기능을 수행합니다.

인수

없음

반환값

쉼표로 구분된 업데이트 핸들러 목록을 포함하는 문자열을 반환합니다. 사용 가능한 업데이트 핸들러의 전체 목록은 172페이지의 “update="update_frequency_list"”를 참조하십시오.

예제

```
function getUpdateFrequency()
{
    return onSelChange";
}
```

isCommandChecked()

지원 버전

Dreamweaver MX

설명

항목의 선택 여부를 지정하는 값을 반환합니다. 버튼의 경우 체크된 상태는 버튼이 설정된 상태 또는 눌러진 상태임을 의미합니다. isCommandChecked() 함수는 툴바 항목 태그의 checked 속성과 동일한 기능을 수행합니다.

인수

팝업 메뉴, 콤보 상자, 텍스트 상자 및 색상 선택기의 경우 첫 번째 인수는 컨트롤 내의 현재 값입니다. getDynamicContent() 함수는 선택적으로 팝업 메뉴 내의 항목에 개별 ID를 첨부할 수 있습니다. 메뉴 내의 선택된 항목에 ID가 첨부되어 있으면 Dreamweaver에서는 값 대신 해당 ID를 isCommandChecked() 함수에 전달합니다. 콤보 상자의 경우 텍스트 상자의 현재 내용이 팝업 메뉴의 항목과 일치하지 않으면 Dreamweaver에서는 텍스트 상자의 내용을 전달합니다. 텍스트 상자가 일치하는지 여부를 확인하기 위해 Dreamweaver에서는 메뉴와 대/소문자 구분 없이 비교합니다.

arguments 속성을 지정하면 해당 인수가 그 다음으로 전달됩니다. arguments 속성을 지정하지 않으면 Dreamweaver에서는 항목의 ID를 전달합니다.

반환값

부울 값을 반환합니다. 항목이 체크 표시되면 true이고, 그렇지 않으면 false입니다.

예제

다음 예제에서는 단락 서식 및 CSS 스타일을 지정하는 팝업 메뉴에서 체크 표시될 항목(있는 경우)을 결정합니다.

```
function isCommandChecked()
{
    var bChecked = false;
    var style = arguments[0];
    var textFormat = dw.getDocumentDOM().getTextFormat();

    if (dw.getDocumentDOM() == null)
        bChecked = false;

    if (style == "(None)")
        bChecked = (dw.cssStylePalette.getSelectedStyle() == '' || textFormat ==
"" || textFormat == "P" || textFormat == "PRE");
    else if (style == "Heading 1")
        bChecked =0(textFormat == "h1");
    else if (style == "Heading 2")
        bChecked =0(textFormat == "h2");
    else if (style == "Heading 3")
        bChecked =0(textFormat == "h3");
    else if (style == "Heading 4")
        bChecked =0(textFormat == "h4");
    else if (style == "Heading 5")
        bChecked =0(textFormat == "h5");
    else if (style == "Heading 6")
        bChecked =0(textFormat == "h6");
    else
        bChecked = (dw.cssStylePalette.getSelectedStyle() == style);
    return bChecked;
}
```

isDOMRequired()

지원 버전
Dreamweaver MX

설명
툴바 명령을 실행하는 데 유효한 DOM이 필요한지 여부를 지정합니다. 이 함수가 true 값을 반환하거나 정의되어 있지 않으면 Dreamweaver에서는 해당 명령에 유효한 DOM이 필요하다고 가정하고 연관된 명령을 실행하기 전에 문서의 [코드] 뷰와 [디자인] 뷰를 동기화합니다. 이 함수는 툴바 항목 태그의 domRequired 속성과 동일한 기능을 수행합니다.

인수
없음

반환값
부울 값을 반환합니다. DOM이 필요하면 true이고, 그렇지 않으면 false입니다.

예제

```
function isDOMRequired()
{
    return false;
}
```

receiveArguments()

지원 버전
Dreamweaver MX

설명
툴바 항목에서 전달된 모든 인수를 처리합니다. receiveArguments() 함수는 툴바 항목 태그의 command 속성과 동일한 기능을 수행합니다.

인수
팝업 메뉴, 콤보 상자, 텍스트 상자 및 색상 선택기의 경우 첫 번째 인수는 컨트롤 내의 현재 값입니다. getDynamicContent() 함수는 선택적으로 팝업 메뉴 내의 항목에 개별 ID를 첨부할 수 있습니다. 팝업 메뉴 내의 선택된 항목에 ID가 첨부되어 있으면 Dreamweaver에서는 값 대신 해당 ID를 receiveArguments() 함수에 전달합니다. 콤보 상자의 경우 텍스트 상자의 현재 내용이 팝업 메뉴의 항목과 일치하지 않으면 Dreamweaver에서는 텍스트 상자의 내용을 전달합니다. 텍스트 상자가 일치하는지 여부를 확인하기 위해 Dreamweaver에서는 팝업 메뉴와 대/소문자 구분 없이 비교합니다.

arguments 속성을 지정하면 해당 인수가 그 다음으로 전달됩니다. arguments 속성을 지정하지 않은 경우 Dreamweaver에서는 항목의 ID를 전달합니다.

반환값
없음

예제
```
function receiveArguments(newTitle)
{
    var dom = dw.getDocumentDOM();
    if (dom)
        dom.setTitle(newTitle);
}
```

showIf()

지원 버전
Dreamweaver MX

설명
함수가 true 값을 반환하는 경우에만 툴바에 항목이 나타나도록 지정합니다. 예를 들어 showIf() 함수를 사용하면 페이지에 특정 서버 모델이 있는 경우에만 특정 버튼이 나타나도록 할 수 있습니다. showif() 함수가 정의되어 있지 않으면 항목이 항상 나타납니다. showIf() 함수는 툴바 항목 태그의 showIf 속성과 동일한 기능을 수행합니다.

showIf() 함수는 해당 항목의 활성자가 실행될 때마다, 즉 getUpdateFrequency() 함수가 반환하는 값에 따라 호출됩니다.

인수
없음

반환값
부울 값을 반환합니다. 항목이 나타나면 true이고, 그렇지 않으면 false입니다.

예제

```
function showif()
{
        var retval = false;
        var dom = dw.getDocumentDOM();

        if(dom)
        {
            var view = dom.getView();
            if(view == 'design')
            {
                retval = true;
            }
        }
        return retval;
}
```

12장: 보고서

Adobe Dreamweaver 에서는 사이트 보고서와 독립 실행형 보고서라는 두 가지 유형의 보고서를 지원합니다.

사이트 보고서

보고서 API를 사용하여 사용자 정의 사이트 보고서를 만들거나 Dreamweaver와 함께 제공되는 미리 작성된 보고서 세트를 수정할 수 있습니다. 사이트 보고서에 액세스하는 유일한 방법은 [보고서] 대화 상자를 사용하는 것입니다.

사이트 보고서는 Dreamweaver Configuration/Reports 폴더에 있습니다. Reports 폴더에는 보고서 범주를 나타내는 하위 폴더가 있습니다. 각 보고서는 한 범주에만 속할 수 있습니다. 범주 이름은 31자를 초과할 수 없습니다. 각 하위 폴더에는 _foldername.txt라는 파일이 포함될 수 있습니다. 이 파일이 있으면 Dreamweaver에서는 이 파일의 내용을 범주 이름으로 사용합니다. _foldername.txt 파일이 없으면 Dreamweaver에서는 해당 폴더 이름을 범주 이름으로 사용합니다.

사용자가 [보고서] 대화 상자에서 여러 개의 사이트 보고서를 선택하면 Dreamweaver에서는 모든 결과를 [결과] 패널에 있는 [사이트 보고서] 탭 아래의 동일한 [결과] 윈도우에 표시하며, 사용자가 다음에 사이트 보고서를 실행할 때 이 결과를 대체합니다.

다음 표에서는 사이트 보고서를 만들 때 사용하는 파일에 대해 설명합니다.

경로	파일	설명
Configuration/Reports/{type/}	**reportname**.js	보고서의 내용을 생성하는 함수가 들어 있습니다.
Configuration/Reports/{type/}	**reportname**.htm	해당 JavaScript 파일을 호출합니다. 필요하면 보고서에 대한 [설정] 대화 상자의 UI(사용자 인터페이스)를 정의합니다.
Configuration/Reports/	Reports.js	보고서 생성에 사용되는 일반 함수를 제공합니다.

사이트 보고서의 작동 방식

1. 보고서는 [사이트] > [보고서] 명령을 통해 액세스할 수 있습니다. 이 명령을 선택하면 선택한 대상에 대해 실행할 보고서를 선택할 수 있는 대화 상자가 표시됩니다.
2. [검색 대상:] 팝업 메뉴를 사용하여 선택한 보고서를 실행할 대상 파일을 선택합니다. 이 메뉴에는 [현재 문서], [현재 로컬 사이트 전체], [사이트에서 파일 선택] 및 [폴더] 명령이 있습니다. [폴더] 명령을 선택하면 [탐색] 버튼과 텍스트 필드가 나타나므로 폴더를 선택할 수 있습니다.
3. [설정] 버튼을 클릭하고 매개 변수의 값을 입력하여 매개 변수가 있는 보고서를 사용자 정의할 수 있습니다. 사용자가 보고서 매개 변수를 직접 설정하려면 보고서에 [설정] 대화 상자가 있어야 합니다. 이 대화 상자는 선택적이며, 모든 보고서에서 보고서 매개 변수를 설정할 필요는 없습니다. 보고서에 [설정] 대화 상자가 없는 경우 목록에서 보고서를 선택하면 [설정] 버튼이 흐리게 표시됩니다.
4. 보고서를 선택하고 설정을 지정한 다음 [실행] 버튼을 클릭합니다.

   참고: 보고서에 preventFileActivity 핸들러가 있으면 Dreamweaver에서 사용자는 해당 보고서가 실행되는 동안에 다른 파일 작업을 수행할 수 없습니다.

   이때 Dreamweaver에서는 [결과] 패널의 [사이트 보고서] 탭에 있는 모든 항목을 지웁니다. Dreamweaver에서는 보고 프로세스를 시작하기 전에 각 보고서에서 beginReporting() 함수를 호출합니다. 이 함수에서 보고서가 false 값을 반환하면 해당 보고서는 보고서 실행에서 제거됩니다.
5. 각 파일이 processFile() 함수를 사용하여 [보고서] 대화 상자에서 선택한 각 보고서에 전달됩니다. 보고서에서 결과 목록에 이 파일에 대한 정보를 포함해야 하는 경우에는 dw.resultsPalette.siteReports.addResultItem() 함수를 호출해야 합니다. 이 과정은

사용자의 선택 영역과 관련된 파일이 모두 처리되거나 사용자가 윈도우 아래쪽에 있는 [중단] 버튼을 클릭할 때까지 계속됩니다. Dreamweaver에서는 처리 중인 각 파일의 이름과 처리해야 할 파일의 수를 표시합니다.

6 모든 파일이 처리되고 보고 프로세스가 완료된 후 Dreamweaver에서는 각 보고서에서 endReporting() 함수를 호출합니다.

간단한 사이트 보고서 예제

이 간단한 Extension 예제에서는 특정 파일, 전체 사이트, 선택한 파일 또는 폴더에서 참조되는 모든 이미지를 나열하고 [사이트 결과] 탭 아래의 [결과] 윈도우에 보고서를 표시합니다.

이 Extension을 만들려면 보고서 정의를 만들고 JavaScript 코드를 작성합니다.

이 예제에서는 HTML Reports 폴더에 두 개의 파일, 즉 보고서 정의가 포함된 List images.htm과 이 보고서에 고유한 JavaScript 코드가 포함된 List Images.js를 만듭니다. 또한 Dreamweaver에 포함된 Reports.js 파일을 참조합니다.

보고서 정의 만들기

보고서 정의에서는 [보고서] 대화 상자에 나타나는 보고서 이름을 지정하고 필요한 모든 JavaScript 파일을 호출하며, 필요하면 [설정] 대화 상자의 사용자 인터페이스를 정의합니다.

1 Configuration/Reports/HTML Reports/List images.htm 파일을 만듭니다.

2 HTML 페이지의 제목에 다음 내용을 추가하여 [보고서] 대화 상자에 표시할 보고서 이름을 지정합니다.

```
<html>
<head>
<title>List Images</title>
```

3 파일의 끝 부분에 script 태그를 추가하고 src 속성에 Reports.js 파일을 지정합니다.

```
<script src="../Reports.js"></script>
```

4 파일의 끝 부분에 다른 script 태그를 추가하고 src 속성에 다음에 만들 List Images.js 파일을 지정합니다.

```
<html>
<head>
<title>List Images</title>
<script src="../Reports.js"></script>
<script src="List Images.js"></script>
```

5 head 태그를 닫고, 열기 및 닫기 body 태그를 포함하고, html 태그를 닫습니다.

```
</head>
<body>
</body>
</html>
```

6 파일을 Configuration/Reports/HTML Reports 폴더에 List Images.js라는 이름으로 저장합니다.

JavaScript 코드 작성

Dreamweaver에는 Reports.js 파일이 포함되어 있습니다. Reports.js에 있는 모든 함수를 호출할 수 있지만, 사용자 정의 사이트 보고서에 고유한 모든 함수가 포함된 JavaScript 파일도 만들어야 합니다.

1 다음 내용이 들어 있는 Configuration/Reports/HTML Reports/List Images.js 파일을 만듭니다.

```
// Function: configureSettings
// Description: Standard report API, used to initialize and load
//the default values. Does not initialize the UI.
//
function configureSettings() {
    return false;
}
// Function: processFile
// Description: Report command API called during file processing.
//
function processFile (fileURL) {
    if (!isHTMLType(fileURL)) //If the file isn't an HTML file
        return;                     //skip it.
    var curDOM = dw.getDocumentDOM(fileURL);     // Variable for DOM
    var tagList = curDOM.getElementsByTagName('img'); // Variable for img tags
    var imgfilename; // Variable for file name specified in img tag
    for (var i=0; i < tagList.length; i++) { // For img tag list
        imgfilename = tagList[i].getAttribute('src'); // Get image filename
        if (imgfilename != null) { // If a filename is specified
        // Print the appropriate icon, HTML filename,
        // image filename, and line number
        reportItem(REP_ITEM_CUSTOM, fileURL, imgfilename,
                curDOM.nodeToSourceViewOffsets(tagList[i])); }
    }
}
```

2 파일을 Configuration/Reports/HTML Reports 폴더에 List Images.js라는 이름으로 저장합니다.

독립 실행형 보고서

결과 윈도우 API를 사용하여 독립 실행형 보고서를 만들 수 있습니다. 독립 실행형 보고서는 보고서 API 대신 결과 윈도우 API를 직접 사용하는 일반 명령입니다. 독립 실행형 보고서는 다른 명령과 마찬가지로 메뉴 또는 다른 명령을 통해 액세스할 수 있습니다.

독립 실행형 보고서는 Dreamweaver Configuration/Commands 폴더에 있습니다. 독립 실행형 보고서에 대한 사용자 정의 명령은 [명령] 메뉴에 나타납니다.

Dreamweaver에서는 사용자가 독립 실행형 보고서를 실행할 때마다 새 결과 윈도우를 만듭니다.

경로	파일	설명
Configuration/Commands	**commandname**.htm	사용자가 명령을 선택할 때 나타나는 대화 상자에 대한 UI를 정의하며, 보고서 생성에 필요한 작업을 수행하는 JavaScript 코드나 JavaScript 파일에 대한 참조가 들어 있습니다.
Configuration/Commands	**commandname.js**	결과 윈도우를 생성하며 이 윈도우 안에 보고서를 넣습니다.

독립 실행형 보고서의 작동 방식

1 사용자 정의 명령은 보고서를 생성하기 위해 만드는 명령으로서, dw.createResultsWindow() 함수를 호출하고 반환된 결과 객체를 윈도우 변수에 저장하여 새 결과 윈도우를 엽니다. 이 프로세스의 나머지 함수는 이 객체의 메서드로 호출해야 합니다.

2 사용자 정의 명령은 setTitle() 및 SetColumnWidths() 함수를 결과 윈도우 객체의 메서드로 호출하여 결과 윈도우의 제목 및 서식을 초기화합니다.

3 이 명령은 addItem() 함수를 호출하여 즉시 결과 윈도우에 항목을 추가할 수도 있고, setFileList() 및 startProcessing() 함수를 결과 윈도우 객체의 메서드로 호출하여 파일 목록의 반복 처리를 시작할 수도 있습니다.

4 이 명령이 resWin.startProcessing()을 호출하면 Dreamweaver에서는 목록에 있는 각 파일 URL에 대해 processFile() 함수를 호출합니다. 독립 실행형 명령에는 processFile() 함수를 정의합니다. 이 함수는 파일 URL을 유일한 인수로 받습니다. 일부 다른 명령에서 processFile() 함수가 호출되도록 하려면 결과 윈도우 객체의 setCallbackCommands() 함수를 사용합니다.

5 addItem() 함수를 호출하려면 processFile() 함수가 독립형 명령에 의해 작성된 결과 윈도우에 액세스해야 합니다. processFile() 함수는 결과 윈도우 객체의 stopProcessing() 함수를 호출하여 파일 목록의 처리를 중단할 수도 있습니다.

간단한 독립 실행형 보고서 예제

이 간단한 독립 실행형 보고서 Extension에서는 특정 파일에서 참조되는 모든 이미지의 목록을 나열하고 결과 윈도우에 보고서를 표시합니다.

대화 상자 UI를 만들고 JavaScript 코드를 작성하여 이 Extension을 만듭니다.

이 예제에서는 Configuration/Commands 폴더에 두 개의 파일, 즉 사용자 정의 명령을 선택할 때 나타나는 대화 상자의 UI를 정의하는 List images.htm 파일과 이 보고서에 고유한 JavaScript 코드가 들어 있는 Listimages.js 파일을 만듭니다.

대화 상자 UI 만들기

HTML 파일의 body 섹션에서는 사용자 정의 명령을 선택할 때 나타나는 대화 상자의 내용을 지정하고 필요한 모든 JavaScript 파일을 호출합니다.

1 Configuration/Commands/Listimages.htm 파일을 만듭니다.

2 Listimages.htm 파일에 다음 내용을 입력합니다.

```
<html>
<head>
<title>Standalone report example</title>
<script src="Listimages.js">
</script>
</head>
<body>
<div name="test">
<form name="myForm">
<table>
<tr>
<td>Click OK to display the standalone report.</td>
</tr>
</table>
</form>
</div>
</body>
```

3 이 파일을 Configuration/Commands 폴더에 Listimages.htm이라는 이름으로 저장합니다.

JavaScript 코드 작성

독립 실행형 보고서에 고유한 모든 함수가 포함된 JavaScript 파일을 만듭니다.

1 Configuration/Commands 폴더에 다음 코드가 들어 있는 Listimages.js 파일을 만듭니다.

```
function stdaloneresultwin()
{
    var curDOM = dw.getDocumentDOM("document");
    var tagList = curDOM.getElementsByTagName('img');
    var imgfilename;
    var iOffset = 0;
    var iLineNumber = 0;

    var resWin = dw.createResultsWindow("Images in File", ["Line", "Image"]);

    for (var i=0; i < tagList.length; i++)
    {
        // Get the name of the source file.
            imgfilename = tagList[i].getAttribute('src');
        // Get the character offset from the start of the file
        // to the start of the img tag.
        iOffset = curDOM.nodeToOffsets(curDOM.images[i]);
        // Based on the offset, figure out what line in the file
        // the img tag is on.
        iLineNumber = curDOM.getLineFromOffset(iOffset[0]);
        // As long as the src attribute specifies a file name,
            if (imgfilename != null)
        { // display the line number, and image path.
            resWin.addItem(resWin, "0", "Images in Current File", null, ¬
                null, null, [iLineNumber, imgfilename]);
            }
    }
    return;
}

// add buttons to dialog
function commandButtons()
{
    return new Array("OK", "stdaloneresultwin()", "Cancel", "window.close()");
}
```

2 파일을 Configuration/Commands 폴더에 Listimages.js라는 이름으로 저장합니다.

보고서 API 함수

보고서 API에 필요한 유일한 함수는 processFile() 함수입니다. 다른 모든 함수는 선택 사항입니다.

processFile()

지원 버전
Dreamweaver 4

설명
이 함수는 처리할 파일이 있을 때 호출됩니다. 보고서 명령은 파일을 수정하지 않고 처리해야 하며, dw.ResultsPalette.SiteReports() 함수, addResultItem() 함수 또는 resWin.addItem() 함수를 사용하여 파일에 대한 정보를 반환해야 합니다. 처리가 끝나면 각 파일의 DOM이 자동으로 해제됩니다.

인수
strFilePath

strFilePath 인수는 처리할 파일의 전체 경로와 파일 이름입니다.

반환값
없음

beginReporting()

지원 버전
Dreamweaver 4

설명
이 함수는 보고서가 실행되기 전에 보고 프로세스가 시작될 때 호출됩니다. 보고서 명령이 이 함수에서 false를 반환하면 해당 보고서 명령은 보고서 실행에서 제외됩니다.

인수
target

target 인수는 보고 세션의 대상을 나타내는 문자열입니다. 사용할 수 있는 값은 "CurrentDoc", "CurrentSite", "CurrentSiteSelection"(사이트의 선택된 파일에 사용) 또는 "Folder:+ 사용자가 선택한 폴더의 경로"(예: "Folder:c:temp")입니다.

반환값
부울 값을 반환합니다. 보고서 실행에 성공하면 true이고, **target**이 보고서 실행에서 제외되면 false입니다.

endReporting()

지원 버전
Dreamweaver 4

설명
이 함수는 보고 프로세스가 끝날 때 호출됩니다.

인수
없음

반환값
없음

commandButtons()

지원 버전
Dreamweaver 4

설명
[옵션] 대화 상자의 오른쪽에 표시할 버튼과 이 버튼을 클릭할 때 발생하는 비헤이비어를 정의합니다. 이 함수가 정의되어 있지 않으면 버튼이 나타나지 않으며 보고서 파일의 body 섹션이 확장되어 대화 상자 전체를 채웁니다.

인수
없음

반환값
짝수 개의 요소가 포함된 배열을 반환합니다. 첫 번째 요소는 맨 위 버튼의 레이블이 포함된 문자열입니다. 두 번째 요소는 맨 위 버튼을 클릭할 때 발생하는 비헤이비어를 정의하는 JavaScript 코드의 문자열입니다. 나머지 요소는 동일한 방법으로 추가 버튼을 정의합니다.

예제
commandButtons() 함수의 다음 인스턴스는 [OK], [Cancel] 및 [Help] 버튼을 정의합니다.

```
function commandButtons(){
    return new Array("OK" , "doCommand()" , "Cancel" , ¬
    "window.close()" , "Help" , "showHelp()");
}
```

configureSettings()

지원 버전
Dreamweaver 4

설명
이 보고서가 선택될 때 [보고서] 대화 상자의 [보고서 설정] 버튼을 활성화할지 여부를 결정합니다.

인수
없음

반환값
부울 값을 반환합니다. [보고서 설정] 버튼이 활성화되면 true이고, 그렇지 않으면 false입니다.

windowDimensions()

지원 버전
Dreamweaver 4

설명
이 함수는 [매개 변수] 대화 상자의 구체적 크기를 설정합니다. 이 함수를 정의하지 않으면 윈도우 크기가 자동으로 계산됩니다.

참고: [옵션] 대화 상자의 크기를 640x480픽셀보다 크게 하려는 경우에만 이 함수를 정의하십시오.

인수
platform

platform 인수의 값은 사용자의 플랫폼에 따라 "macintosh" 또는 "windows"입니다.

반환값

"widthInPixels,heightInPixels" 형식의 문자열을 반환합니다.

반환된 크기는 [확인] 및 [취소] 버튼의 영역을 포함하지 않으므로 전체 대화 상자의 크기보다 작습니다. 반환된 크기에 옵션이 모두 들어가지 않으면 스크롤 막대가 나타납니다.

예제

windowDimensions() 함수의 다음 인스턴스는 [매개 변수] 대화 상자의 크기를 648x520픽셀로 설정합니다.

```
function windowDimensions(){
    return "648,520";
}
```

13장: 태그 라이브러리 및 편집기

Dreamweaver에서는 모든 태그 속성을 비롯하여 각 태그에 대한 정보를 Configuration/TagLibraries 폴더에 있는 일련의 하위 폴더에 저장합니다. 태그 편집기 및 태그 선택기 함수는 태그를 조작 및 편집할 때 이 폴더에 저장된 정보를 사용합니다.

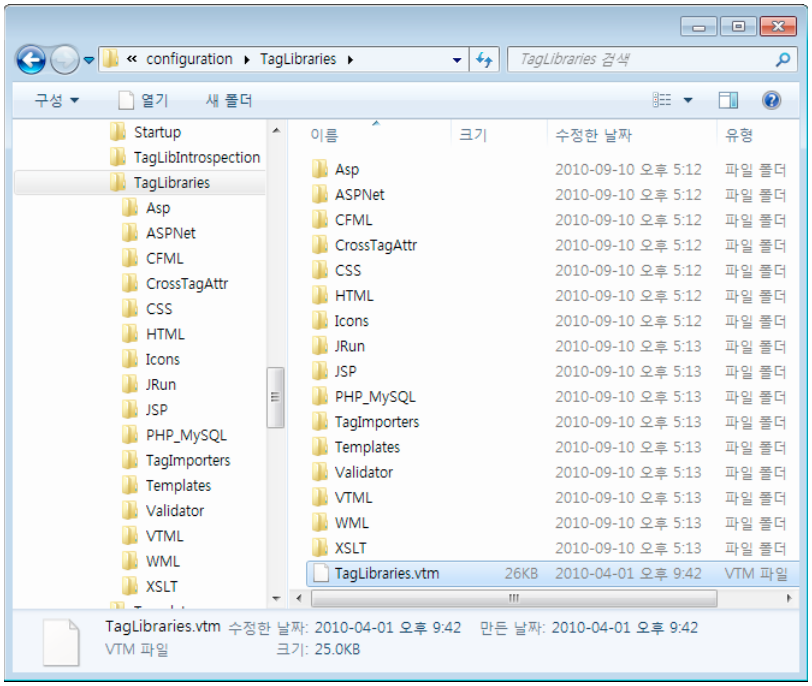
Dreamweaver에는 HTML, ASP.NET, CFML, JRun, JSP 등의 언어에 사용할 수 있는 편집기가 함께 제공됩니다. Dreamweaver에서 제공하는 태그 편집기를 사용자 정의하거나 새로운 태그 편집기를 만들 수 있습니다. 또한 태그 라이브러리에 새 태그를 추가할 수도 있습니다.

사용자 정의 태그 편집기를 만들려면 먼저 태그 라이브러리 구조를 알아야 합니다. 다음 표에서는 태그 라이브러리를 만들 때 사용하는 파일을 설명합니다.

경로	파일	설명
Configuration/TagLibraries/	TagLibraries.vtm	설치된 태그를 모두 나열합니다.
Configuration/TagLibraries/**language**/	**tag**.vtm	태그 속성, 태그에 닫기 태그가 있는지 여부, 서식 규칙 등을 비롯하여 태그에 대한 정보가 포함되어 있습니다.
Configuration/TagLibraries/**language**/	**Tagimagefile**.gif	속성 관리자에 표시할 선택적 파일입니다.

태그 라이브러리 파일 형식

태그 라이브러리는 설치된 태그를 모두 나열하는 단일 루트 파일인 TagLibraries.vtm 파일과 태그 라이브러리에 있는 각 태그에 대한 VTML 파일로 구성됩니다. TagLibraries.vtm 파일은 목차와 같은 역할을 하며 각 개별 태그의 VTML 파일에 대한 포인터를 포함합니다. 다음 그림에서는 Dreamweaver에서 마크업 언어별로 VTML 파일이 구성되는 방식을 보여 줍니다.



Adobe의 Macromedia HomeSite 사용자라면 VTML 파일 구조를 파악할 수는 있지만, Dreamweaver에서는 HomeSite와 동일한 방법으로 VTML 파일을 사용하지 않습니다. 가장 중요한 차이점은 Dreamweaver에는 Extension UI(사용자 인터페이스)를 표시하는 자체 HTML 렌더러가 있으므로 UI(사용자 인터페이스) 렌더링 프로세스에 Dreamweaver VTML 파일이 사용되지 않는다는 점입니다.

다음 예제에서는 TagLibraries.vtm 파일의 구조를 보여 줍니다.

```
<taglibraries>
<taglibrary name="Name of tag library" doctypes="HTML,ASP-JS,ASP-VB" tagchooser="relative
    path to TagChooser.xml file" id="DWTagLibrary_html">
    <tagref name="tag name" file="relative path to tag .vtm file"/>
</taglibrary>
<taglibrary name="CFML Tags" doctypes="ColdFusion" servermodel="Cold Fusion"
    tagchooser="cfml/TagChooser.xml" id="DWTagLibrary_cfml">
    <tagref name="cfabort" file="cfml/cfabort.vtm"/>
</taglibrary>
<taglibrary name="ASP.NET Tags" doctypes="ASP.NET_CSharp,ASP.NET_VB" servermodel="ASPNet"
    prefix="<asp:" tagchooser="ASPNet/TagChooser.xml" id="DWTagLibrary_aspnet">
    <tagref name="dataset" file="aspnet/dataset.vtm" prefix="<mm:dataset"/>
</taglibrary>
</taglibraries>
```

taglibrary 태그는 하나 이상의 태그를 태그 라이브러리로 그룹화합니다. 태그를 가져오거나 새로운 태그 집합을 만들 때 태그를 태그 라이브러리로 그룹화할 수 있습니다. 일반적으로 taglibrary 그룹은 JSP(JavaServer Page) TLD 파일, XML DTD(문서 형식 정의) 파일, ASP.NET 네임스페이스 또는 그 밖의 다른 논리 그룹에 정의되어 있는 태그 집합에 해당됩니다.

다음 표에서는 taglibrary 속성을 설명합니다.

속성	설명	필수/선택 사항
taglibrary.name	이 속성은 UI에서 태그 라이브러리를 참조하는 데 사용됩니다.	필수
taglibrary.doctypes	이 라이브러리가 활성화된 문서 형식을 나타냅니다. 라이브러리가 활성화되어 있으면 코드 힌트 메뉴에 라이브러리 태그가 나타납니다. 이름이 충돌할 수 있으므로 모든 태그 라이브러리가 동시에 활성화될 수는 없습니다. 예를 들어 HTML 파일과 WML 파일은 호환되지 않습니다.	필수
taglibrary.prefix	이 속성이 지정되어 있으면 태그 라이브러리 내의 태그는 taglibrary.prefix + tagref.name 형식으로 사용됩니다. 예를 들어 taglibrary.prefix가 "<jrun:"이고 tagref.name이 "if"이면 해당 태그의 형식은 "<jrun:if"입니다. 특정 태그에서는 이 형식이 무시될 수 있습니다.	선택 사항
taglibrary.servermodel	태그 라이브러리의 태그가 응용 프로그램 서버에서 실행되는 경우 servermodel 속성은 태그의 서버 모델을 식별합니다. 해당 태그가 서버측 태그가 아니라 클라이언트측 태그이면 servermodel 속성은 생략됩니다. 또한 servermodel 속성은 대상 브라우저 확인 기능에 사용됩니다.	선택 사항
taglibrary.id	이 속성은 파일에 있는 다른 태그 라이브러리의 taglibrary.ID 속성과는 다른 문자열일 수 있습니다. Extension Manager에서는 ID 속성을 사용하므로 MXP 파일은 새 taglibrary 및 tags 파일을 TagLibraries.vtm 파일에 삽입할 수 있습니다.	선택 사항
taglibrary.tagchooser	이 태그 라이브러리와 연관된 TagChooser.xml 파일의 상대 경로입니다.	선택 사항

다음 표에서는 tagref 속성을 설명합니다.

속성	설명	필수/선택 사항
tagref.name	이 속성은 UI에서 태그를 참조하는 데 사용됩니다.	필수
tagref.prefix	태그가 [코드] 뷰에 나타나는 방식을 지정합니다. tagref.prefix 속성을 사용하면 현재 태그의 접두어가 결정됩니다. 이 속성이 정의되면 taglibrary.prefix 속성에 대해 지정된 값이 무시됩니다.	선택 사항
tagref.file	태그에 대한 VTML 파일을 참조합니다.	선택 사항

tagref.prefix 속성은 taglibrary.prefix 속성보다 우선할 수 있으므로 두 속성 간의 관계에 혼란이 일어날 수 있습니다. 다음 표에서는 taglibrary.prefix 속성과 tagref.prefix 속성 간의 관계를 보여 줍니다.

taglibrary.prefix 정의 여부	tagref.prefix 정의 여부	생성되는 태그 접두어
아니오	아니오	'<' + tagref.name
예	아니오	taglibrary.prefix + tagref.name
아니오	예	tagref.prefix
예	예	tagref.prefix

태그를 정의하기 위해 Dreamweaver에서는 VTML 파일 형식의 수정된 버전을 사용합니다. 다음 예제에서는 Dreamweaver MX에서 개별 태그를 정의하기 위해 사용해야 하는 모든 요소를 보여 줍니다.

```
<tag name="input" bind="value" casesensitive="no" endtag="no">
    <tagformat indentcontents="yes" formatcontents="yes" nlbeforetag nlbeforecontents=0
    nlaftercontents=0 nlaftertag=1 />
    <tagdialog file = "input.HTM"/>
    <attributes>
        <attrib name="name"/>
        <attrib name="wrap" type="Enumerated">
            <attriboption value="off"/>
            <attriboption value="soft"/>
            <attriboption value="hard"/>
        /attrib>
        <attrib name="onFocus" casesensitive="yes"/>
        <event name="onFocus"/>
    </attributes>
</tag>
```

다음 표에서는 태그를 정의하는 속성을 설명합니다.

속성	설명	필수/선택 사항
tag.bind	[데이터 바인딩] 패널에서 사용됩니다. 이 유형의 태그를 선택하면 bind 속성은 데이터 바인딩에 대한 기본 속성을 나타냅니다.	선택 사항
tag.casesensitive	태그 이름이 대/소문자를 구분할지 여부를 지정합니다. 태그가 대/소문자를 구분하면 이 태그는 태그 라이브러리에 지정된 것과 동일한 대/소문자 형태로 사용자의 문서에 삽입됩니다. 태그가 대/소문자를 구분하지 않으면 [환경 설정] 대화 상자의 [코드 서식] 탭에 지정된 기본 대/소문자를 사용하는 태그가 삽입됩니다. casesensitive가 생략되면 태그가 대/소문자를 구분하지 않는 것으로 간주됩니다.	선택 사항
tag.endtag	이 속성은 태그에 열기 및 닫기 태그가 모두 있는지 여부를 지정합니다. 예를 들어 input 태그에는 닫기 태그가 없으므로 짝을 이루는 /input 태그가 없습니다. 닫기 태그가 선택 사항인 경우에는 ENDTAG 속성을 Yes로 설정해야 합니다. xml을 지정하여 빈 태그에 XML 구문을 적용합니다. 예: <tag name="foo" endtag="xml" tagtype="empty"> inserts <foo/>와 같이 지정합니다.	선택 사항
tagformat	태그의 서식 규칙을 지정합니다. Dreamweaver MX보다 이전 버전의 Dreamweaver에서는 이러한 규칙이 SourceFormat.txt에 저장되었습니다.	선택 사항
tagformat.indentcontents	이 태그의 내용을 들여쓸지 여부를 지정합니다.	선택 사항
tagformat.formatcontents	이 태그의 내용을 파싱할지 여부를 지정합니다. 이 속성은 태그 내용이 HTML 이외의 형식인 SCRIPT 및 STYLE과 같은 태그에 대해 No로 설정됩니다.	선택 사항
tagformat.nlbeforetag	이 태그 앞에 개행 문자를 삽입할지 여부를 지정합니다. 값 0은 아니오를, 값 1은 예를 나타냅니다.	선택 사항
tagformat.nlbeforecontents	이 태그의 내용 앞에 삽입할 개행 문자의 개수입니다.	선택 사항
tagformat.nlaftercontents	이 태그의 내용 뒤에 삽입할 개행 문자의 개수입니다.	선택 사항
tagformat.nlaftertag	이 태그 뒤에 개행 문자를 삽입할지 여부를 지정합니다. 값 0은 아니오를, 값 1은 예를 나타냅니다.	선택 사항
attrib.name	소스 코드에 표시되는 속성 이름입니다.	필수
attrib.type	이 속성이 생략되면 attrib.type은 "text"입니다. 사용할 수 있는 값은 TEXT(사용자 정의 텍스트 내용), ENUMERATED(열거된 값 목록), COLOR(색상 값(이름 또는 16진수)), FONT(글꼴 이름 또는 글꼴군), STYLE(CSS 스타일 속성), CSSSTYLE(CSS 클래스 이름), CSSID(CSS 클래스 ID), FILEPATH(전체 파일 경로), DIRECTORY(폴더 경로), FILENAME(파일 이름 전용), RELATIVEPATH(상대 경로), FLAG(값이 없는 ON/OFF 속성)입니다.	선택 사항
attrib.casesensitive	속성 이름이 대/소문자를 구분할지 여부를 지정합니다. CASESENSITIVE 속성이 없으면 속성 이름은 대/소문자를 구분하지 않습니다.	선택 사항

참고: Dreamweaver MX보다 이전 버전의 Dreamweaver에서는 태그 정보가 Configuration/TagAttributeList.txt 파일에 저장됩니다.

태그 선택기

태그 선택기를 사용하면 태그를 기능별 그룹으로 나누어 볼 수 있으므로 자주 사용하는 태그에 쉽게 액세스할 수 있습니다. 태그 선택기에 태그 또는 일련의 태그를 추가하려면 추가할 태그를 태그 라이브러리에 추가해야 합니다. [태그 라이브러리 편집기] 대화 상자를 사용하거나 MXP 파일로 패키지화된 Dreamweaver Extension을 설치하여 이 작업을 수행할 수 있습니다.

TagChooser.xml 파일

TagChooser.xml 파일은 태그 선택기에 나타나는 태그 그룹을 구성하기 위한 메타데이터를 제공합니다. Dreamweaver에서 제공하는 각 태그는 기능별 그룹에 저장되며 태그 선택기에서 사용할 수 있습니다. TagChooser.xml 파일을 편집하여 새 태그를 그룹화하고 기존 태그를 다시 그룹화할 수 있습니다. 하위 범주를 만들어서 태그를 사용자에 맞도록 구성하는 방식을 사용자 정의하면 가장 중요한 태그에 손쉽게 액세스할 수 있습니다.

TagLibraries.vtm 파일은 TagChooser.xml 파일을 가리키는 taglibrary.tagchooser 속성의 사용을 지원합니다. 기존 TagChooser.xml 파일을 변경하거나 새 파일을 만드는 경우 taglibrary.tagchooser 속성은 태그 선택기가 제 기능을 충분히 발휘할 수 있도록 올바른 위치를 가리켜야 합니다.

taglibrary.tagchooser 속성이 없으면 태그 선택기는 TagLibraries.vtm 파일에 있는 트리 구조를 표시합니다.

TagChooser.xml 파일은 Configuration/TagLibraries/**TagLibraryName** 폴더에 저장됩니다. 다음 예제에서는 TagChooser.xml 파일의 구조를 보여 줍니다.

```
<?xml version="1.0" encoding="iso-8859-1" standalone="yes" ?>
<tclibrary name="Friendly name for library node" desc='Description for incorporated
    reference' reference="Language[,Topic[,Subtopic]]">
    <category name="Friendly name for category node" desc='Description for incorporated
    reference' reference="Language[,Topic[,Subtopic]]" id="Unique id">
        <category name="Friendly name for subcategory node" ICON="Relative path"
        desc='Description for incorporated reference' reference="Language,Topic[,Subtopic]"
        id="Unique id">
            <element name="Friendly name for list item" value='Value to pass to visual dialog
            editors' desc='Description for incorporated reference'
            reference="Language[,Topic[,Subtopic]]" id="Unique id"/>
            ... more elements to display in the list view ...
        </category>
        ... more subcategories ...
    </category>
    ... more categories ...
</tclibrary>
```

다음 표에서는 TagChooser.xml 파일에서 사용할 수 있는 태그를 설명합니다.

태그	설명	필수/선택 사항
tclibrary	이 태그는 해당 태그 라이브러리의 태그 선택기 구조를 캡슐화하는 가장 바깥쪽 태그입니다.	필수
tclibrary.name	트리 보기 노드에 나타나는 값입니다.	필수
tclibrary.desc	[태그 선택기] 대화 상자의 [태그 정보] 섹션에 나타나는 HTML 문자열 값입니다. desc 속성이 없으면 [태그 정보]에 표시할 정보를 [참조] 패널에서 가져옵니다. tclibrary.reference와 상호 교환할 수 있습니다.	선택 사항. desc와 reference는 함께 사용할 수 없습니다.
tclibrary.reference	[태그 선택기] 대화 상자의 [태그 정보] 섹션에 표시할 언어, 항목 및 하위 항목을 설명하는 값입니다. tclibrary.desc와 상호 교환할 수 있습니다.	선택 사항. desc와 reference는 함께 사용할 수 없습니다.

category 태그는 다음 표와 같이 트리 뷰의 tclibrary 노드 아래에 있는 모든 노드를 나타냅니다.

태그	설명	필수/선택 사항
category.name	트리 보기 노드에 나타나는 값입니다.	필수
category.desc	[태그 선택기] 대화 상자의 [태그 정보] 섹션에 나타나는 HTML 문자열 값입니다. desc 및 reference attr을 모두 지정하지 않으면 [태그 정보] 섹션에 아무 것도 나타나지 않습니다.	선택 사항. desc와 reference는 함께 사용할 수 없습니다.
category.reference	[태그 정보] 섹션에 표시할 언어, 항목 및 하위 항목을 설명하는 값입니다.	선택 사항. desc와 reference는 함께 사용할 수 없습니다.
category.icon	GIF 아이콘의 상대 경로 값입니다.	선택 사항
category.id	이 파일에 있는 다른 범주의 category.id 속성과는 다른 문자열입니다.	필수

다음 표에서는 삽입할 태그를 나타내는 element 태그의 속성을 설명합니다.

속성	설명	필수/선택 사항
element.name	목록 보기 항목으로 나타나는 값입니다.	필수
element.value	화면에 표시되는 대화 상자에 전달되는 코드나 매개 변수에 직접 배치되는 값입니다.	필수
element.desc	HTML 문자열 값이며 통합된 [참조] 패널에 나타납니다. 이 속성을 지정하지 않으면 reference 속성은 통합된 [참조] 패널에 참조 내용을 표시합니다.	선택 사항. desc와 reference는 함께 사용할 수 없습니다.
element.reference	쉼표로 구분된 세 개의 문자열로서, 각각 언어, 항목 및 하위 항목을 설명합니다. 이 정보는 [참조] 패널에 나타납니다. 첫 번째 문자열은 필수입니다. 두 번째 문자열은 element 태그에만 필수이고 category 및 tclibrary 태그에는 선택 사항입니다. 세 번째 문자열은 선택 사항입니다.	선택 사항. desc와 reference는 함께 사용할 수 없습니다.
element.id	이 파일에 있는 다른 요소의 element.id 속성과는 다른 문자열입니다.	선택 사항

간단한 새 태그 편집기 만들기 예제

이 단원의 예제에서는 날씨 데이터베이스에서 현재 온도를 추출하기 위해 작성된 가상의 ColdFusion 태그인 cfweather를 사용하여 새 태그 편집기를 만드는 데 필요한 단계를 설명합니다.

다음 표에서는 cfweather 태그의 속성을 설명합니다.

속성	설명
zip	다섯 자리 숫자로 구성된 우편 번호
tempaturescale	온도 척도(Celsius 또는 Farhenheit)

이 새 태그 편집기를 만들려면 태그를 등록하고, 태그 정의를 만들고, 태그 편집기 UI를 만들고, 태그 선택기에 태그를 추가합니다.

태그 라이브러리에 태그 등록

Dreamweaver에서 새 태그가 인식되도록 하려면 Configuration/TagLibraries 폴더에 있는 TagLibraries.vtm 파일에서 이 태그를 식별해야 합니다. 그러나 사용자가 Windows XP, Windows 2000, Windows NT, Mac OS X 등의 다중 사용자 플랫폼에 있을 수도 있습니다. 사용자가 다중 사용자 플랫폼에 있는 경우에는 사용자 Configuration 폴더에 또 다른 TagLibraries.vtm 파일이 있습니다. 이 파일은 Dreamweaver에서 검색하고 파싱하는 인스턴스이므로 업데이트가 필요합니다.

사용자의 Configuration 폴더 위치는 사용자의 플랫폼에 따라 다릅니다.

Windows 2000 및 Windows XP 플랫폼의 경우 Configuration 폴더의 위치는 다음과 같습니다.

```
<drive>:\Documents and Settings\<username>\Application Data\Adobe\\xc2
Dreamweaver CS5\Configuration
```

참고: Windows XP에서는 이 폴더가 숨겨진 폴더 내에 있을 수 있습니다.

Mac OS X 플랫폼의 경우 Configuration 폴더의 위치는 다음과 같습니다.

```
<drive>:Users:<username>:Library:Application Support:Adobe:¬
Dreamweaver CS5:Configuration
```

Dreamweaver에서는 사용자의 Configuration 폴더에서 TagLibraries.vtm 파일을 찾을 수 없으면 Dreamweaver Configuration 폴더에서 해당 파일을 검색합니다.

참고: 다중 사용자 플랫폼에서 Dreamweaver Configuration 폴더에 있는 TagLibraries.vtm의 사본을 편집하는 경우 Dreamweaver에서 변경 내용이 인식되지 않습니다. Dreamweaver에서는 Dreamweaver Configuration 폴더가 아닌 사용자의 Configuration 폴더에 있는 TagLibraries.vtm 파일의 사본을 파싱하기 때문입니다.

cfweather 태그는 ColdFusion 태그이므로 cfweather 태그를 등록하는 데 사용할 수 있는 적절한 태그 라이브러리 그룹이 존재합니다.

1. TagLibraries.vtm 파일을 텍스트 편집기에서 엽니다.
2. 기존 태그 라이브러리를 스크롤하여 CFML 태그를 찾습니다.
3. 다음 예제와 같이 새 태그 참조 요소를 추가합니다.

```
<tagref name="cfweather" file="cfml/cfweather.vtm"/>
```

4. 파일을 저장합니다.

이 태그는 이제 태그 라이브러리에 등록되었으며 cfweather.vtm 태그 정의 파일에 대한 파일 포인터를 가집니다.

태그 정의(VTML) 파일 만들기

사용자가 태그 선택기 또는 태그 편집기를 사용하여 등록된 태그를 선택하면 Dreamweaver에서는 태그 정의에 해당하는 VTML 파일을 검색합니다.

1. 텍스트 편집기에서 다음 내용을 포함하는 파일을 만듭니다.

```
<TAG NAME="cfweather" endtag="no">
            <TAGFORMAT NLBEFORETAG="1" NLAFTERTAG="1"/>
            <TAGDIALOG FILE="cfweather.htm"/>

            <ATTRIBUTES>
                <ATTRIB NAME="zip" TYPE="TEXT"/>
                <ATTRIB NAME="tempaturescale" TYPE="ENUMERATED">
                    <ATTRIBOPTION VALUE="Celsius"/>
                    <ATTRIBOPTION VALUE="Fahrenheit"/>
                </ATTRIB>
            </ATTRIBUTES>
</TAG>
```

2. cfweather.vtm 파일을 Configuration/Taglibraries/CFML 폴더에 저장합니다.

태그 정의 파일을 사용하면 Dreamweaver에서는 cfweather 태그에 대해 코드 힌트, 코드 완성 및 태그 서식 지정 기능을 수행할 수 있습니다.

태그 편집기 UI 만들기

1. cfweather.htm 파일을 Configuration/Taglibraries/CFML 폴더에 저장합니다.

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<html>
<head>
<title>CFWEATHER</title>
<script src="../../Shared/Common/Scripts/dwscripts.js"></script>
<script src="../../Shared/Common/Scripts/ListControlClass.js"></script>
<script src="../../Shared/Common/Scripts/tagDialogsCmn.js"></script>
<script>

/************************ GLOBAL VARS *************************/
var TEMPATURESCALELIST;    // tempaurelist control (initialized in initializeUI())
var theUIObjects;          // array of UI objects used by common API functions
/**************************************************************/

// inspectTag() API function defined (required by all tag editors)
function inspectTag(tagNodeObj)
{
    // call into a common library version of inspectTagCommon defined
    // in tagDialogCmns.js (note that it's been included)
    // For more information about this function, look at the comments
    // for inspectTagCommon in tagDialogCmn.js
    tagDialog.inspectTagCommon(tagNodeObj, theUIObjects);
}
function applyTag(tagNodeObj)
{
    // call into a common library version of applyTagCommon defined
    // in tagDialogCmns.js (note that it's been included)
    // For more information about this function, look at the comments
    // for applyTagCommon in tagDialogCmn.js
    tagDialog.applyTagCommon(tagNodeObj, theUIObjects);
}
function initializeUI()
{
    // define two arrays for the values and display captions for the list
    control
    var theTempatureScaleCap = new Array("celsius","fahrenheit");
    var theTempatureScaleVal = new Array("celsius","fahrenheit");

    // instantiate a new list control
    TEMPATURESCALELIST = new ListControl("thetempaturescale");

    // add the tempaturescalelist dropdown list control to the uiobjects
    theUIObjects0= new Array(TEMPATURESCALELIST);

    // call common populateDropDownList function defined in tagDialogCmn.js to
    // populate the tempaturescale list control
    tagDialog.populateDropDownList(TEMPATURESCALELIST, theTempatureScaleCap,
    theTempatureScaleVal, 1);
}
</script>

</head>
<body onLoad="initializeUI()">
<div name="General">
```

```
    <table border="0" cellspacing="4">
        <tr>
            <td valign="baseline" align="right" nowrap="nowrap">Zip Code: </td>
            <td nowrap="nowrap">
                <input type="text" id="attr:cfargument:zip" name="thezip" attname="zip"
                style="width:100px"0/> 
            </td>
        </tr>
        <tr>
            <td valign="baseline" align="right" nowrap="nowrap">Type: </td>
            <td nowrap="nowrap">
                <select name="thetempaturescale" id="attr:cfargument:tempaturescale"
                attname="tempaturescale" editable="false" style="width:200px">
                </select>
            </td>
        </tr>
    </table>
</div>
</body>
</html>
```

그런 다음 태그 편집기가 작동하는지 확인합니다.

2 Dreamweaver를 시작합니다.

3 [코드] 뷰에 **cfweather**를 입력합니다.

4 태그를 마우스 오른쪽 버튼으로 클릭합니다.

5 컨텍스트 메뉴에서 [태그 편집 cfweather]를 선택합니다.

태그 편집기가 실행되면 성공적으로 만들어진 것입니다.

태그 선택기에 태그 추가

1 다음 예제에서처럼 Configuration/Taglibraries/CFML 폴더의 TagChooser.xml 파일을 수정하여 cfweather 태그를 비롯한 타사 태그를 포함하는 [Third Party Tags]라는 새 범주를 추가합니다.

```
<category name="Third Party Tags" icon="icons/Elements.gif" reference='CFML'>
    <element name="cfweather" value='cfweather zip="" temperaturescale="fahrenheit">' />
</category>
```

참고: 다중 사용자 플랫폼에서는 사용자의 Configuration 폴더에도 TagChooser.xml 파일이 있습니다. 다중 사용자 플랫폼에 대한 자세한 내용은 195페이지의 “태그 라이브러리에 태그 등록”을 참조하십시오.

그런 다음 cfweather 태그가 태그 선택기에 나타나는지 확인합니다.

2 [삽입] > [태그]를 선택합니다.

3 [CFML 태그] 그룹을 확장합니다.

4 태그 선택기 아래쪽에 나타나는 [Third Party Tags] 그룹을 선택합니다. cfweather 태그가 오른쪽의 목록 상자에 나타납니다.

5 cfweather를 선택하고 [삽입] 버튼을 클릭합니다.

태그 편집기가 나타납니다.

태그 편집기 API 함수

새 태그 편집기를 만들려면 inspectTag(), validateTag() 및 applyTag() 함수를 구현해야 합니다. 구현 예제를 보려면 196페이지의 “태그 편집기 UI 만들기”를 참조하십시오.

inspectTag()

지원 버전
Dreamweaver MX

설명
이 함수는 태그 편집기가 처음 나타날 때 호출됩니다. 이 함수는 사용자가 편집하고 있는 태그를 인수로 받으며 이 태그는 dom 객체로 표현됩니다. 이 함수는 편집하고 있는 태그에서 속성 값을 추출하고 이 값을 사용하여 태그 편집기에서 양식 요소를 초기화합니다.

인수
tag

- tag 인수는 편집되는 태그의 DOM 노드입니다.

반환값
없음

예제
사용자가 다음 태그를 편집한다고 가정합니다.

```
<crfweather zip = "94065"/>
```

편집기에 zip 속성을 편집하기 위한 텍스트 필드가 포함되어 있으면 이 함수는 텍스트 필드가 빈 필드로 표시되지 않고 실제 우편 번호를 표시하도록 양식 요소를 초기화해야 합니다.

다음 코드는 초기화를 수행합니다.

```
function inspectTag(tag)
{
    document.forms[0].zip.value = tag.zip
}
```

validateTag()

지원 버전
Dreamweaver MX

설명
사용자가 트리 컨트롤에서 임의의 노드를 클릭하거나 [확인]을 클릭하면 이 함수는 현재 표시된 HTML 양식 요소에 대해 입력 유효성 검사를 수행합니다.

인수
없음.

반환값
부울 값을 반환합니다. HTML 양식 요소에 대한 입력이 올바르면 true이고, 그렇지 않으면 false입니다.

예제
사용자가 표를 만들 때 표의 행 수로 음의 정수를 입력하면 validateTag() 함수는 잘못된 입력을 감지하고 경고 메시지를 표시한 다음 false 값을 반환합니다.

applyTag()

지원 버전
Dreamweaver MX

설명
사용자가 [확인]을 클릭하면 Dreamweaver에서는 validateTag() 함수를 호출합니다. validateTag() 함수가 true 값을 반환하면 Dreamweaver에서는 이 함수를 호출하고 현재 태그, 즉 편집 중인 태그를 나타내는 dom 객체를 전달합니다. 이 함수는 해당 양식 요소의 값을 읽어 dom 객체에 씁니다.

인수
tag

- tag 인수는 편집 중인 태그의 DOM 노드입니다.

반환값
없음

예제
다음 코드에서도 cfweather 예제가 계속 사용됩니다. 사용자가 우편 번호를 94065에서 53402로 변경한 경우, 새 우편 번호가 사용되도록 사용자의 문서를 업데이트하려면 dom 객체를 업데이트해야 합니다.

```
function applyTag(tag)
{
    tag.zip = document.forms[0].zip.value
}
```

14장: 속성 관리자

속성 관리자는 인터페이스에서 가장 익숙한 부동 패널로서, 선택 대상의 이름, 크기, 모양 및 기타 속성을 정의, 검토 및 변경하는 데 필수적이며 선택한 요소의 내부 및 외부 편집기를 실행하는 데도 사용됩니다.

다음 표에서는 속성 관리자를 만들 때 사용하는 파일을 설명합니다.

경로	파일	설명
Configuration/Inspectors/	**Propertyinspectorname**.htm	속성 관리자의 UI(사용자 인터페이스)를 정의합니다.
Configuration/Inspectors/	**Propertyinspectorname**.js	속성 관리자에 필요한 함수가 들어 있습니다.
Configuration/Inspectors/	**Tagimagefile**.gif	속성 관리자에 표시할 선택적 파일입니다.

속성 관리자 파일

Dreamweaver에는 다양한 표준 HTML 태그의 속성을 설정하는 데 사용할 수 있는 속성 관리자에 대한 여러 인터페이스가 내장되어 있습니다. 이러한 내장 관리자는 핵심적인 Dreamweaver 코드의 일부이므로 내장 관리자에 해당하는 속성 관리자 파일은 Configuration 폴더에서 찾을 수 없습니다. 그러나 사용자 정의 속성 관리자 파일을 사용하면 이러한 내장 인터페이스를 재정의하거나 새 인터페이스를 만들어 사용자 정의 태그를 관리할 수 있습니다. 사용자 정의 속성 관리자 파일은 Dreamweaver 응용 프로그램 폴더 내의 Configuration/Inspectors 폴더에 있습니다.

속성 관리자 HTML 파일에는 doctype 주석뿐 아니라 열기 HTML 태그 바로 앞의 주석이 있어야 합니다. 예를 들면 다음과 같습니다.

```
<!-- tag:serverModel:tagNameOrKeyword,priority:1to10,selection:¬ exactOrWithin,hline,vline, serverModel-->
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//pi">
```

이 주석에는 다음 요소가 포함됩니다.

- ***serverModel*** 요소는 지정된 서버 모델이 활성 상태인 경우에만 이 속성 관리자가 Dreamweaver에 로드되도록 지정합니다.
- tagNameOrKeyword 요소는 관리할 태그이거나 *COMMENT*(주석), *LOCKED*(잠긴 영역) 또는 *ASP*(ASP 태그) 키워드 중 하나입니다.
- 1~10 요소는 속성 관리자 파일의 우선 순위입니다. 1은 선택 영역을 관리할 수 있는 다른 관리자가 없는 경우에만 이 관리자가 사용됨을 나타내고, 10은 선택 영역을 관리할 수 있는 다른 모든 관리자보다 이 관리자가 우선적으로 사용됨을 나타냅니다.
- ***exactOrWithin*** 요소는 선택 영역이 태그 내부에 있을 수 있는지(within) 아니면 태그를 정확히 포함해야 하는지(exact)를 나타냅니다.
- ***hline*** 요소(선택 사항)는 확장 모드에서 관리자의 상단과 하단 사이에 회색의 수평선이 표시된다는 것을 나타냅니다.
- vline 요소(선택 사항)는 태그 이름 필드와 관리자의 나머지 속성 사이에 회색의 수직선이 표시된다는 것을 나타냅니다.
- ***serverModel*** 요소(선택 사항)는 속성 관리자의 서버 모델을 나타냅니다. 속성 관리자의 서버 모델은 문서의 속성 관리자 서버 모델과 같아야 합니다. 그렇지 않은 경우 Dreamweaver에서는 현재 선택 영역의 속성을 표시하는 데 속성 관리자를 사용하지 않습니다. 예를 들어 문서의 서버 모델이 Adobe ColdFusion이지만 속성 관리자의 서버 모델은 ASP인 경우, Dreamweaver에서는 문서의 선택 영역에 대해 속성 관리자를 사용하지 않습니다.

다음은 happy 태그를 관리하도록 디자인된 관리자에 적합한 주석입니다.

```
<!-- tag:happy, priority:8,selection:exact,hline,vline,serverModel:ASP -->
```

일부 경우에는 Extension이 이전 렌더링 엔진이 아니라 Dreamweaver Extension 렌더링만 사용하도록 지정해야 합니다. 이렇게 하려면 태그 주석 바로 앞에 다음 예제에 표시된 것과 같은 행을 삽입합니다.

```
<!--DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//pi"-->
```

속성 관리자 파일의 body 섹션에는 HTML 양식이 있습니다. 그러나 Dreamweaver에서는 이 양식 내용을 대화 상자에 표시하는 것이 아니라 이 양식을 사용하여 속성 관리자의 입력 영역 및 레이아웃을 정의합니다.

속성 관리자 파일의 head 섹션에는 JavaScript 함수나 JavaScript 파일에 대한 참조가 있습니다.

속성 관리자 파일의 작동 방식

Dreamweaver는 시작 시 Configuration/Inspectors 폴더에 있는 HTM 및 HTML 파일의 첫 번째 행을 읽어 속성 관리자의 유형, 우선 순위 및 선택 유형을 정의하는 주석 문자열을 검색합니다. 첫 번째 행에 이러한 주석이 없는 파일은 무시됩니다.

사용자가 Dreamweaver에서 항목을 선택하거나 삽입 포인터를 다른 위치로 이동하면 다음과 같은 이벤트가 발생합니다.

1. Dreamweaver가 선택 유형이 within인 관리자를 검색합니다.
2. within 관리자가 있는 경우 Dreamweaver가 현재 선택된 태그에서 문서 트리를 검색하여 선택 영역 바깥쪽의 태그에 대한 관리자가 있는지 확인합니다. within 관리자가 없는 경우에는 선택 유형이 exact인 관리자를 검색합니다.
3. 관리자가 하나 이상 있는 첫 번째 태그에 대해 Dreamweaver가 각 관리자의 canInspectSelection() 함수를 호출합니다. 이 함수가 false 값을 반환하면 Dreamweaver에서는 더 이상 해당 관리자가 선택 영역을 관리하지 않는 것으로 간주합니다.
4. canInspectSelection() 함수를 호출한 후 남아 있는 가능한 관리자가 둘 이상이면 Dreamweaver가 남아 있는 관리자를 우선 순위에 따라 정렬합니다.
5. 우선 순위가 같은 가능한 관리자가 둘 이상이면 Dreamweaver가 이름의 사전순으로 관리자를 선택합니다.
6. 선택한 관리자가 속성 관리자 부동 패널에 나타납니다. 속성 관리자 파일에 displayHelp() 함수가 정의되어 있으면 작은 물음표(?) 아이콘이 관리자의 오른쪽 위에 나타납니다.
7. Dreamweaver가 inspectSelection() 함수를 호출하여 현재 선택 영역에 대한 정보를 수집하고 관리자 필드를 채웁니다.
8. 속성 관리자 인터페이스의 필드에 연결된 이벤트 핸들러는 사용자가 해당 필드에 대한 작업을 수행할 때 실행됩니다. 예를 들어 setAttribute() 함수를 호출하여 사용자가 입력한 값으로 속성을 설정하는 onBlur 이벤트가 있을 수 있습니다.

간단한 속성 관리자 예제

다음 속성 관리자는 Microsoft Internet Explorer에서만 사용할 수 있는 marquee 태그를 관리합니다. 이 예제에서는 속성 관리자에서 direction 속성의 값을 설정할 수 있습니다. marquee 태그의 다른 속성 값을 설정하려면 이 예제를 모델로 사용하십시오.

이 Extension을 만들려면 사용자 인터페이스를 만들고 JavaScript 코드를 작성하고 이미지를 만든 다음 테스트합니다.

사용자 인터페이스 만들기

속성 관리자에 표시되는 양식이 포함된 HTML 파일을 만듭니다.

1. 비어 있는 새 파일을 만듭니다.
2. 다음과 같이 속성 관리자를 식별하는 주석을 파일의 첫 번째 행으로 추가합니다.

   ```
   <!-- tag:MARQUEE,priority:9,selection:exact,vline,hline -->
   ```

3. 문서 제목과 만들려는 JavaScript 파일을 지정하려면 주석 뒤에 다음을 추가합니다.

```
<HTML>
<HEAD>
<TITLE>Marquee Inspector</TITLE>
<SCRIPT src="marquee.js"></SCRIPT>
</HEAD>
<BODY>

</BODY>
</HTML>
```

4 속성 관리자에 나타나는 내용을 지정하려면 열기 및 닫기 body 태그 사이에 다음을 추가합니다.

```
<!-- Specify the image that will appear in the Property inspector -->
<SPAN ID="image" STYLE="position:absolute; width:23px; height:17px; ¬
    z-index:16; left: 3px; top: 2px">
    <IMG SRC="marquee.png" WIDTH="36" HEIGHT="36" NAME="marqueeImage">
</SPAN>
<SPAN ID="label" STYLE="position:absolute; width:23px; height:17px; ¬
    z-index:16; left: 44px; top: 5px">Marquee</SPAN>

<!-- If your form fields are in different AP elements, you must ¬
    create a separate form inside each AP element and reference it as ¬
    shown in the inspectSelection() and setInterjectionTag() functions. -->

<SPAN ID="topLayer" STYLE="position:absolute; z-index:1; left: 125px; ¬
    top: 3px; width: 431px; height: 32px">
<FORM NAME="topLayerForm">
    <TABLE BORDER="0" CELLPADDING="0" CELLSPACING="0">
        <TR>
            <TD VALIGN="baseline" ALIGN="right">Direction:</TD>
            <TD VALIGN="baseline" ALIGN="right">
            <SELECT NAME="marqDirection" STYLE="width:86"
                onChange="setMarqueeTag()">
                <OPTION VALUE="left">Left</OPTION>
                <OPTION VALUE="right">Right</OPTION>
            </SELECT>
        </TR>
    </TABLE>
</FORM>
</SPAN>
```

5 이 파일을 Configuration/Inspectors 폴더에 marquee.htm이라는 이름으로 저장합니다.

JavaScript 코드 작성

선택 영역을 관리하고 속성 관리자에서 적절한 값을 입력할 수 있게 하려면 JavaScript 함수를 추가해야 합니다.

1 비어 있는 새 파일을 만듭니다.

2 선택 영역에 marquee 태그가 포함되어 있을 때마다 속성 관리자가 나타나도록 지정하려면 다음 함수를 추가합니다.

```
function canInspectSelection(){
    return true;
}
```

3 텍스트 필드에 나타나는 direction 속성의 값을 새로 고치려면 파일 끝에 다음 함수를 추가합니다.

```
function inspectSelection(){
    // Get the DOM of the current document.
    var theDOM = dw.getDocumentDOM();
    // Get the selected node.
    var theObj = theDOM.getSelectedNode();

    // Get the value of the DIRECTION attribute on the MARQUEE tag.
    var theDirection = theObj.getAttribute('direction');

    // Initialize a variable for the DIRECTION attribute to -1.
    // This is used to store the menu index that corresponds to
    // the value of the attribute.
    // var typeIndex = -1;
    var directionIndex = -1;

    // If there was a DIRECTION attribute...
    if (theDirection){
        // If the value of DIRECTION is "left", set typeIndex to 0.
        if (theDirection.toLowerCase() == "left"){
            directionIndex = 0;
        // If the value of DIRECTION is "right", set typeIndex to 1.
            }else if (theDirection.toLowerCase() == "right"){
                directionIndex = 1;
                }
            }
    // If the value of the DIRECTION attribute was "left"
    // or "right", choose the corresponding
    // option from the pop-up menu in the interface.
    if (directionIndex != -1){
        document.topLayer.document.topLayerForm.marqDirection.selectedIndex =¬
        directionIndex;
    }
}
```

4 현재 선택 영역을 가져온 다음 속성 관리자의 텍스트 상자에 direction 속성 값이 표시되도록 하려면 파일 끝에 다음 함수를 추가합니다.

```
function setMarqueeTag(){
    // Get the DOM of the current document.
    var theDOM = dw.getDocumentDOM();
    // Get the selected node.
    var theObj = theDOM.getSelectedNode();

    // Get the index of the selected option in the pop-up menu
    // in the interface.
    var directionIndex = ¬
        document.topLayer.document.topLayerForm.marqDirection.selectedIndex;
    // Get the value of the selected option in the pop-up menu
    // in the interface.
    var theDirection = ¬
        document.topLayer.document.topLayerForm.marqDirection.¬
        options[directionIndex].value;

    // Set the value of the direction attribute to theDirection.
    theObj.setAttribute('direction',theDirection);
}
```

5 이 파일을 Configuration/Inspectors 폴더에 marquee.js로 저장합니다.

이미지 만들기

선택적으로 속성 관리자에 나타날 이미지를 만들 수 있습니다.

1 폭과 높이가 각각 36픽셀인 이미지를 만듭니다.

2 이 이미지를 Configuration/Inspectors 폴더에 marquee.gif라는 이름으로 저장합니다.

일반적으로 속성 관리자 이미지는 Dreamweaver에서 지원하는 어느 형식으로나 저장할 수 있습니다.

속성 관리자 테스트

마지막으로 속성 관리자를 테스트할 수 있습니다.

1 Dreamweaver를 다시 시작합니다.

2 새 HTML 페이지를 만들거나 기존 HTML 페이지를 엽니다.

3 페이지의 body 섹션에 다음을 추가합니다.

```
<MARQUEE></MARQUEE>
```

4 방금 추가한 텍스트를 강조 표시합니다.

marquee 태그에 대해 만든 속성 관리자가 나타납니다.

5 속성 관리자에서 direction 속성의 값을 입력합니다.

페이지의 태그가 direction 속성과 속성 관리자에서 입력한 값을 포함하도록 변경됩니다.

속성 관리자 API 함수

두 개의 속성 관리자 API 함수(canInspectSelection() 및 inspectSelection())는 필수입니다.

canInspectSelection()

설명

속성 관리자가 현재 선택 영역에 적합한지 확인합니다.

인수

없음

참고: 현재 선택 영역을 JavaScript 객체로 가져오려면 dom.getSelectedNode()를 사용합니다. dom.getSelectedNode()에 대한 자세한 내용은 Dreamweaver API 참조 설명서를 참조하십시오.

반환값

부울 값을 반환합니다. 관리자가 현재 선택 영역을 관리할 수 있으면 true이고, 그렇지 않으면 false입니다.

예제

선택 영역에 CLASSID 속성이 포함되어 있고 이 속성의 값이 "clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"(Adobe Flash Player의 클래스 ID)이면 canInspectSelection() 함수의 다음 인스턴스는 true 값을 반환합니다.

```
function canInspectSelection(){3
    var theDOM = dw.getDocumentDOM();
    var theObj = theDOM.getSelectedNode();
    return (theObj.nodeType == Node.ELEMENT_NODE && ¬
        theObj.hasAttribute("classid") && ¬
        theObj.getAttribute("classid").toLowerCase()== ¬
        "clsid:D27CDB6E-AE6D-11cf-96B8-444553540000");
}
```

displayHelp()

설명

이 함수가 정의된 경우에는 물음표(?) 아이콘이 속성 관리자의 오른쪽 위에 나타납니다. 사용자가 아이콘을 클릭하면 이 함수가 호출됩니다.

인수

없음

반환값

없음

예제

displayHelp() 함수의 다음 예제는 브라우저 윈도우에서 파일을 엽니다. 이 파일은 속성 관리자의 필드에 대해 설명합니다.

```
function displayHelp(){
    dw.browseDocument('http://www.hooha.com/dw/inspectors/inspHelp.html');
}
```

inspectSelection()

설명

현재 선택 영역의 속성을 기반으로 텍스트 필드의 내용을 새로 고칩니다.

인수

maxOrMin

- **maxOrMin** 인수는 속성 관리자가 확장된 상태인지 축소된 상태인지에 따라 max 또는 min입니다.

반환값

없음

예제

inspectSelection() 함수의 다음 예제는 content 속성의 값을 가져온 다음 이 값을 사용하여 keywords라는 양식 필드를 채웁니다.

```
function inspectSelection(){
    var dom = dreamweaver.getDocumentDOM();
    var theObj = dom.getSelectedNode();
    document.forms[0].keywords.value = theObj.getAttribute("content");
}
```

15장: 부동 패널

속성 관리자의 크기 및 레이아웃 제한이 없는 부동 패널 또는 관리자를 만들 수 있습니다.

현재 선택 항목에 대한 속성을 설정하려면 먼저 사용자 정의 속성 관리자를 선택해야 하지만 사용자 정의 부동 패널을 사용하면 전체 문서 또는 여러 선택 항목에 대한 정보를 표시하기에 충분한 공간과 융통성을 얻을 수 있습니다.

사용자 정의 패널을 만들어 [윈도우] 메뉴에 추가할 수 있습니다. 메뉴 시스템에 항목을 추가하는 방법에 대한 자세한 내용은 133 페이지의 “메뉴 및 메뉴 명령”을 참조하십시오.

다음 표에서는 부동 패널을 만들 때 사용하는 파일을 설명합니다.

경로	파일	설명
Configuration/Floaters/	**panelname**.htm	부동 패널의 제목 막대에 표시되는 텍스트를 지정하고, 부동 패널을 정의하며, 필요한 JavaScript 함수를 포함합니다.
Configuration/Menus/	**menus**.xml	메뉴에 명령을 추가합니다.

부동 패널 파일의 동작 방식

사용자 정의 부동 패널 파일은 응용 프로그램 폴더 내의 Configuration/Floaters 폴더에 있는 HTML 파일입니다. 부동 패널 파일의 body 섹션에는 HTML 양식이 있습니다. 양식 요소에 연결된 이벤트 핸들러는 현재 문서에 대한 임의 편집을 수행하는 JavaScript 코드를 호출할 수 있습니다.

Dreamweaver에는 [윈도우] 메뉴에서 액세스할 수 있는 여러 개의 부동 패널이 내장되어 있습니다. 이러한 내장 패널은 핵심적인 Dreamweaver 코드의 일부이므로 Configuration/Floaters 폴더에는 이 패널에 해당하는 부동 패널 파일이 없습니다.

사용자 정의 부동 패널은 Dreamweaver에 내장된 부동 패널과 같은 방식으로 이동하거나, 크기를 조절하거나, 탭을 지정할 수 있습니다. 사용자 정의 부동 패널과 내장 부동 패널의 차이점은 다음과 같습니다.

- 사용자 정의 부동 패널은 기본적으로 회색으로 표시됩니다. body 태그에 bgcolor 속성을 설정하는 것은 아무 효과가 없습니다.
- 모든 사용자 정의 부동 패널은 [문서] 윈도우의 맨 앞에 나타나거나 비활성 상태인 경우에는 [문서] 윈도우 뒤에 배치됩니다. 부동 패널이 표시되는 위치는 [패널] 환경 설정의 [기타 모든 이동 팔레트] 설정에 따라 달라집니다.

부동 패널 파일은 다른 Extension과는 약간 다릅니다. Dreamweaver는 다른 Extension 파일과는 달리, Dreamweaver를 마지막으로 종료할 때 부동 패널이 표시된 경우에만 시작할 때 부동 패널 파일을 메모리에 로드합니다. Dreamweaver 종료 시 부동 패널이 보이지 않는 경우 다음 함수 중 하나에서 참조하여 부동 패널을 정의하는 파일을 로드합니다.

- dreamweaver.getFloaterVisibility()
- dreamweaver.setFloaterVisibility()
- dreamweaver.toggleFloater()

이러한 함수에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**를 참조하십시오.

Configuration 폴더에 있는 파일 중 하나가 dw.getFloaterVisibility(***floaterName***), dw.setFloaterVisibility(***floaterName***) 또는 dw.toggleFloater(***floaterName***) 함수를 호출하면 다음 이벤트가 발생합니다.

1 ***floaterName***이 예약된 부동 패널 이름 중 하나가 아니면 Dreamweaver에서는 Configuration/Floaters 폴더에서 floaterName.htm이라는 파일을 검색합니다. 예약된 부동 패널 이름의 전체 목록을 보려면 **Dreamweaver API 참조 설명서**에서 dreamweaver.getFloaterVisibility() 함수를 참조하십시오. floaterName.htm이 검색되지 않으면 Dreamweaver에서는 floaterName.html을 검색합니다. 파일이 없으면 아무 작업도 수행되지 않습니다.

2 부동 패널 파일이 처음 로드되는 경우 initialPosition() 함수가 정의되어 있으면 이 함수가 호출되어 화면에서 부동 패널이 나타나는 기본 위치가 결정됩니다. 마찬가지로, initialTabs() 함수가 정의되어 있으면 이 함수가 호출되어 부동 패널의 기본 탭 그룹이 결정됩니다.

3 부동 패널이 숨겨져 있는 동안 변경 사항이 발생했을 것으로 가정하고 selectionChanged() 및 documentEdited() 함수가 호출됩니다.

4 부동 패널이 표시된 경우 다음 액션이 발생합니다.

- 선택 항목이 변경되면 selectionChanged() 함수가 정의된 경우 이 함수가 호출됩니다.
- 사용자가 문서를 변경하면 documentEdited() 함수가 정의된 경우 이 함수가 호출됩니다.
- 부동 패널 인터페이스의 필드에 연결된 이벤트 핸들러는 사용자가 해당 필드에 대한 작업을 수행할 때 실행됩니다. 예를 들어 dw.getDocumentDOM().body.innerHTML="을 실행하는 onClick 이벤트 핸들러가 있는 버튼은 사용자가 이 버튼을 클릭할 때 문서에서 열기 및 닫기 body 태그 사이에 있는 모든 내용을 제거합니다.

  부동 패널은 body 태그에서 onShow() 및 onHide()라는 두 가지 특수 이벤트를 지원합니다.

5 사용자가 Dreamweaver를 종료하면 부동 패널의 현재 가시성, 위치 및 탭 그룹이 저장됩니다. 다음에 Dreamweaver를 시작하면 마지막으로 종료할 때 표시되었던 부동 패널에 대한 부동 패널 파일이 로드되고 마지막 위치 및 탭 그룹에 해당 부동 패널이 표시됩니다.

간단한 부동 패널 예제

이 예제에서는 Script Editor Extension을 만듭니다. 이 Script Editor Extension은 [디자인] 뷰의 선택된 스크립트 표시기에 밑줄을 긋는 JavaScript 코드를 표시할 부동 패널을 만듭니다. Script Editor는 scriptlayer라는 부동 패널에 정의되어 있는 HTML 양식의 textarea 요소에 JavaScript 코드를 표시합니다. 부동 패널에서 선택된 코드를 변경하면 해당 Extension은 updateScript() 함수를 호출하여 변경 내용을 저장합니다. Script Editor를 호출할 때 스크립트 표시기를 선택하지 않았으면 해당 Extension은 blanklayer라는 부동 패널에 (no script selected)라고 표시합니다.

이 Extension을 만들려면 부동 패널을 만들고 JavaScript 코드를 작성한 다음 메뉴 항목을 만듭니다.

부동 패널 만들기

이 Extension에 대한 HTML 파일의 시작 부분에는 표준 문서 머리글 정보가 포함되어 있으며 부동 패널의 제목 막대에 Script Editor라는 단어를 표시하는 title 태그가 포함되어 있습니다.

HTML 파일 헤더 만들기

1 비어 있는 새 문서를 만듭니다.

2 다음을 입력합니다.

```
<!doctype html public "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title>Script Editor</title>
<script language="JavaScript">
```

이 Extension은 두 개의 부동 패널을 정의합니다. 하나는 사용자가 스크립트 표시기를 선택하지 않은 경우 (no script selected)라고 표시하는 패널이고 다른 하나는 선택된 스크립트 표시기에 밑줄을 긋는 JavaScript 코드를 표시하는 패널입니다. 다음 코드에서는 이러한 두 가지 부동 패널, 또는 blanklayer 및 scriptlayer라는 AP(절대 위치) 요소를 정의합니다.

두 개의 부동 패널 만들기

1 다음 코드를 HTML 파일의 헤더 뒤에 추가합니다.

```
<body>
<div id="blanklayer" style="position:absolute; width:422px; ¬
height:181px; z-index:1; left: 8px; top: 11px; ¬
visibility: hidden">
<center>
<br>
<br>
<br>
<br>
<br>
(no script selected)
</center>
</div>

<div id="scriptlayer" style="position:absolute; width:422px; ¬
height:181px; z-index:1; left: 8px; top: 11px; ¬
visibility: visible">
<form name="theForm">
<textarea name="scriptCode" cols="80" rows="20" wrap="VIRTUAL" ¬
onBlur="updateScript()"></textarea>
</form>
</div>

</body>
</html>
```

2 이 파일을 Configuration/Floaters 폴더에 scriptEditor.htm이라는 이름으로 저장합니다.

두 div 태그는 모두 style 속성을 사용하여 부동 패널의 위치(absolute), 크기(width:422px 및 height:181px), 기본 가시성 설정(visible)을 지정합니다. blanklayer 패널은 center 속성과 일련의 분리(br) 태그를 사용하여 패널의 가운데에 텍스트를 배치합니다. scriptlayer 패널은 단일 textarea가 포함된 양식을 만들어 선택된 JavaScript 코드를 표시합니다. textarea 태그는 선택된 코드가 변경되었음을 나타내는 onBlur 이벤트가 발생할 때 updateScript() 함수가 호출되어 변경된 텍스트를 해당 문서에 다시 작성하도록 지정합니다.

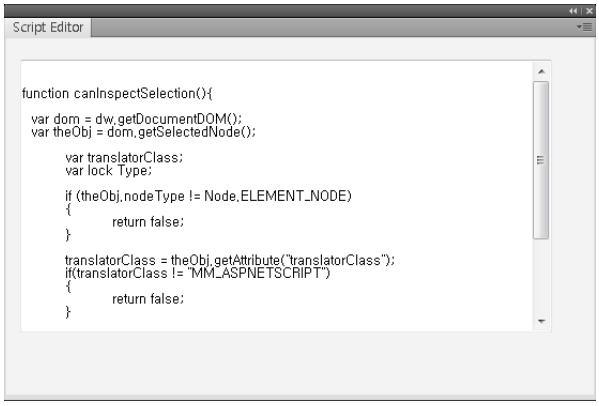
JavaScript 코드 작성

Script Editor에 대한 JavaScript 코드는 Dreamweaver에서 호출하는 부동 패널 함수인 selectionchanged()와 사용자가 정의하는 함수인 updateScript()로 구성되어 있습니다.

selectionChanged(): 스크립트 표시기의 선택 여부 결정

selectionChanged() 함수는 스크립트 표시기가 [디자인] 뷰에서 선택되었는지 여부를 확인합니다. 문서의 body 섹션에 JavaScript 루틴이 있고 [환경 설정] 대화 상자의 [보이지 않는 요소] 섹션에 [스크립트]가 선택되어 있으면 [디자인] 뷰에 스크립트 표시기가 나타납니다. 다음 그림에서는 스크립트 표시기 아이콘을 보여 줍니다.

selectionChanged() 함수는 먼저 dw.getDocumentDOM() 함수를 호출하여 사용자의 문서에 대한 DOM(문서 객체 모델)을 가져옵니다. 그런 다음 getSelectedNode() 함수를 호출하여 해당 문서에 대해 선택된 노드가 요소인지 여부와 SCRIPT 태그인지 여부를 차례로 확인합니다. 두 조건이 모두 true이면 selectionChanged() 함수는 scripteditor 부동 패널을 표시하고 기본 JavaScript 코드와 함께 로드합니다. 또한 blanklayer 부동 패널의 visibility 속성 값을 hidden으로 설정합니다. 다음 그림에서는 선택된 JavaScript 코드와 함께 scriptlayer 부동 패널을 보여 줍니다.



선택된 노드가 요소가 아니거나 script 태그가 아니면 selectionChanged() 함수는 blanklayer 부동 패널이 표시되도록 하고 scriptlayer 패널을 숨깁니다. blanklayer 부동 패널은 다음 그림과 같이 (no script selected)라는 텍스트를 표시합니다.

selectionChanged() 함수 추가

1 Configuration/Floaters 폴더에 있는 scriptEditor.htm 파일을 엽니다.

2 파일의 헤더 섹션에 다음 코드를 입력합니다.

```
function selectionChanged(){
    /* get the selected node */
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();

    /* check to see if the node is a script marker */
    if (theNode.nodeType == Node.ELEMENT_NODE && ¬
    theNode.tagName == "SCRIPT"){
        document.layers['scriptlayer'].visibility = 'visible';
        document.layers['scriptlayer'].document.theForm.¬
        scriptCode.value = theNode.innerHTML;
        document.layers['blanklayer'].visibility = 'hidden';
    }else{
        document.layers['scriptlayer'].visibility = 'hidden';
        document.layers['blanklayer'].visibility = 'visible';
    }
}
```

3 파일을 저장합니다.

updateScript():변경 내용 다시 작성

updateScript() 함수는 scriptlayer 패널의 textarea에서 onBlur 이벤트가 발생할 때 선택된 스크립트를 다시 작성합니다. textarea 양식 요소에는 JavaScript 코드가 포함되어 있으며, textarea가 입력 포커스를 잃으면 onBlur 이벤트가 발생합니다.

1 Configuration/Floaters 폴더에 있는 scriptEditor.htm 파일을 엽니다.

2 파일의 헤더 섹션에 다음 코드를 입력합니다.

```
/* update the document with any changes made by
    the user in the textarea */

    function updateScript(){
    var theDOM = dw.getDocumentDOM();
    var theNode = theDOM.getSelectedNode();
    theNode.innerHTML = document.layers['scriptlayer'].document.¬
    theForm.scriptCode.value;
}

</script>
</head>
```

3 파일을 저장합니다.

메뉴 항목 만들기

Configuration/Floaters 폴더에 Script Editor 코드를 저장하는 것만으로는 충분하지 않습니다. dw.setFloaterVisibility('scriptEditor',true) 함수 또는 dw.toggleFloater('scriptEditor') 함수를 호출하여 부동 패널을 로드하고 화면에 표시되도록 해야 합니다. Script Editor를 가장 확실하게 호출할 수 있는 위치는 menus.xml 파일에 정의되어 있는 [윈도우] 메뉴입니다. 다음 그림과 같이 Script Editor Extension에 대한 항목을 [윈도우] 메뉴에 만드는 menuitem 태그를 만들어야 합니다.

현재 문서의 [디자인] 뷰에서 스크립트 표시기를 선택하고 [Script Editor] 메뉴 항목을 선택하면 [Script Editor] 부동 패널이 호출되고 스크립트 표시기에 밑줄을 긋는 JavaScript 코드가 표시됩니다. 스크립트 표시기가 선택되지 않은 경우 메뉴 항목을 선택하면 (no script selected)라는 텍스트가 포함된 blanklayer 패널이 표시됩니다.

1 Configuration/Menus 폴더에 있는 menus.xml 파일을 엽니다.

2 <menuitem name="Tile _Vertically로 시작하는 태그를 찾은 다음 이 태그의 닫는 /> 뒤에 삽입 포인터를 놓습니다.

3 행을 바꿔 다음을 입력합니다.

```
<menuitem name="Script Editor" enabled="true" ¬
command="dw.toggleFloater('scriptEditor')"¬
checked="dw.getFloaterVisibility('scriptEditor')" />
```

4 파일을 저장합니다.

부동 패널 API 함수

부동 패널 API의 사용자 정의 함수는 모두 선택적입니다.

이 단원에 설명된 일부 함수는 Windows 운영 체제에서만 동작합니다. 이 정보는 함수 설명 부분에서 알 수 있습니다.

displayHelp()

설명

이 함수를 정의하면 대화 상자의 [확인] 및 [취소] 버튼 아래에 [도움말] 버튼이 표시됩니다. 사용자가 [도움말] 버튼을 클릭하면 이 함수가 호출됩니다.

인수

없음

반환값

없음

예제

```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

documentEdited()

설명

이 함수는 부동 패널이 표시되고 현재 수행하는 일련의 편집 작업이 완료되면 호출됩니다. 즉, 이 함수가 호출되기 전에 편집 작업이 여러 번 수행될 수 있습니다. 이 함수는 부동 패널에서 문서의 편집 내용을 추적해야 하는 경우에만 정의해야 합니다.

참고: documentEdited() 함수가 있으면 성능에 영향을 주므로 필요한 경우에만 이 함수를 정의하십시오.

인수
없음

반환값
없음

예제
documentEdited() 함수의 다음 예제에서는 문서에서 AP 요소를 검색하고 문서에 포함된 AP 요소의 수를 표시하는 텍스트 필드를 업데이트합니다.

```
function documentEdited(){
    /* create a list of all the AP elements in the document */
    var theDOM = dw.getDocumentDOM();
    var layersInDoc = theDOM.getElementsByTagName("layer");
    var layerCount = layersInDoc.length;
    /* update the numOfLayers field with the new layercount */
    document.theForm.numOfLayers.value = layerCount;
}
```

getDockingSide()

지원 버전
Dreamweaver MX

설명
부동 패널을 결합할 수 있는 위치를 지정합니다. 이 함수는 "left", "right", "top" 및 "bottom" 등의 몇 가지 단어가 조합된 문자열을 반환합니다. 문자열에 레이블이 있는 경우에는 부동 패널을 해당 면에 결합할 수 있습니다. 이 함수가 없으면 부동 패널을 어느 면에도 결합할 수 없습니다.

특정 패널이 Dreamweaver 작업 영역의 특정 면에 결합되거나 서로 결합되는 것을 방지하려면 이 함수를 사용합니다.

인수
없음

반환값
Dreamweaver에서 부동 패널을 결합할 수 있는 위치를 지정하는 "left", "right", "top" 및 "bottom"이라는 단어가 포함된 문자열이나 이러한 단어의 조합이 포함된 문자열을 반환합니다.

예제

```
getDockingSide()
{
    return dock_side = "left top";
}
```

initialPosition()

설명
부동 패널이 처음 호출될 때의 초기 위치를 결정합니다. 이 함수를 정의하지 않은 경우 기본 위치는 화면의 가운데입니다.

인수
platform

- **platform** 인수의 값은 사용자의 플랫폼에 따라 "Mac" 또는 "Win" 중 하나입니다.

반환값
"leftPosInPixels,topPosInPixels" 형식의 문자열을 반환합니다.

예제
initialPosition() 함수의 다음 예제에서는 부동 패널이 처음으로 나타날 때 Windows의 경우 화면 왼쪽에서 420픽셀, 위쪽에서 20픽셀의 위치에 나타나도록 지정하고 Macintosh의 경우 화면 왼쪽에서 390픽셀, 위쪽에서 20픽셀의 위치에 나타나도록 지정합니다.

```
function initialPosition(platform){
    var initPos = "420,20";
    if (platform == "macintosh"){
        initPos = "390,20";
    }
    return initPos;
}
```

initialTabs()

설명
이 부동 패널이 처음으로 나타날 때 함께 탭으로 표시될 다른 부동 패널을 결정합니다. 목록에 있는 부동 패널이 이전에 나타난 적이 있으면 해당 패널은 탭 그룹에 포함되지 않습니다. 두 개의 사용자 정의 부동 패널이 함께 탭으로 표시되도록 하려면 각 패널이 initialTabs() 함수를 사용하여 다른 패널을 참조해야 합니다.

인수
없음

반환값
"floaterName1,floaterName2,...floaterNameN" 형식의 문자열을 반환합니다.

예제
initialTabs() 함수의 다음 예제에서는 부동 패널이 처음으로 나타날 때 scriptEditor 부동 패널과 함께 탭으로 표시되도록 지정합니다.

```
function initialTabs(){
    return "scriptEditor";
}
```

isATarget()

지원 버전
Dreamweaver MX(Windows 전용)

설명
이 부동 패널에 다른 부동 패널을 결합할 수 있는지 여부를 지정합니다. isATarget() 함수를 지정하지 않으면 이 패널에 다른 패널이 결합되지 못합니다. 사용자가 이 패널을 다른 패널과 결합하려고 하면 이 함수가 호출됩니다.

인수
없음

반환값
부울 값을 반환합니다. 다른 부동 패널을 이 패널에 결합할 수 있으면 true이고, 그렇지 않으면 false입니다.

예제
```
isATarget()
{
    return true;
}
```

isAvailableInCodeView()

설명
[코드] 뷰를 선택할 때 부동 패널이 활성화될지 여부를 결정합니다. 이 함수를 정의하지 않으면 [코드] 뷰에서 부동 패널이 비활성화됩니다.

인수
없음

반환값
부울 값을 반환합니다. 부동 패널이 [코드] 뷰에 활성화되면 true이고, 그렇지 않으면 false입니다.

isResizable()

지원 버전
Dreamweaver 4

설명
사용자가 부동 패널의 크기를 조절할 수 있는지 여부를 결정합니다. 이 함수가 정의되어 있지 않거나 true를 반환하면 사용자가 부동 패널의 크기를 조절할 수 있습니다. 이 함수가 false를 반환하면 사용자가 부동 패널의 크기를 조절할 수 없습니다.

인수
없음

반환값
부울 값을 반환합니다. 사용자가 부동 패널의 크기를 조절할 수 있으면 true이고, 그렇지 않으면 false입니다.

예제
다음 예제에서는 사용자가 부동 패널의 크기를 조절할 수 없도록 합니다.

```
function isResizable()
{
    return false;
}
```

selectionChanged()

설명

부동 패널이 표시되거나 선택 항목이 변경되는 경우, 즉 포커스가 새 문서로 전환되거나 삽입 포인터가 현재 문서의 새 위치로 이동되는 경우에 호출됩니다. 이 함수는 부동 패널에서 선택 항목을 추적해야 하는 경우에만 정의해야 합니다.

참고: selectionChanged() 함수가 있으면 성능에 영향을 주므로 필요한 경우에만 이 함수를 정의하십시오.

인수

없음

반환값

없음

예제

selectionChanged()의 다음 예제에서는 선택 항목이 스크립트 표시기인지 여부에 따라 달라지는 부동 패널의 AP 요소를 보여 줍니다. 선택 항목이 스크립트 표시기이면 Dreamweaver에서는 scriptlayer AP 요소를 화면에 표시하고, 그렇지 않으면 blanklayer AP 요소를 표시합니다.

```
function selectionChanged(){
    /* get the selected node */
        var theDOM = dw.getDocumentDOM();
        var theNode = theDOM.getSelectedNode();
    /* check to see if the node is a script marker */
        if (theNode.nodeType == Node.ELEMENT_NODE && ¬
        theNode.tagName == "SCRIPT"){
            document.layers['blanklayer'].visibility = 'hidden';
            document.layers['scriptlayer'].visibility = 'visible';}
        else{
            document.layers['scriptlayer'].visibility = 'hidden';
            document.layers['blanklayer'].visibility = 'visible';
        }
}
```

성능

사용자 정의 부동 패널에서 selectionChanged() 또는 documentEdited() 함수를 선언하면 Dreamweaver의 성능에 좋지 않은 영향을 줄 수 있습니다. 예를 들어 Dreamweaver가 0.1초 이상 유휴 상태인 경우 사용자가 키보드의 키를 누르거나 마우스를 클릭할 때마다 documentEdited() 및 selectionChanged() 함수가 호출될 수 있습니다. 가능하면 항상 큰 문서(100K 이상의 HTML)를 사용하여 다양한 시나리오로 부동 패널을 테스트함으로써 성능에 어떠한 영향이 있는지 테스트해야 합니다.

성능 저하를 방지하려면 setTimeout() 함수를 사용하십시오. 브라우저에서와 같이 setTimeout() 함수는 두 개의 인수를 사용합니다. 이 인수는 호출할 JavaScript와 호출 전 대기 시간(밀리초)을 나타냅니다.

setTimeout() 메서드를 사용하면 처리 과정에 일시 정지 기능을 포함할 수 있습니다. 이러한 일시 정지 기능을 사용하면 사용자가 응용 프로그램과 계속해서 상호 작용할 수 있습니다. 스크립트 진행 중에는 화면이 정지되어 사용자가 편집 작업을 더 이상 수행할 수 없게 되므로 이러한 일시 정지 기능을 명시적으로 포함해야 합니다. 또한 일시 정지 기능을 사용하면 인터페이스 또는 부동 패널의 업데이트가 방지됩니다.

다음은 문서에 있는 모든 AP 요소에 대한 정보를 표시하는 부동 패널 예제 중 일부입니다. 이 예제에서는 setTimeout() 메서드를 사용하여 각 AP 요소를 처리한 후 0.5초 동안 일시 정지합니다.

```
/* create a flag that specifies whether an edit is being processed, and set it to false.*/
document.running = false;
/* this function called when document is edited */
function documentEdited(){
    /* create a list of all the AP elements to be processed */
    var dom = dw.getDocumentDOM();
    document.layers = dom.getElementsByTagName("layer");
    document.numLayers = document.layers.length;
    document.numProcessed = 0;
    /* set a timer to call processLayer(); if we didn't get
     * to finish processing the previous edit, then the timer
     * is already set. */
    if (document.running = false){
        setTimeout("processLayer()", 500);
    }
    /* set the processing flag to true */
    document.running = true;
}
/* process one AP element*/
function processLayer(){
    /* display information for the next unprocessed AP element.
        displayLayer() is a function you would write to
        perform the "magic".0*/
     displayLayer(document.layers[document.numProcessed]);
    /* if there's more work to do, set a timeout to process
     * the next layer.0.If we're finished, set the document.running
     * flag to false. */
    document.numProcessed = document.numProcessed + 1;
    if (document.numProcessed < document.numLayers){
        setTimeout("processLayer()", 500);
    }else{
        document.running = false;
    }
}
```

16장: 비헤이비어

비헤이비어라는 용어는 이벤트(예: onClick, onLoad 및 onSubmit)와 액션(예: 플러그인 확인, URL로 이동, 이미지 교체)의 조합을 나타냅니다. 어떤 HTML 요소에 어떤 이벤트를 사용할 수 있는지는 브라우저에서 결정합니다. Adobe Dreamweaver 응용 프로그램 폴더의 Configuration/Behaviors/Events 폴더에는 각 브라우저에서 지원하는 이벤트 목록이 포함된 파일이 저장되어 있습니다.

일반적으로 액션 파일의 body 섹션에는 HTML 양식이 포함됩니다. HTML 양식은 예를 들어 표시하거나 숨길 절대 위치 요소를 나타내는 매개 변수와 같이 액션에 대한 매개 변수를 사용합니다. 액션 파일의 head 섹션에는 body 내용에서 입력된 양식을 처리하는 JavaScript 함수가 포함됩니다. 이러한 함수는 사용자의 문서에 삽입된 함수, 인수 및 이벤트 핸들러를 제어하기도 합니다.

다른 사용자와 함수를 공유하거나 동일한 JavaScript 함수를 반복적으로 삽입하려는 경우에는 비헤이비어 액션을 작성합니다. 매개 변수는 매번 변경해야 합니다.

참고: 비헤이비어를 사용하여 VBScript 함수를 직접 삽입할 수는 없습니다. 그러나 applyBehavior() 함수의 DOM(문서 객체 모델)을 편집하여 VBScript 함수를 간접적으로 추가할 수는 있습니다.

다음 표에서는 비헤이비어 액션을 만드는 데 사용하는 파일을 보여 줍니다.

경로	파일	설명
Configuration/Behaviors/Actions/	**behavior action**.htm	이 파일의 본문에는 액션의 매개 변수를 지정하기 위한 HTML 양식이 들어 있습니다. 파일의 헤드에는 JavaScript 함수가 들어 있습니다.

참고: 웹 응용 프로그램 기능을 제공하는 서버 비헤이비어에 대한 자세한 내용은 230페이지의 "서버 비헤이비어"를 참조하십시오.

비헤이비어의 작동 방식

사용자가 Dreamweaver 문서에서 HTML 요소를 선택하고 [비헤이비어] 패널의 플러스(+) 버튼을 클릭하면 다음과 같은 이벤트가 발생합니다.

1. Dreamweaver가 각 액션 파일에서 canAcceptBehavior() 함수를 호출하여 해당 액션이 문서나 선택한 요소에 적합한지 확인합니다.

   이 함수의 반환값이 false이면 Dreamweaver는 [액션] 팝업 메뉴에서 해당 액션을 희미하게 표시합니다. 예를 들어 사용자의 문서에 SWF 파일이 없으면 Shockwave 제어 액션은 희미하게 표시됩니다. 반환값이 이벤트 목록이면 Dreamweaver에서는 현재 선택한 HTML 요소 및 대상 브라우저에 유효한 이벤트와 이 목록의 각 이벤트를 서로 비교하여 일치하는 것을 찾습니다. Dreamweaver는 canAcceptBehavior() 함수에서 찾은 일치하는 이벤트를 [이벤트] 팝업 메뉴 목록의 맨 위에 표시합니다. 일치하는 이벤트가 없으면 HTML 요소에 대한 기본 이벤트(이벤트 파일에서 별표[*]로 표시됨)가 맨 위 항목으로 표시됩니다. 메뉴의 나머지 이벤트는 이벤트 파일에서 수집됩니다.

2. 사용자가 [액션] 팝업 메뉴에서 액션을 선택합니다.

3. Dreamweaver가 windowDimensions() 함수를 호출하여 [매개 변수] 대화 상자의 크기를 결정합니다. windowDimensions() 함수가 정의되어 있지 않으면 크기가 자동으로 결정됩니다.

   대화 상자는 항상 나타나며 body 요소의 내용에 관계없이 오른쪽 가장자리에 [확인] 및 [취소] 버튼이 나타납니다.

4. Dreamweaver가 액션 파일의 BODY 요소가 포함된 대화 상자를 표시합니다. 액션 파일의 body 태그에 onLoad 핸들러가 포함되어 있으면 Dreamweaver에서는 이를 실행합니다.

5. 사용자가 액션의 매개 변수를 채웁니다. 사용자가 양식 필드에서 작업을 수행하면 Dreamweaver에서는 해당 필드와 연관된 이벤트 핸들러를 실행합니다.

6 사용자가 [확인]을 클릭합니다.

7 Dreamweaver가 선택한 액션 파일에서 behaviorFunction() 및 applyBehavior() 함수를 호출합니다. 이러한 함수는 사용자의 문서에 삽입된 문자열을 반환합니다.

8 사용자가 나중에 [액션] 열에서 이 액션을 두 번 클릭하면 Dreamweaver에서는 [매개 변수] 대화 상자를 다시 열고 onLoad 핸들러를 실행합니다. 그런 다음 Dreamweaver는 선택한 액션 파일에서 inspectBehavior() 함수를 호출하여 사용자가 이전에 입력한 데이터로 필드를 채웁니다.

사용자의 파일에 여러 함수 삽입

액션을 통해 여러 개의 함수, 즉 기본 비헤이비어 함수와 여러 개의 보조 함수를 head 섹션에 삽입할 수 있습니다. 각 액션 파일의 함수 정의가 완전히 같은 경우 두 개 이상의 비헤이비어가 보조 함수를 공유할 수도 있습니다. 공유 함수가 동일하도록 하는 한 가지 방법은 각 보조 함수를 외부 JavaScript 파일에 저장하고 <SCRIPT SRC="***externalFile***.js">를 사용하여 이 파일을 적절한 액션 파일에 삽입하는 것입니다.

사용자가 비헤이비어를 삭제하면 Dreamweaver에서는 이 비헤이비어와 연관된 보조 함수 중에서 사용되지 않는 보조 함수를 제거합니다. 다른 비헤이비어가 사용하고 있는 보조 함수는 삭제되지 않습니다. 보조 함수를 삭제하는 알고리즘은 특별히 주의해야 하므로 Dreamweaver에서는 사용되지 않는 함수가 사용자 문서에 그대로 남겨지기도 합니다.

반환값이 필요한 액션 처리

예를 들어 onMouseOver="window.status='This is a link'; return true"와 같이 이벤트 핸들러에 반환값이 있어야 하는 경우가 있습니다. 그러나 Dreamweaver가 이벤트 핸들러에 "return behaviorName(args)" 액션을 삽입하면 목록의 뒷부분에 있는 비헤이비어는 건너뛰게 됩니다.

이러한 제한을 해결하려면 behaviorFunction() 함수가 반환하는 문자열 내에서 document.MM_returnValue 변수를 원하는 반환값으로 설정합니다. 이와 같이 설정하면 Dreamweaver에서 이벤트 핸들러의 액션 목록 끝에 return document.MM_returnValue가 삽입됩니다. MM_returnValue 변수 사용에 대한 예제를 보려면 Dreamweaver 응용 프로그램 폴더 내의 Configuration/Behaviors/Actions 폴더에 있는 Validate Form.js 파일을 참조하십시오.

간단한 비헤이비어 예제

비헤이비어의 작동 방식과 비헤이비어를 만드는 방법을 이해하려는 경우 예제를 보면 도움이 됩니다. Dreamweaver 응용 프로그램 폴더 내의 Configuration/Behaviors/Actions 폴더에 들어 있는 예제는 대부분 상당히 복잡합니다. 이 예제는 더 간단하므로 비헤이비어를 만드는 방법을 쉽게 익힐 수 있습니다. 먼저 간단한 액션 파일인 Call JavaScript.htm과 모든 JavaScript 함수가 포함되어 있는 Call JavaScript.js를 사용하십시오.

비헤이비어를 만들려면 Extension을 만들고 탐색할 HTML 파일을 만든 다음 비헤이비어를 테스트합니다.

비헤이비어 Extension 만들기

다음 코드는 비교적 간단한 예제입니다. 브라우저 이름을 확인하여 Netscape Navigator인 경우에는 특정 페이지를 열고 Microsoft Internet Explorer인 경우에는 다른 페이지를 엽니다. 손쉽게 이 코드를 확장하여 Opera와 WebTV 같은 다른 브라우저를 확인할 수도 있으며 이 코드를 수정하여 URL로 이동하는 것 외의 다른 액션을 수행할 수도 있습니다.

1 비어 있는 새 파일을 만듭니다.

2 다음 내용을 파일에 추가합니다.

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<html>
<head>
<title>behavior "Check Browser Brand"</title>
<meta http-equiv="Content-Type" content="text/html">
<script language="JavaScript">

// The function that will be inserted into the
// HEAD of the user's document
function checkBrowserBrand(netscapeURL,explorerURL) {
    if (navigator.appName == "Netscape") {
        if (netscapeURL) location.href = netscapeURL;
    }else if (navigator.appName == "Microsoft Internet Explorer") {
        if (explorerURL) location.href = explorerURL;
    }
}

//******************* API **********************

function canAcceptBehavior(){
    return true;
}

// Return the name of the function to be inserted into
// the HEAD of the user's document
function behaviorFunction(){
        return "checkBrowserBrand";
}

// Create the function call that will be inserted
// with the event handler
function applyBehavior() {
    var nsURL = escape(document.theForm.nsURL.value);
    var ieURL = escape(document.theForm.ieURL.value);
    if (nsURL && ieURL) {
        return "checkBrowserBrand(\'" + nsURL + "\',\'" + ieURL + "\')";
    }else{
        return "Please enter URLs in both fields."
    }
}

// Extract the arguments from the function call
// in the event handler and repopulate the
// parameters form
function inspectBehavior(fnCall){
    var argArray = getTokens(fnCall, "()',");
    var nsURL = unescape(argArray[1]);
    var ieURL = unescape(argArray[2]);
    document.theForm.nsURL.value = nsURL;
    document.theForm.ieURL.value = ieURL;
}

//***************** LOCAL FUNCTIONS ******************

// Put the pointer in the first text field
// and select the contents, if any
function initializeUI(){
    document.theForm.nsURL.focus();
    document.theForm.nsURL.select();
}

// Let the user browse to the Navigator and
```

```
// IE URLs
function browseForURLs(whichButton){
    var theURL = dreamweaver.browseForFileURL();
    if (whichButton == "nsURL"){
        document.theForm.nsURL.value = theURL;
    }else{
        document.theForm.ieURL.value = theURL;
    }
}

//*************** END OF JAVASCRIPT *****************
</script>
</head>
<body>
<form method="post" action="" name="theForm">
<table border="0" cellpadding="8">
<tr>
<td nowrap="nowrap">  Go to this URL if the browser is ¬
    Netscape Navigator:<br>
<input type="text" name="nsURL" size="50" value="">  
<input type="button" name="nsBrowse" value="Browse..." ¬
    onClick="browseForURLs('nsURL')"></td>
</tr>
<tr>
<td nowrap="nowrap">  Go to this URL is the browser is ¬
    Microsoft Internet Explorer:<br>
<input type="text" name="ieURL" size="50" value="">  
<input type="button" name="ieBrowse" value="Browse..." ¬
    onClick="browseForURLs('ieURL')"></td>
</tr>
</table>
</form>
</body>
</html>
```

3 파일을 Configuration/Behaviors/Actions/BrowserDependentURL.htm으로 저장합니다.

탐색할 HTML 파일 만들기 (사용되지 않음)

탐색할 HTML 파일, 즉 브라우저가 Internet Explorer인 경우에 이동할 파일과 Netscape Navigator인 경우에 이동할 파일을 만듭니다.

1 다음 내용을 사용하여 파일을 만듭니다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Internet Explorer Only</title>
</head>

<body>
This is the page to go to if you are using Internet Explorer.
</body>
</html>
```

2 이 파일을 사용자 컴퓨터의 사이트에 iecontent.htm으로 저장합니다.

3 다음 내용이 포함된 다른 파일을 만듭니다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Netscape Navigator content</title>
</head>

<body>
This is the page to go to if you are using Netscape Navigator.
</body>
</html>
```

4 이 파일을 iecontent.htm 파일과 동일한 폴더에 netscapecontent.htm이라는 이름으로 저장합니다.

5 Dreamweaver를 다시 시작합니다.

6 다음 내용이 포함된 HTML 파일을 만듭니다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Which browser</title>
</head>

<body>
</body>
</html>
```

7 이 파일을 iecontent.htm 파일과 동일한 폴더에 whichbrowser.htm이라는 이름으로 저장합니다.

8 [비헤이비어] 패널의 플러스(+) 버튼을 클릭하고 [Check Browser Brand] 비헤이비어를 선택합니다.

9 [URL로 이동] 옵션 옆의 [탐색] 버튼을 클릭한 다음 브라우저가 Netscape Navigator인 경우 netscapecontent.htm 파일을 선택하고 브라우저가 Internet Explorer인 경우에는 iecontent.htm 파일을 선택합니다.

10 [확인]을 클릭합니다.

지정된 JavaScript는 whichbrowser.htm 파일에 추가되므로 이 파일은 다음과 같이 나타납니다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>Which browser</title>
<script language="JavaScript" type="text/JavaScript">
<!--
function checkBrowserBrand(netscapeURL,explorerURL) {
    if (navigator.appName == "Netscape") {
        if (netscapeURL) location.href = netscapeURL;
    }else if (navigator.appName == "Microsoft Internet Explorer") {
        if (explorerURL) location.href = explorerURL;
    }
}
//-->
</script>
</head>

<body onLoad="checkBrowserBrand('netscaptecontent.htm','iecontent.htm')">
</body>
</html>
```

비헤이비어 테스트

1 브라우저에서 whichbrowser.htm 파일을 엽니다.

2 현재 사용하는 브라우저에 따라 iecontent.htm 또는 netscapecontent.htm이 열립니다.

비헤이비어 API 함수

두 개의 비헤이비어 API 함수(applyBehavior() 및 behaviorFunction())는 필수이고 나머지는 선택 사항입니다.

applyBehavior()

설명

이 함수는 behaviorFunction()이 삽입하는 함수를 호출하는 이벤트 핸들러를 사용자의 문서에 삽입합니다. applyBehavior() 함수는 사용자 문서에서 기타 편집 작업도 수행할 수 있지만, 비헤이비어가 적용될 객체나 액션을 받는 객체를 삭제해서는 안 됩니다.

applyBehavior() 함수를 작성할 때에는 사용자 문서의 편집 방식을 결정해야 합니다. 예를 들어 문서 본문의 스크립트 태그 안에 몇 가지 코드를 삽입할 수 있습니다. 이렇게 하려면 표준 DOM 편집 API를 사용합니다.

인수

uniqueName

이 인수는 사용자 문서에 있는 모든 비헤이비어의 모든 인스턴스에서 고유한 식별자로, **functionNameInteger** 형식으로 사용됩니다. 여기에서 **functionName**은 behaviorFunction()이 삽입하는 함수의 이름입니다. 이 인수는 사용자 문서에 태그를 삽입하고 이 태그의 NAME 속성에 고유한 값을 지정하려는 경우에 유용합니다.

반환값

일반적으로 사용자가 매개 변수를 지정한 후에 사용자 문서에 삽입할 함수 호출을 포함하는 문자열을 반환합니다.
applyBehavior() 함수는 사용자 입력 내용이 잘못된 것으로 판단될 경우 함수 호출 대신 오류 문자열을 반환할 수 있습니다. 문자열이 비어 있으면(return "";) 오류가 보고되지 않습니다. 문자열이 비어 있지 않고 함수 호출이 아니면 **이 비헤이비어에 대한 입력이 잘못되었다는 내용의 대화 상자와 함께 applyBehavior()에서 반환된 문자열이 표시됩니다. 반환값이 null(return;)이면 오류가 발생한 사실은 표시되지만 구체적인 정보는 제공되지 않습니다.

참고: JavaScript 인터프리터가 보고하는 오류를 방지하려면 반환된 문자열 내의 인용 부호("") 앞에 백슬래시(\)가 있어야 합니다.

예제

applyBehavior() 함수의 다음 예제에서는 MM_openBrWindow() 함수에 대한 호출을 반환하고, 윈도우의 높이 및 폭, 윈도우에 스크롤 막대, 툴바, 위치 막대 및 기타 기능을 포함할지 여부, 윈도우에서 열어야 하는 URL 등 사용자가 지정한 매개 변수를 전달합니다.

```
function applyBehavior() {
    var i,theURL,theName,arrayIndex = 0;
    var argArray = new Array(); //use array to produce correct ¬
        number of commas w/o spaces
    var checkBoxNames = new Array("toolbar","location",¬
        "status","menubar","scrollbars","resizable");

    for (i=0; i<checkBoxNames.length; i++) {
        theCheckBox = eval("document.theForm." + checkBoxNames[i]);
        if (theCheckBox.checked) argArray[arrayIndex++] = (checkBoxNames[i] + "=yes");
    }
    if (document.theForm.width.value)
        argArray[arrayIndex++] = ("width=" + document.theForm.width.value);
    if (document.theForm.height.value)
        argArray[arrayIndex++] = ("height=" + document.theForm.height.value);
    theURL = escape(document.theForm.URL.value);
    theName = document.theForm.winName.value;
    return "MM_openBrWindow('"+theURL+"','"+theName+"','"+argArray.join()+"')";
}
```

behaviorFunction()

설명

이 함수는 사용자 문서의 head 섹션에 하나 이상의 함수를 삽입합니다. 이때 다음 태그가 아직 없으면 함수를 이 태그로 묶습니다.

```
<SCRIPT LANGUAGE="JavaScript"></SCRIPT>
```

인수

없음

반환값

JavaScript 함수를 포함하는 문자열이나 사용자 문서에 삽입할 함수의 이름을 포함하는 문자열을 반환합니다. 이 값은 사용자 입력에 따라 달라지지 않고 매번 정확하게 일치해야 합니다. 문서의 요소에 액션이 적용되는 횟수에 관계없이 함수는 한 번만 삽입됩니다.

참고: JavaScript 인터프리터가 보고하는 오류를 방지하려면 반환된 문자열 내의 인용 부호("") 앞에 백슬래시(\) 이스케이프 문자가 있어야 합니다.

예제

behaviorFunction() 함수의 다음 인스턴스는 MM_popupMsg() 함수를 반환합니다.

```
function behaviorFunction(){
    return ""+
    "function MM_popupMsg(theMsg) { //v1.0\n"+
    "alert(theMsg);\n"+
    "}";
}
```

다음 예제는 앞의 behaviorFunction() 선언과 동일하며 Dreamweaver와 함께 제공되는 모든 비헤이비어에서 behaviorFunction() 함수를 선언하는 데는 이 방법을 사용합니다.

```
function MM_popupMsg(theMsg){ //v1.0
    alert(theMsg);
}

function behaviorFunction(){
    return "MM_popupMsg";
}
```

canAcceptBehavior()

설명

이 함수는 선택한 HTML 요소에 액션을 사용할 수 있는지 여부를 확인하고 액션을 트리거할 기본 이벤트를 지정합니다. 또한 사용자 문서에 SWF 파일과 같은 특정 객체가 있는지 확인하고 이러한 객체가 나타나지 않을 경우에는 액션을 허용하지 않을 수도 있습니다.

인수

HTMLElement

이 인수는 선택한 HTML 요소입니다.

반환값

다음 값 중 하나를 반환합니다.

- 액션이 허용되지만 선호하는 이벤트가 없으면 true 값을 반환합니다.
- 이 액션에 대한 선호하는 이벤트 목록을 선호도 기준으로 내림차순으로 반환합니다. 선호하는 이벤트를 지정하면 선택한 객체에 대한 기본 이벤트(이벤트 파일에서 별표[*]로 표시된 이벤트)가 무시됩니다. 자세한 내용은 218페이지의 “비헤이비어의 작동 방식”의 1단계를 참조하십시오.
- 액션이 허용되지 않으면 false 값을 반환합니다.

  canAcceptBehavior() 함수가 false 값을 반환하면 해당 액션이 [비헤이비어] 패널의 [액션] 팝업 메뉴에서 희미하게 표시됩니다.

예제

canAcceptBehavior() 함수의 다음 인스턴스는 문서에 이름이 지정된 이미지가 있을 경우 해당 비헤이비어에 대해 선호하는 이벤트 목록을 반환합니다.

```
function canAcceptBehavior(){
    var theDOM = dreamweaver.getDocumentDOM();
    // Get an array of all images in the document
    var allImages = theDOM.getElementsByTagName('IMG');
    if (allImages.length > 0){
        return "onMouseOver, onClick, onMouseDown";
    }else{
        return false;
    }
}
```

displayHelp()

설명

이 함수를 정의하면 [매개 변수] 대화 상자의 [확인] 및 [취소] 버튼 아래에 [도움말] 버튼이 표시됩니다. 사용자가 [도움말] 버튼을 클릭하면 이 함수가 호출됩니다.

인수
없음

반환값
없음

예제
```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

deleteBehavior()

설명
이 함수는 applyBehavior() 함수가 수행한 편집 작업을 취소합니다.

참고: 사용자가 [비헤이비어] 패널에서 비헤이비어를 삭제하면 해당 비헤이비어와 연관된 함수 선언 및 이벤트 핸들러가 자동으로 삭제됩니다. applyBehavior() 함수가 사용자 문서에 대해 태그 삽입과 같은 추가 편집 작업을 수행하는 경우에만 deleteBehavior() 함수를 정의해야 합니다.

인수
applyBehaviorString

이 인수는 applyBehavior() 함수가 반환하는 문자열입니다.

반환값
없음

identifyBehaviorArguments()

설명
이 함수는 비헤이비어 함수 호출의 인수를 내비게이션 링크, 종속 파일, URL, Netscape 4.0 스타일 참조 또는 객체 이름으로 식별하여 사용자가 문서를 다른 위치에 저장하는 경우 비헤이비어의 URL을 업데이트할 수 있고 참조된 파일을 사이트 맵에 표시하여 서버에 업로드하고 서버에서 다운로드하기 위한 종속 파일로 간주할 수 있도록 합니다.

인수
theFunctionCall

이 인수는 applyBehavior() 함수가 반환하는 문자열입니다.

반환값
함수 호출의 인수 유형이 쉼표로 구분된 목록으로 들어 있는 문자열을 반환합니다. 목록의 길이는 함수 호출의 인수 개수와 같아야 합니다. 인수 유형은 다음 중 하나여야 합니다.

- nav 인수 유형은 해당 인수가 내비게이션 URL이므로 사이트 맵에 표시되어야 함을 나타냅니다.

- dep 인수 유형은 해당 인수가 종속 파일 URL이므로 이 비헤이비어를 포함하는 문서가 서버에서 다운로드되거나 서버로 업로드될 때 다른 모든 종속 파일과 함께 포함되어야 함을 나타냅니다.
- URL 인수 유형은 해당 인수가 내비게이션 URL이자 종속 URL이거나 알 수 없는 유형의 URL로, 사이트 맵에 표시되어야 하며, 서버에서 다운로드되거나 서버로 업로드될 때 종속 파일로 간주되어야 함을 나타냅니다.
- NS4.0ref 인수 유형은 해당 인수가 Netscape Navigator 4.0 스타일의 객체 참조임을 나타냅니다.
- IE4.0ref 인수 유형은 해당 인수가 Internet Explorer DOM 4.0 스타일의 객체 참조임을 나타냅니다.
- objName 인수 유형은 해당 인수가 객체의 NAME 속성에 지정된 단순 객체 이름임을 나타냅니다. 이 유형은 Dreamweaver 3에서 추가되었습니다.
- other 인수 유형은 해당 인수가 위의 유형 중 어느 것에도 속하지 않음을 나타냅니다.

예제

identifyBehaviorArguments() 함수에 대한 간단한 다음 예제는 브라우저 윈도우 열기 비헤이비어 액션에 대해 작동합니다. 이 액션은 항상 세 가지 인수(열려는 URL, 새 윈도우의 이름, 윈도우 속성 목록)가 있는 함수를 반환합니다.

```
function identifyBehaviorArguments(fnCallStr) {
    return "URL,other,other";
}
```

인수의 개수가 유동적인 비헤이비어 함수(예: 레이어 보기/숨김)에는 보다 복잡한 identifyBehaviorArguments() 함수가 필요합니다. 이 예제에 나오는 identifyBehaviorArguments() 함수의 경우에는 최소 개수의 인수가 사용되며 추가 인수는 항상 최소 개수의 배수만큼 사용됩니다. 즉, 최소 인수 개수가 4개인 함수에는 4, 8 또는 12개의 인수를 사용할 수 있지만 10개의 인수를 사용할 수는 없습니다.

```
function identifyBehaviorArguments(fnCallStr) {
    var listOfArgTypes;
    var itemArray = dreamweaver.getTokens(fnCallStr, '(),');

    // The array of items returned by getTokens() includes the
    // function name, so the number of *arguments* in the array
    // is the length of the array minus one. Divide by 4 to get the
    // number of groups of arguments.
    var numArgGroups = ((itemArray.length - 1)/4);
    // For each group of arguments
    for (i=0; i < numArgGroups; i++){

    // Add a comma and "NS4.0ref,IE4.0ref,other,dep" (because this
    // hypothetical behavior function has a minimum of four
    // arguments the Netscape object reference, the IE object
    // reference, a dependent URL, and perhaps a property value
    // such as "show" or "hide") to the existing list of argument
    // types, or if no list yet exists, add only
    // "NS4.0ref,IE4.0ref,other,dep"
    var listOfArgTypes += ((listOfArgTypes)?",":"") + ¬
    "NS4.0ref,IE4.0ref,other,dep";
    }
}
```

inspectBehavior()

설명

이 함수는 이전에 사용자 문서에 적용된 비헤이비어의 함수 호출을 검사하고 그에 따라 [매개 변수] 대화 상자의 옵션 값을 설정합니다. inspectBehavior() 함수가 정의되어 있지 않으면 기본 옵션 값이 나타납니다.

참고: inspectBehavior() 함수의 동작은 applyBehaviorString 인수가 전달하는 정보에 따라서만 달라져야 합니다. 이 함수 내에서 dreamweaver.getDocumentDOM() 등을 통해 사용자의 문서에 대한 다른 정보를 얻으려고 하지 마십시오.

인수
applyBehaviorString

이 인수는 applyBehavior() 함수가 반환하는 문자열입니다.

참고: HTML 요소에 'onClick="someBehavior(); return document.MM_returnValue;"'와 유사한 코드가 포함되어 있는 경우 비헤이비어 메뉴에서 새 비헤이비어를 추가하면 Dreamweaver에서 새 비헤이비어 UI가 나타나는 즉시 inspectBehavior()가 호출되고 빈 문자열이 매개 변수로 전달됩니다. 그러면 다음 예제와 같이 applyBehaviorString 매개 변수를 확인하십시오.

```
function inspectBehavior(enteredStr){
    if(enteredStr){
    //do your work here
    }
}
```

반환값
없음

예제
inspectBehavior() 함수의 다음 인스턴스는 Display Status Message.htm 파일에 포함된 것으로, 비헤이비어가 처음 적용되었을 때 사용자가 선택한 메시지로 [매개 변수] 대화 상자의 [메시지] 필드를 채웁니다.

```
function inspectBehavior(msgStr){
    var startStr = msgStr.indexOf("'") + 1;
    var endStr = msgStr.lastIndexOf("'");
    if (startStr > 0 && endStr > startStr) {
        document.theForm.message.value = ¬
        unescQuotes(msgStr.substring(startStr,endStr));
    }
}
```

참고: unescQuotes() 함수에 대한 자세한 내용은 Configuration/Shared/Common/Scripts/CMN 폴더의 dwscripts.js 파일을 참조하십시오.

windowDimensions()

설명
이 함수는 [매개 변수] 대화 상자의 구체적 크기를 설정합니다. 이 함수를 정의하지 않으면 윈도우 크기가 자동으로 계산됩니다.

참고: [매개 변수] 대화 상자의 크기를 640x480픽셀보다 크게 하려는 경우에만 이 함수를 정의하십시오.

인수
platform

platform 인수의 값은 사용자의 플랫폼에 따라 macintosh 또는 windows입니다.

반환값
widthInPixels,heightInPixels 형식의 문자열을 반환합니다.

반환된 크기는 [확인] 및 [취소] 버튼의 영역을 포함하지 않으므로 전체 대화 상자의 크기보다 작습니다. 반환된 크기에 옵션이 모두 들어가지 않으면 스크롤 막대가 나타납니다.

예제

windowDimensions()의 다음 인스턴스는 [매개 변수] 대화 상자의 크기를 648x520픽셀로 설정합니다.

```
function windowDimensions(){
    return "648,520";
}
```

17장: 서버 비헤이비어

Adobe® Dreamweaver®에서는 다음과 같은 서버측 작업을 수행하기 위해 문서에 서버 비헤이비어를 추가할 때 사용할 수 있는 인터페이스를 제공합니다.

- 사용자의 기준에 따라 레코드 필터링
- 레코드 페이징
- 세부 정보 페이지에 결과 목록 연결
- 결과 집합에 레코드 삽입

Dreamweaver를 사용하는 동안 동일한 런타임 코드를 문서에 반복해서 삽입할 경우 자주 사용되는 코드 블록으로 문서 업데이트 절차를 자동화하는 Extension을 만들 수 있습니다. 사용자 정의 서버 비헤이비어를 구현하는 서버 비헤이비어 구성 관리자 인터페이스를 사용하는 작업에 대한 자세한 내용은 **Dreamweaver 시작하기**에서 "사용자 정의 서버 비헤이비어 추가"를 참조하십시오. 지원 서버 비헤이비어 파일을 사용한 작업 및 설정된 서버 비헤이비어와 상호 작용하는 함수 사용에 대한 자세한 내용은 서버 비헤이비어 장을 참조하십시오. 개별 함수에 대한 정보는 **Dreamweaver API 참조 설명서**에서 "서버 비헤이비어 함수" 및 "Extension 데이터 관리자 함수를 참조하십시오. Dreamweaver에서는 현재 ASP/JavaScript, ASP/VBScript, ColdFusion 및 PHP/MySQL 등의 서버 모델에 대한 런타임 코드를 추가하는 서버 비헤이비어 Extension을 지원합니다.

서버 비헤이비어 용어

다음은 서버 비헤이비어와 관련하여 일반적으로 사용되는 용어입니다.

서버 비헤이비어 Extension 서버 비헤이비어 Extension은 서버측 코드와 Dreamweaver 사이의 인터페이스입니다. 서버 비헤이비어 Extension은 JavaScript와 HTML 그리고 특별히 Extension 데이터용으로 만들어진 Extension Data Markup Language(EDML)로 구성되어 있습니다. 이러한 파일의 예제는 서버 모델에 따라 정렬되어 Configuration/ServerBehaviors 폴더의 설치 디렉토리에 저장됩니다. Extension 스크립트를 작성할 때에는 dwscripts.applySB() 함수를 사용하여 Dreamweaver가 EDML 파일을 읽고, Extension 구성 요소를 검색하고, 적당한 코드 블록을 사용자 문서에 추가하도록 지정합니다.

서버 비헤이비어 인스턴스 Dreamweaver에서 사용자 문서에 코드 블록을 추가하면 삽입된 코드가 서버 비헤이비어의 인스턴스를 구성합니다. 사용자는 대부분의 서버 비헤이비어를 한 번 이상 적용할 수 있기 때문에 결과적으로 여러 서버 비헤이비어 인스턴스가 만들어집니다. 각 서버 비헤이비어 인스턴스는 Dreamweaver 인터페이스의 [서버 비헤이비어] 패널에 나열됩니다.

런타임 코드 런타임 코드는 서버 비헤이비어가 적용될 때 문서에 추가되는 일련의 코드 블록입니다. 이러한 코드 블록에는 보통 <% ... %> 태그에 둘러싸인 ASP 스크립트와 같은 일부 서버측 코드가 포함됩니다.

참여자 서버 비헤이비어 Extension은 코드 블록을 사용자의 문서에 삽입합니다. 코드 블록은 서버측 태그, HTML 태그 또는 웹 페이지에 서버측 기능을 추가하는 속성 등 연속된 단일 스크립트 블록입니다. EDML 파일은 각 코드 블록을 참여자로 정의합니다. 지정된 서버 비헤이비어의 모든 참여자는 하나의 참여자 그룹을 구성합니다.

참고: 참여자, 참여자 그룹 및 Dreamweaver EDML 파일이 구조화되는 방법에 대한 자세한 내용은 231페이지의 “Extension Data Markup Language(EDML)”을 참조하십시오.

Dreamweaver 구조

서버 비헤이비어 구성 관리자를 사용하여 Dreamweaver 고유의 Extension을 만들면 Dreamweaver 문서에 서버 비헤이비어 코드 삽입을 지원하는 여러 파일(EDML 및 HTML 스크립트 파일)이 만들어집니다. 일부 비헤이비어는 추가 기능의 수행을 위해 JavaScript 파일을 참조하기도 합니다. 이 구조는 API의 구현을 단순화하고 Dreamweaver의 배포 방식과 런타임 코드를 분리하기도 합니다. 여기에서는 이 파일을 수정하는 방법에 대해 설명합니다.

서버 비헤이비어 폴더 및 파일

각 서버 비헤이비어의 UI(사용자 인터페이스)는 Configuration/ServerBehaviors/ServerModelName 폴더에 있습니다. 여기서 ServerModelName은 ASP_Js(JavaScript), ASP_Vbs(VBScript), ColdFusion, PHP_MySQL 또는 Shared(크로스 서버 모델 구현) 서버 유형 중 하나입니다.

Extension Data Markup Language(EDML)

서버 비헤이비어 구성 관리자를 사용할 때 생성되는 두 가지 EDML 파일은 서버 비헤이비어 구성 관리자에 제공하는 이름에 대응되는 그룹 EDML 파일과 참여자 EDML 파일입니다. 그룹 파일은 코드 블록을 나타내는 관련 참여자를 정의하고, 그룹은 개별 서버 비헤이비어를 만들기 위해 결합되는 참여자를 정의합니다.

그룹 파일

그룹 파일은 참여자 목록을 포함하고 있으며 참여자 파일은 모든 서버 모델 고유의 코드 데이터를 가지고 있습니다. 참여자 파일은 하나 이상의 Extension에서 사용할 수 있으므로 여러 그룹 파일이 동일한 참여자 파일을 참조할 수 있습니다.

다음 예제는 서버 비헤이비어 그룹 EDML 파일의 고급 뷰입니다. 요소 및 속성의 전체 목록은 242페이지의 “EDML 그룹 파일 태그”를 참조하십시오.

```
<group serverBehavior="Go To Detail Page.htm" dataSource="Recordset.htm">
    <groupParticipants selectParticipant="goToDetailPage_attr">
        <groupParticipant name="moveTo_declareParam"0partType="member"/>
        <groupParticipant name="moveTo_keepParams"0partType="member"/>
        <groupParticipant name="goToDetailPage_attr" partType="identifier" />
    </groupParticipants>
</group>
```

groupParticipants 블록 태그에서 각 groupParticipant 태그는 사용할 코드 블록이 포함된 EDML 참여자 파일을 나타냅니다. name 속성의 값은 참여자 파일 이름에서 확장명 .edml을 뺀 것입니다(예: moveTo_declareParam).

참여자 파일

참여자는 서버 태그, HTML 태그, 속성 등 페이지의 단일 코드 블록을 나타냅니다. 참여자 파일은 Dreamweaver 문서 제작자가 사용하게 될 그룹 파일에 나열되어야 합니다. 서버 그룹 파일은 단일한 참여자 파일을 사용할 수 있습니다.

예를 들어 moveTo_declareParam.edml 파일에는 다음 코드가 포함됩니다.

```
<participant>
    <quickSearch><![CDATA[MM_paramName]]></quickSearch>
    <insertText location="aboveHTML+80">
<![CDATA[
<% var MM_paramName = ""; %>
]]>
    </insertText>
    <searchPatterns whereToSearch="directive">
        <searchPattern><![CDATA[/var\s*MM_paramName/]]></searchPattern>
    </searchPatterns>
</participant>
```

Dreamweaver가 서버 비헤이비어를 문서에 추가할 때는 코드를 삽입할 위치, 코드의 형태, Dreamweaver 제작자 또는 데이터가 런타임에 교체한 매개 변수를 비롯한 자세한 정보가 필요합니다. 각 참여자 EDML 파일에는 각 코드 블록에 대한 이런 세부 정보가 기술되어 있습니다. 특히, 참여자 파일은 다음 데이터를 설명합니다.

- 고유한 인스턴스를 위치시킬 코드 및 장소는 다음 예제에서와 같이 insertText 태그 매개 변수에 의해 정의됩니다.

  ```
  <insertText location="aboveHTML+80">
  ```

- 이미 페이지에 존재하는 인스턴스를 인식하는 방법은 다음 예제와 같이 searchPatterns 태그에 의해 정의됩니다.

```
<searchPatterns whereToSearch="directive">
    <searchPattern><![CDATA[/var\s*MM_paramName/]]></searchPattern>
</searchPatterns>
```

searchPatterns 블록 태그에서 각 searchPattern 태그에는 런타임 코드 인스턴스를 찾아 특정 매개 변수를 추출하는 패턴이 들어 있습니다. 자세한 내용은 264페이지의 “서버 비헤이비어 기술”을 참조하십시오.

스크립트 파일

각 서버 비헤이비어에는 서버 비헤이비어 코드와 Dreamweaver 인터페이스의 통합을 관리하는 스크립트에 대한 링크와 함수를 포함하는 HTML 파일도 있습니다. 이 파일에서 편집에 사용할 수 있는 함수는 239페이지의 “서버 비헤이비어 구현 함수”에서 설명합니다.

간단한 서버 비헤이비어 예제

이 예제에서는 Dreamweaver가 생성하는 파일 및 이 파일의 처리 방법을 볼 수 있도록 새 서버 비헤이비어를 만드는 과정을 보여 줍니다. 서버 비헤이비어 구성 관리자 인터페이스 사용에 대한 자세한 내용은 **Dreamweaver 시작하기**에서 "사용자 정의 서버 비헤이비어 추가"를 참조하십시오. 이 예제에서는 ASP 서버로부터 "Hello World"를 표시합니다. Hello World 비헤이비어는 참여자를 하나만 가지고 있으며(단일 ASP 태그) 수정하거나 페이지에 다른 어떤 것도 추가하지 않습니다.

비헤이비어를 만들려면 동적 페이지 문서를 만들고 새 서버 비헤이비어를 정의하고 삽입할 코드를 정의합니다.

동적 페이지 문서 만들기

1. Dreamweaver에서 [파일] > [새 파일] 메뉴 옵션을 선택합니다.
2. [새 문서] 대화 상자에서 [범주: 동적 페이지]를 선택하고 [동적 페이지: ASP JavaScript]를 선택합니다.
3. [만들기]를 클릭합니다.

새 서버 비헤이비어 정의

참고: [서버 비헤이비어] 패널이 열리지 않거나 표시되지 않는 경우에는 [윈도우] > [서버 비헤이비어] 메뉴 옵션을 선택합니다.

1. [서버 비헤이비어] 패널에서 플러스(+) 버튼을 선택하고 [새 서버 비헤이비어] 메뉴 옵션을 선택합니다.
2. [새 서버 비헤이비어 만들기] 대화 상자의 [문서 형식:]에서 [ASP JavaScript]를 선택하고 [이름:]에서 [Hello World]를 선택합니다. "기존 서버 비헤이비어 복사" 체크 상자는 선택하지 않습니다.
3. [확인]을 클릭합니다.

삽입할 코드 정의

1. [삽입할 코드 블록]의 플러스(+) 버튼을 선택합니다.
2. [새 코드 블록 생성] 대화 상자에 **Hello_World_block1**을 입력합니다. 이 정보는 자동으로 입력될 수도 있습니다.
3. [확인]을 클릭합니다.
4. [코드 블록] 텍스트 필드에 **<% Response.Write(“Hello World”) %>**를 입력합니다.
5. 문서에서의 코드 위치를 사용자가 제어할 수 있도록 [코드 삽입] 팝업 메뉴에서 [선택 영역에 상대적]을 선택합니다.
6. [상대적 위치] 팝업 메뉴에서 [선택 영역 다음]을 선택합니다.
7. [확인]을 클릭합니다.

[서버 비헤이비어] 패널을 보면 플러스(+) 메뉴의 팝업 목록에 새 서버 비헤이비어가 있습니다. 또한 Dreamweaver 파일의 설치 폴더에서 Configuration/ServerBehaviors/ASP_Js 폴더에는 이제 다음 세 파일이 포함됩니다.

- 그룹 파일: Hello World.edml
- 참여자 파일: Hello World_block1.edml
- 스크립트 파일: Hello World.htm

참고: 다중 사용자 구성에서 작업하는 경우, 이 파일은 Application Data 폴더에 나타납니다.

서버 비헤이비어 API 함수가 호출되는 경우

서버 비헤이비어 API 함수는 다음 시나리오에서 호출됩니다.

- 문서가 열리고 다시 참여자가 편집될 때 findServerBehaviors() 함수가 호출됩니다. 사용자의 문서에서 서버 비헤이비어의 인스턴스를 찾습니다. 검색된 인스턴스마다 findServerBehaviors()는 JavaScript 객체를 만들고 JavaScript 속성을 사용하여 이 객체에 대한 상태 정보를 첨부합니다.
- JacaScript 객체가 구현되면 Dreamweaver에서는 findServerBehaviors() 함수가 모두 호출된 후에 사용자의 문서에서 검색된 각 비헤이비어 인스턴스에 대해 analyzeServerBehavior() 함수를 호출합니다.

  findServerBehaviors() 함수는 비헤이비어 객체를 만들 때 대개 네 가지 속성, 즉 incomplete, participants, selectedNode 및 title을 설정합니다. 그러나 다른 모든 서버 비헤이비어가 해당 인스턴스를 찾을 때까지 일부 속성을 설정하는 것을 보류하는 것이 좋을 때도 있습니다. 예를 들어 [다음 레코드로 이동] 비헤이비어에는 링크 객체와 레코드세트 객체라는 두 참여자가 있습니다. 이 경우 레코드세트는 해당 인스턴스를 모두 찾으므로 findServerBehaviors() 함수에서 레코드세트 객체를 찾는 것보다는 레코드세트 비헤이비어의 findServerBehaviors() 함수가 실행될 때까지 기다리는 것이 좋습니다.

  [다음 레코드로 이동] 비헤이비어의 analyzeServerBehavior() 함수가 호출되면 문서의 서버 비헤이비어 객체가 모두 포함된 배열을 얻게 됩니다. 함수는 이 배열에서 자신의 레코드세트 객체를 찾을 수 있습니다.

  분석하는 동안 사용자 문서의 단일 태그가 둘 이상의 비헤이비어에서 해당 비헤이비어의 인스턴스로 식별되는 경우도 있습니다. 예를 들어 [동적 속성] 비헤이비어에 대한 findServerBehaviors() 함수는 사용자 문서의 input 태그와 연관된 [동적 속성] 비헤이비어의 인스턴스를 찾아낼 수 있습니다. 동시에, [동적 텍스트 필드] 비헤이비어에 대한 findServerBehaviors() 함수는 동일한 input 태그를 통해 [동적 텍스트 필드] 비헤이비어의 인스턴스를 찾아낼 수 있습니다. 결과적으로 [서버 비헤이비어] 패널에는 [동적 속성] 블록과 [동적 텍스트 필드]가 표시됩니다. 이 문제를 해결하려면 analyzeServerBehavior() 함수는 중복된 서버 비헤이비어 중 하나를 제외하고 모두 삭제해야 합니다.

  특정 서버 비헤이비어를 삭제하기 위해 analyzeServerBehavior() 함수는 서버 비헤이비어의 deleted 속성을 true로 설정할 수 있습니다. analyzeServerBehavior() 함수 호출이 종료된 후에도 deleted 속성이 계속 true이면 이 비헤이비어는 목록에서 삭제됩니다.

- 사용자가 [서버 비헤이비어] 패널에서 플러스(+) 버튼을 클릭하면 팝업 메뉴가 나타납니다.

  메뉴의 내용을 결정하기 위해 Dreamweaver에서는 먼저 비헤이비어와 같은 폴더에서 ServerBehaviors.xml 파일을 찾습니다. ServerBehaviors.xml은 해당 메뉴에 나타나야 하는 HTML 파일을 참조합니다.

  참조된 HTML 파일에 제목 태그가 있으면 해당 제목 태그 내용이 메뉴에 나타납니다. 예를 들어 ServerBehaviors/ASP_Js/GetRecords.htm 파일에 <title>Get More Records</title> 태그가 있으면 **Get More Records**라는 텍스트가 메뉴에 나타납니다.

  파일에 title 태그가 없으면 파일 이름이 메뉴에 나타납니다. 예를 들어 GetRecords.htm에 제목 태그가 없으면 GetRecords라는 텍스트가 메뉴에 나타납니다.

  ServerBehaviors.xml 파일이 없거나 폴더에 ServerBehaviors.xml에서 언급되지 않은 HTML 파일이 하나 이상 있으면 Dreamweaver에서는 각 파일의 제목 태그를 확인하고 해당 제목 태그나 파일 이름을 사용하여 메뉴를 채웁니다.

  ServerBehaviors 폴더의 파일을 메뉴에 표시하지 않으려면 다음 명령문을 HTML 파일의 첫 번째 행에 추가합니다.

```
<!-- MENU-LOCATION=NONE -->
```

- 사용자가 메뉴에서 한 항목을 선택하면 canApplyServerBehavior() 함수가 호출됩니다. 이 함수가 true를 반환하면 대화 상자가 나타납니다. 사용자가 [확인]을 클릭하면 applyServerBehavior() 함수가 호출됩니다.
- 사용자가 기존 서버 비헤이비어를 두 번 클릭하여 편집하면 Dreamweaver에서는 해당 대화 상자를 표시하고 body 태그에 onLoad 핸들러가 있으면 이 핸들러를 실행한 다음 inspectServerBehavior() 함수를 호출합니다. inspectServerBehavior() 함수는 양식 요소를 현재 인수 값으로 채웁니다. 사용자가 [확인]을 클릭하면 Dreamweaver에서는 applyServerBehavior() 함수를 다시 호출합니다.
- 사용자가 마이너스(-) 버튼을 클릭하면 deleteServerBehavior() 함수가 호출됩니다. deleteServerBehavior() 함수는 문서에서 비헤이비어를 제거합니다.
- 사용자가 특정 서버 비헤이비어를 선택하고 [잘라내기] 또는 [복사] 명령을 사용하면 Dreamweaver에서는 선택한 서버 비헤이비어를 나타내는 객체를 copyServerBehavior() 함수로 전달합니다. copyServerBehavior() 함수는 나중에 서버 비헤이비어 객체를 붙여넣을 때 필요한 기타 속성을 해당 서버 비헤이비어 객체에 추가합니다.

  copyServerBehavior() 함수가 반환된 후, Dreamweaver에서는 서버 비헤이비어 객체를 클립보드에 넣을 수 있는 형태로 변환합니다. Dreamweaver에서는 객체를 변환할 때 객체를 참조하는 모든 속성을 삭제합니다. 숫자, 부울 값 또는 문자열이 아닌 해당 객체의 모든 속성을 잃게 됩니다.

  사용자가 [붙여넣기] 명령을 사용하면 Dreamweaver에서는 클립보드의 내용을 언패킹하여 새 서버 비헤이비어 객체를 생성합니다. 새 객체는 객체를 참조하는 속성이 없는 것을 제외하고는 원래 객체와 똑같습니다. Dreamweaver에서는 pasteServerBehavior() 함수에 새 서버 비헤이비어 객체를 전달합니다. pasteServerBehavior() 함수는 비헤이비어를 사용자의 문서에 추가합니다. pasteServerBehavior() 함수가 반환된 후 Dreamweaver에서는 findServerBehaviors() 함수를 호출하여 사용자 문서에 있는 모든 서버 비헤이비어가 포함된 새 목록을 가져옵니다.

  사용자는 한 문서의 비헤이비어를 복사하여 다른 문서에 붙여넣을 수 있습니다. copyServerBehavior() 및 pasteServerBehavior() 함수는 지정된 비헤이비어 객체의 속성에 따라 정보를 교환합니다.

서버 비헤이비어 API

다음 API 함수로 서버 비헤이비어를 관리할 수 있습니다.

analyzeServerBehavior()

지원 버전
Dreamweaver UltraDev 1

설명
서버 비헤이비어가 incomplete 및 deleted 속성을 설정할 수 있도록 합니다.

페이지의 모든 서버 비헤이비어에 대해 findServerBehaviors() 함수가 호출된 후 사용자 문서에 포함된 모든 비헤이비어의 배열이 나타납니다. 이 배열의 각 JavaScript 객체에 대해 analyzeServerBehavior() 함수가 호출됩니다. 예를 들어 [동적 텍스트] 비헤이비어의 경우 Dreamweaver에서는 DynamicText.htm 또는 DynamicText.js에서 analyzeServerBehavior() 함수를 호출합니다.

analyzeServerBehavior() 함수의 용도 중 하나는 비헤이비어 객체에 incomplete, participants, selectedNode 및 title과 같은 속성을 설정하는 것입니다. findServerBehaviors() 함수가 사용자의 문서에서 전체 서버 비헤이비어 목록을 생성한 후에 이 작업을 수행하는 것이 보다 쉬울 수 있습니다.

analyzeServerBehavior() 함수의 다른 용도는 사용자의 문서에서 두 개 이상의 비헤이비어가 동일한 태그를 참조할 때 알리는 것입니다. 이 경우, deleted 속성은 배열에서 한 비헤이비어를 제외한 모든 비헤이비어를 제거합니다.

Recordset1, DynamicText1 및 DynamicText2 서버 비헤이비어가 페이지에 존재한다고 가정해 보십시오. 이 중 두 개 DynamicText 서버 비헤이비어 모두에 대해 Recordset1이 해당 페이지에 있어야 합니다. findServerBehaviors() 함수를 사용하여 서버 비헤이비어를 찾은 후 Dreamweaver에서는 이 세 서버 비헤이비어에 대해 analyzeServerBehavior() 함수를 호출합니다. DynamicText1에 대해 analyzeServerBehavior()가 호출되면 이 함수는 Recordset1에 속한 비헤이비어를 찾기 위해 페이지에 있는 모든 서버 비헤이비어의 배열을 검색합니다. Recordset1에 속하는 서버 비헤이비어 객체를 찾을 수 없으면 incomplete 속성이 true로 설정되고 [서버 비헤이비어] 패널에 느낌표가 나타나도록 하여 문제가 있음을 사용자에게 알립니다. 마찬가지로 DynamicText2에 대해 analyzeServerBehavior()가 호출되면 함수는 Recordset1에 속하는 객체를 찾습니다. Recordset1은 다른 서버 비헤이비어에 종속되어 있지 않기 때문에 이 예제에서 analyzeServerBehavior() 함수를 정의할 필요는 없습니다.

인수
serverBehavior, {**serverBehaviorArray**}

- **serverBehavior** 인수는 분석할 비헤이비어를 나타내는 JavaScript 객체입니다.
- {***serverBehaviorArray***} 인수는 특정 페이지에서 발견된 모든 서버 비헤이비어를 나타내는 JavaScript 객체의 배열입니다.

반환값
없음

applyServerBehavior()

지원 버전
Dreamweaver UltraDev 1

설명
대화 상자의 양식 요소에서 값을 읽어 해당 비헤이비어를 사용자의 문서에 추가합니다. 사용자가 [서버 비헤이비어] 대화 상자에서 [확인]을 클릭하면 이 함수가 호출됩니다. 이 함수가 성공적으로 반환하면 [서버 비헤이비어] 대화 상자가 닫힙니다. 이 함수가 실패하면 [서버 비헤이비어] 대화 상자를 닫지 않은 채 오류 메시지가 표시됩니다. 이 함수는 사용자의 문서를 편집할 수 있습니다.

자세한 내용은 239페이지의 “dwscripts.applySB()”를 참조하십시오.

인수
serverBehavior

serverBehavior JavaScript 객체는 해당 서버 비헤이비어를 나타납니다. 그 이유는 기존 서버 비헤이비어를 수정할 필요가 있기 때문입니다. 이것이 새 비헤이비어이면 인수는 null이 됩니다.

반환값
이 함수가 성공하면 빈 문자열을 반환하고 이 함수가 실패하면 오류 메시지를 반환합니다.

canApplyServerBehavior()

지원 버전
Dreamweaver UltraDev 1

설명
비헤이비어가 적용될 수 있는지 여부를 판별합니다. Dreamweaver에서는 [서버 비헤이비어] 대화 상자가 나타나기 전에 이 함수를 호출합니다. 이 함수가 true를 반환하면 [서버 비헤이비어] 대화 상자가 나타납니다. 이 함수가 false를 반환하면 [서버 비헤이비어] 대화 상자도 나타나지 않고 서버 비헤이비어를 추가하려는 시도도 중단됩니다.

인수
serverBehavior

serverBehavior JavaScript 객체는 해당 비헤이비어를 나타냅니다. 그 이유는 기존 비헤이비어를 수정할 필요가 있기 때문입니다. 이것이 새 비헤이비어이면 인수는 null이 됩니다.

반환값
부울 값을 반환합니다. 비헤이비어를 적용할 수 있으면 true이고, 그렇지 않으면 false입니다.

copyServerBehavior()

지원 버전
Dreamweaver UltraDev 1

설명
copyServerBehavior() 함수 구현은 선택 사항입니다. 사용자는 지정된 서버 비헤이비어의 인스턴스를 복사할 수 있습니다. 다음 예제에서는 레코드세트에 대해 이 함수가 구현되어 있습니다. 사용자가 [서버 비헤이비어] 패널이나 [데이터 바인딩] 패널에서 레코드세트를 선택한 경우 [복사] 명령을 사용하면 비헤이비어가 클립보드에 복사되고 [잘라내기] 명령을 사용하면 비헤이비어가 잘려 클립보드에 놓입니다. 이 함수를 구현하지 않은 서버 비헤이비어의 경우 [복사]와 [잘라내기] 명령은 아무 것도 하지 않습니다. 자세한 내용은 233페이지의 “서버 비헤이비어 API 함수가 호출되는 경우”를 참조하십시오.

copyServerBehavior() 함수는 문자열로 변환될 수 있는 비헤이비어 객체 속성에 따라 pasteServerBehavior() 함수를 사용하여 정보를 교환합니다. 클립보드에는 순수 텍스트만 보관되므로 문서의 participant 노드를 분석하여 결과 텍스트를 보조 속성에 저장해야 합니다.

참고: 또한 pasteServerBehavior() 함수도 구현되어야 사용자가 Dreamweaver 문서에 비헤이비어를 붙여넣을 수 있습니다.

인수
serverBehavior

- **serverBehavior** JavaScript 객체는 해당 비헤이비어를 나타냅니다.

반환값
부울 값을 반환합니다. 비헤이비어가 클립보드에 복사되면 true이고, 그렇지 않으면 false입니다.

deleteServerBehavior()

지원 버전
Dreamweaver UltraDev 1

설명
사용자의 문서에서 비헤이비어를 제거합니다. 사용자가 [서버 비헤이비어] 패널에서 마이너스(-) 버튼을 클릭하면 이 함수가 호출됩니다. 이 함수는 사용자의 문서를 편집할 수 있습니다.

자세한 내용은 240페이지의 “dwscripts.deleteSB()”를 참조하십시오.

인수
serverBehavior

- **serverBehavior** JavaScript 객체는 해당 비헤이비어를 나타냅니다.

반환값
없음

displayHelp()

설명
이 함수를 정의하면 대화 상자의 [확인] 및 [취소] 버튼 아래에 [도움말] 버튼이 표시됩니다. 사용자가 [도움말] 버튼을 클릭하면 이 함수가 호출됩니다.

인수
없음

반환값
없음

예제

```
// the following instance of displayHelp() opens
// in a browser a file that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

findServerBehaviors()

지원 버전
Dreamweaver UltraDev 1

설명
사용자의 문서에서 해당 인스턴스를 검색합니다. 검색된 각 인스턴스에 대해 findServerBehaviors()는 JavaScript 객체를 만들고 이 객체의 JavaScript 속성으로 상태 정보를 첨부합니다.

필수 속성 네 가지는 incomplete, participants, title 및 selectedNode입니다. 필요에 따라 추가 속성을 설정할 수 있습니다.

자세한 내용은 **Dreamweaver API** 참조 설명서에서 239페이지의 “dwscripts.findSBs()” 및 dreamweaver.getParticipants()를 참조하십시오.

인수
없음

반환값
JavaScript 객체의 배열을 반환합니다. 이 배열의 길이는 해당 페이지에서 검색되는 비헤이비어 인스턴스의 수와 같습니다.

inspectServerBehavior()

지원 버전
Dreamweaver UltraDev 1

설명
지정된 비헤이비어 객체에 따라 서버 비헤이비어 대화 상자의 설정을 확인합니다. 사용자가 [서버 비헤이비어] 대화 상자를 열 때 Dreamweaver에서는 inspectServerBehavior() 함수를 호출합니다. 사용자가 기존 비헤이비어를 편집할 때만 이 함수가 호출됩니다.

인수
serverBehavior

serverBehavior 인수는 비헤이비어를 나타내는 JavaScript 객체입니다. 이 객체는 findServerBehaviors()가 반환하는 객체와 동일합니다.

반환값
없음

pasteServerBehavior()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수가 구현되면 사용자는 pasteServerBehavior()함수를 사용하여 지정된 서버 비헤이비어의 인스턴스를 붙여넣습니다. 사용자가 서버 비헤이비어를 붙여넣으면 Dreamweaver에서는 클립보드의 내용을 구성하고 새 비헤이비어 객체를 생성합니다. 새 객체는 포인터 속성이 부족한 것을 제외하고는 원래 객체와 같습니다. Dreamweaver에서는 새 서버 비헤이비어 객체를 pasteServerBehavior() 함수에 전달합니다. pasteServerBehavior() 함수는 비헤이비어 객체의 속성에 따라 사용자의 문서에 추가할 내용을 결정합니다. 그런 다음 pasteServerBehavior() 함수는 사용자의 문서에 비헤이비어를 추가합니다. pasteServerBehavior()가 반환한 후 Dreamweaver에서는 findServerBehaviors() 함수를 호출하여 사용자 문서에 있는 모든 서버 비헤이비어가 포함된 새 목록을 가져옵니다.

pasteServerBehavior() 함수 구현은 선택 사항입니다. 자세한 내용은 233페이지의 “서버 비헤이비어 API 함수가 호출되는 경우”를 참조하십시오.

참고: 이 함수를 구현하면 copyServerBehavior() 함수도 구현해야 합니다.

인수
behavior

behavior JavaScript 객체는 해당 비헤이비어를 나타냅니다.

반환값
부울 값을 반환합니다. 클립보드에서 비헤이비어를 붙여넣으면 true이고, 그렇지 않으면 false입니다.

서버 비헤이비어 구현 함수

이 함수는 HTML 스크립트 파일에서 또는 HTML 스크립트 파일 목록의 지정된 JavaScript 파일에서 추가하거나 편집할 수 있습니다.

dwscripts.findSBs()

지원 버전
Dreamweaver MX. 이전 버전의 Dreamweaver에 사용된 findSBs() 함수가 이 함수로 대체되었습니다.

설명
현재 페이지에서 서버 비헤이비어의 모든 인스턴스와 모든 참여자를 찾습니다. 제목, 유형, 참여자 배열, 무게 배열, 형식 배열, selectedNode 값 및 불완전한 플래그를 설정합니다. 이 함수는 레코드세트, 이름 및 열 이름 등과 같은 사용자 정의 가능한 속성의 배열을 저장하는 매개 변수 객체도 만듭니다. findServerBehaviors() 함수에서 이 배열을 반환할 수 있습니다.

인수
serverBehaviorTitle

serverBehaviorTitle 인수는 EDML 제목에 아무 제목도 지정되지 않았을 때 사용되는 제목 문자열 선택 사항입니다. 이 기능은 지역화 작업에 유용합니다.

반환값
필수 속성이 정의되는 JavaScript 객체의 배열을 반환합니다. 서버 비헤이비어의 인스턴스가 페이지에 나타나지 않을 때는 빈 문자열을 반환합니다.

예제
다음 예제에서는 현재 사용자 문서에서 특정 서버 비헤이비어의 모든 인스턴스를 찾습니다.

```
function findServerBehaviors() {
    allMySBs = dwscripts.findSBs();
    return allMySBs;
}
```

dwscripts.applySB()

지원 버전
Dreamweaver MX. 이전 버전의 Dreamweaver에 사용된 applySB() 함수가 이 함수로 대체되었습니다.

설명
서버 비헤이비어의 런타임 코드를 삽입하거나 업데이트합니다. **sbObj** 인수에 null 값이 있으면 새 런타임 코드가 삽입됩니다. 그렇지 않으면 **sbObj** 객체가 알려 주는 기존 런타임 코드로 업데이트됩니다. 사용자 설정은 JavaScript 객체에 속성으로 설정되고 **paramObj**로 전달되어야 합니다. 이 설정은 EDML 삽입 텍스트에 @@paramName@@으로 선언된 모든 인수와 일치해야 합니다.

인수
paramObj, sbObj

- **paramObj** 인수는 사용자 설정을 포함하는 객체입니다.
- **sbObj**는 기존 서버 비헤이비어를 업데이트할 경우, 이전 서버 비헤이비어 객체가 됩니다. 그렇지 않으면 null이 됩니다.

반환값
부울 값을 반환합니다. 서버 비헤이비어가 사용자 문서에 추가되면 true이고, 그렇지 않으면 false입니다.

예제
다음 예제에서는 paramObj 객체를 사용자의 입력으로 채우고 dwscripts.applySB() 함수를 호출하여 입력 내용 및 서버 비헤이비어 sbObj를 전달합니다.

```
function applyServerBehaviors(sbObj) {
    // get all UI values here...
    paramObj = new Object();
    paramObj.rs= rsName.value;
    paramObj.col = colName.value;
    paramObj.url = urlPath.value;
    paramObj.form__tag = formObj;
     dwscripts.applySB(paramObj, sbObj);
}
```

dwscripts.deleteSB()

지원 버전
Dreamweaver MX. Dreamweaver의 이전 버전에 있는 deleteSB()가 이 함수로 대체되었습니다.

설명
sbObj 서버 비헤이비어 인스턴스의 모든 참여자를 삭제합니다. EDML 파일이 delete 태그로 특별한 삭제 지침을 지정하지 않으면 참여자 전체가 삭제됩니다. 둘 이상의 서버 비헤이비어 인스턴스(참조 수 > 1)에 속하는 참여자는 삭제되지 않습니다.

인수
sbObj

- **sbObj** 인수는 사용자의 문서에서 제거할 서버 비헤이비어 객체의 인스턴스입니다.

반환값
없음

예제
다음 예제에서는 EDML 파일의 delete 태그로 보호되는 참여자를 제외한 sbObj 서버 비헤이비어의 모든 참여자를 삭제합니다.

```
function deleteServerBehavior(sbObj) {
    dwscripts.deleteSB(sbObj);
}
```

EDML 파일

파일을 편집할 때는 Dreamweaver 코딩 규칙을 따라야 합니다. 한 요소가 다른 요소에 종속되는 경우가 있으므로 주의해야 합니다. 예를 들어 삽입한 태그를 업데이트할 때는 검색 패턴도 업데이트해야 할 경우가 있습니다.

참고: EDML 파일은 Dreamweaver MX의 새 기능입니다. 레거시 서버 비헤이비어로 작업하는 경우에는 Dreamweaver 확장 설명서의 이전 버전을 참조하십시오.

정규식

JavaScript 1.5에서 구현되는 정규식을 이해해야 합니다. 또한 서버 비헤이비어 EDML 파일에 언제 사용하는 것이 적합한지 알아야 합니다. 예를 들어 정규식은 quickSearch 값에서는 사용될 수 없습니다. 하지만 데이터 검색 및 추출 작업을 위해 searchPattern 태그의 내용에는 사용됩니다.

정규식은 특별한 의미가 할당된 문자(메타 문자)를 사용하여 미리 정의된 규칙에 따라 텍스트를 나타내고 분할하고 처리하여 텍스트 문자열을 설명합니다. 정규식은 패턴을 나타내는 표준 방법을 제공하기 때문에 강력한 파싱 및 처리 도구입니다.

JavaScript 1.5에 대한 참조 설명서에는 정규식을 다루는 부분이 있습니다. 이 절에서는 Dreamweaver 서버 비헤이비어 EDML 파일이 런타임 코드에서 인수를 찾아 해당 값을 추출하기 위해 정규식을 사용하는 방식을 알아봅니다. 사용자가 서버 비헤이비어를 편집할 때마다 이전의 인수 값은 런타임 코드의 인스턴스에서 추출되어야 합니다. 추출 과정에 정규식을 사용할 수 있습니다.

다음 표에 설명된 대로 서버 비헤이비어 EDML 파일에 유용한 일부 메타 문자와 메타 시퀀스(특수 문자 그룹화)를 이해해야 합니다.

정규식	설명
\	이스케이프 특수 문자. 예를 들어 \.는 메타 문자를 다시 리터럴 마침표로 되돌리고, \/는 정방향 슬래시를 해당 리터럴이 의미하는 것으로 되돌리며, \)는 괄호를 해당 리터럴이 의미하는 것으로 되돌립니다.
/ ... /i	대소문자 구분 없이 메타 시퀀스를 찾습니다
(...)	메타 시퀀스 안에 하위 표현식을 만들어 괄호로 묶어 표시합니다.
\s*	공백을 찾습니다.

EDML 태그 <searchPatterns whereToSearch="directive">는 런타임 코드를 검색해야 함을 선언합니다. 각 <searchPattern>...</searchPattern> 하위 태그는 식별해야 할 패턴을 런타임 코드에 정의합니다. Redirect If Empty 예제에 대해 두 가지 패턴이 있습니다.

다음 예제에서는 <% if (@@rs@@.EOF) Response.Redirect("@@new__url@@"); %>에서 인수 값을 추출하기 위해 문자열 rs 및 new__url을 식별하는 정규식을 작성합니다.

```
<searchPattern paramNames="rs,new__url">
    /if d ((\w+)\.EOF\) Response\.Redirect\("([^\r\n]*)"\)/i
</searchPattern>
```

이 과정에서 사용자의 문서를 검색하여 일치하는 것이 있으면 인수 값을 추출합니다. 첫 번째 괄호 하위 표현식(\w+)은 rs에 대한 값을 추출합니다. 두 번째 하위 표현식([^\r\n]*)은 new_url에 대한 값을 추출합니다.

참고: 문자 시퀀스 "[^\r\n]*"는 Macintosh 및 Windows에서 줄 바꿈을 제외한 모든 문자에 대응됩니다.

EDML 구조

서버 비헤이비어 그룹에는 고유한 파일 이름을 사용해야 합니다. 연관된 참여자 파일을 한 그룹 파일만 사용하는 경우, 그 참여자 파일 이름을 해당 그룹 이름과 일치시킵니다. 이 규칙을 사용할 경우, 서버 비헤이비어 그룹 파일 updateRecord.edml은 참여자 파일 updateRecord_init.edml과 함께 동작합니다. 참여자 파일이 서버 비헤이비어 그룹에서 공유될 가능성이 있으면 이 파일에 대해 설명하는 고유한 이름을 지정합니다.

참고: EDML 이름 공간은 폴더 구조에 상관없이 공유되므로 반드시 고유한 파일 이름을 사용해야 합니다. 파일 이름은 Macintosh 제한 사항에 따라 확장명 .edml을 포함하여 31자를 넘어서는 안됩니다.

서버 비헤이비어의 런타임 코드는 EDML 파일 안에 있습니다. EDML 파서는 런타임 코드와 EDML 마크업을 혼동해서는 안되므로, 런타임 코드 앞뒤에 CDATA를 첨부해야 합니다. CDATA 태그는 문자 데이터를 나타내며 EDML 마크업이 아닌 텍스트는 무엇이든 가능합니다. CDATA 태그를 사용할 때 EDML 파서는 이 태그 안의 내용을 마크업으로 해석하는 대신, 일반 텍스트 블록으로 인식합니다. CDATA로 표시된 블록은 <![CDATA[로 시작하고]]>로 끝납니다.

Hello, World 텍스트를 삽입할 때는 다음 예제와 같이 EDML을 지정하는 것이 간단합니다.

```
<insertText>Hello, World</insertText>
```

그러나 <img src='foo.gif'>와 같은 태그가 포함된 내용을 삽입할 때는 EDML 파서를 혼동할 수 있습니다. 이 경우 다음 예제에서와 같이 CDATA 구조에 내용을 포함합니다.

```
<insertText><![CDATA[<img src='foo.gif'>]]></insertText>
```

ASP 런타임 코드는 다음 예제와 같이 CDATA 태그로 묶여 있습니다.

```
<![CDATA[
    <% if (@@rs@@.EOF) Response.Redirect("@@new__url@@"); %>
]]
```

CDATA 태그 때문에 태그 내의 다른 내용과 함께 ASP 태그 <%= %>는 처리되지 않습니다. 대신, EDM(Extension Data Manager)은 다음 예제에 표시된 대로 해석되지 않은 텍스트를 받습니다.

```
<% if (Recordset1.EOF) Response.Redirect("http://www.Adobe.com"); %>
```

다음 EDML 정의에서 CDATA 태그가 표시될 적절한 위치가 예제에 표시됩니다.

EDML 그룹 파일 태그

이러한 태그와 속성은 EDML 그룹 파일 내에서 유효합니다.

<group>

설명
이 태그에는 참여자 그룹에 대한 모든 세부 사항이 포함됩니다.

부모
없음

유형
블록 태그

필수
예

<group> 속성

다음 항목은 해당 그룹 태그의 유효한 속성입니다.

version

설명
이 속성은 현재의 서버 비헤이비어를 처리하는 Dreamweaver 서버 비헤이비어가 대상으로 하는 Dreamweaver 버전을 정의합니다. Dreamweaver CS3의 버전 번호는 9입니다. 버전이 지정되어 있지 않으면 Dreamweaver에서는 버전 7을 기본값으로 사용합니다. Dreamweaver 버전 9.0의 경우 서버 비헤이비어 구성 관리자가 만드는 모든 그룹 및 참여자는 버전 속성을 9.0으로 설정합니다. 이 속성의 그룹 버전은 현재 아무 영향을 주지 않습니다.

부모
group

유형
속성

필수
아니오

serverBehavior

설명
serverBehavior 속성은 그룹을 사용할 수 있는 서버 비헤이비어를 지정합니다. 그룹의 참여자인 quickSearch 문자열이 문서에서 검색되면 serverBehavior 속성에 표시된 서버 비헤이비어는 Dreamweaver에서 findServerBehaviors() 함수를 호출하도록 합니다.

경우에 따라, 여러 그룹이 단일 서버 비헤이비어와 연결되어 있으면 해당 서버 비헤이비어는 사용할 특정 그룹을 결정해야 합니다.

부모
group

유형
속성

필수
아니오

값
이 값은 다음 예제에서와 같이 Configuration/ServerBehaviors 폴더에 있는 서버 비헤이비어 HTML 파일의 정확한 이름이며, 경로는 표시되지 않습니다.

```
<group serverBehavior="redirectIfEmpty.htm">
```

dataSource

설명
이 고급 기능은 Dreamweaver에 추가될 수 있는 새 데이터 소스를 지원합니다.

서버 비헤이비어의 여러 버전은 사용하는 데이터 소스에 따라 달라질 수 있습니다. 예를 들어 반복 영역 서버 비헤이비어는 표준 Recordset.htm 데이터 소스에 맞게 디자인되었습니다. Dreamweaver가 COM 객체와 같은 새 데이터 소스 유형을 지원하도록 확장되었으면 반복 영역의 다른 구현을 사용하여 그룹 파일에 dataSource="COM.htm"을 설정할 수 있습니다. 그런 다음, 새 데이터 소스가 선택되면 반복 영역 서버 비헤이비어는 반복 영역의 새 구현을 적용합니다.

부모
group

유형
속성

필수
아니오

값
다음 예제에 표시된 대로 Configuration/DataSources 폴더에 있는 데이터 소스 파일의 정확한 이름입니다.

```
<group serverBehavior="Repeat Region.htm" dataSource="myCOMdataSource.htm">
```

이 그룹은 COM 데이터 소스를 사용하는 경우에 사용할 반복 영역 서버 비헤이비어의 새 구현을 정의합니다. applyServerBehaviors()에서 다음 예제에서와 같이 매개 변수 객체에 MM_dataSource 속성을 설정하여 이 그룹이 적용되도록 지정할 수 있습니다.

```
function applyServerBehavior(ssRec) {
    var paramObj = new Object();
    paramObj.rs = getComObjectName();
    paramObj.MM_dataSource = "myCOMdataSource.htm";

    dwscripts.applySB(paramObj, sbObj);
}
```

subType

설명
이 고급 기능은 서버 비헤이비어의 여러 구현을 지원합니다.

서버 비헤이비어의 여러 버전은 사용자의 선택에 따라 달라질 수 있습니다. 서버 비헤이비어가 적용되어 있지만 여러 그룹 파일이 관련된 경우에는 subType 값에 전달하여 정확한 그룹 파일을 선택할 수 있습니다. 이 경우, 해당 subType 값을 가진 그룹이 적용됩니다.

부모
group

유형
속성

필수
아니오

값
이 값은 다음 예제에서와 같이 적용할 그룹을 결정하는 고유한 문자열입니다.

```
<group serverBehavior="myServerBehavior.htm" subType="longVersion">
```

이 그룹 속성은 myServerBehavior 하위 유형의 긴 버전을 정의합니다. 또한 subType="shortVersion" 속성을 가진 버전도 만들어집니다. applyServerBehaviors()에서 다음 예제에서와 같이 매개 변수 객체에 MM_subType 속성을 설정하여 적용되어야 할 그룹을 나타낼 수 있습니다.

```
function applyServerBehavior(ssRec) {
    var paramObj = new Object();
    if (longVersionChecked) {
        paramObj.MM_subType = "longVersion";
    } else {
        paramObj.MM_subType = "shortVersion";
    }
    dwscripts.applySB(paramObj, sbObj);
}
```

<title>

설명
이 문자열은 현재 문서에서 검색된 각 서버 비헤이비어 인스턴스에 대해 [서버 비헤이비어] 패널에 표시되는 문자열입니다.

부모
group

유형
블록 태그

필수
아니오

값
이 값은 다음 예제에서와 같이 각 인스턴스를 고유하게 하는 인수 이름을 포함할 수 있는 일반 텍스트 문자열입니다.

```
<title>Redirect If Empty (@@recordsetName@@)</title>
```

<groupParticipants>

설명
이 태그에는 groupParticipant 선언의 배열이 포함됩니다.

부모
group

유형
블록 태그

필수
예

<groupParticipants> 속성

다음 항목은 groupParticipants 태그의 유효한 속성입니다.

selectParticipant

설명
[서버 비헤이비어] 패널에서 인스턴스를 선택한 경우 문서에서 선택되어 강조 표시될 참여자를 지정합니다. 이 패널에 나열되는 서버 비헤이비어 인스턴스는 선택된 참여자에 의해 순서가 정해지므로 참여자가 표시되지 않은 경우에도 selectParticipant 속성을 설정합니다.

부모
groupParticipants

유형
속성

필수
아니오

값
participantName 값은 다음 예제에서와 같이 하나의 그룹 참여자로 표시된 단일 참여자 파일의 정확한 이름이며, 확장명 .edml은 표시되지 않습니다. 자세한 내용은 246페이지의 "name"을 참조하십시오.

```
<groupParticipants selectParticipant="redirectIfEmpty_link">
```

<groupParticipant>

설명
이 태그는 해당 그룹 내에 단일 참여자가 포함된다는 것을 나타냅니다.

부모
groupParticipants

유형
태그

필수
예. 적어도 하나는 필요합니다.

<groupParticipant> 속성

다음 항목은 groupParticipant 태그의 유효한 속성입니다.

name

설명
이 속성은 특정 참여자가 해당 그룹에 포함되도록 명명합니다. groupParticipant 태그에 있는 groupParticipant 속성은 참여자의 파일 이름과 동일해야 하며 파일 확장명 .edml은 표시되지 않습니다.

부모
groupParticipant

유형
속성

필수
예

값
이 값은 다음 예제에서와 같이 임의의 참여자 파일의 정확한 이름이며, 확장명 .edml는 표시되지 않습니다.

```
<groupParticipant name="redirectIfEmpty_init">
```

이 예제는 redirectIfEmpty_init.edml 파일을 참조합니다.

partType

설명
이 속성은 참여자의 유형을 나타냅니다.

부모
groupParticipant

유형
속성

필수
아니오

값
identifier, member, option, multiple, data

- **identifier** 값은 전체 그룹을 식별하는 참여자입니다. 이 참여자가 문서에 있으면 그룹이 있는 것으로 간주됩니다. partType 속성이 지정되어 있지 않으면 이것이 기본값입니다.
- **member** 값은 특정 그룹의 정규 멤버입니다. 이것만 단독으로 검색되면 그룹을 나타내지 않습니다. 그룹에 없으면 해당 그룹은 불완전한 것으로 간주됩니다.
- **option** 값은 해당 참여자가 선택 사항이라는 것을 알려 줍니다. 이것이 없는 경우에도 그룹은 완전한 것으로 간주되며 [서버 비헤이비어] 패널에 불완전 플래그가 설정되지 않습니다.
- **multiple**은 참여자가 선택 사항이며 해당 사본 여러 개가 서버 비헤이비어와 연결될 수 있다는 것을 나타냅니다. 이 참여자 고유의 인수는 값이 다를 수 있기 때문에 참여자들을 그룹화할 때 사용되지 않습니다.
- **data** 값은 프로그래머가 추가 그룹 데이터를 위한 저장소로 사용하는 비표준 참여자입니다. 다른 모든 경우에 무시됩니다.

참여자 EDML 파일

이러한 태그와 속성은 EDML 참여자 파일에서 유효합니다.

<participant>

설명
이 태그에는 단일 참여자의 모든 세부 사항이 포함됩니다.

부모
없음

유형
블록 태그

필수
예

<participant> 속성

다음 항목은 해당 참여자 태그의 유효한 속성입니다.

version

설명
이 속성은 현재의 서버 비헤이비어를 처리하는 Dreamweaver 서버 비헤이비어가 대상으로 하는 Dreamweaver 버전을 정의합니다. Dreamweaver CS3의 버전 번호는 9입니다. 버전이 지정되어 있지 않으면 Dreamweaver에서는 버전 7을 기본값으로 사용합니다. Dreamweaver 버전 0.0의 경우 서버 비헤이비어 구성 관리자가 만드는 모든 그룹 및 참여자는 버전 속성을 9.0으로 설정합니다.

참고: 참여자 버전 속성과 그룹 버전 속성이 다른 경우, 참여자 버전 속성은 그룹 버전 속성을 무시합니다. 그러나 참여자가 버전을 지정하지 않은 경우, 참여자 파일은 그룹 버전 속성을 사용합니다.

참여자 파일의 경우, 이 속성에 따라 코드 블록 병합이 일어날지 결정됩니다. 이 속성이 없거나 4 이전의 버전으로 설정된 참여자의 경우 삽입된 코드 블록이 페이지의 다른 코드 블록과 병합되지 않습니다. 5 이상 버전으로 설정된 참여자의 경우, 페이지의 다른 코드 블록과 병합될 수도 있습니다. HTML 태그 위와 아래에 있는 참여자의 경우에만 코드 블록 병합이 이루어진다는 점에 주의하십시오.

부모
participant

유형
속성

필수
아니오

<quickSearch>

설명
이 태그는 성능 이유에 사용되는 단순한 검색 문자열입니다. 이 경우 정규식은 사용할 수 없습니다. 문자열이 현재 문서에 있으면 특정 인스턴스를 찾기 위해 나머지 검색 패턴이 호출됩니다. 항상 검색 패턴을 사용하기 위해 이 문자열을 비워둘 수 있습니다.

부모
participant

유형
블록 태그

필수
아니오

값

searchString 값은 참여자가 존재하는 경우 페이지에 있는 리터럴 문자열입니다. 성능을 최대화하기 위해서는 이 문자열이 가능한 고유성을 유지하는 것이 좋지만 반드시 고유해야 하는 것은 아닙니다. 다음 예제에 표시된 대로 대/소문자는 구분되지 않지만 사용자가 변경할 수 있는 불필요한 공백은 주의해야 합니다.

```
<quickSearch>Response.Redirect</quickSearch>
```

quickSearch 태그가 비어 있으면 일치하는 것으로 간주되며, 더 정확한 검색에서는 searchPattern 태그에 정의된 정규식을 사용합니다. 이것은 간단한 문자열로는 신뢰할만한 검색 패턴을 표현할 수 없고 정규식이 필요한 경우 유용합니다.

<insertText>

설명

문서에 삽입할 항목과 위치에 대한 정보를 제공합니다. 이 태그에는 삽입할 텍스트가 포함됩니다. 사용자 정의된 일부 텍스트는 @@parameterName@@ 형식으로 나타내야 합니다.

translator-only 참여자 같은 경우에는 이 태그가 필요하지 않을 수도 있습니다.

부모

implementation

유형

블록 태그

필수

아니오

값

이 값은 문서에 삽입할 텍스트입니다. 텍스트의 부분들을 사용자 정의해야 할 경우 나중에 매개 변수로 전달할 수 있습니다. 인수를 포함할 때에는 두 개의 at 기호(@@)를 사용해야 합니다. 이 텍스트는 EDML 구조와 충돌할 수 있으므로 다음 예제에서와 같이 CDATA 구조를 사용해야 합니다.

```
<insertText location="aboveHTML">
    <![CDATA[<%= @@recordset@@).cursorType %>]]>
</insertText>
```

텍스트가 삽입되면 @@recordset@@ 인수는 사용자가 제공하는 레코드세트 이름으로 바뀝니다. 조건 및 반복 코드 블록에 대한 자세한 내용은 **Dreamweaver 시작하기**에서 "사용자 정의 서버 비헤이비어 추가"를 참조하십시오.

<insertText> 속성

다음 항목은 insertText 태그의 유효한 속성입니다.

location

설명

이 속성은 참여자 텍스트가 삽입될지 여부를 지정합니다. 삽입 위치는 searchPatterns 태그의 whereToSearch 속성과 연관되므로 두 가지 모두 신중하게 설정해야 합니다. 자세한 내용은 252페이지의 “whereToSearch”를 참조하십시오.

부모

insertText

유형
속성

필수
예

값
aboveHTML[+***weight***], **belowHTML**[+***weight***], **beforeSelection**, **replaceSelection**, **wrapSelection**, **afterSelection**, **beforeNode**, **replaceNode**, **afterNode**, **firstChildOfNode**, **lastChildOfNode**, **nodeAttribute[+attribute]**

- **aboveHTML**[+***weight***] 값은 텍스트를 HTML 태그(서버 코드에만 사용 가능) 위에 삽입합니다. 가중치는 1부터 99 사이의 정수일 수 있으며 다른 여러 참여자들 사이에 상대적인 순서를 유지하기 위해 사용됩니다. 규칙에 따라, 레코드세트의 가중치는 50이므로 참여자가 레코드세트 변수를 참조하면 60과 같은 더 큰 가중치가 필요하므로 다음 예제에서와 같이 레코드세트 아래에 코드가 삽입됩니다.

  ```
  <insert location="aboveHTML+60">
  ```

  가중치가 제공되지 않으면 내부적으로 100의 가중치가 지정되며, 이것은 다음 예제와 같이 특히 가중치가 적용된 모든 참여자 아래에 추가됩니다.

  ```
  <insert location="aboveHTML">
  ```

- **belowHTML**[+***weight***] 값은 참여자가 닫기 /HTML 태그 아래에 추가된다는 점만 제외하면 aboveHTML 값과 유사합니다.
- **beforeSelection** 값은 현재 선택 영역 또는 삽입 포인터 앞에 텍스트를 삽입합니다. 선택한 것이 없으면 body 태그의 끝에 텍스트를 삽입합니다.
- **replaceSelection** 값은 현재 선택 영역을 이 텍스트로 대체합니다. 선택한 것이 없으면 body 태그의 끝에 텍스트를 삽입합니다.
- **wrapSelection** 값은 현재 선택 영역에 포함된 태그의 짝을 맞추고 선택 영역 앞에 블록 태그를 삽입하고 해당 선택 영역 뒤에 적절한 닫기 태그를 추가합니다.
- **afterSelection** 값은 현재 선택 영역 또는 삽입 포인터 뒤에 텍스트를 삽입합니다. 선택한 것이 없으면 body 태그의 끝에 텍스트를 삽입합니다.
- **beforeNode** 값은 DOM의 특정 위치인 노드의 앞에 텍스트를 삽입합니다. 삽입 작업을 위해 dwscripts.applySB()와 같은 함수가 호출되면 노드 포인터는 **paramObj** 매개 변수로 전달되어야 합니다. 사용자가 정의할 수 있는 이 매개 변수의 이름은 nodeParamName 속성을 사용하여 지정해야 합니다. 자세한 내용은 251페이지의 “nodeParamName”을 참조하십시오.

  요약하면, 해당 위치에 node라는 단어가 있으면 반드시 nodeParamName 태그를 선언해야 합니다.
- **replaceNode** 값은 특정 노드를 이 텍스트로 대체합니다.
- **afterNode** 값은 특정 노드 뒤에 이 텍스트를 삽입합니다.
- **firstChildOfNode** 값은 텍스트를 블록 태그의 첫 번째 자식으로 삽입합니다. 예를 들어 FORM 태그의 시작 부분에 무언가를 삽입하려고 할 때 사용합니다.
- **lastChildOfNode**는 텍스트를 블록 태그의 마지막 자식으로 삽입합니다. 예를 들어 FORM 태그 끝 부분에 코드를 삽입하는 것은 숨겨진 양식 필드를 추가할 때 유용합니다.
- **nodeAttribute[+attribute]**는 태그 노드의 속성을 설정합니다. 이 속성이 존재하지 않으면 이 값은 해당 속성을 만듭니다.

  예를 들어 <insert location="nodeAttribute+ACTION" nodeParamName="form">을 사용하여 양식의 ACTION 속성을 설정합니다. 이렇게 하면 사용자의 FORM 태그는 <form>에서 <form action="myText">로 변경됩니다.

  속성을 지정하지 않으면 **nodeAttribute** 위치는 해당 텍스트가 열기 태그에 직접 추가되도록 합니다. 예를 들어 insert location="nodeAttribute"를 사용하여 태그에 선택적 속성을 추가합니다. 이 속성은 <input type="checkbox">에서 <input type="checkbox" <%if(foo)Reponse.Write("CHECKED")%>>로 사용자의 INPUT 태그를 변경하는 데 사용할 수 있습니다.

참고: location="nodeAttribute" 속성 값의 경우, 마지막 검색 패턴은 해당 속성이 시작하고 끝나는 위치를 결정합니다. 마지막 패턴이 명령문 전체를 찾는지 확인합니다.

nodeParamName

설명
이 속성은 노드 상대적 삽입 위치에만 사용됩니다. 이 속성은 삽입 시점에 노드를 전달하는 매개 변수의 이름을 알려 줍니다.

부모
insertText

유형
속성

필수
이 속성은 삽입 위치에 node라는 단어가 있는 경우에만 필요합니다.

값
tagtype_Tag 값은 매개 변수 객체와 함께 dwscripts.applySB() 함수에 전달되는 노드 매개 변수의 사용자 지정 이름입니다. 예를 들어 양식에 텍스트를 삽입할 때 **form_tag** 매개 변수를 사용할 수 있습니다. 서버 비헤이비어 applyServerBehavior() 함수에서 다음 예제에서와 같이 **form_tag** 매개 변수를 사용하여 업데이트할 정확한 양식을 표시할 수 있습니다.

```
function applyServerBehavior(ssRec) {
    var paramObj = new Object();
    paramObj.rs = getRecordsetName();
    paramObj.form_tag = getFormNode();
    dwscripts.applySB(paramObj, sbObj);
}
```

다음 예제와 같이 EDML 파일에 form_tag 노드 매개 변수를 표시할 수 있습니다.

```
<insertText location="lastChildOfNode" nodeParamName="form_tag">
    <![CDATA[<input type="hidden" name="MY_DATA">]]>
</insertText>
```

텍스트는 lastChildOfNode 값으로 삽입되고 특정 노드는 해당 매개 변수 객체의 form_tag 속성을 사용하여 전달됩니다.

<searchPatterns>

설명
이 태그는 문서에서 참여자 텍스트를 찾는 것에 대한 정보를 제공합니다. 이 태그에는 참여자를 검색할 때 사용되는 패턴 목록이 포함되어 있습니다. 여러 검색 패턴이 정의되어 있는 경우, isOptional 플래그를 사용하여 선택적이라고 표시되어 있지만 않으면, 검색 중인 텍스트에서 해당 패턴을 모두 찾을 수 있습니다. 검색 패턴은 논리 AND 관계를 가지고 있습니다.

부모
implementation

유형
블록 태그

필수
아니오

<searchPatterns> 속성

다음 항목은 searchPatterns 태그의 유효한 속성입니다.

whereToSearch

설명
이 속성은 참여자 텍스트를 검색할 위치를 지정합니다. 이 속성은 삽입 위치와 연관되므로 각 속성을 신중하게 설정해야 합니다. 자세한 내용은 249페이지의 “location”을 참조하십시오.

부모
searchPatterns

유형
속성

필수
예

값
***directive*, *tag+tagName*, tag+*, comment, text**

- **directive** 값은 모든 서버 지시문(서버 전용 태그)을 검색합니다. ASP 및 JSP의 경우에는 모든 <% ... %> 스크립트 블록을 검색한다는 의미입니다.

  참고: 태그 속성은 지시문에 포함되어 있는 경우에도 검색되지 않습니다.

- **tag+tagName** 값은 다음 예제와 같이 지정된 태그의 내용을 검색합니다.

  ```
  <searchPatterns whereToSearch="tag+FORM">
  ```

  이 예제에서는 form 태그만 검색합니다. 기본적으로, 전체 outerHTML 노드가 검색됩니다. INPUT 태그의 경우, 슬래시(/) 뒤에 유형을 지정합니다. 다음 예제에서 전송 버튼을 모두 검색하려면 다음 코드를 사용합니다.

  ```
  <searchPatterns whereToSearch="tag+INPUT/SUBMIT">.
  ```

- **tag+*** 값은 다음 예제와 같이 임의의 태그 내용을 검색합니다.

  ```
  <searchPatterns whereToSearch="tag+*">
  ```

  이 예제에서는 모든 태그를 검색합니다.

- 다음 예제와 같이 **comment** 값은 HTML 주석 <! ... >에서만 검색합니다.

  ```
  <searchPatterns whereToSearch="comment">
  ```

  이 예제에서는 <!-- my comment here -->와 같은 태그를 검색합니다.

- **text** 값은 다음 예제에서와 같이 순수한 텍스트 섹션에서만 검색합니다.

  ```
  <searchPatterns whereToSearch="text">
      <searchPattern>XYZ</searchPattern>
  </searchPatterns>
  ```

  이 예제에서는 XYZ 텍스트가 포함된 텍스트 노드를 찾습니다.

<searchPattern>

설명

이 태그는 참여자 텍스트를 식별하고 이 텍스트에서 매개 변수 값을 추출하는 패턴입니다. 각 매개 변수 하위 표현식은 괄호()에 포함해야 합니다.

참여자 텍스트를 식별하기 위해 사용하는 매개 변수가 없는 패턴, 매개 변수가 하나인 패턴, 매개 변수가 여러 개인 패턴을 사용할 수 있습니다. 선택적이지 않은 모든 패턴을 찾고, 각 매개 변수는 이름이 지정되고 정확히 한 번만 있어야 합니다.

searchPattern 태그 사용에 대한 자세한 내용은 264페이지의 “서버 비헤이비어 찾기”를 참조하십시오.

부모

searchPatterns

유형

블록 태그

필수

예

값

searchString, /regularExpression/, <empty>

- **searchString** 값은 대소문자가 구분되는 간단한 검색 문자열로, 매개 변수를 추출할 수는 없습니다.
- **/regularExpression/** 값은 정규식 검색 패턴입니다.
- **<empty>** 값은 패턴이 지정되지 않은 경우에 사용됩니다. 항상 일치하는 것으로 간주되며 값 전체가 첫 번째 매개 변수에 지정됩니다.

예를 들어 참여자 텍스트인 <%= RS1.Field.Items("author_id") %>를 식별하려면 간단한 패턴 다음에 두 개의 매개 변수 값을 추출하는 정확한 패턴을 정의합니다.

```
<searchPattern>Field.Items</searchPattern>
<searchPattern paramNames="rs,col">
    <![CDATA[
    /<%=\s*(\w+)\.Field\.Items\("(\w+)"\)/
    ]]>
</searchPattern>
```

이 예제는 해당 패턴과 정확하게 일치하며 첫 번째 하위 표현식(\w+)의 값을 매개 변수 rs에 지정하고 두 번째 하위 표현식(\w+)의 값을 매개 변수 col에 지정합니다.

참고: 정규식은 슬래시(/)로 시작하고 끝내야 합니다. 그렇지 않으면 표현식이 리터럴 문자열 검색으로 사용됩니다. /pattern/i처럼 정규식 뒤에 정규식 수정자 /pattern/i를 지정하여 대/소문자 구분 여부를 지정할 수 있습니다. 예를 들어 VBScript에서는 대/소문자가 구분되지 않으므로 /pattern/i를 사용해야 합니다. JavaScript에서는 대/소문자가 구분되므로 /pattern/을 사용해야 합니다.

매개 변수에 제한된 검색 위치의 내용 전체를 지정하려고 할 때도 있습니다. 그런 경우에는 다음 예제에 표시된 대로 패턴을 제공하지 않습니다.

```
<searchPatterns whereToSearch="tag+OPTION">
    <searchPattern>MY_OPTION_NAME</searchPattern>
    <searchPattern paramNames="optionLabel" limitSearch="innerOnly">
    </searchPattern>
</searchPatterns>
```

이 예제에서는 OPTION태그의 전체 innerHTML 내용에 optionLabel 매개 변수를 설정합니다.

<searchPattern> 속성

다음 항목은 searchPattern 태그의 유효한 속성입니다.

paramNames

설명

이 속성은 값을 추출할 매개 변수 이름이 들어 있는 쉼표로 구분된 목록입니다. 이 매개 변수는 하위 표현식의 순서대로 지정됩니다. 단일 매개 변수를 지정할 수도 있고 쉼표로 구분된 목록을 사용하여 여러 매개 변수를 지정할 수도 있습니다. 괄호로 묶는 다른 표현식이 사용되지만 매개 변수를 나타내지 않으면 [매개 변수 이름] 목록에서 추가 쉼표를 자리 표시자로 사용할 수 있습니다.

매개 변수 이름은 삽입 텍스트에 지정된 이름 및 업데이트 매개 변수와 일치해야 합니다.

부모

searchPattern

유형

속성

필수

예

값

paramName1, **paramName2**, ...

각 매개 변수 이름은 삽입 텍스트에 사용된 매개 변수의 이름과 똑같아야 합니다. 예를 들어 삽입 텍스트에 @@p1@@가 있으면 해당 이름을 사용하여 한 개의 매개 변수를 정의해야 합니다.

```
<searchPattern paramNames="p1">patterns</searchPattern>
```

단일 패턴을 사용하여 매개 변수를 여러 개 추출하려면 패턴에 나타나는 하위 표현식의 순서대로 쉼표로 구분된 매개 변수 이름 목록을 사용합니다. 다음 예제에서는 사용자의 검색 패턴을 표시한다고 가정합니다.

```
<searchPattern paramName="p1,,p2">/(\w+)_(BIG|SMALL)_(\w+)/</searchPattern>
```

추출할 매개 변수가 두 개(그 사이에 텍스트가 포함됨) 있습니다. 텍스트가 <%= a_BIG_b %>일 경우 검색 패턴의 첫 번째 하위 표현식은 a와 일치하므로 p1="a"입니다. 두 번째 하위 표현식은 무시됩니다. paramName 값에서 ,, 에 주의하십시오. 세 번째 하위 표현식은 b에 대응되므로 p2="b"입니다.

limitSearch

설명

이 속성은 검색을 whereToSearch 태그의 일부분으로 제한합니다.

부모

searchPattern

유형

속성

필수

아니오

값
all, attribute+attribName, tagOnly, innerOnly

- **all** 값(기본값)은 whereToSearch 속성에 지정된 전체 태그를 검색합니다.
- **attribute+attribName** 값은 다음 예제에서와 같이 지정된 속성의 값에서만 검색합니다.

```
<searchPatterns whereToSearch="tag+FORM">
    <searchPattern limitSearch="attribute+ACTION">
        /MY_PATTERN/
    </searchPattern>
</searchPatterns>
```

  이 예제에서는 FORM 태그의 ACTION 속성 값만 검색된다는 것을 알려 줍니다. 해당 속성이 정의되어 있지 않으면 이 태그가 무시됩니다.
- **tagOnly** 값은 바깥쪽 태그만 검색하며 innerHTML 태그는 무시합니다. 이 값은 whereToSearch가 태그인 경우에만 유효합니다.
- **innerOnly** 값은 innerHTML 태그만 검색하고 바깥쪽 태그는 무시합니다. 이 값은 whereToSearch가 태그인 경우에만 유효합니다.

isOptional

설명
이 속성은 해당 참여자를 찾는 작업에 검색 패턴이 필요하지 않다는 것을 알려 주는 플래그입니다. 이것은 중요하지 않은 추출 매개 변수를 가질 수도 있는 복잡한 참여자에게 유용합니다. 참여자를 확실하게 구별하는 일부 패턴을 만들 수도 있고 중요하지 않은 매개 변수를 추출하는 선택적 패턴들도 사용할 수 있습니다.

부모
searchPattern

유형
속성

필수
아니오

값
true, false

- 이 값은 searchPattern이 해당 참여자를 식별하는 데 필요하지 않으면 **true**입니다.
- 이 값은 searchPattern 태그가 필요하면 **false**(기본값)입니다.

예를 들어 다음과 같은 간단한 레코드세트 문자열을 생각해 볼 수 있습니다.

```
<%
var Recordset1 = Server.CreateObject("ADODB.Recordset");
Recordset1.ActiveConnection = "dsn=andescoffee;";
Recordset1.Source = "SELECT * FROM PressReleases";
Recordset1.CursorType = 3;
Recordset1.Open();
%>
```

이 검색 패턴은 참여자를 식별하고 여러 매개 변수를 추출합니다. 그러나 **cursorType**과 같은 매개 변수가 없는 경우에는 이 패턴이 레코드세트로 인식됩니다. **cursor** 매개 변수는 선택 사항입니다. EDML에서 검색 패턴은 다음 예제와 유사합니다.

```
<searchPattern paramNames="rs">/var (\w+) = Server.CreateObject/
</searchPattern>
<searchPattern paramNames="src">/ActiveConnection = "([^\r\n]*)"/</searchPattern>
<searchPattern paramNames="conn">/Source = "([^\r\n]*)"/</searchPattern>
<searchPattern paramNames="cursor" isOptional="true">/CursorType = (\d+)/
</searchPattern>
```

레코드세트를 식별하기 위해서는 처음의 세 패턴이 필요합니다. 마지막 매개 변수를 찾을 수 없어도 레코드세트를 식별할 수 있습니다.

<updatePatterns>

설명
이 선택적인 고급 기능을 사용하면 참여자를 정확하게 업데이트할 수 있습니다. 이 태그가 없으면 참여자는 매번 참여자 텍스트 전체를 바꿔 자동으로 업데이트됩니다. updatePatterns 태그를 지정하는 경우, 참여자 내에서 각 매개 변수를 찾아서 대체하는 특정 패턴이 이 태그에 포함되어 있어야 합니다.

이 태그는 사용자가 참여자 텍스트를 편집할 때 효과적입니다. 이는 해당 태그를 사용한 경우 텍스트에서 변경해야 할 부분만 정확히 업데이트하기 때문입니다.

부모
implementation

유형
블록 태그

필수
아니오

<updatePattern>

설명
이 태그는 특정 유형을 가진 정규식으로, 참여자 텍스트를 정확하게 업데이트할 수 있도록 해 줍니다. 양식 @@paramName@@의 삽입 텍스트에 선언된 모든 고유한 매개 변수에는 업데이트 패턴 정의가 적어도 하나 이상 있어야 합니다.

부모
updatePatterns

유형
블록 태그

필수
예(updatePatterns 태그를 선언하는 경우에는 하나 이상 필요)

값
이 값은 양식 **/(pre-pattern)parameter-pattern(post-pattern)/**에서 괄호 안에 포함된 두 개의 하위 표현식 사이에서 매개 변수를 찾는 정규식입니다. 삽입 텍스트에서 각각의 고유한 @@paramName@@에 대해 업데이트 패턴을 적어도 하나 이상 정의해야 합니다. 다음 예제에서는 삽입 텍스트가 어떻게 표시되는지 보여 줍니다.

```
<insertText location="afterSelection">
    <![CDATA[<%= @@rs@@.Field.Items("@@col@@") %>]]>
</insertText>
```

페이지에 있는 삽입 텍스트의 특별한 인스턴스는 다음 예제와 같습니다.

```
<%= RS1.Field.Items("author_id") %>
```

두 매개 변수인 rs 및 col이 있습니다. 페이지에 텍스트를 삽입한 후에 이 텍스트를 업데이트하려면 다음과 같은 두 개의 업데이트 패턴 정의가 필요합니다.

```
<updatePattern paramName="rs" >
    /(\b)\w+(\.Field\.Items)/
</updatePattern>
<updatePattern paramName="col">
    /(\bItems\(")\w+("\))/
</updatePattern>
```

다른 특수 정규식 문자 외에 리터럴 괄호도 앞에 백슬래시(\)를 붙여야 합니다. \w+로 정의된 가운데 표현식은 매개 변수 rs 및 col에 각각 전달된 최신 값으로 업데이트됩니다. RS1 및 author_id 값은 새 값으로 정확하게 업데이트될 수 있습니다.

동일한 패턴이 여러 번 나오는 경우, **/pattern/g**와 같이 닫기 슬래시 뒤에 정규식 전역 플래그 g를 사용하여 동시에 업데이트할 수 있습니다.

참여자 텍스트가 길고 복잡하다면 다음 예제에 표시된 대로 단일 매개 변수를 업데이트하기 위해 패턴이 여러 개 필요합니다.

```
<% ...
    Recordset1.CursorType = 0;
    Recordset1.CursorLocation = 2;
    Recordset1.LockType = 3;
%>
```

세 위치에서 모두 레코드세트 이름을 업데이트하려면 다음 예제와 같이 단일 매개 변수에 대한 업데이트 패턴이 세 개 필요합니다.

```
<updatePattern paramName="rs">
    /(\b)\w+(\.CursorType)/
</updatePattern>
<updatePattern paramName="rs">
    /(\b)\w+(\.CursorLocation)/
</updatePattern>
<updatePattern paramName="rs">
    /(\b)\w+(\.LockType)/
</updatePattern>
```

이제 레코드세트의 새 값에 전달할 수 있으며 세 위치에서 정확하게 업데이트됩니다.

<updatePattern> 속성

다음 항목은 updatePattern 태그의 유효한 속성입니다.

paramName

설명

이 속성은 참여자를 업데이트하는 데 사용되는 값을 갖는 매개 변수의 이름을 나타냅니다. 이 매개 변수는 삽입 텍스트에 지정된 매개 변수 및 검색 매개 변수와 일치해야 합니다.

부모

updatePattern

유형
속성

필수
예

값
이 값은 삽입 텍스트에서 사용되는 매개 변수의 정확한 이름입니다. 다음 예제에서 삽입 텍스트에 @@rs@@ 값이 있으면 해당 이름을 가진 매개 변수가 있어야 합니다.

```
<updatePattern paramName="rs">pattern</updatePattern>
```

<delete>

설명
이 태그는 참여자를 삭제하는 방법을 제어할 수 있도록 해주는 선택적인 고급 기능입니다. 이 태그가 없으면 제거하는 것으로 참여자가 완전히 삭제되지만 참조하는 서버 비헤이비어가 없을 때만 가능합니다. delete 태그를 지정하면 절대로 삭제되지 않게 하거나 일부만 삭제되도록 지정할 수 있습니다.

부모
implementation

유형
태그

필수
아니오

<delete> 속성

다음 항목은 delete 태그의 유효한 속성입니다.

deleteType

설명
이 속성은 수행할 삭제 유형을 나타내는 데 사용됩니다. 참여자가 지시문, 태그 또는 속성이냐 여부에 따라 다른 의미를 갖습니다. 기본적으로, 참여자 전체가 삭제됩니다.

부모
delete

유형
속성

필수
아니오

값
all, **none**, **tagOnly**, **innerOnly**, **attribute+attribName**, **attribute+***

- **all**(기본값)은 전체 지시문이나 태그를 삭제합니다. 속성의 경우, 정의 전체를 삭제합니다.
- **none** 값은 결코 자동으로 삭제되지 않습니다.
- **tagOnly** 값은 바깥쪽 태그만 제거하고 innerHTML 태그의 내용은 그대로 둡니다. 속성의 경우, 블록 태그이면 바깥쪽 태그도 제거합니다. 지시문에는 의미가 없습니다.
- 태그에 적용될 때 **innerOnly** 값은 내용(innerHTML 태그)만 제거합니다. 속성의 경우 값만 제거합니다. 지시문에는 의미가 없습니다.
- 태그에 적용될 때 **attribute+attribName** 값은 지정된 속성만 제거합니다. 지시문과 속성에는 의미가 없습니다.
- **attribute+*** 값은 태그에 대한 모든 속성을 제거합니다. 지시문과 속성에는 의미가 없습니다.

서버 비헤이비어가 선택된 텍스트를 링크로 변환하면 다음 예제에서와 같이 바깥쪽 태그만 제거하여 해당 링크를 제거합니다.

```
<delete deleteType="tagOnly"/>
```

이 예제에서는 링크 참여자를 <A HREF="...">HELLO</A>에서 HELLO로 변경합니다.

<translator>

설명
이 태그는 참여자가 다르게 렌더링되고 사용자 정의 속성 관리자를 가질 수 있도록 참여자를 변환하기 위한 정보를 제공합니다.

부모
implementation

유형
블록 태그

필수
아니오

<searchPatterns>

설명
이 태그는 Dreamweaver가 지정된 인스턴스를 문서에서 찾을 수 있도록 합니다. 여러 검색 패턴이 정의되어 있는 경우, isOptional 플래그를 사용하여 선택적이라고 표시되어 있지만 않으면, 검색 중인 텍스트에서 해당 패턴을 모두 찾을 수 있습니다. 검색 패턴은 논리 AND 관계를 가지고 있습니다.

부모
translator

유형
블록 태그

필수
예

<translations>

설명
이 태그에는 변환 지침 목록이 포함되어 있습니다. 목록에 있는 각 지침은 해당 참여자를 찾을 위치 및 해당 참여자로 사용 가능한 작업을 알려 줍니다.

부모
translator

유형
블록 태그

필수
아니오

<translation>

설명
이 태그에는 참여자의 위치, 수행할 변환 유형 및 참여자 텍스트를 대체하는 내용이 포함된 단일 변환 지침이 포함되어 있습니다.

부모
translations

유형
블록 태그

필수
아니오

<translation> 속성

다음 항목은 translation 태그의 유효한 속성입니다.

whereToSearch

설명
이 속성은 텍스트를 찾을 위치를 지정합니다. 이때, 이 위치는 삽입 위치와 연관되므로 각 위치를 신중하게 설정해야 합니다. 자세한 내용은 249페이지의 “location”을 참조하십시오.

부모
translation

유형
속성

필수
예

limitSearch

설명
이 속성은 검색을 whereToSearch 태그의 일부분으로 제한합니다.

부모
translation

유형
속성

필수
아니오

translationType

설명
이 속성은 수행할 변환 유형을 알려 줍니다. 이 유형은 미리 설정되어 있고 변환에 해당되는 기능성을 제공합니다. 예를 들어 dynamic data를 지정하면, 변환된 모든 데이터는 Dreamweaver 동적 데이터와 동일하게 동작해야 합니다. 즉, 변환된 데이터는 [디자인] 뷰에서 동적 데이터 자리 표시자와 동일하게 표시되어야 하며(동적 배경색을 가진 중괄호({}) 표기) [서버 비헤이비어] 패널에 나타나야 합니다.

부모
translation

유형
속성

필수
예

값
dynamic data, **dynamic image**, **dynamic source**, **tabbed region start**, **tabbed region end**, **custom**

- 다음 예제에서 **dynamic data** 값은 변환된 지시문이 Dreamweaver 동적 데이터처럼 표시되고 동작한다는 것을 나타냅니다.

```
<translation whereToSearch="tag+IMAGE"
    limitSearch="attribute+SRC"
    translationType="dynamic data">
```

- 다음 예제에서 **dynamic image** 값은 변환된 속성이 Dreamweaver 동적 이미지처럼 표시되고 작동한다는 것을 나타냅니다.

```
<translation whereToSearch="IMAGE+SRC"
    translationType="dynamic image">
```

- 다음 예제에서 **dynamic source** 값은 변환된 지시문이 Dreamweaver 동적 소스처럼 작동한다는 것을 나타냅니다.

```
<translation whereToSearch="directive"
    translationType="dynamic source">
```

- 다음 예제에서 ***tabbed region start*** 값은 변환된 <CFLOOP> 태그가 탭 외곽선의 시작 부분을 정의한다는 것을 나타냅니다.

```
<translation whereToSearch="CFLOOP"
    translationType="tabbed region start">
```

- 다음 예제에서 ***tabbed region end*** 값은 변환된 </CFLOOP> 태그가 탭 외곽선의 끝 부분을 정의한다는 것을 나타냅니다.

```
<translation whereToSearch="CFLOOP"
    translationType="tabbed region end">
```

- **custom** 값은 변환 작업에 내부 Dreamweaver 기능이 전혀 추가되지 않는 기본적인 경우입니다. 다음 예제에 표시된 대로 사용자 정의 [속성 관리자]에 대해 삽입할 태그를 지정할 때 사용되기도 합니다.

```
<translation whereToSearch="directive"
    translationType="custom">
```

<openTag>

설명
이 태그는 변환 섹션의 시작에 삽입될 수 있는 선택적인 태그입니다. 이 태그를 사용하면 사용자 정의 속성 관리자와 같은 특정 Extension이 변환을 찾을 수 있습니다.

부모
translation

유형
블록 태그

필수
아니오

값
tagName 값은 유효한 태그 이름입니다. 이 경우 알려진 태그 유형과 충돌하지 않도록 고유한 이름을 지정해야 합니다. 예를 들어 <openTag>MM_DYNAMIC_CONTENT</openTag>를 지정하면 동적 데이터는 MM_DYNAMIC_CONTENT 태그로 변환됩니다.

<attributes>

설명
이 태그에는 변환된 태그에 추가될 속성 목록이 포함되어 있습니다. 이때 변환된 태그는 openTag 태그에 의해 지정됩니다. 만약 openTag 태그가 정의되어 있지 않고 searchPattern 태그가 tag를 지정하면 이 태그에는 검색된 태그에 추가할 변환된 속성 목록이 포함됩니다.

부모
translation

유형
블록 태그

필수
아니오

<attribute>

설명
이 태그는 변환된 태그에 추가할 단일 속성 또는 변환된 속성을 지정합니다.

부모
attributes

유형
블록 태그

필수
예. 적어도 하나는 필요합니다.

값
The **attributeName="attributeValue**" 사양은 값에 속성을 설정합니다. 대개, 다음 예제에 표시된 대로 속성 이름은 고정되어 있으며, 값에는 매개 변수 패턴으로 추출된 몇 가지 매개 변수 참조가 들어 있습니다.

```
<attribute>SOURCE="@@rs@@"</attribute>
<attribute>BINDING="@@col@@"</attribute>
```

또는

```
<attribute>
    mmTranslatedValueDynValue="VALUE={@@rs@@.@@col@@}"
</attribute>
```

<display>

설명
이 태그는 선택적 표시 문자열으로, 변환 작업 시 삽입되어야 합니다.

부모
translation

유형
블록 태그

필수
아니오

값
displayString 값은 텍스트 및 HTML로 구성된 문자열입니다. 매개 변수 패턴으로 추출되는 매개 변수 참조를 포함할 수 있습니다. 예를 들어 <display>{@@rs@@.@@col@@}</display>로 인해 변환은 {myRecordset.myCol}로 렌더링됩니다.

<closeTag>

설명
이 태그는 변환된 섹션의 끝에 삽입되는 선택적인 태그입니다. 이 태그를 사용하면 사용자 정의 속성 관리자와 같은 특정 Extension이 변환을 찾을 수 있습니다.

부모
translation

유형
블록 태그

필수
아니오

값
tagName 값은 유효한 태그 이름입니다. 이 값은 변환 **openTag** 태그와 일치해야 합니다.

예제
값 <closeTag>MM_DYNAMIC_CONTENT</closeTag>를 지정하면 동적 데이터는 </MM_DYNAMIC_CONTENT> 태그를 사용하여 끝내도록 변환됩니다.

서버 비헤이비어 기술

이 단원에서는 서버 비헤이비어를 만들고 편집하는 일반 및 고급 기술에 대해 다룹니다. 대부분의 제안 사항은 EDML 파일의 특정 설정과 관련되어 있습니다.

서버 비헤이비어 찾기

다음과 같은 방법으로 서버 비헤이비어를 찾을 수 있습니다.

- 검색 패턴 작성
- 선택적 검색 패턴 사용

검색 패턴 작성
서버 비헤이비어를 업데이트하거나 삭제하려면 Dreamweaver가 문서에서 각 인스턴스를 찾을 수 있는 방법을 제공합니다. 이렇게 하려면 quickSearch 태그와 searchPatterns 태그에 포함된 searchPattern 태그(하나 이상)가 필요합니다.

quickSearch 태그는 정규식이 아니라 문자열이어야 합니다. 이때 해당 문자열은 서버 비헤이비어가 페이지에 존재함을 나타냅니다. 대/소문자는 구분되지 않습니다. 이 태그는 짧고 고유해야 하며 사용자가 변경할 수 있는 공백 및 기타 섹션은 사용하지 말아야 합니다. 다음 예제에서는 간단한 ASP JavaScript 태그로 구성된 참여자를 보여 줍니다.

```
<% if (Recordset1.EOF) Response.Redirect("some_url_here") %>
```

다음 예제에서 quickSearch 문자열은 해당 태그를 검색합니다.

```
<quickSearch>Response.Redirect</quickSearch>
```

quickSearch 패턴은 서버 비헤이비어 인스턴스를 찾는 과정의 시작이며 성능 향상을 위한 것입니다. 이 문자열이 해당 문서에 있고 참여자가 서버 비헤이비어를 식별하면(그룹 파일에서는 이 참여자의 경우 partType="identifier"), 관련된 서버 비헤이비어 파일이 로드되고 findServerBehaviors() 함수가 호출됩니다. 참여자에 검색하거나 디버깅할만한 문자열이 없으면 다음 예제에서와 같이 quickSearch 문자열을 공백으로 둘 수 있습니다.

```
<quickSearch></quickSearch>
```

이 예제에서 서버 비헤이비어는 항상 로드되고 문서를 검색할 수 있습니다.

다음으로, searchPattern 태그는 quickSearch 태그보다 문서를 더 정확하게 검색하고 참여자 코드에서 매개 변수 값을 추출합니다. 검색 패턴은 특정 패턴이 포함된 일련의 searchPattern 태그를 사용하여 검색할 위치를 지정합니다. 이때 whereToSearch 속성이 사용됩니다. 이러한 패턴은 간단한 문자열이나 정규식을 사용할 수 있습니다. 앞의 예제 코드는 ASP 지시문이므로 whereToSearch="directive" 사양과 정규식은 다음 예제에 표시된 대로 지시문을 식별하고 매개 변수를 추출합니다.

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="rs,new__url">
        /if\s*\((\w+)\.EOF\)\s*Response\.Redirect\("([^\r\n]*)"\)/i
    </searchPattern>
</searchPatterns>
```

검색 문자열은 앞뒤에 슬래시(/)가 있는 정규식으로 정의되며 대/소문자를 구분하지 않는다는 의미의 i가 슬래시 뒤에 옵니다. 정규식에서 괄호() 및 마침표(.)와 같은 특수 문자는 앞에 백슬래시(\)를 붙여야 합니다. 두 매개 변수 rs와 new_url은 괄호 안의 하위 표현식을 사용하여 문자열에서 추출됩니다. 이때 매개 변수는 괄호 안에 포함되어야 합니다. 이 예제에서 (\w+) 및 ([^\r\n]*)는 매개 변수를 나타냅니다. 이러한 값은 보통 $1 및 $2로 반환되는 정규식 값에 해당합니다.

선택적 검색 패턴

일부 매개 변수가 검색되지 않은 경우에도 참여자를 식별하려는 경우가 있습니다. 참여자는 전화 번호와 같은 일부 선택적 정보를 저장합니다. 이러한 예로 다음 ASP 코드를 사용할 수 있습니다.

```
<% //address block
    LNAME = "joe";
    FNAME = "smith";
    PHONE = "123-4567";
%>
```

다음과 같은 검색 패턴을 사용할 수 있습니다.

```
<quickSearch>address</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="lname">/LNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="fname">/FNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="phone">/PHONE\s*=\s*"([^\r\n]*)"/i</searchPattern>
</searchPatterns>
```

앞의 예제에서는 전화 번호를 지정해야 합니다. 그러나 다음 예제와 같이 isOptional 속성을 추가하여 전화 번호를 선택 사항으로 만들 수 있습니다.

```
<quickSearch>address</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="lname">/LNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="fname">/FNAME\s*=\s*"([^\r\n]*)"/i</searchPattern>
    <searchPattern paramNames="phone" isOptional="true">/PHONE\s*=\s*"([^\r\n]*)"/i¬
    </searchPattern>
</searchPatterns>
```

이제 전화 번호를 찾을 수 없어도 해당 참여자는 인식됩니다.

참여자가 일치되는 방식

서버 비헤이비어에 참여자가 둘 이상 있으면 해당 참여자는 사용자 문서에서 식별되고 일치되어야 합니다. 사용자가 문서에 서버 비헤이비어의 여러 인스턴스를 적용하면 그에 따라 참여자의 각 그룹이 일치되어야 합니다. 참여자가 정확하게 일치되는지 확인하려면 매개 변수를 변경하거나 추가하고 고유하게 식별되도록 참여자를 구성합니다.

일치 작업에는 몇 가지 규칙이 필요합니다. 참여자는 이름이 같은 매개 변수의 값이 모두 같을 경우 일치됩니다. html 태그의 위와 아래에는 일련의 지정된 매개 변수 값을 가진 참여자의 한 인스턴스만 나타날 수 있습니다. html.../html 태그 내에서 참여자는 삽입에 사용되는 선택 영역 또는 공통 노드에 상대적인 위치에 따라 일치됩니다.

매개 변수가 없는 참여자는 그룹 파일이 있는 서버 비헤이비어를 사용하는 다음 예제에서와 같이 자동으로 일치됩니다.

```
<group serverBehavior="test.htm">
    <title>Test</title>
    <groupParticipants>
        <groupParticipant name="test_p1" partType="identifier" />
        <groupParticipant name="test_p2" partType="identifier" />
    </groupParticipants>
</group>
```

다음 예제에서는 html 태그 위에 간단한 두 참여자를 삽입합니다.

```
<% //test_p1 %>
<% //test_p2 %>
<html>
```

이 참여자가 검색되어 일치된 다음, [서버 비헤이비어] 패널에 **Test**가 한 번 나타납니다. 서버 비헤이비어를 다시 추가하는 경우 참여자가 있기 때문에 아무 것도 추가되지 않습니다.

참여자에 고유한 매개 변수가 있으면 html 태그의 위에 여러 인스턴스가 삽입될 수 있습니다. 예를 들어 참여자에 이름 매개 변수를 추가하여 사용자는 [서버 비헤이비어 테스트] 대화 상자에 고유한 이름을 입력할 수 있습니다. 사용자가 **aaa**라는 이름을 입력하면 다음과 같은 참여자가 삽입됩니다.

이 서버 비헤이비어를 **bbb**와 같은 다른 이름으로 다시 추가하면 문서는 다음 예제와 같이 나타납니다.

```
<% //test_p1 name="aaa" %>
<% //test_p2 name="aaa" %>
<html>
```

[서버 비헤이비어] 패널에 Test의 인스턴스가 두 개 나열되어 있습니다. 사용자가 페이지에 세 번째 인스턴스를 추가하고 이름을 **aaa**로 지정하려고 하면 해당 이름이 있기 때문에 아무 것도 추가되지 않습니다.

html 태그 내에서는 일치 작업을 할 때 위치 정보도 사용될 수 있습니다. 다음 예제에는 참여자가 두 개 있는데, 하나는 선택 영역 앞에 추가되고 다른 하나는 선택 영역 뒤에 추가됩니다.

```
<% if (expression) { //mySBName %>
```

임의의 HTML 선택 영역:

```
<% } //end mySBName %>
```

이러한 두 참여자는 매개 변수가 없으므로 함께 그룹화됩니다. 그러나 다음 예제에서처럼, HTML의 다른 위치에 이 서버 비헤이비어의 다른 인스턴스를 추가할 수 있습니다.

```
<% if (expression) { //mySBName %>
```

임의의 HTML 선택 영역:

```
<% } //end mySBName %>
```

추가 HTML:

```
<% if (expression) { //mySBName %>
```

또 다른 HTML 선택 영역:

```
<% } //end mySBName %>
```

이제 HTML에서 허용되는 각 참여자의 동일한 인스턴스가 두 개 있습니다. 이러한 두 인스턴스는 문서에서 발생하는 순서에 따라 일치됩니다.

다음 예제에서는 이러한 일치상의 문제점과 이 문제를 방지하는 방법을 보여 줍니다. 일부 동적 데이터의 세금을 계산하고 선택 영역에 결과를 표시하는 참여자를 만들 수 있습니다.

```
<% total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<html>
<body>
    The total (with taxes) is $<%=total%>
</body>
</html>
```

두 참여자는 공통 매개 변수가 없기 때문에 일치합니다. 그러나 이 서버 비헤이비어의 두 번째 인스턴스를 추가하는 경우 다음 코드가 있어야 합니다.

```
<% total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<% total = Recordset1.Fields.Item("salePrice").Value * 1.0825 %>
<html>
<body>
    The total0(with taxes) is $<%=total%>
    Sale price (with taxes) is $<%=total%>
</body>
</html>
```

한 매개 변수에만 total이라는 이름이 지정되어 있으므로 이 서버 비헤이비어는 더 이상 제대로 작동하지 않습니다. 이 문제를 해결하기 위해 고유 값을 가진 매개 변수가 있는지 확인하고 참여자를 일치시키는 데 해당 매개 변수를 사용할 수 있습니다. 다음 예제에서 열 이름을 사용하여 total 변수 이름을 고유하게 만들 수 있습니다.

```
<% itemPrice_total = Recordset1.Fields.Item("itemPrice").Value * 1.0825 %>
<% salePrice_total = Recordset1.Fields.Item("salePrice").Value * 1.0825 %>
<html>
<body>
    The total0(with taxes) is $<%=itemPrice_total%>
    Sale price (with taxes) is $<%=salePrice_total%>
</body>
</html>
```

이제 검색 패턴은 참여자를 고유하게 식별하고 대응시킵니다.

검색 패턴 분석

Dreamweaver에서는 참여자 searchPatterns 기능을 사용하여 다음 액션을 지원합니다.

- 파일 전송 의존성
- 포함 파일에 대한 것처럼 모든 파일 참조에 대한 파일 경로 업데이트

Dreamweaver에서는 서버 모델을 만들 때 모든 참여자를 검색하여 특별한 paramNames 속성을 찾아 패턴 목록을 구성합니다. URL을 찾아 파일 종속을 확인하고 경로명을 수정하기 위해 Dreamweaver에서는 paramNames 속성 중 하나가 _url로 끝나는 각 searchPattern 태그를 사용합니다. 단일한 searchPattern 태그에 URL을 여러 개 지정할 수 있습니다.

_includeUrl로 끝나는 paramNames 속성 값이 포함된 각 변환기 searchPattern 태그의 경우, Dreamweaver에서는 searchPattern 태그를 사용하여 페이지에 포함 파일 명령문을 변환합니다. 모든 URL 참조가 변환되지는 않기 때문에 Dreamweaver 는 다른 접미어 문자열을 사용하여 포함 파일 URL을 식별합니다. 또한, 단일 URL만 포함 파일로 변환될 수 있습니다.

searchPatterns 태그를 확인할 때 Dreamweaver에서는 다음 알고리즘을 사용합니다.

1 searchPatterns 태그에서 whereToSearch 속성을 찾습니다.

2 속성 값이 tag+로 시작하면 나머지 문자열은 태그 이름으로 간주됩니다. 태그 이름에는 공백을 사용할 수 없습니다.

3 searchPattern 태그에서 limitSearch 속성을 찾습니다.

4 속성 값이 attribute+로 시작하면 나머지 문자열은 속성 이름으로 간주됩니다. 속성 이름에는 공백을 사용할 수 없습니다.

이 네 단계가 성공적으로 수행되었으면 Dreamweaver에서는 특정 태그/속성 조합을 가정합니다. 그렇지 않으면, Dreamweaver에서는 _url 접미어 및 정의된 정규식이 포함된 paramName 속성을 사용하여 searchPattern 태그 검색을 시작합니다. 정규식에 대한 자세한 내용은 241페이지의 “정규식”을 참조하십시오.

다음 searchPatterns 태그의 예제에는 검색 패턴이 없습니다. 태그(cfinclude)를 속성(template)과 결합하여 종속 파일 확인 및 경로 수정 등과 같은 작업을 위해 URL을 분리하기 때문입니다.

```
<searchPatterns whereToSearch="tag+cfinclude">
    <searchPattern paramNames="include_url" limitSearch="attribute+template" />
</searchPatterns>
```

Dreamweaver는 JavaScript 레이어에서 항상 일반 텍스트로 변환되므로 태그/속성 조합(앞 예제 참조)은 변환에 적용되지 않습니다. 파일 종속 확인 및 경로 수정 등과 같은 작업은 C 레이어에서 수행됩니다. C 레이어에서 Dreamweaver는 내부적으로 문서를 지시문(일반 텍스트)와 효율적인 트리 구조로 파싱된 태그로 분할합니다.

서버 비헤이비어 업데이트

다음과 같은 방법으로 서버 비헤이비어를 업데이트할 수 있습니다.

- 바꾸기 업데이트
- 정밀 업데이트

바꾸기 업데이트

기본적으로 참여자 EDML 파일에는 <updatePatterns> 태그가 없으며 참여자의 인스턴스는 문서에서 전체적으로 대체되어 업데이트됩니다. 사용자가 기존 서버 비헤이비어를 편집하고 [확인]을 클릭하면 값이 변경된 매개 변수를 포함하는 참여자가 제거된 다음 같은 위치에 새 값으로 다시 삽입됩니다.

사용자가 문서에서 참여자 코드를 사용자 정의하면 참여자는 검색 패턴이 이전 코드를 찾을 경우 인식되지 못할 수도 있습니다. 더 단순화된 검색 패턴을 사용하여 문서에서 참여자 코드를 사용자 정의할 수 있습니다. 그러나 서버 비헤이비어 인스턴스를 업데이트하면 참여자가 바뀔 수 있으며 이 경우 사용자 정의 편집 내용이 손실됩니다.

정밀 업데이트

경우에 따라, 문서에 삽입된 참여자 코드는 사용자가 원하는 대로 정의하도록 허용하는 것이 더 좋을 수도 있습니다. 이와 같이 하려면 EDML 파일에서 검색 패턴을 제한하고 업데이트 패턴을 제공합니다. 참여자가 페이지에 추가된 후 서버 비헤이비어는 이 참여자의 특정 부분만 업데이트합니다. 다음 예제에서는 두 매개 변수를 가진 간단한 참여자를 보여 줍니다.

```
<% if (Recordset1.EOF) Response.Redirect("some_url_here") %>
```

이 예제에서는 다음 검색 패턴을 사용할 수 있습니다.

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="rs,new__url">
        /if\s*\((\w+)\.EOF\)\s*Response\.Redirect\("([^\r\n]*)"\)/i
    </searchPattern>
</searchPatterns>
```

사용자는 다음 예제와 같이 이 코드의 특정 인스턴스에 다른 테스트를 추가할 수 있습니다.

```
<% if (Recordset1.EOF || x > 2) Response.Redirect("some_url_here") %>
```

검색 패턴은 EOF 매개 변수 뒤에서 괄호를 검색하기 때문에 실패합니다. 검색 패턴의 범위를 더욱 확장하려면 다음 예제에서와 같이 검색 패턴을 분할하여 짧게 만드십시오.

```
<quickSearch>Response.Write</quickSearch>
<searchPatterns whereToSearch="directive">
    <searchPattern paramNames="rs">/(\w+)\.EOF/</searchPattern>
    <searchPattern paramNames="new__url">
        /if\s*\([^\r\n]*\)\s*Response\.Redirect\("([^\r\n]*)"/i
    </searchPattern>
</searchPatterns>
```

단순화된 이 검색 패턴은 매우 유연하므로 사용자가 이 패턴을 코드에 추가할 수 있습니다. 그러나 사용자가 [확인]을 클릭할 때 서버 비헤이비어가 URL을 변경한 경우 참여자가 바뀌고 사용자 정의 내용을 잃게 됩니다. 더 정확하게 업데이트하려면 각 매개 변수를 업데이트하기 위한 패턴이 들어 있는 updatePatterns 태그를 추가합니다.

```
<updatePatterns>
    <updatePattern paramNames="rs">/(\b)\w+(\.EOF)/\</updatePattern>
    <updatePattern paramNames="new__url">
        /(Response\.Redirect\(")[^\r\n]*("/i
    </updatePattern>
</updatePatterns>
```

업데이트 패턴에서 괄호는 순서가 바뀌어 매개 변수 앞과 뒤의 텍스트를 둘러쌉니다. 검색 패턴인 경우에는 textBeforeParam(param)textAfterParam 매개 변수를 사용합니다. 업데이트 패턴인 경우에는 (textBeforeParam)param(textAfterParam) 매개 변수를 사용합니다. 괄호로 묶인 두 개 하위 표현식 사이의 모든 텍스트는 매개 변수의 새 값으로 바뀝니다.

서버 비헤이비어 삭제

다음과 같은 방법으로 서버 비헤이비어를 삭제할 수 있습니다.

- 기본 삭제 및 종속 수
- 삭제 플래그를 사용하여 참여자 삭제 제한

기본 삭제 및 종속 수

사용자는 마이너스(-) 버튼을 클릭하거나 [삭제]를 눌러 [서버 비헤이비어] 패널에서 선택한 인스턴스를 삭제할 수 있습니다. 다른 서버 비헤이비어가 공유하는 참여자를 제외하고 모든 참여자가 제거됩니다. 특히, 하나 이상의 서버 비헤이비어가 같은 노드에 대한 참여자 포인터를 갖고 있으면 노드는 삭제되지 않습니다.

기본적으로, 참여자는 태그 전체를 제거하여 삭제됩니다. 삽입 위치가 wrapSelection이면 바깥쪽 태그만 제거됩니다. 속성의 경우 속성 선언 전체가 제거됩니다. 다음 예제에서는 form 태그의 ACTION 속성에 있는 속성 참여자를 보여 줍니다.

```
<form action="<% my_participant %>">
```

속성이 삭제된 후 form만 남게 됩니다.

삭제 플래그를 사용하여 참여자 삭제 제한

참여자가 삭제되는 방식을 제한하려면 EDML 파일에 delete 태그를 추가합니다. 다음 예제에서는 링크의 href 속성인 참여자를 보여 줍니다.

```
<a href="<%=MY_URL%>">Link Text</a>
```

이 속성 참여자가 삭제되면 <a>Link Text 태그만 남게 됩니다. 이 태그는 Dreamweaver에서 더 이상 링크로 나타나지 않습니다. 속성 값만 삭제하는 것이 좋습니다. 참여자 EDML 파일에 다음 태그를 추가하여 삭제가 수행됩니다.

```
<delete deleteType="innerOnly"/>
```

다른 방법은 <delete deleteType="tagOnly"/>를 입력하여 해당 속성을 삭제할 때 전체 태그를 제거하는 것입니다. 그렇게 되면, **Link Text** 텍스트만 남게 됩니다.

Share-in-memory JavaScript 파일

여러 HTML 파일이 특정 JavaScript 파일을 참조하면 Dreamweaver에서는 HTML 파일이 같은 JavaScript 소스를 공유할 수 있는 특정 위치로 JavaScript를 로드합니다. 이러한 파일에는 다음과 같은 코드가 있습니다.

```
//SHARE-IN-MEMORY=true
```

JavaScript 파일에 SHARE-IN-MEMORY 지시문이 있고 SRC 속성을 가진 SCRIPT 태그를 사용하여 HTML 파일이 해당 지시문을 참조하면 Dreamweaver에서는 코드가 모든 HTML 파일에 묵시적으로 포함되는 메모리 위치로 JavaScript를 로드합니다.

참고: 이 중심 위치로 로드된 JavaScript 파일들이 메모리를 공유하기 때문에 해당 파일은 어느 선언도 복제할 수 없습니다. share-in-memory 파일이 변수 또는 함수를 정의하고 다른 임의의 JavaScript 파일이 같은 변수 또는 함수를 정의하면 이름이 충돌합니다. 새 JavaScript 파일을 작성할 때는 이 파일들과 이름 지정 규칙을 알아야 합니다.

18장: 데이터 소스

데이터 소스 파일은 Configuration/DataSources/**ServerModelName** 폴더에 저장됩니다. Dreamweaver에서는 현재 ASP.NET/C#, ASP.NET/VisualBasic, ASP/JavaScript, ASP/VBScript, Adobe ColdFusion, JSP 및 PHP/MySQL 서버 모델을 지원합니다. 각 서버 모델의 하위 폴더에는 해당 서버 모델의 데이터 소스와 연관된 HTML 및 EDML 파일이 있습니다.

다음 표에서는 데이터 소스를 만들 때 사용하는 파일을 설명합니다.

경로	파일	설명
Configuration/DataSources/**ServerModelName**	**datasourcename**.htm	데이터 소스 이름과 지원 JavaScript 파일을 찾을 위치를 지정합니다.
Configuration/DataSources/**ServerModelName**	**datasourcename**.edml	데이터 소스 값을 나타내기 위해 문서에 삽입되는 코드를 정의합니다.
Configuration/DataSources/**ServerModelName**	**datasourcename**.js	필요한 코드를 문서에 추가, 삽입 및 삭제하는 JavaScript 함수가 포함되어 있습니다.

데이터 소스 작동 방식

Dreamweaver 사용자는 [바인딩] 패널을 사용하여 동적 데이터를 추가할 수 있습니다. 플러스(+) 메뉴에 나타나는 동적 데이터 객체는 페이지에 대해 지정된 서버 모델을 기반으로 합니다. 예를 들어 사용자는 ASP 응용 프로그램용 레코드세트, 명령, 요청 변수, 세션 변수, 응용 프로그램 변수 등을 삽입할 수 있습니다. 자세한 내용은 **Dreamweaver API 참조** 설명서에서 dreamweaver.dbi.getDataSources() 함수를 참조하십시오.

다음 단계에서는 동적 데이터 추가에 관련된 과정을 설명합니다.

1 사용자가 [바인딩] 패널에서 플러스(+) 메뉴를 클릭하면 팝업 메뉴가 나타납니다.

먼저 Dreamweaver에서는 메뉴의 내용을 확인하기 위해 데이터 소스와 동일한 폴더에서 DataSources.xml 파일(예: Configuration/DataSources/ASP_Js/DataSources.xml)을 찾습니다. DataSources.xml 파일은 팝업 메뉴의 내용을 설명합니다. 이 파일에는 팝업 메뉴에 표시할 HTML 파일에 대한 참조가 들어 있습니다.

Dreamweaver에서는 참조된 각 HTML 파일에서 title 태그를 찾습니다. 파일에 title 태그가 있는 경우에는 title 태그의 내용이 메뉴에 나타나고, 없는 경우에는 파일 이름이 메뉴에 사용됩니다.

Dreamweaver에서는 DataSources.xml 파일을 읽은 후 폴더의 나머지 파일을 검색하여 메뉴에 표시할 다른 항목을 찾습니다. DataSources.xml 파일이 없는 경우에도 동일한 작업을 수행합니다. Dreamweaver에서는 메뉴에 없는 파일을 주 폴더에서 찾으면 해당 파일을 메뉴에 추가합니다. 메뉴에 없는 파일이 하위 폴더에 있으면 Dreamweaver에서는 하위 메뉴를 만들고 해당 파일을 하위 메뉴에 추가합니다.

2 사용자가 플러스(+) 메뉴에서 항목을 선택하면 Dreamweaver에서는 addDynamicSource() 함수를 호출하여 데이터 소스의 코드를 사용자의 문서에 추가합니다.

3 Dreamweaver에서는 적절한 서버 모델 폴더에서 각 파일을 살펴보고 각 파일에서 findDynamicSources() 함수를 호출합니다. Dreamweaver에서는 반환된 배열의 각 값에 대해 동일한 파일에 있는 generateDynamicSourceBindings() 함수를 호출하여 사용자의 문서에 대한 각 데이터 소스의 모든 필드가 포함된 새 목록을 가져옵니다. 해당 필드는 사용자에게 [동적 데이터] 또는 [동적 텍스트] 대화 상자나 [바인딩] 패널에서 트리 컨트롤로 표시됩니다. ASP 문서의 데이터 소스 트리는 다음 예제와 같이 나타납니다.

```
Recordset (Recordset1)
    ColumnOneInRecordset
    ColumnTwoInRecordset
Recordset (Recordset2)
    ColumnOfRecordset
Request
    NameOfRequestVariable
    NameOfAnotherRequestVariable
Session
    NameOfSessionVariable
```

4 사용자가 [바인딩] 패널에서 데이터 소스 이름을 두 번 클릭하면 Dreamweaver에서는 editDynamicSource() 함수를 호출하여 트리 내에서의 편집 내용을 처리합니다.

5 사용자가 마이너스(-) 버튼을 클릭하면 Dreamweaver에서는 트리에서 현재 선택된 노드를 가져와서 deleteDynamicSource() 함수에 전달합니다. 이 함수는 이전에 addDynamicSource() 함수를 사용하여 추가된 코드를 삭제합니다. 현재 선택 항목을 삭제할 수 없으면 오류 메시지가 반환됩니다. deleteDynamicSource() 함수가 반환된 후, Dreamweaver에서는 findDynamicSources() 및 generateDynamicSourceBindings() 함수를 호출하여 데이터 소스 트리를 새로 고칩니다.

6 사용자가 데이터 소스를 선택하고 [동적 데이터] 또는 [동적 텍스트] 대화 상자에서 [확인]을 클릭하면 Dreamweaver에서는 generateDynamicDataRef() 함수를 호출합니다. 사용자가 [바인딩] 패널에서 [삽입] 또는 [바인딩]을 클릭할 경우에도 이 함수가 호출됩니다. 반환값은 문서에서 현재 삽입 포인터가 있는 위치에 삽입됩니다.

7 사용자가 [동적 데이터] 대화 상자를 사용하여 동적 데이터 객체를 편집하면 데이터 소스 트리의 선택 항목은 해당 객체로 초기화되어야 합니다. 이는 사용자가 [동적 텍스트] 대화 상자를 사용하여 동적 데이터 객체를 편집하는 경우에도 동일합니다. 트리 컨트롤을 초기화하기 위해 Dreamweaver에서는 해당 서버 모델 폴더(예: Configuration/DataSources/ASP_Js 폴더)의 각 파일을 검색하고 각 파일에서 inspectDynamicDataRef() 함수의 구현을 호출합니다.

   Dreamweaver에서는 inspectDynamicDataRef() 함수를 호출하여 동적 데이터 객체를 사용자 문서의 코드에서 트리의 항목으로 다시 변환합니다. 이 과정은 generateDynamicDataRef() 함수가 호출될 때의 상황과 정반대입니다. inspectDynamicDataRef() 함수가 두 개의 요소가 포함된 배열을 반환하면 Dreamweaver에서는 현재 선택 항목에 바인딩된 트리 항목에 시각적인 신호를 제공합니다.

8 사용자가 선택 항목을 변경할 때마다 Dreamweaver에서는 inspectDynamicDataRef() 함수를 호출하여 새로 선택한 항목이 동적 텍스트인지 동적 속성이 있는 태그인지를 확인합니다. 동적 텍스트인 경우 [바인딩] 패널에 현재 선택 항목에 대한 바인딩이 표시됩니다.

9 사용자가 이미 페이지에 추가한 동적 텍스트 객체 또는 동적 속성의 데이터 형식을 변경할 수도 있습니다. 이렇게 하려면 [동적 데이터] 또는 [동적 텍스트] 대화 상자나 [바인딩] 패널을 사용합니다. 형식이 변경되면 Dreamweaver에서는 generateDynamicDataRef() 함수를 호출하여 사용자의 문서에 삽입할 문자열을 가져옵니다. 그런 다음 해당 문자열을 formatDynamicDataRef() 함수에 전달합니다. 자세한 내용은 286페이지의 “formatDynamicDataRef()”를 참조하십시오. formatDynamicDataRef() 함수가 반환하는 문자열은 사용자의 문서에 삽입됩니다.

간단한 데이터 소스 예제

이 Extension은 Adobe ColdFusion 문서에 대한 [바인딩] 패널에 사용자 정의 데이터 소스를 추가합니다. 사용자는 새 데이터 소스에서 원하는 변수를 지정할 수 있습니다.

이 예제에서는 MyDatasource.js JavaScript 파일, MyDatasource_DataRef.edml 파일 및 MyDatasource Variable 명령 파일이 포함된 MyDatasource라는 데이터 소스를 만들어 사용자가 특정 변수의 이름을 입력할 수 있는 대화 상자를 구현합니다. MyDatasource 예제는 Cookie Variable 데이터 소스 및 URL Variable 데이터 소스의 구현을 기반으로 합니다. 이러한 데이터 소스의 파일은 Configuration/DataSources/ColdFusion 폴더에 있습니다.

정의 파일, EDML 파일, JavaScript 파일 및 지원 명령 파일을 만들어 이 데이터 소스를 만든 다음 새 데이터 소스를 테스트합니다.

데이터 소스 정의 파일 만들기

데이터 소스 정의 파일은 데이터 소스가 [바인딩] 패널의 플러스(+) 메뉴에 나타날 때 데이터 소스의 이름을 Dreamweaver에 알려 줍니다. 또한 데이터 소스 구현을 위해 지원 JavaScript 파일이 있는 위치를 알려 줍니다.

사용자가 [바인딩] 패널의 플러스(+) 메뉴를 클릭하면 Dreamweaver에서는 DataSources 폴더에서 현재 서버 모델을 검색하고 이 폴더의 HTML(HTM) 파일에 정의된 사용 가능한 모든 데이터 소스를 수집합니다. 그러므로 새 데이터 소스를 사용자가 사용할 수 있도록 하려면 데이터 소스 정의 파일을 만들어야 합니다. 이 데이터 소스 정의 파일은 title 태그를 사용하여 데이터 소스의 이름을 제공하고, script 태그를 사용하여 모든 지원 JavaScript 파일의 위치를 제공합니다.

또한 이 데이터 소스를 구현하려면 여러 개의 지원 파일이 필요합니다. 일반적으로 이러한 지원 파일의 함수를 사용할 필요가 없는 경우도 있지만 때로는 이러한 함수가 유용하며 이 예제와 같은 경우에는 이러한 함수가 필요합니다. 예를 들어 dwscriptsServer.js 파일에는 이 데이터 소스를 구현하는 데 사용되는 dwscripts.stripCFOutputTags() 함수가 포함되어 있습니다. 또한 DataSourceClass.js 파일을 사용하면 DataSource 클래스의 인스턴스를 만들어 findDynamicSources() 함수의 반환값으로 사용할 수 있습니다.

1 비어 있는 새 파일을 만듭니다.

2 다음을 입력합니다.

```
<HTML>
<HEAD>
<TITLE>MyDatasource</TITLE>
<SCRIPT SRC="../../Shared/Common/Scripts/dwscripts.js"></SCRIPT>
<SCRIPT SRC="../../Shared/Common/Scripts/dwscriptsServer.js"></SCRIPT>
<SCRIPT SRC="../../Shared/Common/Scripts/DataSourceClass.js"></SCRIPT>
<SCRIPT SRC="MyDatasource.js"></SCRIPT>
</HEAD>
<body></body>
</HTML>
```

3 이 파일을 Configuration/DataSources/ColdFusion 폴더에 MyDatasource.htm이라는 이름으로 저장합니다.

EDML 파일 만들기

EDML 파일은 데이터 소스 값을 나타내기 위해 문서에 삽입되는 코드를 정의합니다. EDML 파일에 대한 자세한 내용은 230페이지의 “서버 비헤이비어”를 참조하십시오. 사용자가 데이터 소스의 특정 값을 문서에 추가할 때 런타임에 실제 값으로 변환하는 코드가 삽입됩니다. 참여자 EDML 파일은 문서의 코드를 정의합니다. 자세한 내용은 247페이지의 “참여자 EDML 파일”을 참조하십시오.

MyDatasource Variable의 경우 Dreamweaver에서 ColdFusion 코드 <cfoutput>#MyXML.***variable***#</cfoutput>을 삽입하도록 할 수 있습니다. 여기서 **variable**은 사용자가 데이터 소스에서 원하는 값입니다.

1 비어 있는 새 파일을 만듭니다.

2 다음을 입력합니다.

```
<participant>
    <quickSearch><![CDATA[#]]></quickSearch>
    <insertText location="replaceSelection"><![CDATA[<cfoutput>#MyDatasource.
        @@bindingName@@#</cfoutput>]]></insertText>
    <searchPatterns whereToSearch="tag+cfoutput">
        <searchPattern paramNames="sourceName,bindingName"><![CDATA[/#(?:\s*\w+\s*\()?
        (MyDatasource)\.(\w+)\b[^#]*#/i]]></searchPattern>
    </searchPatterns>
</participant>
```

3 이 파일을 Configuration/DataSources/ColdFusion 폴더에 MyDatasource_DataRef.edml이라는 이름으로 저장합니다.

데이터 소스 API 함수를 구현하는 JavaScript 파일 만들기

데이터 소스의 이름, 지원 스크립트 파일의 이름 및 작업 중인 Dreamweaver 문서의 코드를 정의한 후, 필요한 코드를 문서에 추가, 삽입 및 삭제하는 기능을 사용자에게 제공하는 JavaScript 함수를 Dreamweaver에서 지정해야 합니다.

Cookie Variable 데이터 소스의 구성을 기반으로 다음 예제와 같이 MyXML 데이터 소스를 구현할 수 있습니다. addDynamicSource() 함수에서 사용되는 MyDatasource_Variable 명령은 276페이지의 "사용자 입력을 위한 지원 명령 파일 만들기"에 정의되어 있습니다.

1 비어 있는 새 파일을 만듭니다.

2 다음을 입력합니다.

```
//************** GLOBALS VARS *****************
var MyDatasource_FILENAME = "REQ_D.gif";
var DATASOURCELEAF_FILENAME = "DSL_D.gif";

//****************** API **********************
function addDynamicSource()
{
    MM.retVal = "";
    MM.MyDatasourceContents = "";
    dw.popupCommand("MyDatasource_Variable");
    if (MM.retVal == "OK")
    {
        var theResponse = MM.MyDatasourceContents;
        if (theResponse.length)
        {
            var siteURL = dw.getSiteRoot();
            if (siteURL.length)
            {
                dwscripts.addListValueToNote(siteURL, "MyDatasource", theResponse);
            }
            else
            {
                alert(MM.MSG_DefineSite);
            }
        }
        else
        {
        alert(MM.MSG_DefineMyDatasource);
        }
    }
}

function findDynamicSources()
{
    var retList = new Array();

    var siteURL = dw.getSiteRoot()

    if (siteURL.length)
    {
        var bindingsArray = dwscripts.getListValuesFromNote(siteURL, "MyDatasource");
        if (bindingsArray.length > 0)
        {

    // Here you create an instance of the DataSource class as defined in the
    // DataSourceClass.js file to store the return values.

        retList.push(new DataSource("MyDatasource",
                                    MyDatasource_FILENAME,
                                    false,
```

```
                                        "MyDatasource.htm"))
            }
        }

        return retList;
    }

    function generateDynamicSourceBindings(sourceName)
    {
        var retVal = new Array();

        var siteURL = dw.getSiteRoot();

        // For localized object name...
        if (sourceName != "MyDatasource")
        {
            sourceName = "MyDatasource";
        }

        if (siteURL.length)
        {
            var bindingsArray = dwscripts.getListValuesFromNote(siteURL, sourceName);
            retVal = getDataSourceBindingList(bindingsArray,
                                        DATASOURCELEAF_FILENAME,
                                        true,
                                        "MyDatasource.htm");
        }

        return retVal;
    }

    function generateDynamicDataRef(sourceName, bindingName, dropObject)
    {
        var paramObj = new Object();
        paramObj.bindingName = bindingName;
        var retStr = extPart.getInsertString("", "MyDatasource_DataRef", paramObj);

        // We need to strip the cfoutput tags if we are inserting into a CFOUTPUT tag
        // or binding to the attributes of a ColdFusion tag. So, we use the
        // dwscripts.canStripCfOutputTags() function from dwscriptsServer.js

        if (dwscripts.canStripCfOutputTags(dropObject, true))
        {
            retStr = dwscripts.stripCFOutputTags(retStr, true);
        }
        return retStr;
    }

    function inspectDynamicDataRef(expression)
    {
        var retArray = new Array();

        if(expression.length)
        {
            var params = extPart.findInString("MyDatasource_DataRef", expression);
            if (params)
            {
                retArray[0] = params.sourceName;
                retArray[1] = params.bindingName;
```

```
            }
        }

        return retArray;
    }
    function deleteDynamicSource(sourceName, bindingName)
    {
        var siteURL = dw.getSiteRoot();

        if (siteURL.length)
        {
            //For localized object name
            if (sourceName != "MyDatasource")
            {
                sourceName = "MyDatasource";
            }

            dwscripts.deleteListValueFromNote(siteURL, sourceName, bindingName);
        }
    }
```

3 이 파일을 Configuration/DataSources/ColdFusion 폴더에 MyDatasource.js라는 이름으로 저장합니다.

사용자 입력을 위한 지원 명령 파일 만들기

addDynamicSource() 함수에는 dw.popupCommand("MyDatasource_Variable") 명령이 포함되어 있습니다. 이 명령을 실행하면 사용자가 특정 변수 이름을 입력할 수 있는 대화 상자가 열립니다. 하지만 MyDatasource Variable에 사용할 대화 상자를 만들어야 합니다.

사용자에게 대화 상자를 제공하려면 일련의 명령 파일을 만듭니다. 명령 파일에는 HTML 형식의 명령 정의 파일 및 JavaScript 형식의 명령 구현 파일이 있습니다. 명령 파일에 대한 자세한 내용은 124페이지의 “명령의 작동 방식”을 참조하십시오.

명령 정의 파일은 지원하는 구현 JavaScript 파일의 위치와 사용자가 참조하는 대화 상자의 양식에 대한 정보를 Dreamweaver에 제공합니다. 지원 JavaScript 파일에 따라 대화 상자의 버튼과 대화 상자에서 사용자 입력을 지정하는 방법이 결정됩니다.

명령 정의 파일 만들기

1 비어 있는 새 파일을 만듭니다.

2 다음을 입력합니다.

```
<!DOCTYPE HTML SYSTEM "-//Adobe//DWExtension layout-engine 10.0//dialog">
<html>
<head>
<title>MyDatasource Variable</title>
<script src="MyDatasource_Variable.js"></script>
<SCRIPT SRC="../Shared/MM/Scripts/CMN/displayHelp.js"></SCRIPT>
<SCRIPT SRC="../Shared/MM/Scripts/CMN/string.js"></SCRIPT>
<link href="../fields.css" rel="stylesheet" type="text/css">
</head>
<body>
<form>
    <div ALIGN="center">
        <table border="0" cellpadding="2" cellspacing="4">
            <tr>
                <td align="right" valign="baseline" nowrap>Name:</td>
                <td valign="baseline" nowrap>
                    <input name="theName" type="text" class="medTField">
                </td>
            </tr>
        </table>
    </div>
</form>
</body>
</html>
```

3 이 파일을 Configuration/Commands 폴더에 MyDatasource_Variable.htm이라는 이름으로 저장합니다.

참고: MyDatasource_Variable.js 파일은 다음 절차에서 만들 구현 파일입니다.

지원 JavaScript 파일 만들기

1 비어 있는 새 파일을 만듭니다.

2 다음을 입력합니다.

```
//******************* API **********************

function commandButtons(){
    return new Array(MM.BTN_OK,"okClicked()",MM.BTN_Cancel,"window.close()");
}

//***************** LOCAL FUNCTIONS*****************

function okClicked(){
    var nameObj = document.forms[0].theName;

    if (nameObj.value) {
        if (IsValidVarName(nameObj.value)) {
            MM.MyDatasourceContents = nameObj.value;
            MM.retVal = "OK";
            window.close();
        } else {
            alert(nameObj.value + " " + MM.MSG_InvalidParamName);
        }
    } else {
        alert(MM.MSG_NoName);
    }
}
```

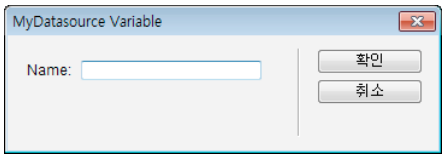
3 이 파일을 Configuration/Commands 폴더에 MyDatasource_Variable.js라는 이름으로 저장합니다.

새 데이터 소스 테스트

이제 Dreamweaver를 열거나 이미 열려 있는 경우 다시 시작하고, ColdFusion 파일을 열거나 새로 만들 수 있습니다.

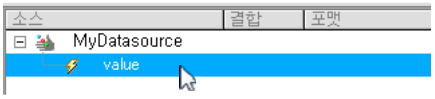
1 문서에 포인터가 있는 상태에서 [바인딩] 패널의 플러스(+) 메뉴를 클릭하여 사용 가능한 데이터 소스를 표시합니다. MyDatasource는 목록의 맨 아래에 나타납니다.

2 [MyDatasource] 데이터 소스 옵션을 클릭하여 앞에서 만든 MyDatasource Variable 대화 상자를 엽니다.

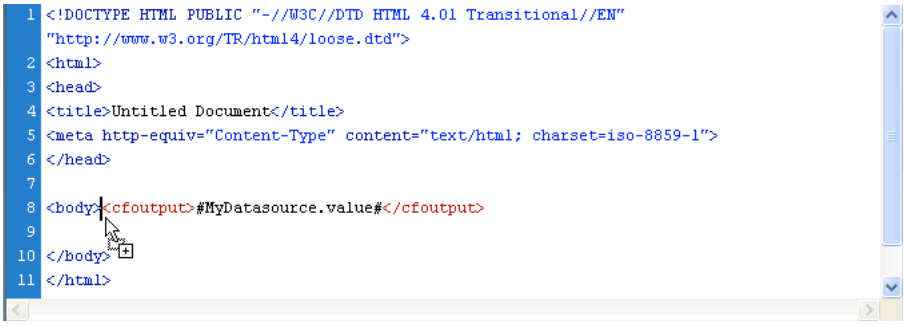


3 대화 상자에 값을 입력하고 [확인]을 클릭합니다.

[바인딩] 패널에는 다음 그림과 같이 데이터 소스가 트리 구조로 표시되며 데이터 소스 이름 아래에는 대화 상자의 변수가 표시됩니다.



4 변수를 문서로 드래그하면 Dreamweaver에서는 다음과 같이 EDML 파일로부터 해당 코드를 삽입합니다.



데이터 소스 API 함수

데이터 소스 API의 함수를 사용하여 데이터 소스를 검색, 추가, 편집 및 삭제하고 동적 데이터 객체를 생성하거나 관리할 수 있습니다.

addDynamicSource()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 동적 데이터 소스를 추가합니다. 이 함수는 각 데이터 소스 파일에서 한 번씩 구현되므로 플러스(+) 메뉴에서 데이터 소스를 선택하면 addDynamicSource() 함수의 해당 구현이 호출됩니다.

예를 들어 레코드세트 또는 명령의 경우 새 서버 비헤이비어를 문서에 삽입하는 dw.serverBehaviorInspector.popupServerBehavior() 함수가 호출됩니다. 요청, 세션 및 응용 프로그램 변수의 경우 Dreamweaver에서는 HTML/JavaScript 대화 상자를 표시하여 변수의 이름을 수집합니다. 이 비헤이비어는 나중에 사용할 수 있도록 변수 이름을 저장합니다.

addDynamicSource() 함수가 반환된 후, Dreamweaver에서는 데이터 소스 트리의 내용을 지우고 findDynamicSources() 및 generateDynamicSourceBindings() 함수를 호출하여 데이터 소스 트리를 다시 채웁니다.

반환값
없음

deleteDynamicSource()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 사용자가 트리에서 데이터 소스를 선택하고 마이너스(-) 버튼을 클릭할 때 호출됩니다.

예를 들어 Dreamweaver에서 선택한 항목이 레코드세트이거나 명령이면 deleteDynamicSource() 함수는 dw.serverBehaviorInspector.deleteServerBehavior() 함수를 호출합니다. 요청, 세션 또는 응용 프로그램 변수가 선택된 경우 이 함수는 해당 변수가 삭제되었다는 것을 기억하고 더 이상 표시하지 않습니다. deleteDynamicSource() 함수가 반환된 후, Dreamweaver에서는 데이터 소스 트리의 내용을 지우고 findDynamicSources() 및 generateDynamicSourceBindings() 함수를 호출하여 사용자의 문서에 대한 모든 데이터 소스가 포함된 새 목록을 가져옵니다.

인수
sourceName, bindingName

- **sourceName** 인수는 자식 노드가 연관된 최상위 노드의 이름입니다.
- **bindingName** 인수는 자식 노드의 이름입니다.

반환값
없음

displayHelp()

설명
이 함수를 정의하면 대화 상자의 [확인] 및 [취소] 버튼 아래에 [도움말] 버튼이 표시됩니다. 사용자가 [도움말] 버튼을 클릭하면 이 함수가 호출됩니다.

인수
없음

반환값
없음

예제

```
// The following instance of displayHelp() opens
// a file (in a browser) that explains how to use
// the extension.
function displayHelp(){
    var myHelpFile = dw.getConfigurationPath() +
        '/ExtensionsHelp/superDuperHelp.htm';
    dw.browseDocument(myHelpFile);
}
```

editDynamicSource()

지원 버전
Dreamweaver MX

설명
이 함수는 사용자가 [바인딩] 패널에서 데이터 소스 이름을 두 번 클릭하여 데이터 소스를 편집할 때 호출됩니다. 이 함수를 구현하여 트리에서 사용자가 편집한 내용을 처리할 수 있습니다. 그렇지 않으면 해당 데이터 소스에 맞는 서버 비헤이비어가 자동으로 호출됩니다. Extension 개발자는 이 함수를 사용하여 서버 비헤이비어의 기본 구현을 재정의하고 사용자 정의 핸들러를 제공할 수 있습니다.

인수
sourceName, **bindingName**

- **sourceName** 인수는 자식 노드가 연관된 최상위 노드의 이름입니다.
- **bindingName** 인수는 자식 노드의 이름입니다.

반환값
부울 값을 반환합니다. 이 함수가 편집 내용을 처리했으면 true이고, 그렇지 않으면 false입니다.

findDynamicSources()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 [동적 데이터] 또는 [동적 텍스트] 대화 상자나 [바인딩] 패널에 나타나는 데이터 소스 트리의 최상위 노드를 반환합니다. 각 데이터 소스 파일에는 findDynamicSources() 함수가 구현되어 있습니다. Dreamweaver에서는 트리를 새로 고칠 때 DataSources 폴더의 모든 파일을 읽고 각 파일에서 findDynamicSources() 함수를 호출합니다.

반환값
JavaScript 객체의 배열을 반환합니다. 각 객체에는 최대 다섯 개의 속성이 있을 수 있습니다. 이 속성에 대해서는 다음 목록에서 설명합니다.

- title 속성은 각 부모 노드 아이콘의 오른쪽에 나타나는 레이블 문자열입니다. title 속성은 항상 필수입니다.
- imageFile 속성은 [동적 데이터] 또는 [동적 텍스트] 대화 상자나 [바인딩] 패널의 트리 컨트롤에서 부모 노드를 나타내는 아이콘(GIF 이미지)이 들어 있는 파일의 경로입니다. 이 속성은 필수입니다.

- allowDelete 속성은 선택 사항입니다. 이 속성을 false로 설정하면 사용자가 [바인딩] 패널에서 이 노드를 클릭할 때 마이너스(-) 버튼이 비활성화됩니다. 이 속성을 true로 설정하면 마이너스(-) 버튼이 활성화됩니다. 이 속성을 정의하지 않을 경우 기본값은 true입니다.
- dataSource 속성은 findDynamicSources() 함수가 정의되어 있는 파일의 간단한 이름입니다. 예를 들어 Configuration/DataSources/ASP_Js 폴더의 Session.htm 파일에서 findDynamicSources() 함수는 dataSource 속성을 session.htm으로 설정합니다. 이 속성은 필수입니다.
- name 속성은 데이터 소스와 연관된 서버 비헤이비어가 있는 경우 이 서버 비헤이비어의 이름이 됩니다. 레코드세트 같은 일부 데이터 소스는 서버 비헤이비어와 연관되어 있습니다. 레코드세트를 만들고 rsAuthors라는 이름을 지정하는 경우, name 속성은 rsAuthors와 동일해야 합니다. name 속성은 항상 정의되지만 세션 변수와 같은 데이터 소스와 연관된 서버 비헤이비어가 없는 경우에는 빈 문자열("")이 될 수도 있습니다.

참고: 이러한 속성을 정의하는 JavaScript 클래스는 DataSourceClass.js 파일에 존재합니다. 이 파일은 Configuration/Shared/Common/Scripts 폴더에 저장되어 있습니다.

generateDynamicDataRef()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 자식 노드의 동적 데이터 객체를 생성합니다.

인수
sourceName, **bindingName**

- **sourceName** 인수는 자식 노드와 연관된 최상위 노드의 이름입니다.
- **bindingName** 인수는 동적 데이터 객체를 생성하려는 자식 노드의 이름입니다.

반환값
문자열을 반환하며, 이 문자열은 사용자의 문서에 삽입되기 전에 서식 지정을 위해 formatDynamicDataRef() 함수에 전달될 수 있습니다.

generateDynamicSourceBindings()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 최상위 노드의 자식을 반환합니다.

인수
sourceName

- **sourceName** 인수는 반환하려는 자식 노드의 최상위 노드 이름입니다.

반환값
JavaScript 객체의 배열을 반환합니다. 각 객체에는 최대 네 개의 속성이 있을 수 있습니다. 이 속성에 대해서는 다음 목록에서 설명합니다.

- title 속성은 각 부모 노드 아이콘의 오른쪽에 나타나는 레이블 문자열입니다. 이 속성은 필수입니다.
- allowDelete 속성은 선택 사항입니다. 이 속성을 false 값으로 설정하면 사용자가 [바인딩] 패널에서 이 노드를 클릭할 때 마이너스(-) 버튼이 비활성화됩니다. 이 속성을 true 값으로 설정하면 마이너스(-) 버튼이 활성화됩니다. 이 속성을 정의하지 않을 경우 기본값은 true입니다.
- dataSource 속성은 findDynamicSources() 함수가 정의되어 있는 파일의 간단한 이름입니다. 예를 들어 Configuration/DataSources/ASP_Js 폴더의 Session.htm 파일에서 findDynamicSources() 함수는 dataSource 속성을 session.htm으로 설정합니다. 이 속성은 필수입니다.
- name 속성은 데이터 소스와 연관된 서버 비헤이비어가 있는 경우 이 서버 비헤이비어의 이름이 됩니다. 이 속성은 필수 속성입니다. 레코드세트 같은 일부 데이터 소스는 서버 비헤이비어와 연관되어 있습니다. 레코드세트를 만들고 rsAuthors라는 이름을 지정하는 경우, name 속성은 rsAuthors와 동일해야 합니다. 세션 변수 등의 다른 데이터 소스에는 해당하는 서버 비헤이비어가 없습니다. 이 경우 이름 속성은 빈 문자열("")이어야 합니다.

참고: 이러한 속성을 정의하는 JavaScript 클래스는 DataSourceClass.js 파일에 존재합니다. 이 파일은 Configuration/Shared/Common/Scripts 폴더에 저장되어 있습니다.

inspectDynamicDataRef()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 동적 데이터 객체의 데이터 소스 트리에서 해당 노드를 판별합니다. inspectDynamicDataRef() 함수는 Dreamweaver에서 전달하는 문자열을 받아 generateDynamicDataRef() 함수가 트리의 각 노드에 대해 반환하는 문자열과 비교합니다. 일치하는 문자열이 없으면 inspectDynamicDataRef() 함수는 전달된 문자열과 일치하는 트리의 노드를 표시합니다. 이 함수는 두 개의 요소로 구성된 배열을 사용하여 노드를 표시합니다. 첫 번째 요소는 부모 노드의 이름이고 두 번째 요소는 자식 노드의 이름입니다. 일치하는 노드가 없으면 inspectDynamicDataRef() 함수는 빈 배열을 반환합니다.

inspectDynamicDataRef() 함수의 각 구현에서는 이 함수의 객체 형식과 일치하는 항목만 검색합니다. 예를 들어 inspectDynamicDataRef() 함수의 레코드세트 구현은 전달된 문자열이 트리의 레코드세트 노드와 일치하는 경우에만 일치하는 항목을 검색합니다.

인수
string

- **string** 인수는 동적 데이터 객체입니다.

반환값
일치하는 노드에 대해 두 개의 요소(부모 이름과 자식 이름)로 구성된 배열을 반환합니다. 일치하는 노드가 없으면 null을 반환합니다.

19장: 서버 서식

271페이지의 “데이터 소스”에서는 Adobe Dreamweaver에서 적절한 위치에 서버 표현식을 추가하여 사용자 문서에 동적 데이터를 삽입하는 방법에 대해 설명합니다. 방문자가 웹 서버에서 문서를 요청하면 해당 서버 표현식은 데이터베이스의 값, 요청 변수의 내용 또는 기타 동적인 값으로 변환됩니다. Dreamweaver 서버 서식을 사용하면 이러한 동적인 값이 방문자에게 제공되는 방법의 서식을 지정할 수 있습니다.

데이터 서식 지정 작동 방식

Dreamweaver 사용자는 내장된 서식을 사용하여 데이터에 서식을 지정하거나 내장된 서식 유형을 기반으로 새 서식을 만들거나 사용자 정의 서식 유형을 기반으로 새 서식을 만들 수 있습니다.

사용자는 여러 가지 방법으로 동적 데이터의 서식을 지정할 수 있습니다. HTML 문서에 데이터를 삽입하기 전에 서식을 지정하려면 [서식] 메뉴를 사용합니다. 이 메뉴는 [동적 데이터] 또는 [동적 텍스트] 대화 상자나 [바인딩] 패널에 있습니다. 서식을 만들려면 [서식] 메뉴에서 [서식 목록 편집] 명령을 선택하거나 플러스(+) 메뉴에서 서식 유형을 선택합니다. 플러스(+) 메뉴에는 서식 유형 목록이 포함되어 있습니다. 서식 유형은 [통화], [날짜/시간] 또는 [대/소문자 구분] 같은 기본 서식 범주입니다. 서식 유형은 서식의 범주에 대한 일반적인 매개 변수를 모두 모아서 제공하므로 간단하게 서식을 만들 수 있습니다.

한 예로 통화 서식을 만들어 봅니다. 통화 서식 지정 작업은 숫자를 문자열로 변환, 쉼표 및 소수점 삽입, 달러($) 기호 같은 통화 기호 삽입 등으로 구성됩니다. 통화 서식 데이터 형식은 일반적인 매개 변수를 모두 제공하며 사용자는 각 매개 변수의 값을 지정할 수 있습니다.

서버 서식 API에서는 271페이지의 “데이터 소스”에 설명된 함수가 반환하는 동적 데이터의 서식을 지정하는 API에 대해 설명합니다. 이 두 항목에 설명된 함수는 함께 사용되어 동적 데이터의 서식을 지정합니다.

모든 서식 파일은 Configuration/ServerFormats/**currentServerModel** 폴더에 있습니다. 각 하위 폴더에는 하나의 XML 파일과 여러 개의 HTML 파일이 포함되어 있습니다.

Formats.xml 파일은 [서식] 메뉴에 있는 모든 선택 사항에 대해 설명합니다. Dreamweaver에서는 [서식 목록 편집] 및 [없음] 옵션을 자동으로 추가합니다.

또한 이 폴더에는 현재 설치된 각 서식 유형에 대한 HTML 파일이 하나씩 포함됩니다. 현재 설치된 서식 유형에는 AlphaCase, Currency, DateTime, Math, Number, Percent, Simple 및 Trim이 있습니다.

참고: 통화 서식은 PHP 서버 모델에 사용할 수 없습니다.

Formats.xml 파일

Formats.xml 파일에는 [서식] 메뉴의 각 항목에 대한 format 태그가 하나씩 있습니다. 각 format 태그에는 다음과 같은 필수 속성이 들어 있습니다.

- file=***fileName*** 속성은 "Currency"와 같은 서식 유형에 대한 HTML 파일입니다.
- title=***string*** 속성은 "Currency - default"처럼 [서식] 메뉴에 나타나는 문자열입니다.
- expression=***regexp*** 속성은 이 서식을 사용하는 동적 데이터 객체에 상응하는 일반 표현식입니다. 이 표현식은 동적 데이터 객체에 현재 적용될 서식을 결정합니다. 예를 들어 "Currency - default" 서식의 표현식은 "<%\s*=\s*FormatCurrency\(.*, -1, -2, -2,-2\)\s*%>|<%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>"입니다. expression 속성의 값은 파일의 모든 format 태그에서 고유해야 합니다. 즉, 이 서식의 인스턴스만 해당 표현식과 일치하도록 속성 값이 구체적이어야 합니다.
- visibility=***[hidden | visible]*** 속성은 값이 [서식] 메뉴에 표시되는지 여부를 나타냅니다. visibility 속성의 값이 hidden이면 해당 서식이 [서식] 메뉴에 나타나지 않습니다.

이름이 지정된 임의의 속성이 format 태그에 추가로 포함될 수도 있습니다.

일부 데이터 서식 지정 함수는 JavaScript 객체인 **format** 인수를 필요로 합니다. 이 객체는 Formats.xml 파일의 format 태그에 해당하는 노드입니다. 이 객체에는 해당하는 format 태그의 각 속성에 대한 JavaScript 속성이 있습니다.

다음 예제에서는 "Currency - default" 문자열의 format 태그를 보여 줍니다.

```
<format file="Currency" title="Currency - default" ¬
expression="<%\s*=\s*FormatCurrency\(.*, -1, -2, -2, -2\)\s*%>|¬
    <%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>"
NumDigitsAfterDecimal=-1 IncludeLeadingDigit=-2 ¬
    UseParensForNegativeNumbers=-2 GroupDigits=-2/>
```

이 서식의 서식 유형은 Currency이며, "Currency - default" 문자열이 [서식] 메뉴에 나타납니다. 표현식 <%\s*=\s*FormatCurrency\(.*, -1, -2, -2,-2\)\s*%>|<%\s*=\s*DoCurrency\(.*, -1, -2, -2, -2\)\s*%>는 사용자 문서에서 이 서식이 적용된 경우를 찾습니다.

NumDigitsAfterDecimal, IncludeLeadingDigit, UseParensForNegativeNumbers 및 GroupDigits 매개 변수는 Currency 서식 유형에 사용되며 필수는 아닙니다. 이 매개 변수는 Currency 서식 유형에 대한 [매개 변수] 대화 상자에 나타납니다. [매개 변수] 대화 상자는 사용자가 [서식 목록 편집] 대화 상자의 플러스(+) 메뉴에서 Currency 서식 유형을 선택할 때 나타납니다. 이러한 매개 변수에 지정된 값은 새 서식을 정의합니다.

서식 목록 편집 플러스(+) 메뉴

ServerFormats 폴더의 파일이 [서식 목록 편집] 대화 상자의 플러스(+) 메뉴에 나타나지 않도록 하려면 HTML 파일의 첫 번째 행에 다음 명령문을 추가합니다.

```
<!-- MENU-LOCATION=NONE -->
```

메뉴의 내용을 결정하기 위해 Dreamweaver에서는 먼저 데이터 서식이 포함된 폴더와 동일한 폴더(예: Configuration/ServerFormats/ASP/)에서 ServerFormats.xml 파일을 검색합니다. ServerFormats.xml 파일은 [서식 목록 편집] 대화 상자의 플러스(+) 메뉴에 포함된 내용에 대해 설명합니다. 이 파일에는 메뉴에 나열되는 HTML 파일에 대한 참조가 들어 있습니다.

Dreamweaver에서는 참조된 각 HTML 파일에서 title 태그를 찾습니다. 파일에 title 태그가 있는 경우에는 title 태그의 내용이 메뉴에 나타나고, 없는 경우에는 파일 이름이 메뉴에 사용됩니다.

Dreamweaver에서 DataSources.xml 파일을 모두 검색한 후 또는 해당 파일이 없는 경우, Dreamweaver에서는 폴더의 나머지 부분에서 메뉴에 나타나는 다른 항목을 찾습니다. Dreamweaver에서 메뉴에 없는 파일을 기본 폴더에서 찾은 경우 이 파일을 추가합니다. 아직 메뉴에 없는 파일이 하위 폴더에 포함되어 있으면 Dreamweaver에서는 하위 메뉴를 만들어 해당 파일을 추가합니다.

데이터 서식 지정 함수가 호출되는 경우

데이터 서식 함수는 다음과 같은 경우에 호출됩니다.

- [동적 데이터] 또는 [동적 텍스트] 대화 상자에서 사용자는 데이터 소스 트리에서 특정 노드를 선택하고 [서식] 메뉴에서 특정 서식을 선택합니다. 사용자가 서식을 선택하면 Dreamweaver에서는 generateDynamicDataRef() 함수를 호출하여 generateDynamicDataRef() 함수에서 formatDynamicDataRef() 함수로 그 반환값을 전달합니다. formatDynamicDataRef() 함수에서 반환된 값은 대화 상자의 [코드] 설정에 나타납니다. 사용자가 [확인]을 클릭하면 코드의 문자열이 사용자 문서에 삽입됩니다. 다음으로, Dreamweaver에서는 applyFormat() 함수를 호출하여 함수 선언을 삽입합니다. 자세한 내용은 281페이지의 “generateDynamicDataRef()”를 참조하십시오. 사용자가 [바인딩] 패널에서 작업하는 경우에도 유사한 프로세스가 수행됩니다.
- 사용자가 서식을 변경하거나 동적 데이터 항목을 삭제하면 deleteFormat() 함수가 호출됩니다. deleteFormat() 함수는 문서에서 지원 스크립트를 제거합니다.

- 사용자가 [서식 목록 편집] 대화 상자의 플러스(+) 버튼을 클릭하면 지정된 서버 모델에 대한 모든 서식 유형이 포함된 메뉴가 표시됩니다. 각 서식 유형은 Configuration/ServerFormats/**currentServerModel** 폴더의 파일에 해당됩니다.

  사용자가 플러스(+) 메뉴에서 사용자 지정 매개 변수가 필요한 서식을 선택하면 Dreamweaver에서는 body 태그에서 onload 핸들러를 실행하고 [매개 변수] 대화 상자를 표시하여 해당 서식 유형의 매개 변수를 보여 줍니다. 이 대화 상자에서 사용자가 해당 서식에 사용할 매개 변수를 선택하고 [확인]을 클릭하면 applyFormatDefinition() 함수가 호출됩니다.

  선택된 서식이 [매개 변수] 대화 상자를 표시할 필요가 없으면 Dreamweaver에서는 사용자가 플러스(+) 메뉴에서 서식 유형을 선택할 때 applyFormatDefinition() 함수를 호출합니다.

- 나중에 사용자가 [서식 목록 편집] 대화 상자에서 서식을 선택하고 [편집] 버튼을 클릭하여 서식을 편집하면 Dreamweaver에서는 [매개 변수] 대화 상자를 표시하기 전에 inspectFormatDefinition() 함수를 호출합니다. 따라서 양식 컨트롤은 올바른 값으로 초기화됩니다.

서버 서식 API 함수

서버 서식 API는 다음과 같은 데이터 서식 함수로 구성되어 있습니다.

applyFormat()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 사용자의 문서에 서식 함수 선언을 추가하여 해당 문서를 편집할 수 있습니다. 사용자가 [동적 데이터] 또는 [동적 텍스트] 대화 상자의 [서식] 텍스트 필드나 [바인딩] 패널에서 서식을 선택하면 Dreamweaver에서는 사용자 문서에서 두 가지 사항을 변경합니다. 사용자 문서에 적절한 서식 함수가 없는 경우에는 HTML 태그 앞에 해당 함수를 추가하고 동적 데이터 객체를 변경하여 적절한 서식 함수를 호출합니다.

Dreamweaver에서는 데이터 서식 파일에서 applyFormat() JavaScript 함수를 호출하여 함수 선언을 추가하고, formatDynamicDataRef() 함수를 호출하여 동적 데이터 객체를 변경합니다.

applyFormat() 함수는 사용자 문서의 맨 위에 함수 선언을 추가할 때 DOM을 사용해야 합니다. 예를 들어 사용자가 [통화] < [기본값]을 선택하면 이 함수는 Currency 함수 선언을 추가합니다.

인수
format

- **format** 인수는 적용할 서식을 설명하는 JavaScript 객체입니다. JavaScript 객체는 Formats.xml 파일의 format 태그에 해당하는 노드입니다. 이 객체에는 해당하는 format 태그의 각 속성에 대한 JavaScript 속성이 있습니다.

반환값
없음

applyFormatDefinition()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 [서식 편집] 대화 상자에서 작성된 서식에 대한 변경 사항을 반영합니다.

사용자는 [서식 목록 편집] 대화 상자를 사용하여 서식을 만들거나 편집 또는 삭제할 수 있습니다. 이 함수는 서식에 대한 수정 작업을 적용하기 위해 호출됩니다. 또한 객체에 이름이 지정된 임의의 다른 속성을 설정할 수도 있습니다. 각 속성은 Formats.xml 파일에 있는 format 태그의 속성으로 저장됩니다.

인수
format

- **format** 인수는 JavaScript format 객체에 해당합니다. 이 함수는 JavaScript 객체의 expression 속성이 서식에 대한 일반 표현식이 되도록 설정해야 합니다. 또한 이 함수는 객체의 이름이 지정된 임의의 다른 속성을 설정할 수도 있습니다. 각 속성은 format 태그의 속성으로 저장됩니다.

반환값
함수가 성공적으로 완료되면 서식 객체를 반환합니다. 오류가 발생한 경우 함수는 오류 문자열을 반환합니다. 빈 문자열을 반환하는 경우에는 양식이 닫히지만 새로운 서식이 만들어지지는 않습니다. 즉, [취소] 작업과 같습니다.

deleteFormat()

지원 버전
Dreamweaver UltraDev 1

설명
사용자 문서 맨 위에서 서식 함수 선언을 제거합니다.

사용자가 [동적 데이터] 또는 [동적 텍스트] 대화 상자나 [바인딩] 패널에서 동적 데이터 객체의 서식을 변경하거나 서식이 지정된 동적 데이터 객체를 삭제하면 Dreamweaver에서는 deleteFormat() 함수를 호출하여 문서의 맨 위에서 함수 선언을 제거하고 동적 데이터 객체에서 함수 호출을 제거합니다.

현재 문서의 맨 위에서 함수 선언을 제거하려면 deleteFormat() 함수와 DOM을 함께 사용합니다.

인수
format

format 인수는 제거할 서식을 설명하는 JavaScript 객체입니다. JavaScript 객체는 Formats.xml 파일의 format 태그에 해당하는 노드입니다.

반환값
없음

formatDynamicDataRef()

지원 버전
Dreamweaver UltraDev 1

설명

동적 데이터 객체에 서식 함수 호출을 추가합니다. 사용자가 [동적 데이터] 또는 [동적 텍스트] 대화 상자의 [서식] 텍스트 상자나 [바인딩] 패널에서 서식을 선택하면 Dreamweaver에서는 사용자 문서에서 두 가지 사항을 변경합니다. 사용자 문서에 적절한 서식 함수가 없는 경우에는 HTML 태그 앞에 해당 함수를 추가하고 동적 데이터 객체를 변경하여 적절한 서식 함수를 호출합니다.

Dreamweaver에서는 데이터 서식 파일에서 applyFormat() JavaScript 함수를 호출하여 함수 선언을 추가하고, formatDynamicDataRef() 함수를 호출하여 동적 데이터 객체를 변경합니다.

formatDynamicDataRef() 함수는 사용자가 [동적 데이터] 또는 [동적 텍스트] 대화 상자의 [서식] 텍스트 상자 또는 [바인딩] 패널에서 특정 서식을 선택할 때 호출됩니다. 사용자 문서를 편집할 수는 없습니다.

인수

dynamicDataObject, format

- **dynamicDataObject** 인수는 동적 데이터 객체를 포함하는 문자열입니다.
- **format** 인수는 적용할 서식을 설명하는 JavaScript 객체입니다. JavaScript 객체는 Formats.xml 파일의 format 태그에 해당하는 노드입니다. 이 객체에는 해당하는 format 태그의 각 속성에 대한 JavaScript 속성이 있습니다.

반환값

동적 데이터 객체의 새 값을 반환합니다.

오류가 발생하는 경우 이 함수는 특정 조건에서 경고 메시지를 표시합니다. 함수가 빈 문자열을 반환하면 [서식] 텍스트 필드는 None으로 설정됩니다.

inspectFormatDefinition()

지원 버전

Dreamweaver UltraDev 1

설명

사용자가 [서식 목록 편집] 대화 상자에서 서식을 편집할 때 양식 컨트롤이 초기화됩니다.

인수

format

format 인수는 적용할 서식을 설명하는 JavaScript 객체입니다. JavaScript 객체는 Formats.xml 파일의 format 태그에 해당하는 노드입니다. 이 객체에는 해당하는 format 태그의 각 속성에 대한 JavaScript 속성이 있습니다.

반환값

없음

20장: 구성 요소

Adobe Dreamweaver에서는 가장 일반적인 유형의 구성 요소를 대부분 만들 수 있습니다. 또한 [구성 요소] 패널에 나타나는 구성 요소의 유형을 확장할 수도 있습니다.

구성 요소 기본 사항

프로그래머는 작업을 캡슐화하기 위해 다양한 전략을 사용합니다. **캡슐화**란 가상의 블랙 박스에 존재하는 엔터티를 만드는 것입니다. 캡슐화의 작동 방식을 모르더라도 캡슐화 기능을 사용할 수 있습니다. 하지만 캡슐화 작업을 수행하는 데 필요한 정보와 캡슐화 작업이 완료되었을 때의 결과는 반드시 알고 있어야 합니다. 예를 들어 한 프로그래머가 직원 데이터베이스에서 정보를 가져오는 프로그램을 만들 경우, 다른 프로그래머나 다른 프로그램이 이 프로그램을 사용하여 직원 데이터베이스를 쿼리할 수 있습니다. 따라서 이 프로그램은 재사용이 가능합니다.

캡슐화를 사용하는 잘 짜여진 프로그램은 유지 관리, 향상 및 재사용이 보다 쉽습니다. 사용하는 기술에 따라 캡슐화를 구현하는 방법이 다양하며 이러한 전략을 나타내는 이름도 **함수**, **모듈** 등으로 다양합니다. Dreamweaver에서는 CFC(ColdFusion 구성 요소)와 같이 보다 대중적이고 현대적인 캡슐화 전략에 **구성 요소**라는 용어를 사용합니다. 그러므로 사용자가 Dreamweaver에서 웹 응용 프로그램을 만들 때 [구성 요소] 패널을 이용하면 CFC를 사용할 수 있습니다.

웹 서비스, JavaBeans 또는 CFC와 같은 최신 기술이 제공하는 구성 요소를 살펴보면 이 사실을 확인할 수 있습니다. 대개 구성 요소를 구성하고 있는 파일에는 구성 요소에 대한 정보가 포함되어 있는데, 이 정보를 게시하거나 공유하는 구성 요소 기능을 검사라고 합니다. 따라서 Dreamweaver와 같은 프로그램에서는 구성 요소가 노출하는 함수 목록을 구성 요소에 요청할 수 있습니다. 노출되는 함수는 다른 프로그램에서 로드할 수 있는 함수입니다. 사용되는 기술에 따라 구성 요소는 해당 구성 요소에 대한 다른 정보도 노출할 수 있습니다.

구성 요소 패널 확장

현재 [구성 요소] 패널에 없는 구성 요소 전략을 사용하려면 [구성 요소] 패널의 논리를 확장합니다. 이렇게 하면 새로운 종류의 구성 요소를 이 패널에서 처리할 수 있습니다.

Dreamweaver의 [구성 요소] 패널에 새 구성 요소를 추가하려면 사용자의 환경에서 사용 가능한 구성 요소를 찾습니다. 그런 다음 각 구성 요소에 설명을 요청하거나, 설명이 ASCII 파일을 사용하여 작성된 경우 해당 설명을 파싱합니다.

구성 요소 위치 및 구성 요소 세부 사항을 검색하는 방법은 기술에 따라 달라지며 ASP.NET, JSP/J2EE, ColdFusion 등의 서버 모델에 따라서도 달라집니다. 그러므로 [구성 요소] 패널을 확장하기 위해 작성하는 JavaScript는 추가할 구성 요소 기술에 따라 달라집니다. 여기에 설명된 함수를 사용하면 [구성 요소] 패널에 나타나는 정보를 쉽게 가져올 수 있습니다. 하지만 구성 요소를 찾고 검사하는 데 필요한 논리는 대부분 개발자가 직접 작성해야 합니다. 여기에는 구성 요소의 내부 구조를 쿼리하고 구성 요소의 필드, 메서드 및 속성을 Dreamweaver에서 사용할 수 있도록 하는 작업이 포함됩니다.

마지막으로, ASP.NET, JSP/J2EE 또는 ColdFusion과 같은 서버 모델은 전부는 아니지만 일부 구성 요소 유형을 지원합니다. 예를 들어 ASP.NET은 웹 서비스를 지원하지만 JavaBeans는 지원하지 않습니다. 또한 Adobe ColdFusion은 웹 서비스와 CFC를 지원합니다. [구성 요소] 패널에 새로운 구성 요소 유형을 추가하면 해당 구성 요소 유형은 특정 서버 모델과 연관됩니다. 예를 들어 Dreamweaver 사용자가 ColdFusion 사이트에 대해 작업하는 경우에는 [구성 요소] 패널의 팝업 메뉴에 CF 구성 요소가 나타납니다.

경우에 따라서는 파일을 변경하기 위해 특정 구성 요소 관련 함수를 호출하는 JavaScript 코드를 작성해야 할 수도 있습니다.

구성 요소 패널 사용자 정의

사용자는 Dreamweaver [구성 요소] 패널에서 구성 요소를 로드하여 작업할 수 있습니다. 이 패널에는 사용 가능한 각 서버 모델과 호환되는 사용 가능한 구성 요소 유형이 모두 나열됩니다. 예를 들어 CFC는 Adobe ColdFusion 페이지에서만 동작할 수 있으므로 [구성 요소] 패널에서 Adobe ColdFusion 서버 모델에만 나타납니다.

확장성은 패널에 새 구성 요소 유형을 추가할 수 있는 기능입니다. [구성 요소] 패널에 새 구성 요소 유형을 추가하는 일반적인 단계는 다음과 같습니다.

1 적절한 서버 모델의 사용 가능한 구성 요소 유형 목록에 구성 요소를 추가합니다.

2 이 단계가 구현되는 Extension에 따라 [구성 요소] 패널이나 대화 상자에서 구성 요소를 설정하기 위한 지침을 추가합니다. 해당 지침은 알아보기 쉽게 번호가 매겨진 단계로 나타나며 설정 단계라고도 합니다. 사용자가 완료한 단계 옆에 체크 표시가 나타나는지 확인합니다.

3 사용자의 컴퓨터나 현재 사이트에만 존재하는 구성 요소 유형의 구성 요소를 나열합니다.

4 사용자가 [구성 요소] 패널에서 플러스(+) 버튼을 클릭하면 구성 요소를 만듭니다.

또한 사용자가 구성 요소를 편집하고 삭제할 수 있도록 할 수도 있습니다.

구성 요소 패널 파일 사용자 정의

Configuration/Components 폴더에는 구현된 각 서버 모델의 하위 폴더가 있습니다. 구성 요소 파일은 Configuration/Components/**server-model/ComponentType** 폴더에 저장됩니다. 다른 서버 모델 및 지원 서버 Extension을 추가할 수 있습니다. 자세한 내용은 300페이지의 "서버 모델" 및 230페이지의 "서버 비헤이비어"를 참조하십시오.

구성 요소 패널에서 동작할 수 있는 사용자 정의 구성 요소 만들기

1 지원 JavaScript 및 이미지 파일의 위치를 알려 주는 HTML 파일을 만듭니다.

2 JavaScript를 작성하여 구성 요소를 활성화합니다.

3 [구성 요소] 패널에서 해당 구성 요소를 나타낼 GIF 이미지 파일을 만들거나 기존 파일로 지정합니다.

구성 요소 유형을 트리 컨트롤 뷰에 표시하려면 연관된 옵션 파일을 만들고 트리 컨트롤을 채웁니다.

구성 요소 유형은 세 가지 수준, 즉 개별 웹 페이지, 일련의 웹 페이지 또는 사이트 전체에서 동작하도록 설정할 수 있습니다. JavaScript 코드에는 구성 요소가 세션 간에 저장되고 새로운 세션이 시작될 때 다시 로드되도록 구성 요소를 보존하기 위한 논리가 들어 있어야 합니다.

새로운 LDAP(Lightweight Directory Access Protocol) 서비스 구성 요소 추가

1 Dreamweaver의 [구성 요소] 패널에 새 구성 요소 유형을 표시하려면 응용 프로그램 폴더 내의 Configuration/Components/Common/WebServices에 있는 파일과 같은 기존 구성 요소 유형 파일을 모델로 사용하여 다음 표에서와 같이 모든 필수 파일과 원하는 선택적 파일을 만듭니다.

파일 이름	설명	필수/선택
*.htm	그 밖의 지원 JavaScript 및 GIF 파일을 식별하는 Extension 파일	필수
*.js	구성 요소 API 콜백을 구현하는 Extension 파일	필수

파일 이름	설명	필수/선택
*.gif	[구성 요소] 팝업 메뉴에 나타나는 이미지	필수
*Menus.xml	[구성 요소] 패널 구조를 구성하는 메타데이터의 저장소. 일반적인 WebServices 구성 요소는 이 파일을 사용하지 않지만 응용 프로그램 폴더 내의 Components/ColdFusion/WebServices에 있는 WebServicesMenus.xml 파일을 예제로 참조할 수 있습니다.	선택 사항
*.gif	활성화하거나 비활성화할 수 있는 툴바 이미지(예: ToolBarImageUp.gif ToolBarImageDown.gif ToolBarImageDisabled.gif) 또는 트리 노드 이미지	선택 사항

참고: 각 파일과 해당 구성 요소를 쉽게 식별할 수 있도록 한 구성 요소에 해당하는 모든 파일에는 같은 접두어를 사용하는 것이 좋습니다.

2 새 서버 구성 요소를 구현하는 JavaScript 코드를 작성합니다.

Extension 파일(HTM)의 SCRIPT 태그에서는 JavaScript 코드의 위치를 정의합니다. 이러한 JavaScript 파일은 Shared 폴더, Extension 파일과 동일한 폴더, 또는 여러 서버 모델에 적용되는 코드를 저장하는 Common 폴더에 저장될 수 있습니다.

예를 들어 Configuration/Components/Common/WebServices/ 폴더의 WebServices.htm 파일에는 다음과 같은 행이 포함되어 있습니다.

```
<SCRIPT SRC="../../Common/WebServices/WebServicesCommon.js"></SCRIPT>
```

사용 가능한 구성 요소 API 함수에 대한 자세한 내용은 291페이지의 “구성 요소 패널 API 함수”를 참조하십시오.

새 서비스를 추가하는 경우에는 Extension 작성 시 정보를 쉽게 사용할 수 있도록 [구성 요소] 패널을 사용하여 메타 정보를 찾을 수 있습니다. Dreamweaver에서는 추가된 구성 요소를 찾고 구성 요소 트리에 노드를 표시할 수 있습니다. [구성 요소] 패널에서는 [코드] 뷰를 통해 드래그 앤 드롭 및 키보드 지원 기능을 제공합니다.

트리 컨트롤의 속성

[구성 요소] 패널의 트리 컨트롤이 [구성 요소] 패널 내의 적절한 위치에 나타나도록 ComponentRec 속성을 사용하여 이 컨트롤을 채울 수 있습니다. 트리 컨트롤의 모든 노드에는 다음과 같은 속성이 있어야 합니다.

속성 이름	설명	필수/선택
name	트리 노드 항목의 이름	필수
image	트리 노드 항목의 아이콘. 아이콘을 지정하지 않으면 기본 아이콘이 사용됩니다.	선택 사항
hasChildren	자식을 로드하여 트리 컨트롤에서 플러스(+) 및 마이너스(-) 버튼의 클릭에 응답합니다. 미리 채워지지 않은 트리로 작업할 수 있습니다.	필수
toolTipText	트리 노드 항목의 도구 설명 텍스트	선택 사항
isCodeViewDraggable	항목을 [코드] 뷰에 드래그하여 드롭할 수 있는지 확인합니다.	선택 사항
isDesignViewDraggable	항목을 [디자인] 뷰에 드래그하여 드롭할 수 있는지 확인합니다.	선택 사항

예를 들어 다음 WebServicesClass 노드에는 웹 메서드가 자식 항목으로 포함되어 있습니다.

```
this.name = "TrafficLocatorWebService";
this.image = "Components/Common/WebServices/WebServices.gif";
this.hasChildren = true;
this.toolTipText = "TrafficLocatorWebService";
this.isCodeViewDraggable = true;
// the following allows of enabling/disabling of the button that appears
// above the Component Tree
this.allowDelete = true;
this.isDesignViewDraggable = false;
```

구성 요소 패널 API 함수

이 단원에서는 [구성 요소] 패널을 채우는 API 함수에 대해 설명합니다.

getComponentChildren()

지원 버전
Dreamweaver MX

설명
이 함수는 활성 상태인 부모 ComponentRec 객체의 자식 ComponentRec 객체 목록을 반환합니다. 루트 수준의 트리 항목을 로드하려는 경우 이 함수는 영구 저장소에서 메타데이터를 읽어야 합니다.

인수
{parentComponentRec}

parentComponentRec 인수는 부모의 componentRec 객체입니다. 이 인수를 생략하면 루트 노드의 ComponentRec 객체 목록이 사용됩니다.

반환값
ComponentRec 객체의 배열을 반환합니다.

예제
자세한 내용은 Configuration/Components/Common/WebServices 폴더의 WebServices.js 파일에서 function getComponentChildren(componentRec)을 참조하십시오.

getContextMenuId()

지원 버전
Dreamweaver MX

설명
구성 요소 유형의 컨텍스트 메뉴 ID를 반환합니다. 모든 구성 요소 유형에는 컨텍스트 메뉴가 연관되어 있을 수 있습니다. 팝업 메뉴인 컨텍스트 메뉴는 **ComponentName**Menus.xml 파일에 정의되어 있으며 menu.xml 파일과 동일하게 동작합니다. 메뉴 문자열은 정적이거나 동적일 수 있으며, 키보드 단축키가 지원됩니다.

인수
없음

반환값
컨텍스트 메뉴 ID를 정의하는 문자열을 반환합니다.

예제
다음 예제에서는 Adobe ColdFusion 서버 모델과 연결된 CFC에 대한 [구성 요소] 패널의 [옵션] 메뉴를 설정하고 이 메뉴의 키보드 단축키도 정의합니다.

```
function getContextMenuId()
{
    return "DWCFCsContext";
}
```

이 메뉴의 DWWebServicesContext는 다음과 같이 Configuration/Components/ColdFusion/CFCs/CFCsMenus.xml로 파일에 정의됩니다.

```
<menubar xmlns:MMString="http://www.macromedia.com/schemes/dat/string/" name="" id="DWCFCsContext">
    <menu MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs/menu/name"
id="DWContext_CFCs">
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_createNewCFC/menuitem/name"
domRequired="false" enabled="true" command="createCFC()" id="DWContext_CFCs_createNewCFC" />
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_editCode/menuitem/name"
domRequired="false" enabled="canGetSelectedCFC()" command="editCFC();" id="DWContext_CFCs_editCode" />
<separator/>
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_getDetails/menuitem/name"
domRequired="false" enabled="canGetDetails()" command="getDetails()" id="DWContext_CFCs_getDetails" />
    <menuitem
MMString:name="Components/ColdFusion/CFCs/CFCsMenus_xml/DWContext_CFCs_getDescription/menuitem/name"
domRequired="false" enabled="canGetSelectedCFC()" command="getDescription()"
id="DWContext_CFCs_getDescription" />
    <separator/>
    <menuitem
MMString:name="Components/ASP_NET_Csharp/Connections/ConnectionsMenus_xml/DWShortcuts_ServerComponent_Ins
ert/menuitem/name" domRequired="false" enabled="insertCFCEnabled();" command="clickedInsertCFC();"
id="DWShortcuts_ServerComponent_Insert" />
    </menu>
</menubar>
```

getCodeViewDropCode()

지원 버전
Dreamweaver MX

설명
이 함수는 [구성 요소] 패널에서 [코드] 뷰로 드래그하거나 잘라내거나 복사한 코드를 가져옵니다.

인수
componentRec

- ***componentRec*** 인수는 객체입니다.

반환값
해당 구성 요소의 코드가 포함된 문자열을 반환합니다.

예제

다음 예제에서는 Adobe ColdFusion 구성 요소(CFC)의 코드를 식별합니다.

```
  function getCodeVi e wDropCode(compone nt  Rec )
{
    var codeToDrop="";
    if (componentRec)
     {
        if  (componentRec.objectType == "Connection")
        {
            v ar  connPart = new Participant("d at asource_tag");
            var paramObj = new Obje ct( );
            paramObj.datasource = componentRec.name;
            codeToDrop =  con nPart.getInsertString(paramObj, "aboveHTML");}
        el se if ((componentRec.objectType == "Column")| |
                 (componentRec.objectType == "Paramete r "))
        {
            c  odeToDrop = componentRec.dropcode;
        }
        e ls e{
            codeToDrop =  co mponentRec.name;
            }
        }
        return codeToDrop;
}
```

getSetupSteps()

지원 버전

Dreamweaver MX

설명

Dreamweaver에서는 setupStepsCompleted() 함수가 0 또는 양의 정수를 반환할 경우 이 함수가 호출됩니다. 이 함수는 모달 대화 상자를 사용하는 Extension과 서버 구성 요소를 사용하는 Extension을 사용하여 구현할 수 있는 서버측 설정 지침을 제어합니다.

이 함수는 Extension 유형에 따라 Dreamweaver의 [설정 단계] 대화 상자 또는 [구성 요소] 패널에 표시될 문자열의 배열을 반환합니다.

인수

없음

반환값

n+1개의 문자열이 들어 있는 배열을 반환합니다. 여기에서 **n**은 다음 목록에 설명된 것과 같은 단계 수입니다.

- 설정 단계 목록 위에 나타나는 제목
- 단계마다 텍스트 지침이 있으며 이 텍스트 지침에는 li 태그 내에서 유효한 HTML 마크업이 포함될 수 있습니다.

  다음 양식을 사용하여 단계 목록에 링크(a 태그)를 포함할 수 있습니다.

  ```
  <a href="#" onMouseDown="handler">Blue Underlined Text</a>
  ```

  "**handler**" 값은 다음 문자열 중 하나나 "dw.browseDocument('http://www.adobe.com')"와 같은 JavaScript 표현식으로 대체할 수 있습니다.
- "Event:SetCurSite" 핸들러는 현재 사이트를 설정할 대화 상자를 엽니다.

- "Event:CreateSite" 핸들러는 새 사이트를 만들 대화 상자를 엽니다.
- "Event:SetDocType" 핸들러는 사용자 문서의 문서 형식을 변경할 대화 상자를 엽니다.
- "Event:CreateConnection" 핸들러는 새 데이터베이스 연결을 만들 대화 상자를 엽니다.
- "Event:SetRDSPassword" 핸들러는 RDS(Remote Development Service) 사용자 이름 및 암호(ColdFusion만 해당)를 설정할 대화 상자를 엽니다.
- "Event:CreateCFDataSource" 핸들러는 브라우저에서 Adobe ColdFusion 관리자를 엽니다.

예제

다음 예제에서는 Adobe ColdFusion 구성 요소에 대한 4단계를 설정하고 사용자가 RDS 사용자 이름 및 암호를 입력할 수 있도록 4번째 단계에서 하이퍼텍스트 링크를 제공합니다.

```
function getSetupSteps()
{
    var doSDK = false;
    dom = dw.getDocumentDOM();
    if (dom && dom.serverModel)
    {
        var aServerModelName = dom.serverModel.getDisplayName();
    }
    else
    {
        var aServerModelName = site.getServerDisplayNameForSite();
    }
    if (aServerModelName.length)
    {
        if(aServerModelName != "ColdFusion")
        {
            if(needsSDKInstalled != null)
            {
                doSDK = needsSDKInstalled();
            }
        }
    }

    var someSteps = new Array();
    someSteps.push(MM.MSG_WebService_InstructionsTitle);
    someSteps.push(MM.MSG_Dynamic_InstructionsStep1);
    someSteps.push(MM.MSG_Dynamic_InstructionsStep2);
    if(doSDK == true)
    {
        someSteps.push(MM.MSG_WebService_InstructionsStep3);
    }
    someSteps.push(MM.MSG_WebService_InstructionsStep4);

    return someSteps;
}
```

setupStepsCompleted()

지원 버전

Dreamweaver MX

설명

Dreamweaver에서 이 함수는 [구성 요소] 탭이 나타나기 전에 호출됩니다. 그런 다음 setupStepsCompleted() 함수가 0 또는 양의 정수를 반환하면 getSetupSteps() 함수가 호출됩니다.

인수
없음

반환값
다음 목록에 설명된 것과 같이 사용자가 완료한 설정 단계의 수를 나타내는 정수를 반환합니다.

- 0 또는 양의 정수는 완료된 단계의 수를 나타냅니다.
- -1 값은 필요한 모든 단계가 완료되었음을 나타내므로 지침 목록이 나타나지 않습니다.

handleDesignViewDrop()

지원 버전
Dreamweaver MX

설명
사용자가 [데이터베이스] 패널의 테이블 또는 뷰나 [구성 요소] 패널의 구성 요소를 [디자인] 뷰로 드래그할 때 드롭 작업을 처리합니다.

인수
componentRec

- **componentRec** 인수는 다음 속성이 포함된 객체입니다.
- name 속성은 트리 노드 항목의 이름입니다.
- image 속성은 트리 노드 항목에 대한 선택적 아이콘입니다. 이 속성을 생략하면 기본 아이콘이 사용됩니다.
- hasChildren 속성은 트리 노드 항목이 확장 가능한지를 나타내는 부울 값입니다. true이면 Dreamweaver MX에서 해당 트리 노드 항목에 플러스(+) 및 마이너스(-) 버튼이 표시되고, false이면 해당 항목은 확장할 수 없습니다.
- toolTipText 속성은 트리 노드 항목에 대한 선택적 도구 설명 텍스트입니다.
- isCodeViewDraggable 속성은 해당 트리 노드 항목을 [코드] 뷰로 드래그하여 드롭할 수 있는지 여부를 나타내는 부울 값입니다.
- isDesignViewDraggable 속성은 해당 트리 노드 항목을 [디자인] 뷰로 드래그하여 드롭할 수 있는지 여부를 나타내는 부울 값입니다.

반환값
드롭 작업이 성공적으로 완료되었는지 여부를 나타내는 부울 값을 반환합니다. 성공하면 true를 반환하고, 그렇지 않으면 false를 반환합니다.

예제
다음 예제에서는 구성 요소가 테이블인지 또는 뷰인지 결정한 다음, 적절한 bHandled 값을 반환합니다.

```
function handleDesignViewDrop(componentRec)
{
    var bHandled = false;
    if (componentRec)
    {
        if ((componentRec.objectType == "Table")||
            (componentRec.objectType == "View"))
        {
            alert("popup Recordset Server Behavior");
            bHandled = true;
        }
    }
    return bHandled;
}
```

handleDoubleClick()

지원 버전
Dreamweaver MX

설명
사용자가 트리의 노드를 두 번 클릭하면 편집을 허용하도록 이벤트 핸들러가 호출됩니다. 이 함수는 선택 사항입니다. 이 함수는 이벤트 핸들러가 정의되어 있지 않음을 나타내는 false 값을 반환할 수 있습니다. 이러한 경우 트리 노드를 두 번 클릭하면 기본 비헤이비어가 실행되어 해당 트리 노드가 확장되거나 축소됩니다.

인수
componentRec

- **componentRec** 인수는 다음 속성이 포함된 객체입니다.
- name 속성은 트리 노드 항목의 이름입니다.
- image 속성은 트리 노드 항목에 대한 선택적 아이콘입니다. 이 아이콘을 생략하면 기본 아이콘이 사용됩니다.
- hasChildren 속성은 트리 노드 항목이 확장 가능한지를 나타내는 부울 값입니다. true이면 Dreamweaver에서 해당 트리 노드 항목에 플러스(+) 및 마이너스(-) 버튼이 표시되고, false이면 해당 항목은 확장할 수 없습니다.
- toolTipText 속성은 트리 노드 항목에 대한 선택적 도구 설명 텍스트입니다.
- isCodeViewDraggable 속성은 해당 트리 노드 항목을 [코드] 뷰로 드래그하여 드롭할 수 있는지 여부를 나타내는 부울 값입니다.
- isDesignViewDraggable 속성은 해당 트리 노드 항목을 [디자인] 뷰로 드래그하여 드롭할 수 있는지 여부를 나타내는 부울 값입니다.

반환값
없음

예제
다음 예제의 Extension은 사용자가 트리 노드 항목을 두 번 클릭할 때 이를 처리할 수 있습니다. false 값이 반환될 경우 기본 비헤이비어는 노드를 확장하거나 축소하는 것입니다.

```
function handleDoubleClick(componentRec)
{
    var selectedObj = dw.serverComponentsPalette.getSelectedNode();
    if(dwscripts.IS_WIN)
    {
        if (selectedObj && selectedObj.wsRec && selectedObj.wsRec[ProxyGeneratorNamePropName])
        {
            if (selectedObj.objectType == "Root")
            {
                editWebService();
                return true;
            }
            else if (selectedObj.objectType == "MissingProxyGen")
            {
                displayMissingProxyGenMessage(componentRec);
                editWebService();
                return true;
            }
        }
    }
    return false;
}
```

toolbarControls()

지원 버전
Dreamweaver MX

설명
모든 구성 요소 유형은 왼쪽에서 오른쪽 방향으로 툴바 아이콘을 나타내는 toolBarButtonRec 객체 목록을 반환합니다. 각 toolBarButtonRec 객체에는 다음 속성이 포함되어 있습니다.

속성 이름	설명
image	이미지 파일의 경로입니다.
disabledImage	선택 사항으로, 툴바 버튼의 비활성화된 이미지를 검색하는 경로입니다.
pressedImage	선택 사항으로, 툴바 버튼의 누른 이미지를 검색하는 경로입니다.
toolTipText	툴바 버튼의 도구 설명입니다.
toolStyle	왼쪽/오른쪽
enabled	부울 값(true 또는 false)을 반환하는 JavaScript 코드입니다. 다음 조건을 만족할 경우 활성자가 호출됩니다. • dreamweaver.serverComponents.refresh() 함수가 호출되는 경우 • 트리의 선택 영역이 변경되는 경우 • 서버 모델이 변경되는 경우
command	실행할 JavaScript 코드입니다. 명령 핸들러는 dreamweaver. serverComponents.refresh() 함수를 사용하여 새로 고침을 수행할 수 있습니다.
menuId	팝업 메뉴 버튼을 클릭할 때 해당 버튼에 대해 표시되는 고유한 메뉴 ID입니다. 현재 이 ID가 있으면 이 ID가 명령 핸들러보다 우선합니다. 즉, 이 버튼은 명령과 연관된 버튼이거나 명령과 연관된 팝업 메뉴가 있는 버튼입니다. 하지만, 동시에 두 역할을 다 하지는 못합니다.

인수
없음

반환값
왼쪽에서 오른쪽으로 나타나는 툴바 버튼의 배열을 반환합니다.

예제
다음 예제에서는 툴바 버튼에 속성을 지정합니다.

```
function toolbarControls()
{
    var toolBarBtnArray = new Array();
    dom = dw.getDocumentDOM();
    var plusButton = new ToolbarControlRec();
    var aServerModelName = null;
    if (dom && dom.serverModel)
    {
        aServerModelName = dom.serverModel.getDisplayName();
    }
    else
    {
        //look in the site for potential server model
        aServerModelName = site.getServerDisplayNameForSite();
    }
    if (aServerModelName.length)
    {
        if(aServerModelName == "ColdFusion")
        {
            plusButton.image           = PLUS_BUTTON_UP;
            plusButton.pressedImage    = PLUS_BUTTON_DOWN;
            plusButton.disabledImage   = PLUS_BUTTON_UP;
            plusButton.toolStyle       = "left";
            plusButton.toolTipText    = MM.MSG_WebServicesAddToolTipText;
            plusButton.enabled        = "dwscripts.IS_WIN";
            plusButton.command        = "invokeWebService()";
        }
        else
        {
            plusButton.image          = PLUSDROPBUTTONUP;
            plusButton.pressedImage   = PLUSDROPBUTTONDOWN;
            plusButton.disabledImage  = PLUSDROPBUTTONUP;
            plusButton.toolStyle      = "left";
            plusButton.toolTipText    = MM.MSG_WebServicesAddToolTipText;
            plusButton.enabled        = "dwscripts.IS_WIN";
            plusButton.menuId         = "DWWebServicesChoosersContext";
        }
        toolBarBtnArray.push(plusButton);

        var minusButton            = new ToolbarControlRec();
        minusButton.image          = MINUSBUTTONUP;
        minusButton.pressedImage   = MINUSBUTTONDOWN;
        minusButton.disabledImage  = MINUSBUTTONDISABLED;
        minusButton.toolStyle      = "left";
        minusButton.toolTipText    = MM.MSG_WebServicesDeleteToolTipText;
        minusButton.command        = "clickedDelete()";
        minusButton.enabled  = "(dw.serverComponentsPalette.getSelectedNode() != null &&
            dw.serverComponentsPalette.getSelectedNode() &&
            ((dw.serverComponentsPalette.getSelectedNode().objectType=='Root') ||
            (dw.serverComponentsPalette.getSelectedNode().objectType == 'Error') ||
            (dw.serverComponentsPalette.getSelectedNode().objectType ==
```

```
            'MissingProxyGen')))";
        toolBarBtnArray.push(minusButton);

        if(aServerModelName != null && aServerModelName.indexOf(".NET") >= 0)
        {
            var deployWServiceButton           = new ToolbarControlRec();
            deployWServiceButton.image         = DEPLOYSUPPORTBUTTONUP;
            deployWServiceButton.pressedImage  = DEPLOYSUPPORTBUTTONDOWN;
            deployWServiceButton.disabledImage = DEPLOYSUPPORTBUTTONUP;
            deployWServiceButton.toolStyle     = "right";
            deployWServiceButton.toolTipText   = MM.MSG_WebServicesDeployToolTipText;
            deployWServiceButton.command  = "site.showTestingServerBinDeployDialog()";
            deployWServiceButton.enabled  = true;
            toolBarBtnArray.push(deployWServiceButton);
        }
        //add the rebuild proxy button for windows only.
        //bug 45552:
        if(navigator.platform.charAt(0) !="M")
        {
            var proxyButton            = new ToolbarControlRec();
            proxyButton.image          = PROXYBUTTONUP;
            proxyButton.pressedImage   = PROXYBUTTONDOWN;
            proxyButton.disabledImage  = PROXYBUTTONDISABLED;
            proxyButton.toolStyle      = "right";
            proxyButton.toolTipText    = MM.MSG_WebServicesRegenToolTipText;
            proxyButton.command        = "reGenerateProxy()";
            proxyButton.enabled        = "enableRegenerateProxyButton()";
            toolBarBtnArray.push(proxyButton);
        }
    }
    return toolBarBtnArray;
}
```

21장: 서버 모델

서버 모델은 서버에서 스크립트를 실행하는 기술입니다. 사용자는 새로운 사이트를 정의할 때 사이트 수준 또는 개별 문서 수준에서 사용할 서버 모델을 지정할 수 있습니다. 이 서버 모델은 사용자가 문서에 추가하는 동적 요소를 처리합니다.

서버 모델 구성 파일은 Configuration/ServerModels 폴더에 저장됩니다. 이 폴더 내에서 각 서버 모델은 서버 모델에 필요한 일련의 함수를 구현하는 고유한 HTML 파일을 가집니다.

서버 모델 사용자 정의

서버 모델 API에서 사용할 수 있는 함수를 사용하여 서버 모델의 일부 기능을 사용자 정의할 수 있습니다.

Adobe Dreamweaver에서는 새로운 사용자가 Dreamweaver를 처음 시작할 때 서버 모델을 지정하라는 메시지가 표시됩니다. 사용자가 서버 모델을 지정하지 않은 경우에는 필요한 단계를 완료하라는 메시지를 표시하는 동적 대화 상자를 만들 수 있습니다. 사용자가 서버 객체를 삽입하려고 할 때 이 대화 상자가 나타납니다. 이 대화 상자 만들기에 대한 자세한 내용은 293페이지의 “getSetupSteps()” 및 294페이지의 “setupStepsCompleted()” 함수를 참조하십시오.

특별한 서버 모델을 만들어야 하는 경우도 있습니다. Adobe에서는 Dreamweaver와 함께 제공되는 서버 모델을 편집하는 대신 새 서버 모델을 만들 것을 권장합니다. 사용하는 서버 모델에서 지원되는 새 문서 형식을 만드는 방법에 대한 자세한 내용은 12페이지의 “Dreamweaver의 확장 가능한 문서 형식”을 참조하십시오.

새 서버 모델을 만들 때 사용하는 서버 모델 파일에 canRecognizeDocument() 함수의 구현을 포함해야 합니다. 이 함수는 특정 파일 확장명을 사용하는 서버 모델이 많은 경우, 해당 파일 확장명을 처리하기 위해 서버 모델에 제공하는 우선 순위 레벨을 Dreamweaver에 알려 줍니다.

서버 모델 API 함수

이 단원에서는 Dreamweaver의 서버 모델을 구성하는 함수에 대해 설명합니다.

canRecognizeDocument()

지원 버전
Dreamweaver MX

설명
문서를 열 때 두 개 이상의 서버 모델이 특정 파일 확장명을 사용하면 Dreamweaver에서는 이 확장명과 연결된 각 서버 모델에 대해 이 함수를 호출하여 이 문서가 어떤 서버 모델의 문서인지 식별하도록 합니다. 하나 이상의 서버 모델이 같은 파일 확장명을 사용할 경우 가장 큰 정수를 반환하는 서버 모델에 우선 순위가 부여됩니다.

참고: Dreamweaver에 정의된 모든 서버 모델은 1의 값을 반환하므로 타사 서버 모델이 이 파일 확장명에 대한 연결을 무시할 수 있습니다.

인수
dom

dom 인수는 dreamweaver.getDocumentDOM() 함수에 의해 반환되는 Adobe 문서 객체입니다.

반환값
파일 확장명에 대해 서버 모델에 지정된 우선 순위를 나타내는 정수를 반환합니다. 이 함수는 서버 모델이 파일 확장명을 사용하지 않으면 -1을 반환하고 그렇지 않으면 0보다 큰 값을 반환합니다.

예제
다음 예제에서 사용자가 현재 서버 모델에 대한 JavaScript 문서를 열면 샘플 코드는 값 2를 반환합니다. 이 값은 현재 서버 모델이 Dreamweaver 기본 서버 모델보다 우선하도록 지정합니다.

```
var retVal = -1;
var langRE = /@\s*language\s*=\s*(\"|\')?javascript(\"|\')?/i;
// Search for the string language="javascript"
var oHTML = dom.documentElement.outerHTML;
if (oHTML.search(langRE) > -1)
    retVal = 2;
return retVal;
```

getFileExtensions()

지원 버전
Dreamweaver UltraDev 1. Dreamweaver MX에서 사용되지 않음

설명
서버 모델이 작업할 수 있는 문서 파일 확장명을 반환합니다. 예를 들어 ASP 서버 모델은 .asp 및 .htm 파일 확장명을 지원합니다. 이 함수는 문자열 배열을 반환하고, Dreamweaver에서는 이 문자열을 사용하여 [사이트 정의] 대화 상자의 [App Server] 범주에 있는 [Default Page Extension] 목록을 채웁니다.

참고: [Default Page Extension] 목록은 Dreamweaver 4 이하 버전 버전에만 있습니다. Dreamweaver MX 이상 버전의 경우, [사이트 정의] 대화 상자에 파일 확장명 설정이 나열되지 않습니다. 대신 Dreamweaver에서는 Extensions.txt 파일을 읽고 mmDocumentTypes.xml 파일에서 요소를 파싱합니다. 이러한 두 파일 및 documenttype 요소에 대한 자세한 내용은 12페이지의 “Dreamweaver의 확장 가능한 문서 형식”을 참조하십시오.

인수
없음

반환값
허용된 파일 확장명을 나타내는 문자열 배열을 반환합니다.

getLanguageSignatures()

지원 버전
Dreamweaver MX

설명
이 함수는 스크립팅 언어가 사용하는 메서드 및 배열 서명을 설명하는 객체를 반환합니다. getLanguageSignatures() 함수는 다음 요소에 대해 특정 언어 전용 매핑에 일반 서명 매핑을 매핑하는 데 유용합니다.

- 함수
- 생성자
- 드롭 코드(반환값)

- 배열
- 예외
- 원시 데이터 형식에 대한 데이터 형식 매핑

getLanguageSignatures() 함수는 이러한 서명 선언의 맵을 반환합니다. Extension 개발자는 이 맵을 사용하여 사용자가 드래그 앤 드롭할 때 페이지에 해당되는 서버 모델에 따라 Dreamweaver가 페이지에 드롭하는 특정 언어 전용 코드 블록을 생성할 수 있습니다. Web Services 메서드가 그 예입니다.

이 함수를 작성하는 방법의 예는 JSP 및 ASP.NET 서버 모델의 HTML 구현 파일을 참조하십시오. 서버 모델 구현 파일은 Configuration/ServerModels 폴더에 있습니다.

인수
없음

반환값
스크립팅 언어 서명을 정의하는 객체를 반환합니다. 이 객체는 일반 서명을 특정 언어 서명에 매핑합니다.

getServerExtension()

지원 버전
Dreamweaver UltraDev 4, Dreamweaver MX에서 사용되지 않음

설명
이 함수는 현재 서버 모델을 사용하는 파일의 기본 파일 확장명을 반환합니다. serverModel 객체는 현재 선택된 사용자 문서가 없는 경우, 현재 선택된 사이트의 서버 모델로 설정됩니다.

인수
없음

반환값
지원되는 파일 확장명을 나타내는 문자열을 반환합니다.

getServerInfo()

지원 버전
Dreamweaver MX

설명
이 함수는 JavaScript 코드 내에서 액세스 가능한 JavaScript 객체를 반환합니다. dom.serverModel.getServerInfo() JavaScript 함수를 호출하여 이 객체를 조회할 수 있습니다. 또한 serverName, serverLanguage 및 serverVersion은 다음과 같은 JavaScript 함수를 통해 액세스할 수 있는 특별한 속성입니다.

```
dom.serverModel.getServerName()
dom.serverModel.getServerLanguage()
dom.serverModel.getServerVersion()
```

인수
없음

반환값
서버 모델의 속성이 들어 있는 객체를 반환합니다.

예제
```
var obj = new Object();
obj.serverName = "ASP";
obj.serverLanguage = "JavaScript";
obj.serverVersion = "2.0";
...
return obj;
```

getServerLanguages()

지원 버전
Dreamweaver UltraDev 1. Dreamweaver MX에서 사용되지 않음

설명
이 함수는 특정 서버 모델에서 지원되는 스크립팅 언어를 문자열 배열로 반환합니다. Dreamweaver에서는 이 문자열을 사용하여 [사이트 정의] 대화 상자의 [App Server] 범주에 있는 [Default Scripting Language] 목록을 채웁니다.

참고: [Default Scripting Language] 목록은 Dreamweaver 4 이하 버전에만 있습니다. Dreamweaver MX 이상 버전의 경우 지원되는 스크립팅 언어가 [사이트 정의] 대화 상자에 표시되지도 않고 getServerLanguages() 함수가 사용되지도 않습니다. Dreamweaver에서 각 서버 모델은 서버 언어를 하나만 가지고 있으므로 Dreamweaver에서는 이 함수를 사용하지 않습니다.

이전 버전의 Dreamweaver에서는 서버 모델이 여러 스크립팅 언어를 지원했습니다. 예를 들어 ASP 서버 모델은 JavaScript와 VBScript를 지원합니다.

ServerFormats 폴더의 파일을 특정 스크립팅 언어에만 적용하려면 다음 명령문을 HTML 파일의 맨 위에 추가해야 합니다.

```
<!-- SCRIPTING-LANGUAGE=XXX -->
```

이 예제에서 XXX는 스크립팅 언어를 나타냅니다. 이 명령문으로 인해 서버 비헤이비어는 현재 선택된 스크립팅 언어가 XXX인 경우에만 [서버 비헤이비어] 패널의 플러스(+) 메뉴에 나타납니다.

인수
없음

반환값
지원되는 스크립팅 언어를 나타내는 문자열 배열을 반환합니다.

getServerModelExtDataNameUD4()

지원 버전
Dreamweaver MX

설명
이 함수는 Configurations/ExtensionData 폴더에 있는 UltraDev 4 Extension 데이터 파일을 액세스할 때 Dreamweaver가 사용하는 서버 모델 구현 이름을 반환합니다.

인수
없음

반환값
"ASP/JavaScript" 등과 같은 문자열을 반환합니다.

getServerModelDelimiters()

지원 버전
Dreamweaver MX

설명
이 함수는 응용 프로그램 서버가 사용하는 스크립트 구분 기호를 반환하고 각 구분 기호가 코드 블록 병합에 참여할 수 있는지 여부를 알려 줍니다. dom.serverModel.getDelimiters() 함수를 호출하여 JavaScript에서 이 반환값을 액세스할 수 있습니다.

인수
없음

반환값
각 객체에 다음 세 가지 속성이 들어 있는 객체 배열을 반환합니다.

- **startPattern** 속성은 열기 스크립트 구분 기호(예: <%)에 해당하는 일반 표현식입니다.
- **endPattern** 속성은 닫기 스크립트 구분 기호(예: %>)에 해당하는 일반 표현식입니다.
- **participateInMerge** 속성은 목록의 구분 기호로 둘러싸인 내용이 블록 병합에 참여하는지 여부를 지정하는 부울 값입니다. 참여할 수 있으면 true이고, 그렇지 않으면 false입니다.

getServerModelDisplayName()

지원 버전
Dreamweaver MX

설명
이 함수는 사용자 인터페이스에 나타날 이 서버 모델의 이름을 반환합니다. dom.serverModel.getDisplayName() 함수를 호출하여 JavaScript에서 이 값을 액세스할 수 있습니다.

인수
없음

반환값
"ASP JavaScript" 등과 같은 문자열을 반환합니다.

getServerModelFolderName()

지원 버전
Dreamweaver MX

설명
이 함수는 Configuration 폴더 내에서 이 서버 모델에 사용할 폴더 이름을 반환합니다. dom.serverModel.getFolderName() 함수를 호출하여 JavaScript에서 이 값에 액세스할 수 있습니다.

인수
없음

반환값
"ASP_JS" 등과 같은 문자열을 반환합니다.

getServerSupportsCharset()

지원 버전
Dreamweaver MX

설명
이 함수는 현재 서버가 지정된 문자 세트를 지원하면 true를 반환합니다. JavaScript에서는 dom.serverModel.getServerSupportsCharset() 함수를 호출하여 서버 모델이 특정 문자 집합을 지원하는지 여부를 판별할 수 있습니다.

인수
metaCharSetString

metaCharSetString 인수는 문서의 "charset=" 속성 값이 들어 있는 문자열입니다.

반환값
부울 값을 반환합니다.

getVersionArray()

지원 버전
Dreamweaver UltraDev 1. Dreamweaver MX에서 사용되지 않음

설명
이 함수는 서버 기술과 버전 번호의 매핑을 조회합니다. 이 함수는 dom.serverModel.getServerVersion() 함수에 의해 호출됩니다.

인수
없음

반환값
다음 예제와 같이 각 버전 이름과 버전 값이 들어 있는 버전 객체 배열을 반환합니다.

- ASP 2.0
- ADODB 2.1

22장: 데이터 변환기

데이터 변환기를 사용하면 특수화된 마크업을 Adobe Dreamweaver에서 읽고 표시할 수 있는 코드로 변환할 수 있습니다. 특수화된 마크업의 예로는 서버측 포함 파일, JavaScript 조건문 및 PHP3, JSP, CFML, ASP 등의 기타 코드가 있습니다. Dreamweaver에서는 코드 블록이나 태그 전체뿐만 아니라 태그 내의 속성도 변환할 수 있습니다. 모든 데이터 변환기, 블록/태그나 속성은 HTML 파일입니다.

특히 전체 태그나 코드 블록의 경우 데이터 변환 작업을 수행할 때 JavaScript로 수행할 수 없거나 C를 사용하여 보다 효과적으로 수행할 수 있는 복잡한 작업이 포함됩니다. 또한 C나 C++에 친숙한 경우 323페이지의 “C 레벨 확장성”을 참조하십시오.

다음 표에서는 데이터 변환기를 만들 때 사용하는 파일을 설명합니다.

경로	파일	설명
Configuration/ThirdPartyTags/	**language**.xml	마크업 언어 태그에 대한 정보가 포함되어 있습니다.
Configuration/ThirdPartyTags/	**language**.gif	언어 태그에 대한 아이콘입니다.
Configuration/Translators/	**language**.htm	데이터 변환기에 대한 JavaScript 함수가 포함되어 있습니다.

데이터 변환기 작업 방법

Dreamweaver에서는 태그 전체를 변환하든지 또는 속성만 변환하든지 관계없이 모든 변환기 파일을 같은 방식으로 처리합니다. Dreamweaver는 시작할 때 Configuration/Translator 폴더의 파일을 모두 읽고 getTranslatorInfo() 함수를 호출하여 변환기에 대한 정보를 얻습니다. Dreamweaver에서는 getTranslatorInfo() 함수가 존재하지 않는 파일이나 이 함수가 정의되지 않도록 하는 오류를 포함하는 파일을 무시합니다.

참고: JavaScript 오류가 발생하여 변환기를 실행하지 못하는 경우가 발생하지 않게 하기 위해 변환기를 모두 불러온 후에만 변환기 파일의 오류가 보고됩니다. 변환기 디버깅에 대한 자세한 내용은 311페이지의 “사용하는 변환기에서 버그 찾기”를 참조하십시오.

또한 Dreamweaver에서는 사용자가 새 내용이나 변환이 필요한 기존 내용을 변경할 때마다 해당하는 모든 변환기 파일([변환] 환경 설정에 지정되어 있음)에서 translateMarkup() 함수를 호출합니다. Dreamweaver에서는 사용자가 다음 작업 중 하나를 수행할 때 translateMarkup() 함수를 호출합니다.

- Dreamweaver에서 파일을 엽니다.
- [HTML] 패널 또는 [코드] 뷰에서 변경 작업을 수행한 다음 [디자인] 뷰로 다시 전환합니다.
- 현재 문서에서 객체의 속성을 변경합니다.
- 삽입 막대 또는 [삽입] 메뉴 사용하여 객체를 삽입합니다.
- 현재 문서를 다른 응용 프로그램에서 변경한 다음 새로 고칩니다.
- 템플릿을 문서에 적용합니다.
- [문서] 윈도우로 또는 해당 윈도우 내에서 내용을 붙여넣거나 드래그합니다.
- 종속 파일에 변경 사항을 저장합니다.
- 태그 객체의 innerHTML 또는 outerHTML 속성 또는 주석 객체의 data 속성을 설정하는 명령, 비헤이비어, 서버 비헤이비어, 속성 관리자 및 기타 Extension을 호출합니다.
- [파일] > [변환] > [3.0 브라우저 호환]을 선택합니다.
- [수정] > [변환] > [표를 AP div로 변환]을 선택합니다.

- [수정] > [변환] > [AP div를 표로 변환]을 선택합니다.
- 퀵 태그 편집기에서 태그나 속성을 변경하고 Tab 키 또는 Enter 키를 누릅니다.

사용할 변환기 종류 확인

모든 변환기에는 getTranslatorInfo() 및 translateMarkup() 함수가 포함되어 있어야 하며 모든 변환기는 Configuration/Translators 폴더에 있어야 합니다. 그러나 변환기는 다음 목록에 설명된 것처럼 사용자의 문서에 삽입되는 코드 종류 및 코드 관리 방법에 따라 다릅니다.

- 속성값을 판별하거나 표준 HTML 태그에 속성을 조건부로 추가하는 서버 마크업을 변환하려면 속성 변환기를 작성합니다. 변환된 속성이 포함된 표준 HTML 태그는 Dreamweaver에 내장된 속성 관리자로 관리할 수 있습니다. 사용자 정의 속성 관리자를 작성할 필요가 없습니다. 자세한 내용은 307페이지의 “변환된 속성을 태그에 추가”를 참조하십시오.
- SSI(서버측 포함)와 같은 전체 태그 또는 코드 블록(예: JavaScript, ColdFusion, PHP 또는 기타 스크립팅)을 변환하려면 블록/태그 변환기를 만들어야 합니다. 블록/태그 변환기에 의해 생성되는 코드는 Dreamweaver에 내장된 속성 관리자로 관리할 수 없습니다. 사용자가 원본 코드의 속성을 변경할 수 있도록 하려면 변환된 내용에 대한 사용자 정의 속성 관리자를 작성해야 합니다. 자세한 내용은 308페이지의 “변환된 태그 또는 코드 블록 잠그기”를 참조하십시오.

변환된 속성을 태그에 추가

속성 변환 여부는 서버 마크업을 무시하는 Dreamweaver 파서에 따라 결정됩니다. 기본적으로 Dreamweaver에서는 일반적인 서버 마크업(예: ASP, CFML, PHP 등)을 무시합니다. 열기 및 닫기 표시 기호가 서로 다른 서버 마크업을 사용하고 있으면 타사 태그 데이터베이스를 수정해야 변환기가 제대로 작동합니다. 타사 태그 데이터베이스 수정에 대한 자세한 내용은 **Dreamweaver 사용 설명서**에서 "Dreamweaver 사용자 정의"를 참조하십시오.

Dreamweaver에서 원본 서버 마크업을 보존하면 변환기는 [문서] 윈도우에서 볼 수 있는 유효한 속성값을 생성합니다. 사용자 화면 표시에 영향을 주지 않는 속성에 대해서만 서버 마크업을 사용하는 경우 변환기가 필요하지 않습니다.

변환기는 서버 마크업이 포함된 태그에 mmTranslatedValue라는 특정 속성을 추가하여 [문서] 윈도우의 화면 표시에 영향을 주는 속성값을 만듭니다. mmTranslatedValue 속성과 이 속성값은 [HTML] 패널 또는 [코드] 뷰에 표시되지 않으며 문서와 함께 저장되지도 않습니다.

mmTranslatedValue 속성은 태그 내에서 고유해야 합니다. 변환기가 한 태그에서 속성을 두 개 이상 변환해야 한다고 판단되면 변환기에 mmTranslatedValue 속성 뒤에 숫자를 덧붙이는(예: mmTranslatedValue1, mmTranslatedValue2 등) 루틴을 추가해야 합니다.

mmTranslatedValue 속성의 값은 유효한 속성-값 쌍이 적어도 하나 이상 포함된 URL 인코딩 문자열이어야 합니다. 즉 mmTranslatedValue="src=%22open.jpg%22"는 src="<? if (dayType == weekday) then open.jpg else closed.jpg" ?> 및 <? if (dayType == weekday) then src="open.jpg" else src="closed.jpg" ?> 모두에 사용할 수 있는 유효한 변환입니다. mmTranslatedValue="%22open.jpg%22"가 속성을 제외하고 값만 포함하고 있으므로 두 가지 경우 모두 유효하지 않습니다.

동시에 둘 이상의 속성 변환

mmTranslatedValue 속성에는 유효한 속성-값 쌍이 두 개 이상 포함될 수 있습니다. 다음 변환되지 않은 코드를 참조하십시오.

```
<img <? if (dayType==weekday) then src="open.jpg" width="320" height="100" else
src="closed.jpg" width="100" height="320" ?> alt="We're open 24 hours a day from
12:01am Monday until 11:59pm Friday">
```

다음 예제에서는 변환된 마크업의 모습을 보여 줍니다.

```
<img <? if (dayType==weekday) then src="open.jpg" width="320" height="100" else
src="closed.jpg" width="100" height="320" ?>
mmTranslatedValue="src=%22open.jpg%22 width=%22320%22 height=%22100%22"
alt="We're open 24 hours a day from 12:01am Monday until 11:59pm Friday">
```

mmTranslatedValue 속성의 속성-값 쌍 사이의 공백은 인코딩되지 않습니다. Dreamweaver에서는 변환된 값을 렌더링할 때 이러한 공백을 찾으므로 mmTranslatedValue의 각 속성-값 쌍을 별도로 인코딩하고 나중에 합쳐 완전한 mmTranslatedValue 속성을 구성해야 합니다. 이 과정의 예제를 보려면 312페이지의 “간단한 속성 변환기 예제”를 참조하십시오.

변환된 속성 관리

변환된 속성은 잠금을 요청하지 않으므로 속성 내에 포함된 태그를 관리하기가 쉽습니다.

서버 마크업이 속성 관리자에 표시되는 단일 속성을 지정하면 속성 관리자에 해당 마크업이 표시됩니다.

마크업은 변환기가 연결되어 있는지 여부에 관계없이 나타납니다. 변환기는 속성 관리자에 나타나는 서버 마크업을 사용자가 편집할 때마다 실행됩니다.

서버 마크업이 태그의 속성을 둘 이상 조절하는 경우 서버 마크업은 속성 관리자에 나타나지 않습니다. 그러나 번개 모양의 아이콘은 선택한 요소에 대한 변환된 마크업이 존재함을 보여 줍니다.

참고: 텍스트나 표 셀, 행 또는 열을 선택하면 번개 모양의 아이콘이 나타나지 않습니다. 사용자가 패널에서 서버 마크업을 편집하고 해당 마크업 유형을 처리할 변환기가 존재하면 변환이 계속됩니다.

속성 관리자의 텍스트 상자는 편집 가능합니다. 이 경우, 사용자가 서버 마크업이 제어하는 속성의 값을 입력하면 속성이 중복될 수 있습니다. 변환된 값과 일반 값 둘 다 특정 속성에 대해 설정된 경우 변환된 값이 [문서] 윈도우에 표시되고 변환기가 중복 속성을 검색하는지 여부가 확인된 다음 중복 속성이 제거됩니다.

변환된 태그 또는 코드 블록 잠그기

대부분의 경우 변환기가 마크업을 변경하여 Dreamweaver에 변경된 마크업이 표시되도록 합니다. 하지만 저장하는 대상은 변경 내용이 아니라 원래의 마크업입니다. 이러한 경우 Dreamweaver에서는 변환된 내용을 묶고 원본 코드를 참조할 위치에 특별한 XML 태그를 제공합니다.

이러한 XML 태그를 사용하면 원본 속성의 내용이 [코드] 뷰에 복제됩니다. 파일을 저장하면 변환되지 않은 원본 마크업이 파일에 기록됩니다. [코드] 뷰에는 변환되지 않은 내용이 표시됩니다.

XML 태그의 구문은 다음 예제와 같습니다.

```
<MM:BeginLock translatorClass="translatorClass" ¬
type="tagNameOrType" depFiles="dependentFilesList" ¬
orig="encodedOriginalMarkup">
Translated content
<MM:EndLock>
```

이 예제의 값이 의미하는 것은 다음과 같습니다.

- **translatorClass** 값은 변환기에 대한 고유 식별자입니다. 즉, 이 값은 getTranslatorInfo() 함수가 반환하는 배열의 첫 번째 문자열입니다.
- **tagNameOrType** 값은 잠금에 포함되어 있는 마크업 형식 또는 마크업에 연결된 태그 이름을 식별하는 문자열입니다. 문자열에는 영숫자, 하이픈(-) 또는 밑줄(_) 문자만 포함될 수 있습니다. 사용자 정의 속성 관리자의 canInspectSelection() 함수에서 이 값을 확인하여 속성 관리자가 내용에 적합한 것인지 판별할 수 있습니다. 자세한 내용은 309페이지의 “잠긴 내용에 대한 속성 관리자 만들기”를 참조하십시오. 잠긴 내용은 Dreamweaver에 내장된 속성 관리자에서 관리할 수 없습니다. 예를 들어 type="IMG"를 지정해도 [이미지] 패널은 나타나지 않습니다.

- **dependentFilesList** 값은 잠긴 마크업이 의존하는 파일 목록이 쉼표로 구분되어 들어 있는 문자열입니다. 파일은 사용자 문서에 상대적인 URL로 참조됩니다. 사용자가 **dependentFilesList**의 파일 중 하나를 업데이트하면 목록이 포함된 문서의 내용이 자동으로 다시 변환됩니다.
- **encodedOriginalMarkup** 값은 URL 인코딩의 일부를 사용(" 대신 %22, < 대신 %3C, > 대신 %3E, % 대신 %25를 사용)하여 인코딩된 변환되지 않은 원본 마크업이 포함된 문자열입니다. escape() 메서드를 사용하면 가장 빠르게 문자열을 URL 인코딩할 수 있습니다. 예를 들어 myString이 '<img src="foo.gif">'이면 escape(myString)은 %3Cimg%20src=%22foo.gif%22%3E를 반환합니다.

다음 예제에서는 서버측 포함 <!--#include virtual="/footer.html" -->의 변환에서 생성될 수 있는 코드의 잠긴 부분을 보여 줍니다.

```
<MM:BeginLock translatorClass="MM_SSI" type="ssi" ¬
depFiles="C:\sites\webdev\footer.html" orig="%3C!--#include ¬
virtual=%22/footer.html%22%20--%3E">
<!-- begin footer -->
<CENTER>
<HR SIZE=1 NOSHADE WIDTH=100%>

<BR>

[<A TARGET="_top" HREF="/">home</A>]
[<A TARGET="_top" HREF="/products/">products</A>]
[<A TARGET="_top" HREF="/services/">services</A>]
[<A TARGET="_top" HREF="/support/">support</A>]
[<A TARGET="_top" HREF="/company/">about us</A>]
[<A TARGET="_top" HREF="/help/">help</A>]
</CENTER>
<!-- end footer -->
<MM:EndLock>
```

script 태그 내에서 코드를 잠그는 변환기를 만들면 변환기에 문제가 발생할 수 있습니다. 예를 들어 다음과 같은 코드가 있다고 가정합니다.

```
<script language="javascript">
<!--
function foo() {
alert('<bean:message key="show.message"/>');
}
// -->
</script>
```

그런 다음 bean:message 받침 태그에 대한 변환기를 만들면 MM:BeginLock 섹션 내에 MM:BeginLock 섹션을 만드는 것이므로 변환기가 실패합니다. 이 문제를 해결하려면 <%= My_lookup.lookup("show.message") %> 등의 정규 JSP 태그를 사용하는 bean:message 태그 주위에 JSP 래퍼를 만듭니다. 이렇게 하면 변환기가 이 코드를 건너뛰어 변환에 성공합니다.

잠긴 내용에 대한 속성 관리자 만들기

변환기를 만든 후에는 사용자가 해당 속성(예: 포함할 파일이나 조건문의 조건 중 하나)을 변경할 수 있도록 하는 내용이 구현된 속성 관리자를 만들어야 합니다. 변환된 내용의 관리는 다음과 같은 몇 가지 경우에만 필요합니다.

- 변환된 내용의 속성을 변경하려고 하며 이러한 변경 사항이 변환되지 않은 내용에 반영되어야 할 경우입니다.
- lock 태그 및 이 태그 내의 태그는 DOM의 노드이므로 DOM(문서 객체 모델)에는 변환된 내용이 포함되어 있지만, documentElement 객체, dreamweaver.getSelection() 및 dreamweaver.nodeToOffsets() 함수의 outerHTML 속성은 변환되지 않은 소스에 적용됩니다.
- 관리 대상 태그는 변환 전과 후에 다릅니다.

  HAPPY 태그에 대한 속성 관리자에는 다음 코드와 비슷한 주석이 있을 수 있습니다.

```
<!-- tag:HAPPY,priority:5,selection:exact,hline,vline, attrName:xxx,¬ attrValue:yyy -->
```

그러나 변환된 HAPPY 태그에 대한 속성 관리자의 주석은 다음 코드와 비슷합니다.

```
<!-- tag:*LOCKED*,priority:5,selection:within,hline,vline -->
```

변환되지 않은 HAPPY 속성 관리자에 대한 canInspectSelection() 함수는 간단합니다. selection 유형이 exact이므로 이 함수는 더 이상 분석하지 않고 true 값을 반환합니다. 변환된 HAPPY 속성 관리자의 경우, 이 함수는 복잡합니다. 키워드 *LOCKED* 는 선택이 잠긴 영역 내에 있을 때에는 속성 관리자가 적절하다고 나타냅니다. 하지만 문서에 잠긴 영역이 여러 개 있을 수 있기 때문에 속성 관리자가 이 잠긴 영역과 일치하는지 판별하기 위해 확인 작업을 더 수행해야 합니다.

또 다른 문제는 변환된 내용 관리의 고유한 문제입니다. dom.getSelection() 함수를 호출할 때 기본적으로 반환되는 값은 변환되지 않은 소스로의 오프셋입니다. 잠긴 영역만 선택되도록 선택 영역을 적절히 확장하려면 다음 메서드를 사용합니다.

```
var currentDOM = dw.getDocumentDOM();
var offsets = currentDOM.getSelection();
var theSelection = currentDOM.offsetsToNode(offsets[0],offsets[0]+1);
```

offsets[0]+1을 두 번째 인수로 사용하면 오프셋을 노드로 변환할 때 열기 잠금 태그에서 벗어나지 않을 수 있습니다. offsets[1]을 두 번째 인수로 사용하면 잠금 상태가 아닌 노드를 선택하는 위험을 감수해야 합니다.

nodeType 값이 node.ELEMENT_NODE인지 확인한 후 선택을 하고 나면, 다음 예제에서처럼 type 속성을 검사하여 잠긴 영역이 이 속성 관리자와 일치하는지 확인할 수 있습니다.

```
if (theSelection.nodeType == node.ELEMENT_NODE && ¬
theSelection.getAttribute('type') == 'happy'){
    return true;
}else{
    return false
}
```

변환된 태그의 속성 관리자에 있는 텍스트 상자를 채우려면 orig 속성의 값을 파싱해야 합니다. 예를 들어 변환되지 않은 코드가 <HAPPY TIME="22">이고 속성 관리자에 [시간] 텍스트 상자가 있으면 orig 문자열에서 TIME 속성의 값을 추출해야 합니다.

```
function inspectSelection() {
    var currentDOM = dw.getDocumentDOM();
    var currSelection = currentDOM.getSelection();
    var theObj = currentDOM.offsetsToNode(curSelection[0],curSelection[0]+1);

    if (theObj.nodeType != Node.ELEMENT_NODE) {
        return;
    }

    // To convert the encoded characters back to their
    // original values, use the unescape() method.
    var origAtt = unescape(theObj.getAttribute("ORIG"));

    // Convert the string to lowercase for processing
    var origAttLC = origAtt.toLowerCase();

    var timeStart = origAttLC.indexOf('time="');
    var timeEnd = origAttLC.indexOf('"',timeStart+6);
    var timeValue = origAtt.substring(timeStart+6,timeEnd);

    document.layers['timelayer'].document.timeForm.timefield.value = timeValue;
}
```

변환된 태그의 속성 관리자의 텍스트 상자를 채우기 위해 orig 속성을 파싱했으면, 다음 단계는 사용자가 임의의 텍스트 상자 값을 변경할 경우 orig 속성의 값을 설정하는 것입니다. 잠긴 영역이 변경할 수 없도록 제한되어 있을 수 있습니다. 원본 마크업을 변경하고 다시 변환하면 이 문제를 방지할 수 있습니다.

변환된 서버측 포함(Configuration/Inspectors 폴더의 ssi_translated.js 파일)에 대한 속성 관리자는 setComment() 함수에서 이러한 기술을 보여 줍니다. orig 속성을 다시 작성하는 대신 속성 관리자는 새 서버측 포함 주석을 구성합니다. 그런 다음 해당 주석을 문서에 삽입하고 문서의 내용을 다시 작성하여 기존의 내용을 대체합니다. 이렇게 하여 새 orig 속성이 생성됩니다. 다음은 이 방법을 요약한 코드입니다.

```
// Assemble the new include comment. radioStr and URL are
// variables defined earlier in the code.
newInc = "<!--#include " + radioStr + "=" + '"' + URL + '"' + " -->";
// Get the contents of the document.
var entireDocObj = dreamweaver.getDocumentDOM();
var docSrc = entireDocObj.documentElement.outerHTML;
// Store everything up to the SSI comment and everything after
// the SSI comment in the beforeSelStr and afterSelStr variables.
var beforeSelStr = docSrc.substring(0, curSelection[0] );
var afterSelStr= docSrc.substring(curSelection[1]);
// Assemble the new contents of the document.
docSrc = beforeSelStr + newInc + afterSelStr;
// Set the outerHTML of the HTML tag (represented by
// the documentElement object) to the new contents,
// and then set the selection back to the locked region
// surrounding the SSI comment.
entireDocObj.documentElement.outerHTML = docSrc;
entireDocObj.setSelection(curSelection[0], curSelection[0]+1);
```

사용하는 변환기에서 버그 찾기

translateMarkup() 함수에 특정 유형의 오류가 포함되어 있더라도 변환기는 제대로 로드됩니다. 그러나 이 함수를 호출하면 오류 메시지 없이 호출되지 않습니다. 반응 없이 실패하면 Dreamweaver가 불안정을 방지할 수는 있지만 여러 줄의 코드에서 사소한 구문 오류 하나를 찾아야 하는 등의 개발 작업을 할 때는 문제가 될 수 있습니다.

변환기의 실행이 실패로 끝날 때 효과적인 디버그 방법은 다음 단계와 같이 변환기를 명령어로 전환하는 것입니다.

1. 변환기 파일의 내용 전체를 새 문서에 복사하고 Dreamweaver 응용 프로그램 폴더 내의 Configuration/Commands 폴더에 저장합니다.
2. 문서 위쪽에 있는 SCRIPT 태그 사이에 다음 함수를 추가합니다.

```
function commandButtons(){
    return new Array( "OK","translateMarkup(dreamweaver.¬
    getDocumentPath('document'), dreamweaver.getSiteRoot(), ¬
    dreamweaver.getDocumentDOM().documentElement.outerHTML); ¬
    window.close()", "Cancel", "window.close()");
}
```

3. 다음 예제에서와 같이 translateMarkup() 함수의 끝에서 return**whateverTheReturnValueIs** 행을 주석 처리하고 이 행을 dreamweaver.getDocumentDOM().documentElement.outerHTML = ***whateverTheReturnValueIs***로 대체합니다.

```
    // return theCode;
    dreamweaver.getDocumentDOM().documentElement.outerHTML = theCode;
}
/* end of translateMarkup() */
```

4. 문서의 body에 다음과 같이 텍스트 상자가 없는 양식을 추가합니다.

```
<body>
<form>
Hello.
</form>
</body>
```

5 Dreamweaver를 다시 시작하고 [명령] 메뉴에서 변환기 명령을 선택합니다. [확인]을 클릭하면 translateMarkup() 함수가 호출되어 변환을 시뮬레이션합니다.

오류 메시지가 나타나지 않는데도 변환이 계속 실패할 경우 코드에 논리 오류가 있을 수 있습니다.

6 해당 분기가 맞는지 확인하고 다른 지점의 속성과 변수 값을 확인할 수 있도록 translateMarkup() 함수 내의 특정 위치에 alert() 명령문을 추가합니다.

```
for (var i=0; i< foo.length; i++){
    alert("we're at the top of foo.length array, and the value of i is " + i);
    /* rest of loop */
}
```

7 alert() 명령문을 추가한 후, [명령] 메뉴에서 사용할 명령을 선택하고 [취소]를 클릭한 다음 다시 해당 명령을 선택합니다. 이렇게 하면 명령 파일이 다시 로드되고 변경 사항이 통합됩니다.

간단한 속성 변환기 예제

예제를 살펴보면 속성 변환에 대한 이해를 높이는 데 도움이 됩니다. 다음 변환기는 ASP 또는 PHP와 다소 유사한 구문인 Poco(Pound Conditional) 마크업입니다.

속성 변환기를 만들려면 tagspec 태그와 아이콘을 만든 다음 속성 변환기를 만듭니다.

tagspec 태그 만들기

이 변환기가 제대로 작동하기 위한 첫 번째 단계는 Poco 마크업의 tagspec를 만드는 것입니다. 이렇게 하면 변환되지 않은 Poco 명령문이 파싱되지 않습니다.

1 비어 있는 새 파일을 만듭니다.

2 다음을 입력합니다.

```
<tagspec tag_name="poco" start_string="<#" end_string="#>" ¬
detect_in_attribute="true" icon="poco.gif" icon_width="17" ¬
icon_height="15"></tagspec>
```

3 이 파일을 Configuration/ThirdPartyTags 폴더에 poco.xml로 저장합니다.

tagspec 태그 작성 예제는 Configuration/ThirdPartyTags 폴더에서 Tags.xml 파일을 참조하십시오.

아이콘 만들기

다음으로 Poco 태그에 대한 아이콘을 만듭니다.

1 18 x 18픽셀의 Poco 태그 아이콘에 대한 이미지 파일을 만듭니다.

2 이 파일을 Configuration/ThirdPartyTags 폴더에 poco.gif로 저장합니다.

속성 변환기 만들기

속성 변환기에 필요한 함수를 포함하는 HTML 파일을 만듭니다.

1 비어 있는 새 파일을 만듭니다.

2 다음을 입력합니다.

```
<html>
<head>
<title>Conditional Translator</title>
<meta http-equiv="Content-Type" content="text/html; charset=">
<script language="JavaScript">

/************************************************************
 * This translator handles the following statement syntaxes: *
 * <# if (condition) then foo else bar #>                    *
 * <# if (condition) then att="foo" else att="bar" #>        *
 * <# if (condition) then att1="foo" att2="jinkies"          *
 * att3="jeepers" else att1="bar" att2="zoinks" #>           *
 *                                                          *
 * It does not handle statements with no else clause.       *
 ************************************************************/

var count = 1;

function translateMarkup(docNameStr, siteRootStr, inStr){
var count = 1;
// Counter to ensure unique mmTranslatedValues
var outStr = inStr;
// String that will be manipulated
var spacer = "";
// String to manage space between encoded attributes
var start = inStr.indexOf('<# if');   // 1st instance of Pound Conditional code
// Declared but not initalized //
var attAndValue;
// Boolean indicating whether the attribute is part of
// the conditional statement
var trueStart;
// The beginning of the true case
var falseStart;
// The beginning of the false case
var trueValue;
// The HTML that would render in the true case
var attName;
// The name of the attribute that is being
// set conditionally.
var equalSign;
// The position of the equal sign just to the
// left of the <#, if there is one
var transAtt;
// The entire translated attribute
var transValue;
// The value that must be URL-encoded
var back3FromStart;
// Three characters back from the start position
// (used to find equal sign to the left of <#
var tokens;
// An array of all the attributes set in the true case
var end;
// The end of the current conditional statement.
// As long as there's still a <# conditional that hasn't been
// translated.
while (start != -1){
    back3FromStart = start-3;
    end = outStr.indexOf(' #>',start);
    equalSign = outStr.indexOf('="<# if',back3FromStart);
    attAndValue = (equalSign != -1)?false:true;
    trueStart = outStr.indexOf('then', start);
    falseStart = outStr.indexOf(' else', start);
```

```
    trueValue = outStr.substring(trueStart+5, falseStart);
    tokens = dreamweaver.getTokens(trueValue,' ');

    // If attAndValue is false, find out what attribute you're
    // translating by backing up from the equal sign to the
    // first space. The substring between the space and the
    // equal sign is the attribute.
    if (!attAndValue){
        for (var i=equalSign; i > 0; i--){
            if (outStr.charAt(i) == " "){
                attName = outStr.substring(i+1,equalSign);
                break;
            }
        }
        transValue = attName + '="' + trueValue + '"';
        transAtt = ' mmTranslatedValue' + count + '="' + ¬
        escape(transValue) + '"';
        outStr = outStr.substring(0,end+4) + transAtt + ¬
        outStr.substring(end+4);
    // If attAndValue is true, and tokens is greater than
    // 1, then trueValue is a series of attribute/value
    // pairs, not just one. In that case, each attribute/value
    // pair must be encoded separately and then added back
    // together to make the translated value.
    }else if (tokens.length > 1){
        transAtt = ' mmTranslatedValue' + count + '="'
        for (var j=0; j < tokens.length; j++){
            tokens[j] = escape(tokens[j]);
            if (j>0){
                spacer=" ";
            }
            transAtt += spacer + tokens[j];
        }
        transAtt += '"';
        outStr = outStr.substring(0,end+3) + transAtt + ¬
        outStr.substring(end+3)

    // If attAndValue is true and tokens is not greater
    // than 1, then trueValue is a single attribute/value pair.
    // This is the simplest case, where all that is necessary is
    // to encode trueValue.
    }else{
        transValue = trueValue;
        transAtt = ' mmTranslatedValue' + count + '="' + ¬
        escape(transValue) + '"';
        outStr = outStr.substring(0,end+3) + transAtt + ¬
        outStr.substring(end+3);
    }
    // Increment the counter so that the next instance
    // of mmTranslatedValue will have a unique name, and
    // then find the next <# conditional in the code.
    count++;
    start = outStr.indexOf('<# if',end);
}
// Return the translated string.
return outStr
}
```

```
function getTranslatorInfo(){
    returnArray = new Array(7);

    returnArray[0] = "Pound_Conditional";               // The translatorClass
    returnArray[1] = "Pound Conditional Translator";    // The title
    returnArray[2] = "2";                               // The number of extensions
    returnArray[3] = "html";                            // The first extension
    returnArray[4] = "htm";                             // The second extension
    returnArray[5] = "1";                               // The number of expressions
    returnArray[6] = "<#";                              // The first expression
    returnArray[7] = "byString";                        //
    returnArray[8] = "50";                                  //
    return returnArray
}
</script>
</head>

<body>
</body>
</html>
```

3 이 파일을 Configuration/Translators 폴더에 Poco.htm으로 저장합니다.

간단한 블록/태그 변환기 예제

변환에 대해 이해하려면 완전히 JavaScript로 작성된 변환기를 살펴 보십시오. 이 변환기는 C 라이브러리에 구애받지 않고 모든 기능이 동작합니다. 다음 예제의 변환기는 C로 작성하면 더욱 효율적으로 작성할 수 있습니다. 하지만 JavaScript로 작성하면 더욱 간단하게 작성할 수 있으므로 변환기의 작동 방식을 설명하기에 적합합니다.

대부분의 변환기처럼 이 변환기는 서버 비헤이비어를 모방하도록 설계되었습니다. 요일, 시간 및 사용자의 플랫폼에 따라 KENT 태그가 엔지니어의 다른 사진으로 바뀌도록 웹 서버를 구성했다고 가정합니다. 변환기는 로컬에서만 이와 동일한 작업을 합니다.

블록/태그 변환기 만들기

1 비어 있는 새 파일을 만듭니다.

2 다음 코드를 입력합니다.

```
<html>
<head>
<title>Kent Tag Translator</title>
<meta http-equiv="Content-Type" content="text/html; charset=">
<script language="JavaScript">
/*********************************************************
 * The getTranslatorInfo() function provides information  *
 * about the translator, including its class and name,    *
 * the types of documents that are likely to contain the  *
 * markup to be translated, the regular expressions that  *
 * a document containing the markup to be translated      *
 * would match (whether the translator should run on all  *
 * files, no files, in files with the specified           *
 * extensions, or in files matching the specified         *
 * expressions).                                          *
 *********************************************************/
function getTranslatorInfo(){
    //Create a new array with 6 slots in it
    returnArray = new Array(6);

    returnArray[0] = "DREAMWEAVER_TEAM";           // The translatorClass
    returnArray[1] = "Kent Tags";                  // The title
```

```
    returnArray[2] = "0";                          // The number of extensions
    returnArray[3] = "1";                          // The number of expressions
    returnArray[4] = "<kent";                      // Expression
    returnArray[5] = "byExpression";               // run if the file contains "<kent"
    return returnArray;
}

/****************************************************************************
 * The translateMarkup() function performs the actual translation.        *
 * In this translator, the translateMarkup() function is written          *
 * entirely in JavaScript (that is, it does not rely on a C library) --   *
 * and it's also extremely inefficient. It's a simple example, however,   *
 * which is good for learning.                                            *
****************************************************************************/
function translateMarkup(docNameStr, siteRootStr, inStr){
    var outStr = "";                      // The string to be returned after translation
    var start = inStr.indexOf('<kent>');  // The first position of the KENT tag
                                          // in the document.
    var replCode = replaceKentTag();      // Calls the replaceKentTag() function
                                          // to get the code that will replace KENT.
        var outStr = "";                  // The string to be returned after translation
     //If the document does not contain any content, terminate the translation.
    if ( inStr.length <= 0 ){
         return "";
    }

    // As long as start, which is equal to the location in inStr of the
    // KENT tag, is not equal to -1 (that is, as long as there is another
    // KENT tag in the document)
    while (start != -1){
         // Copy everything up to the start of the KENT tag.
         // This is very important, as translators should never change
         // anything other than the markup that is to be translated.
         outStr = inStr.substring(0, start);
         // Replace the KENT tag with the translated HTML, wrapped in special
         // locking tags. For more information on the repacement operation, see
         // the comments in the replaceKentTag() function.
         outStr = outStr + replCode;

         // Copy everything after the KENT tag.
         outStr = outStr + inStr.substring(start+6);

         // Use the string you just created for the next trip through
         // the document. This is the most inefficient part of all.
         inStr = outStr;
         start = inStr.indexOf('<kent>');

    }
    // When there are no more KENT tags in the document, return outStr.
    return outStr;
}

/************************************************************
* The replaceKentTag() function assembles the HTML that will  *
* replace the KENT tag and the special locking tags that will *
* surround the HTML. It calls the getImage() function to      *
* determine the SRC of the IMG tag.                           *
************************************************************/
function replaceKentTag(){
    // The image to display.
    var image = getImage();
    // The location of the image on the local disk.
```

```
    var depFiles = dreamweaver.getSiteRoot() + image;
    // The IMG tag that will be inserted between the lock tags.
    var imgTag = '<IMG SRC="/' + image + '" WIDTH="320" HEIGHT="240" ALT="Kent">\n';
    // 1st part of the opening lock tag. The remainder of the tag is assembled
    below.
    var start = '<MM:BeginLock translatorClass="DREAMWEAVER_TEAM" type="kent"';
    // The closing lock tag.
    var end = '<MM:EndLock>';

    //Assemble the lock tags and the replacement HTML.
    var replCode = start + ' depFiles="' + depFiles + '"';
    replCode = replCode + ' orig="%3Ckent%3E">\n';
    replCode = replCode + imgTag;
    replCode = replCode + end;

    return replCode;
}

/*****************************************************************
 * The getImage() function determines which image to display      *
 * based on the day of the week, the time of day and the          *
 * user's platform. The day and time are figured based on UTC     *
 * time (Greenwich Mean Time) minus 8 hours, which gives          *
 * Pacific Standard Time (PST). No allowance is made for Daylight *
 * Savings Time in this routine.                                  *
*****************************************************************/
function getImage(){
    var today = new Date();                  // Today's date & time.
    var day = today.getUTCDay();             // Day of the week in the GMT time zone.
                                             // 0=Sunday, 1=Monday, and so on.
        var hour = today.getUTCHours();      // The current hour in GMT, based on the
                                             // 24-hour clock.
    var SFhour = hour - 8;                   // The time in San Francisco, based on the
                                             // 24-hour clock.
    var platform = navigator.platform;       // User's platform. All Windows computers
                                             // are identified by Dreamweaver as "Win32",
                                             // all Macs as "MacPPC".
        var imageRef;                        // The image reference to be returned.
            // If SFhour is negative, you have two adjustments to make.
            // First, subtract one from the day count because it is already the wee
            // hours of the next day in GMT. Second, add SFhour to 24 to
            // give a valid hour in the 24-hour clock.
     if (SFhour < 0){
         day = day - 1;
              // The day count back one would make it negative, and it's Saturday,
              // so set the count to 6.
              if (day < 0){
                      day = 6;
              }
          SFhour = SFhour + 24;
     }

    // Now determine which photo to show based on whether it's a work day or a
    // weekend; what time it is; and, if it's a time and day when Kent is
    // working, what platform the user is on.

    //If it's not Sunday
    if (day != 0){
         //And it's between 10am and noon, inclusive
         if (SFhour >= 10 && SFhour <= 12){
                imageRef = "images/kent_tiredAndIrritated.jpg";
            //Or else it's between 1pm and 3pm, inclusive
```

```
            }else if (SFhour >= 13 && SFhour <= 15){
                   imageRef ="images/kent_hungry.jpg";
            //Or else it's between 4pm and 5pm, inclusive
            }else if (SFhour >= 16 && SFhour <= 17){
                   //If user is on Mac, show Kent working on Mac
                   if (platform == "MacPPC"){
                        imageRef = "images/kent_gettingStartedOnMac.jpg";
                   //If user is on Win, show Kent working on Win
                   }else{
                        imageRef = "images/kent_gettingStartedOnWin.jpg";
                   }
            //Or else it's after 6pm but before the stroke of midnight
            }else if (SFhour >= 18){
                    //If it's Saturday
                    if (day == 6){
                        imageRef = "images/kent_dancing.jpg";
                   //If it's not Saturday, check the user's platform
                   }else if (platform == "MacPPC"){
                        imageRef = "images/kent_hardAtWorkOnMac.jpg";
                   }else{
                        imageRef = "images/kent_hardAtWorkOnWin.jpg";
                   }
            }else{
                   imageRef = "images/kent_sleeping.jpg";
            }
      //If it's after midnight and before 10am, or anytime on Sunday
      }else{
            imageRef = "images/kent_sleeping.jpg";
      }

      return imageRef;
}

</script>
</head>

<body>
</body>
</html>
```

3 이 파일을 Configuration/Translators 폴더에 kent.htm으로 저장합니다.

데이터 변환기 API 함수

이 단원에서는 Dreamweaver의 변환기를 정의하는 데 사용되는 함수에 대해 설명합니다.

getTranslatorInfo()

설명

이 함수는 변환기 및 변환기가 영향을 주는 파일에 대한 정보를 제공합니다.

인수

없음

반환값

문자열 배열을 반환합니다. 배열의 요소는 다음 순서로 나타나야 합니다.

1 **translatorClass** 문자열은 변환기를 고유하게 식별합니다. 이 문자열은 문자로 시작하고 영숫자, 하이픈(-) 및 밑줄(_)만 포함할 수 있습니다.

2 **title** 문자열은 변환기를 40자 이하로 설명합니다.

3 **nExtensions** 문자열은 파일 확장명의 수를 지정합니다. **nExtensions** 값이 0이면 변환기는 어떤 파일에 대해서도 실행될 수 없습니다. **nExtensions** 값이 0이면 **nRegExps**의 값은 배열의 다음 요소입니다.

4 **extension** 문자열은 이 변환기를 사용하여 동작하는 파일 확장명(예: "htm" 또는 "SHTML")을 지정합니다. 이 문자열은 대소문자를 구분하지 않으며 선행 마침표를 포함하고 있지 않습니다. 배열에는 **nExtensions**에서 지정한 것과 동일한 수의 **Extension** 요소가 들어 있어야 합니다.

5 **nRegExps** 문자열은 일반 표현식의 수를 지정합니다. **nRegExps** 값이 0이면 **runDefault**는 배열의 다음 요소입니다.

6 **regExps** 문자열은 확인할 수 있는 일반 표현식을 지정합니다. 배열에는 **nRegExps**에서 지정한 것과 동일한 수의 **regExps** 요소가 들어 있어야 하며, 변환기에서 파일에 대한 작업을 수행하기 전에 **regExps** 요소가 문서 소스 코드와 적어도 하나 이상 일치해야 합니다.

7 **runDefault** 문자열은 이 변환기가 실행되는 시간을 지정합니다. 다음 목록에서는 가능한 문자열 값을 보여 줍니다.

문자열	정의
"allFiles"	변환기가 항상 실행하도록 설정합니다.
"noFiles"	변환기가 전혀 실행되지 않도록 설정합니다.
"byExtension"	파일 확장명이 Extension에 지정되어 있는 파일에 대해 변환기가 실행되도록 설정합니다.
"byExpression"	지정된 일반 표현식 중 하나와 일치하는 항목이 문서에 있으면 변환기가 실행되도록 설정합니다.
"bystring"	지정한 문자열 중 하나와 일치하는 항목이 문서에 있으면 변환기가 실행되도록 설정합니다.

참고: runDefault를 "byExtension"으로 설정하고 확장명을 지정(4단계 참조)하지 않으면 "allFiles" 설정과 동일한 효과가 나타납니다. runDefault를 "byExpression"으로 설정하고 표현식을 지정(6단계 참조)하지 않으면 "noFiles" 설정과 동일한 효과가 나타납니다.

8 **priority** 문자열은 이 변환기를 실행할 기본 우선 순위를 지정합니다. 우선 순위는 0과 100 사이의 수입니다. 우선 순위를 지정하지 않으면 기본 우선 순위는 100이 됩니다. 가장 높은 우선 순위는 0이고 가장 낮은 우선 순위는 100입니다. 문서에 여러 변환기가 적용될 때 이 설정에 따라 변환기의 적용 순서가 조절됩니다. 가장 높은 우선 순위가 가장 먼저 적용됩니다. 여러 변환기의 우선 순위가 같은 경우 translatorClass에 의해 사전순으로 적용됩니다.

예제

getTranslatorInfo() 함수의 다음 인스턴스는 서버측 포함 파일의 변환기에 대한 정보를 제공합니다.

```
function getTranslatorInfo(){
    var transArray = new Array(11);
    transArray[0] = "SSI";
    transArray[1] = "Server-Side Includes";
    transArray[2] = "4";
    transArray[3] = "htm";
    transArray[4] = "stm";
    transArray[5] = "html";
    transArray[6] = "shtml";
    transArray[7] = "2";
    transArray[8] = "<!--#include file";
    transArray[9] = "<!--#include virtual";
    transArray[10] = "byExtension";
    transArray[11] = "50";
    return transArray;
}
```

translateDOM()

지원 버전
Dreamweaver CS3

설명
Dreamweaver는 두 번의 변환 과정을 수행합니다. 첫 번째 과정에서는 모든 변환기를 검색하여 translateMarkup() 함수를 호출합니다. 해당 함수를 호출한 후, 두 번째 과정에서는 translateDOM() 함수를 호출합니다. 전달되는 dom이 변환 대상 dom입니다. 두 번째 과정 동안에는 변환된 속성에 대한 편집만 허용됩니다.

인수
dom, sourceStr

- **dom** 인수입니다.
- **sourceStr** 인수는 translateMarkup에 전달되는 동일한 문자열로, 참조용으로 제공됩니다. 모든 변환은 **sourceStr** 인수가 아닌 **dom** 인수에 대해 수행해야 합니다.

반환값
없음

예제
translateDOM(dom, sourceStr); //반환되는 값이 없습니다.

translateDOM() 함수의 다음 인스턴스는 문서에서 div1 ID를 가진 태그를 숨깁니다.

```
function translateDOM(dom, sourceStr){
    var div1 = dom.getAttributeById("div1");
    if (div1){
    div1.style.display = "none";
    }
}
```

translateMarkup()

설명
이 함수는 변환을 수행합니다.

인수
docName, siteRoot, docContent

- **docName** 인수는 변환될 문서의 file:// URL이 포함된 문자열입니다.
- **siteRoot** 인수는 변환될 문서가 포함된 사이트의 루트에 대한 file:// URL이 포함된 문자열입니다. 문서가 사이트 밖에 있는 경우 이 문자열은 비어 있을 수도 있습니다.
- **docContent** 인수는 문서 내용이 포함된 문자열입니다.

반환값
변환된 문서가 포함된 문자열 또는 변환된 내용이 없으면 빈 문자열을 반환합니다.

예제
translateMarkup() 함수의 다음 인스턴스는 C 함수 translateASP()를 호출합니다. 이 함수는 DLL(Windows) 또는 ASPTrans라는 코드 라이브러리(Macintosh)에 들어 있습니다.

```
function translateMarkup(docName, siteRoot, docContent){
    var translatedString = "";
    if (docContent.length > 0){
    translatedString = ASPTrans.translateASP(docName, siteRoot, ¬
    docContent);
    }
    return translatedString;
}
```

모든 JavaScript 예제는 312페이지의 “간단한 속성 변환기 예제” 또는 315페이지의 “간단한 블록/태그 변환기 예제”를 참조하십시오.

liveDataTranslateMarkup()

지원 버전
Dreamweaver UltraDev 1

설명
이 함수는 사용자가 라이브 데이터 윈도우를 사용하고 있을 때 문서를 변환합니다. 사용자가 [보기] > [라이브 데이터] 명령을 선택하거나 [새로 고침] 버튼을 클릭하면 Dreamweaver에서는 translateMarkup() 함수 대신, liveDataTranslateMarkup() 함수를 호출합니다.

인수
docName, siteRoot, docContent

- **docName** 인수는 변환될 문서의 file:// URL이 포함된 문자열입니다.
- **siteRoot** 인수는 변환될 문서가 포함된 사이트의 루트에 대한 file:// URL이 포함된 문자열입니다. 문서가 사이트 밖에 있는 경우 이 문자열은 비어 있을 수도 있습니다.
- **docContent** 인수는 문서 내용이 포함된 문자열입니다.

반환값
변환된 문서가 포함된 문자열 또는 변환된 내용이 없으면 빈 문자열을 반환합니다.

예제

liveDataTranslateMarkup()의 다음 인스턴스는 C 함수 translateASP()를 호출합니다. 이 함수는 DLL(Windows) 또는 ASPTrans라는 코드 라이브러리(Macintosh)에 들어 있습니다.

```
function liveDataTranslateMarkup(docName, siteRoot, docContent){
    var translatedString = "";
    if (docContent.length > 0){
    translatedString = ASPTrans.translateASP(docName, siteRoot, docContent);
    }
    return translatedString;
}
```

23장: C 레벨 확장성

C 레벨 확장성 메커니즘을 활용하면 JavaScript와 사용자 정의 C 코드를 함께 사용하여 Adobe Dreamweaver 확장 파일을 구현할 수 있습니다. C 언어로 함수를 정의하고 이 함수를 DLL(동적 링크 라이브러리)이나 공유 라이브러리로 만들어 Dreamweaver 응용 프로그램 폴더 내의 Configuration/JSExtensions 폴더에 저장한 다음, Dreamweaver JavaScript 인터프리터를 사용하여 JavaScript에서 해당 함수를 호출합니다.

예를 들어 사용자가 지정한 파일의 내용을 현재 문서에 삽입하는 Dreamweaver 객체를 정의하려는 경우, 클라이언트측 JavaScript는 파일 I/O(입/출력)를 지원하지 않기 때문에 C 언어로 함수를 작성하여 이 기능을 제공해야 합니다.

C 함수의 통합 방식

다음과 같은 HTML 및 JavaScript를 사용하여 파일의 텍스트를 삽입하는 간단한 객체를 만들 수 있습니다. objectTag() 함수는 myLibrary 라이브러리에 저장되어 있는 C 함수 readContentsOfFile()을 호출합니다.

```
<HTML>
<HEAD>
<SCRIPT>
function objectTag() {
    fileName = document.forms[0].myFile.value;
    return myLibrary.readContentsOfFile(fileName);
}
</SCRIPT>
</HEAD>

<BODY>
<FORM>
Enter the name of the file to be inserted:
<INPUT TYPE="file" NAME="myFile">
</FORM>
</BODY>
</HTML>
```

readContentsOfFile() 함수는 사용자로부터 인수 목록을 받아 filename 인수를 검색하고 해당 파일의 내용을 읽은 다음 그 내용을 반환합니다. readContentsOfFile() 함수에 나타나는 JavaScript 데이터 구조 및 함수에 대한 자세한 내용은 325페이지의 “C 레벨 확장성 및 JavaScript 인터프리터”를 참조하십시오.

```
JSBool
readContentsOfFile(JSContext *cx, JSObject *obj, unsigned int ¬
argc, jsval *argv, jsval *rval)
{
    char *fileName, *fileContents;
    JSBool success;
    unsigned int length;

    /* Make sure caller passed in exactly one argument. If not,
     * then tell the interpreter to abort script execution.*/
    if (argc != 1){
        JS_ReportError(cx, "Wrong number of arguments", 0);
    return JS_FALSE;
    }

    /* Convert the argument to a string */
    fileName = JS_ValueToString(cx, argv[0], &length);
    if (fileName == NULL){
        JS_ReportError(cx, "The argument must be a string", 0);
        return JS_FALSE;
    }

    /* Use the string (the file name) to open and read a file */
    fileContents = exerciseLeftToTheReader(fileName);

    /* Store file contents in rval, which is the return value passed
    * back to the caller */
    success = JS_StringToValue(cx, fileContents, 0, *rval);
    free(fileContents);

    /* Return true to continue or false to abort the script */
    return success;
}
```

readContentsOfFile() 함수가 제대로 실행되어 JavaScript 오류가 발생하지 않도록 하려면 JavaScript 인터프리터에 해당 함수를 등록해야 합니다. 이렇게 하려면 라이브러리에 MM_Init() 함수를 포함합니다. 그러면 Dreamweaver에서는 시작 시 라이브러리를 로드하면서 MM_Init() 함수를 호출하여 다음 세 가지 정보를 가져옵니다.

- 함수의 JavaScript 이름
- 함수에 대한 포인터
- 함수에 필요한 인수 개수

다음 예제에서는 myLibrary 라이브러리에 대한 MM_Init() 함수가 어떻게 나타나는지 보여 줍니다.

```
void
MM_Init()
{
    JS_DefineFunction("readContentsOfFile", readContentsOfFile, 1);
}
```

라이브러리에는 다음 매크로의 인스턴스가 한 개만 들어 있어야 합니다.

```
/* MM_STATE is a macro that expands to some definitions that are
 * needed to interact with Dreamweaver. This macro must
 * be defined exactly once in your library. */
MM_STATE
```

참고: 이 라이브러리는 C 또는 C++ 중 하나로 구현할 수 있지만 MM_Init() 함수와 MM_STATE 매크로를 포함하는 파일은 C로 구현해야 합니다. C++ 컴파일러에서는 함수 이름이 다른 방식으로 처리되기 때문에 Dreamweaver에서 MM_Init() 함수를 찾을 수 없게 됩니다.

C 레벨 확장성 및 JavaScript 인터프리터

라이브러리의 C 코드는 다음과 같은 여러 경우에 Dreamweaver JavaScript 인터프리터와 상호 작용해야 합니다.

- 시작 시 라이브러리 함수를 등록할 때
- 함수 호출 시 JavaScript가 C로 전달하는 인수를 파싱할 때
- 함수 반환 전 반환값을 패키지화할 때

이러한 작업을 위해 인터프리터에서는 몇 가지 데이터 유형을 정의하고 API를 노출합니다. 이 단원에서 다루는 데이터 유형 및 함수에 대한 정의는 mm_jsapi.h 파일에 들어 있습니다. 라이브러리가 제대로 동작하도록 하려면 라이브러리에 있는 각 파일의 맨 위에 다음 행을 추가하여 mm_jsapi.h 파일을 포함해야 합니다.

```
#include "mm_jsapi.h"
```

mm_jsapi.h 파일을 포함하면 MM_Environment 구조를 정의하는 mm_jsapi_environment.h도 포함됩니다.

데이터 유형

JavaScript 인터프리터는 다음 데이터 유형을 정의합니다.

typedef struct JSContext JSContext
이 불투명 데이터 유형에 대한 포인터는 C 레벨 함수에 전달됩니다. API의 일부 함수는 이 포인터를 인수 중 하나로 받습니다.

typedef struct JSObject JSObject
이 불투명 데이터 유형에 대한 포인터는 C 레벨 함수에 전달됩니다. 이 데이터 유형은 객체를 나타냅니다. 이 객체는 배열 객체이거나 다른 객체 유형일 수 있습니다.

typedef struct JSVal JSVal
부동 소수점 숫자, 문자열 또는 객체에 대한 포인터나 정수를 포함할 수 있는 불투명한 데이터 구조입니다. API에 있는 일부 함수는 JSVal 구조의 내용을 읽는 방법으로 함수 인수의 값을 읽을 수 있으며, 일부 함수는 JSVal 구조를 작성하는 방법으로 함수의 반환값을 작성하는 데 사용될 수 있습니다.

typedef enum { JS_FALSE = 0, JS_TRUE = 1 } JSBool
부울 값을 저장하는 단순 데이터 유형입니다.

C 레벨 API

C 레벨 확장성 API는 다음과 같은 함수로 구성됩니다.

typedef JSBool (*JSNative)(JSContext *cx, JSObject *obj, unsigned int argc, jsval *argv, jsval *rval)

설명
이 함수 서명은 다음과 같은 상황에서 JavaScript 함수의 C 레벨 구현을 나타냅니다.

- **cx** 포인터는 JavaScript API의 일부 함수에 전달해야 하는 불투명 JSContext 구조에 대한 포인터입니다. 이 변수에는 인터프리터의 실행 컨텍스트가 포함됩니다.

- **obj** 포인터는 해당 컨텍스트에서 스크립트가 실행되는 객체에 대한 포인터입니다. 스크립트가 실행되는 동안에는 this 키워드가 이 객체를 나타냅니다.
- **argc** 정수는 함수에 전달되는 인수의 수입니다.
- **argv** 포인터는 JSVal 구조의 배열에 대한 포인터입니다. 이 배열은 argc개의 요소를 가집니다.
- **rval** 포인터는 단일 JSVal 구조에 대한 포인터입니다. 함수의 반환값은 *rval에 기록되어야 합니다.

성공적으로 실행되면 JS_TRUE가 반환되고 그렇지 않으면 JS_FALSE가 반환됩니다. JS_FALSE가 반환되면 현재 스크립트 실행이 중지되고 오류 메시지가 나타납니다.

JSBool JS_DefineFunction()

설명

Dreamweaver에서 이 함수는 JavaScript 인터프리터에 C 레벨 함수를 등록합니다. JS_DefineFunction() 함수가 **call** 인수에 지정된 C 레벨 함수를 등록하고 나면 이 함수를 **name** 인수에 지정된 이름으로 참조하여 JavaScript 스크립트에서 호출할 수 있습니다. **name** 인수는 대/소문자를 구분합니다.

일반적으로 이 함수는 Dreamweaver 시작 시 호출되는 MM_Init() 함수에서 호출됩니다.

인수

char ***name***, JSNative**call**, unsigned int nargs

- **name** 인수는 JavaScript에 공개되는 함수의 이름입니다.
- **call** 인수는 C 레벨 함수에 대한 포인터입니다. 이 함수는 readContentsOfFile과 동일한 인수를 받아야 하며 성공 또는 실패를 나타내는 JSBool을 반환해야 합니다.
- **nargs** 인수는 함수에 전달되는 인수의 수입니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

char *JS_ValueToString()

설명

이 함수는 JSVal 구조에서 함수 인수를 추출하고 가능한 경우 이를 문자열로 변환한 다음 변환된 값을 호출자에게 다시 전달합니다.

참고: 반환된 버퍼 포인터는 수정하지 말아야 합니다. 이 버퍼 포인터를 수정하면 JavaScript 인터프리터의 데이터 구조가 손상될 수 있습니다. 문자열을 변경하려면 문자를 다른 버퍼에 복사하고 JavaScript 문자열을 만듭니다.

인수

JSContext *cx, JSVal v, unsigned integer *pLength

- ***cx*** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- ***v*** 인수는 문자열을 추출할 JSVal 구조입니다.
- ***pLength*** 인수는 부호 없는 정수에 대한 포인터입니다. 이 함수는 *pLength를 문자열 길이(바이트)와 같게 설정합니다.

반환값

성공하면 null 종료 문자열을 가리키는 포인터를 반환하고, 실패하면 null 값을 가리키는 포인터를 반환합니다. 호출 루틴은 종료 시 이 문자열을 해제하지 말아야 합니다.

JSBool JS_ValueToInteger()

설명

이 함수는 JSVal 구조에서 함수 인수를 추출하고 가능한 경우 이를 정수로 변환한 다음 변환된 값을 호출자에게 다시 전달합니다.

인수

JSContext ***cx***, JSVal ***v***, long ***lp***

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **v** 인수는 정수를 추출할 JSVal 구조입니다.
- **lp** 인수는 4바이트 정수에 대한 포인터입니다. 이 함수는 변환된 값을 ***lp**에 저장합니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSBool JS_ValueToDouble()

설명

이 함수는 JSVal 구조에서 함수 인수를 추출하고 가능한 경우 이를 double 형식으로 변환한 다음 변환된 값을 호출자에게 다시 전달합니다.

인수

JSContext ***cx***, JSVal ***v***, double *dp

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **v** 인수는 double 형식을 추출할 JSVal 구조입니다.
- **dp** 인수는 8바이트 double에 대한 포인터입니다. 이 함수는 변환된 값을 ***dp**에 저장합니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSBool JS_ValueToBoolean()

설명

이 함수는 JSVal 구조에서 함수 인수를 추출하고 가능한 경우 이를 부울 값으로 변환한 다음 변환된 값을 호출자에게 다시 전달합니다.

인수

JSContext ***cx***, JSVal ***v***, JSBool *bp

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **v** 인수는 부울 값을 추출할 JSVal 구조입니다.
- **bp** 인수는 JSBool 부울 값에 대한 포인터입니다. 이 함수는 변환된 값을 ***bp**에 저장합니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSBool JS_ValueToObject()

설명

이 함수는 JSVal 구조에서 함수 인수를 추출하고 가능한 경우 이를 객체로 변환한 다음 변환된 값을 호출자에게 다시 전달합니다. 객체가 배열일 경우 JS_GetArrayLength() 및 JS_GetElement()를 사용하여 해당 내용을 읽습니다.

인수

JSContext ***cx***, JSVal ***v***, JSObject **op

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **v** 인수는 객체를 추출할 JSVal 구조입니다.
- **op** 인수는 JSObject 포인터에 대한 포인터입니다. 이 함수는 변환된 값을 ***op**에 저장합니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JS_ValueToUCString()

설명

이 함수는 JSVal 구조에서 함수 인수를 추출하고 가능한 경우 이를 문자열로 변환한 다음 변환된 값을 호출자에게 다시 전달합니다.

참고: 반환된 버퍼 포인터는 수정하지 말아야 합니다. 이 버퍼 포인터를 수정하면 JavaScript 인터프리터의 데이터 구조가 손상될 수 있습니다. 문자열을 변경하려면 문자를 다른 버퍼에 복사하고 JavaScript 문자열을 만듭니다.

인수

JSContext *cx, JSVal v, unsigned int *pLength

- ***cx*** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- ***v*** 인수는 문자열을 추출할 JSVal 구조입니다.
- ***pLength*** 인수는 부호 없는 정수에 대한 포인터입니다. 이 함수는 *pLength를 문자열 길이(바이트)와 같게 설정합니다.

반환값

성공하면 null 종료 UTF-8 문자열을 가리키는 포인터를 반환하고, 실패하면 null 값을 가리키는 포인터를 반환합니다. 호출 루틴은 종료 시 이 문자열을 해제하지 말아야 합니다.

JSBool JS_StringToValue()

설명

이 함수는 문자열 반환값을 JSVal 구조에 저장하고 새 JavaScript 문자열 객체를 할당합니다.

인수

JSContext *cx, JSVal *bytes, **size_t sz**, JSVal*vp

- ***cx*** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.

- ***bytes*** 인수는 JSVal 구조에 저장되는 문자열입니다. 이 문자열 데이터는 단지 복사되는 것이므로 호출자는 더 이상 필요하지 않을 경우 해당 문자열을 해제해야 합니다. 문자열 크기가 지정되지 않은 경우(***sz*** 인수 참조) 문자열은 null로 종료되어야 합니다.
- ***sz*** 인수는 문자열의 크기(바이트 단위)입니다. ***sz***가 0이면 null 종료 문자열의 길이가 자동으로 계산됩니다.
- *vp 인수는 문자열의 내용이 복사되는 대상 JSVal 구조에 대한 포인터입니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSBool JS_UCStringToValue()

설명
이 함수는 문자열 반환값을 JSVal 구조에 저장하고 새 JavaScript 문자열 객체를 할당합니다.

인수
JSContext ***cx*** , JSVal *bytes, size_t***sz**, JSVal ***vp***

- ***cx*** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- ***bytes*** 인수는 JSVal 구조에 저장되는 문자열입니다. 이 문자열 데이터는 단지 복사되는 것이므로 호출자는 더 이상 필요하지 않을 경우 해당 문자열을 해제해야 합니다. 문자열 크기가 지정되지 않은 경우(***sz*** 인수 참조) 문자열은 null로 종료되어야 합니다.
- ***sz*** 인수는 문자열의 크기(바이트 단위)입니다. ***sz***가 0이면 null 종료 문자열의 길이가 자동으로 계산됩니다.
- *vp 인수는 문자열의 내용이 복사되는 대상 JSVal 구조에 대한 포인터입니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

참고: JS_UCStringToValue() 메서드는 UTF-8 문자열을 반환한다는 점을 제외하면 모든 면에서 JSBool JS_StringToValue()와 유사합니다.

JSBool JS_DoubleToValue()

설명
이 함수는 부동 소수점 숫자 반환값을 JSVal 구조에 저장합니다.

인수
JSContext ***cx***, double ***dv***, JSVal ***vp***

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **dv** 인수는 8바이트 부동 소수점 숫자입니다.
- **vp** 인수는 double 형식의 내용이 복사되는 대상 JSVal 구조에 대한 포인터입니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSVal JS_BooleanToValue()

설명
이 함수는 부울 반환값을 JSVal 구조에 저장합니다.

인수
JSBool bv

- **bv** 인수는 부울 값입니다. JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

반환값
함수에 인수로 전달되는 부울 값이 포함된 JSVal 구조입니다.

JSVal JS_IntegerToValue()

설명
이 함수는 정수(long) 값을 JSVal 구조로 변환합니다.

인수
lv

lv 인수는 JSVal 구조로 변환할 정수(long) 값입니다.

반환값
함수에 인수로 전달되는 정수 값이 포함된 JSVal 구조입니다.

JSVal JS_ObjectToValue()

설명
이 함수는 객체 반환값을 JSVal에 저장합니다. 배열 객체를 만들려면 JS_ NewArrayObject()를 사용하고 배열 객체의 내용을 정의하려면 JS_SetElement()를 사용합니다.

인수
JSObject ***obj***

obj 인수는 JSVal 구조로 변환할 JSObject 객체에 대한 포인터입니다.

반환값
함수에 인수로 전달되는 객체가 포함된 JSVal 구조입니다.

char *JS_ObjectType()

설명
객체 참조의 경우 JS_ObjectType() 함수는 객체의 클래스 이름을 반환합니다. 예를 들어 객체가 DOM 객체이면 "Document"를 반환하고 객체가 문서의 노드이면 "Element"를 반환합니다. 배열 객체일 경우 "Array"를 반환합니다.

참고: 반환된 버퍼 포인터는 수정하지 말아야 합니다. 버퍼 포인터를 수정하면 JavaScript 인터프리터의 데이터 구조가 손상될 수 있습니다.

인수
JSObject *obj

일반적으로 이 인수는 JS_ValueToObject() 함수를 통해 전달되고 변환됩니다.

반환값
Null 종료 문자열에 대한 포인터를 반환합니다. 호출자는 종료 시 이 문자열을 해제하지 말아야 합니다.

JSObject *JS_NewArrayObject()

설명
이 함수는 JSVals의 배열을 포함하는 새 객체를 만듭니다.

인수
JSContext ***cx***, unsigned int ***length***, JSVal *v

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **length** 인수는 배열에 포함할 수 있는 요소의 수입니다.
- **v** 인수는 배열에 저장할 JSVals에 대한 선택적 포인터입니다. 반환값이 null이 아닐 경우 **v**는 **length** 요소가 포함된 배열입니다. 반환값이 null일 경우 배열 객체의 초기 내용은 정의되지 않고 JS_SetElement() 함수를 사용하여 설정할 수 있습니다.

반환값
새 배열 객체에 대한 포인터를 반환하거나 실패할 경우 null 값을 반환합니다.

long JS_GetArrayLength()

설명
배열 객체에 대한 포인터의 경우 이 함수는 배열의 요소 수를 가져옵니다.

인수
JSContext ***cx***, JSObject***obj**

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **obj** 인수는 배열 객체에 대한 포인터입니다.

반환값
배열 요소의 수를 반환하거나 실패할 경우 -1을 반환합니다.

JSBool JS_GetElement()

설명
이 함수는 배열 객체의 단일 요소를 읽습니다.

인수
JSContext ***cx***, JSObject ****obj***, unsigned int ***index***, JSVal *v

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.

- **obj** 인수는 배열 객체에 대한 포인터입니다.
- **index** 인수는 배열 내에서의 정수 인덱스입니다. 첫 번째 요소의 index는 0이고 마지막 요소의 index는 length - 1입니다.
- **v** 인수는 배열에 있는 JSVal 구조의 내용을 복사할 jsval에 대한 포인터입니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSBool JS_SetElement()

설명
이 함수는 배열 객체의 단일 요소를 기록합니다.

인수
JSContext *cx, JSObject *obj, unsigned int index, JSVal *v

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **obj** 인수는 배열 객체에 대한 포인터입니다.
- **index** 인수는 배열 내에서의 정수 인덱스입니다. 첫 번째 요소의 index는 0이고 마지막 요소의 index는 length - 1입니다.
- **v** 인수는 배열의 JSVal에 복사할 내용이 포함된 JSVal 구조에 대한 포인터입니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSBool JS_ExecuteScript()

설명
이 함수는 JavaScript 문자열을 컴파일하고 실행합니다. 스크립트에서 반환값을 생성하면 ***rval**에 반환됩니다.

인수
JSContext ***cx***, JSObject ****obj***, **char *script**, **unsigned int*sz***, JSVal *rval

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **obj** 인수는 해당 컨텍스트에서 스크립트가 실행되는 객체에 대한 포인터입니다. 스크립트가 실행되는 동안에는 this 키워드가 이 객체를 나타냅니다. 일반적으로 이것은 JavaScript 함수에 전달되는 JSObject 포인터입니다.
- **script** 인수는 JavaScript 코드가 포함된 문자열입니다. 문자열 크기가 지정되지 않은 경우(**sz** 인수 참조) 문자열은 null로 종료되어야 합니다.
- **sz** 인수는 문자열의 크기(바이트 단위)입니다. **sz**가 0이면 null 종료 문자열의 길이가 자동으로 계산됩니다.
- **rval** 인수는 단일 JSVal 구조에 대한 포인터입니다. 함수의 반환값은 ***rval**에 저장됩니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

JSBool JS_ReportError()

설명

이 함수는 스크립트 오류가 발생한 원인을 보고합니다. 스크립트 오류가 발생할 경우 JS_FALSE 값을 반환하기 전에 이 함수를 호출하여 인수 개수가 잘못된 경우와 같이 스크립트가 실패한 원인에 대한 정보를 사용자에게 제공합니다.

인수

JSContext ***cx***, char ***error***, size_t sz

- **cx** 인수는 JavaScript 함수에 전달되는 불투명 JSContext 포인터입니다.
- **error** 인수는 오류 메시지가 포함된 문자열입니다. 이 문자열은 단지 복사되는 것이므로 호출자는 더 이상 필요하지 않을 경우 해당 문자열을 해제해야 합니다. 문자열 크기가 지정되지 않은 경우(**sz** 인수 참조) 문자열은 null로 종료되어야 합니다.
- **sz** 인수는 문자열의 크기(바이트 단위)입니다. **sz**가 0이면 null 종료 문자열의 길이가 자동으로 계산됩니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

파일 액세스 및 다중 사용자 구성 API

C 레벨 확장성을 통해 파일 시스템에 액세스하려면 항상 파일 액세스 및 다중 사용자 구성 API를 사용하는 것이 좋습니다. 이 함수는 구성 파일 이외의 파일을 사용할 때는 지정된 파일이나 폴더에 액세스합니다.

Dreamweaver는 Windows XP, Windows 2000 및 Mac OS X 운영 체제에서 다중 사용자 구성을 지원합니다.

Dreamweaver는 일반적으로 Windows의 C:\Program Files와 같은 제한된 폴더에 설치됩니다. 따라서 Administrator 권한이 있는 사용자만 Dreamweaver Configuration 폴더에서 변경 작업을 수행할 수 있습니다. 다중 사용자 운영 체제에서 각 사용자별 구성을 만들고 관리할 수 있도록 Dreamweaver에서는 각 사용자별로 별도의 Configuration 폴더를 만듭니다. Dreamweaver나 JavaScript Extension에서 Dreamweaver Configuration 폴더에 대해 쓰기 작업을 수행할 때마다 해당 작업은 자동으로 이 폴더 대신 사용자의 Configuration 폴더에 대해 수행됩니다. 따라서 각 사용자는 다른 사용자의 사용자 정의 구성에 영향을 주지 않고 Dreamweaver 구성 설정을 사용자 정의할 수 있습니다.

Dreamweaver에서는 사용자가 읽기 및 쓰기 권한을 갖고 있는 위치에 해당 사용자의 Configuration 폴더가 만들어집니다. 사용자의 Configuration 폴더 위치는 사용자의 플랫폼에 따라 다릅니다.

Windows 2000 및 Windows XP 플랫폼의 경우 Configuration 폴더의 위치는 다음과 같습니다.

```
<drive>:\Documents and Settings\<username>\Application Data\Adobe\\
Dreamweaver CS5\Configuration
```

참고: Windows XP에서는 이 폴더가 숨겨진 폴더 내에 있을 수 있습니다.

Mac OS X 플랫폼의 경우 Configuration 폴더의 위치는 다음과 같습니다.

```
<drive>:Users:<username>:Library:Application Support:Adobe:Dreamweaver CS5:Configuration
```

대부분의 경우 JavaScript Extension에서는 Configuration 폴더에 대해 파일 열기 및 쓰기 작업을 수행합니다. JavaScript Extension에서는 DWFile 또는 MMNotes를 사용하거나 dreamweaver.getDocumentDOM() 함수에 URL을 전달하여 파일 시스템에 액세스할 수 있습니다. Extension에서 Configuration 폴더의 파일 시스템에 액세스할 때는 일반적으로 dw.getConfigurationPath() 함수를 사용하고 해당 파일 이름을 추가하거나, 열려 있는 문서의 dom.URL 속성에 액세스하고 해당 파일 이름을 추가하여 경로를 가져옵니다. 또한 Extension에서는 dom.URL에 액세스하고 해당 파일 이름을 제거하여 경로를 가져올 수도 있습니다. 문서가 사용자의 Configuration 폴더에 있는 경우에도 dw.getConfigurationPath() 함수와 dom.URL 속성은 항상 Dreamweaver Configuration 폴더의 URL을 반환합니다.

JavaScript Extension에서 Dreamweaver Configuration 폴더의 파일을 열 때마다 Dreamweaver에서는 파일 액세스를 가로채어 사용자의 Configuration 폴더를 먼저 확인합니다. JavaScript Extension에서 DWFile 또는 MMNotes를 통해 디스크의 Dreamweaver Configuration 폴더에 데이터를 저장할 경우 Dreamweaver에서는 호출을 가로챈 다음 사용자의 Configuration 폴더로 리디렉션합니다.

예를 들어 Windows 2000 또는 Windows XP에서 사용자가 file:///C|/Program Files/Adobe/Adobe Dreamweaver CS5/Configuration/Objects/Common/Table.htm을 요구하는 경우, Dreamweaver에서는 C:\Documents and Settings**username**\adobe\Dreamweaver CS5\Configuration\Objects\Common 폴더에서 Table.htm 파일을 찾고 이 파일이 존재하면 이 파일을 대신 사용합니다.

C 레벨 Extension 또는 공유 라이브러리에서는 Dreamweaver Configuration 폴더에 대한 읽기 및 쓰기 작업을 수행할 때 파일 액세스 및 다중 사용자 구성 API를 사용해야 합니다. 파일 액세스 및 다중 사용자 구성 API를 사용하면 Dreamweaver에서 사용자의 Configuration 폴더에 대한 읽기 및 쓰기 작업을 수행할 수 있으며 액세스 권한이 충분하지 않아 파일 작업이 실패하는 것을 방지할 수 있습니다. C 레벨 Extension에서 액세스하는 Dreamweaver Configuration 폴더의 파일이 DWFile, MMNotes 또는 DOM 조작을 사용하여 JavaScript를 통해 만들어진 경우 파일 액세스 및 다중 사용자 구성 API를 사용합니다. 이러한 파일은 사용자의 Configuration 폴더에 있을 수 있습니다.

참고: 대부분의 JavaScript Extension은 변경하지 않아도 사용자의 Configuration 폴더에 대해 쓰기 작업을 수행할 수 있습니다. 파일 액세스 및 다중 사용자 구성 API 함수를 사용하기 위해 업데이트해야 하는 것은 Configuration 폴더에 대해 쓰기 작업을 수행하는 C 공유 라이브러리뿐입니다.

Dreamweaver Configuration 폴더에서 파일을 삭제하면 마스크 파일에 항목이 추가됩니다. 이 항목은 Configuration 폴더의 파일 중 사용자 인터페이스에 표시되지 않아야 하는 파일을 알려 주기 위한 것입니다. 마스크 처리된 파일 또는 폴더는 실제로 존재하더라도 Dreamweaver에는 존재하지 않는 것으로 표시됩니다.

예를 들어 [코드 단편] 패널에 있는 휴지통 아이콘을 사용하여 javascript라는 코드 단편 폴더와 onepixelborder.csn이라는 파일을 삭제할 경우 사용자의 Configuration 폴더에 mm_deleted_files.xml이라는 파일이 만들어집니다. 이 파일의 내용은 다음과 같습니다.

```
<?xml version = "1.0" encoding="utf-8" ?>
    <deleteditems>
    <item name="snippets/javascript/" />
    <item name="snippets/html/onepixelborder.csn" />
    </deleteditems>
```

Dreamweaver에서는 [코드 단편] 패널을 표시할 때 사용자의 Configuration/Snippets 폴더에 있는 모든 파일을 읽습니다. 또한 Dreamweaver Configuration/Snippets 폴더에 있는 모든 파일도 읽습니다. 이때 Configuration/Snippets/javascript 폴더와 Configuration/Snippets/html/onepixelborder.csn 파일은 제외됩니다. 그런 다음 결과 파일 목록을 [코드 단편] 패널 목록에 추가합니다.

C 레벨 Extension에서 file:///c|Program Files/Adobe/Adobe Dreamweaver CS5/Configuration/Snippets/javascript/onepixelborder.csn URL에 대해 MM_ConfigFileExists() 함수를 호출하면 false 값이 반환됩니다. 마찬가지로 JavaScript Extension에서 dw.getDocumentDom(file:///c|Program Files/Adobe/Adobe Dreamweaver CS5/Configuration/Snippets/javascript/onepixelborder.csn)을 호출하면 null 값이 반환됩니다.

객체, 새 대화 상자의 미리 준비된 내용 등이 Dreamweaver의 사용자 인터페이스에 표시되지 않도록 하려면 mm_deleted_files.xml 파일을 편집합니다. MM_DeleteConfigfile() 함수를 호출하여 mm_deleted_files.xml 파일에 파일 경로를 추가할 수 있습니다.

JS_Object MM_GetConfigFolderList()

지원 버전
Dreamweaver MX

설명

이 함수는 지정된 폴더 내의 파일, 폴더 또는 둘 모두가 들어 있는 목록을 가져옵니다. 구성 폴더를 지정하면 이 함수는 사용자의 Configuration 폴더와 Dreamweaver Configuration 폴더에 있는 폴더 목록을 가져옵니다. 이 목록은 mm_deleted_files.xml 파일에 따라 필터링됩니다.

인수

char *fileURL, char *constraints

- **char *fileUrl** 인수는 내용 목록을 얻으려는 폴더의 이름을 나타내는 문자열에 대한 포인터입니다. 이 문자열은 file:// URL 형식이어야 합니다. 이 함수에서는 file:// URL 문자열에 유효한 와일드카드 문자인 별표(*)와 물음표(?)를 사용할 수 있습니다. 하나 이상의 임의 문자를 대신하려면 별표(*)를 사용하고 하나의 임의 문자를 대신하려면 물음표(?)를 사용합니다.
- **char *contstraints** 인수의 값은 files, directories 또는 null일 수 있습니다. null이 지정되면 MM_GetConfigFolderList() 함수는 파일과 폴더를 반환합니다.

반환값

JSObject는 사용자의 Configuration 폴더나 Dreamweaver Configuration 폴더에 있는 파일 또는 폴더의 목록이 들어 있는 배열을 반환합니다. 이 목록은 mm_deleted_files.xml 파일에 따라 필터링됩니다.

예제

```
JSObject *jsobj_array;
jsobj_array = MM_GetConfigFolderList("file:///¬
    c|/Program Files/Adobe/Adobe Dreamweaver CS3/Configuration", "directories" );
```

JSBool MM_ConfigFileExists()

지원 버전

Dreamweaver MX

설명

이 함수는 지정된 파일이 존재하는지 확인합니다. 파일이 구성 폴더에 존재하면 이 함수는 사용자의 Configuration 폴더나 Dreamweaver Configuration 폴더에서 해당 파일을 검색합니다. 이 함수는 또한 파일 이름이 mm_deleted_files.xml 파일에 나열되어 있는지도 확인합니다. 이 파일에 이름이 들어 있으면 이 함수는 false 값을 반환합니다.

인수

char *fileUrl

char *fileUrl 인수는 원하는 파일의 이름을 나타내는 문자열에 대한 포인터로, file:// URL의 형식으로 제공됩니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

예제

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
Configuration/Extensions.txt";
int fileno = 0;
if(MM_ConfigFileExists(dwConfig))
{
    fileno = MM_OpenConfigFile(dwConfig, "read");
}
```

int MM_OpenConfigFile()

지원 버전
Dreamweaver MX

설명
이 함수는 파일을 열고 운영 체제 파일 핸들을 반환합니다. 운영 체제 파일 핸들은 시스템 파일 함수를 호출할 때 사용할 수 있습니다. 시스템 _close 함수를 호출할 때는 파일 핸들을 닫아야 합니다.

파일이 구성 파일인 경우 이 함수는 사용자의 Configuration 폴더나 Dreamweaver Configuration 폴더에서 해당 파일을 찾습니다. 쓰기 작업을 위해 구성 파일을 열면 이 함수는 파일이 Dreamweaver Configuration 폴더에 있더라도 사용자의 Configuration 폴더에 해당 파일을 만듭니다.

참고: 파일에 쓰기 전에 파일을 읽으려면 "read" 모드로 파일을 엽니다. 파일에 쓰려고 할 때는 read 핸들을 닫고 "write" 또는 "append" 모드로 파일을 다시 엽니다.

인수
char *fileURL, char *mode

- **char *fileURL** 인수는 열려고 하는 파일의 이름을 나타내는 문자열에 대한 포인터로, file:// URL의 형식으로 제공됩니다. 이 인수에 Dreamweaver Configuration 폴더 내의 경로를 지정하면 MM_OpenConfigFile() 함수는 파일을 열기 전에 해당 경로를 해석합니다.
- **char *mode** 인수는 파일을 여는 방법을 지정하는 문자열을 나타냅니다. null, "read", "write" 또는 "append" 모드를 지정할 수 있습니다. "write"를 지정했지만 해당 파일이 존재하지 않으면 MM_OpenconfigFile() 함수는 해당 파일을 만듭니다. "write"를 지정하면 MM_OpenConfigFile() 함수는 파일을 배타적 공유 상태로 엽니다. read를 지정하면 MM_OpenConfigFile() 함수는 파일을 비배타적 공유 상태로 엽니다.

  "write" 모드로 파일을 열면 새 데이터를 쓰기 전에 해당 파일의 기존 데이터가 잘립니다. "append" 모드로 파일을 열면 사용자가 쓰는 모든 데이터가 해당 파일의 끝에 추가됩니다.

반환값
이 파일에 대한 운영 체제 파일 핸들을 나타내는 정수를 반환합니다. 파일을 찾을 수 없거나 파일이 존재하지 않으면 -1을 반환합니다.

예제

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
    Configuration/Extensions.txt";
int = fileno;
if(MM_ConfigFileExists(dwConfig))
{
    fileno = MM_OpenConfigFile(dwConfig, "read");
}
```

JSBool MM_GetConfigFileAttributes()

지원 버전
Dreamweaver MX

설명
이 함수는 파일을 찾고 파일의 속성을 반환합니다. 값이 필요하지 않으면 **fileURL**을 제외한 모든 인수를 null로 설정할 수 있습니다.

인수
char *fileURL, unsigned long *attrs, unsigned long *filesize, unsigned long *modtime, unsigned long *createtime

- **char *fileURL** 인수는 속성을 가져오려는 파일의 이름을 나타내는 문자열에 대한 포인터입니다. 이 인수는 file:// URL 형식으로 지정해야 합니다. **fileURL**에 Dreamweaver Configuration 폴더 내의 경로를 지정하면 MM_GetConfigFileAttributes() 함수는 파일을 열기 전에 해당 경로를 해석합니다.
- **unsigned long *attrs** 인수는 반환되는 속성 비트가 포함되는 정수의 주소입니다. 사용할 수 있는 속성은 337페이지의 “JSBool MM_SetConfigFileAttributes()”를 참조하십시오.
- **unsigned long *filesize** 인수는 이 함수가 파일 크기(바이트)를 반환하는 정수의 주소입니다.
- **unsigned long *modtime** 인수는 이 함수가 파일을 마지막으로 수정한 시간을 반환하는 정수의 주소입니다. 이 시간은 운영 체제 시간 값으로 지정합니다. 운영 체제 시간 값에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**에서 DWfile.getModificationDate()를 참조하십시오.
- **unsigned long *createtime** 인수는 이 함수가 파일을 만든 시간을 반환하는 정수의 주소입니다. 이 시간은 운영 체제 시간 값으로 지정합니다. 운영 체제 시간 값에 대한 자세한 내용은 **Dreamweaver API 참조 설명서**에서 DWfile.getCreationDate()를 참조하십시오.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다. 파일이 없거나 속성을 가져올 때 오류가 발생하면 JS_FALSE가 반환됩니다.

예제

```
char dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
    Configuration/Extensions.txt";
unsigned long attrs;
unsigned long filesize;
unsigned long modtime;
unsigned long createtime;
MM_GetConfigAttributes(dwConfig, &attrs, &filesize, &modtime, &createtime);
```

JSBool MM_SetConfigFileAttributes()

지원 버전
Dreamweaver MX

설명
이 함수는 파일에 대해 사용자가 지정하는 속성(현재 속성과 다른 경우)을 설정합니다.

지정한 파일 URL이 Dreamweaver Configuration 폴더 내의 URL이면 이 함수는 속성을 설정하기 전에 먼저 해당 파일을 사용자의 Configuration 폴더에 복사합니다. 지정된 속성이 현재 파일 속성과 동일하면 파일이 복사되지 않습니다.

인수
char *fileURL, unsigned long attrs

- **char *fileURL** 인수는 속성을 설정하려는 파일의 이름을 나타내는 문자열에 대한 포인터로, file:// URL 형식으로 지정됩니다.
- **unsigned long attrs** 인수는 파일에 대해 설정할 속성 비트를 지정합니다. 다음 상수에 논리 OR를 사용하여 속성을 설정할 수 있습니다.

```
MM_FILEATTR_NORMAL
MM_FILEATTR_RDONLY
MM_FILEATTR_HIDDEN
MM_FILEATTR_SYSTEM
MM_FILEATTR_SUBDIR
```

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다. 파일이 없거나 삭제된 것으로 표시되어 있으면 JS_FALSE를 반환합니다.

예제

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3/
    Configuration/Extensions.txt";
unsigned long attrs;
attrs = (MM_FILEATTR_NORMAL | MM_FILEATTR_RDONLY);
int fileno = 0;
if(MM_SetConfigFileAttrs(dwConfig, attrs))
{
    fileno = MM_OpenConfigFile(dwConfig);
}
```

JSBool MM_CreateConfigFolder()

지원 버전

Dreamweaver MX

설명

이 함수는 지정된 위치에 폴더를 만듭니다.

fileURL 인수에 Dreamweaver Configuration 폴더 내의 폴더를 지정하면 이 함수는 사용자의 Configuration 폴더 내에 해당 폴더를 만듭니다. **fileURL**에 Dreamweaver Configuration 폴더 내의 폴더를 지정하지 않으면 이 함수는 지정된 폴더를 만들며 이때 경로의 상위 폴더가 아직 없으면 해당 폴더도 모두 만듭니다.

인수

char *fileURL

- **char *fileURL** 인수는 만들려는 구성 폴더의 이름을 나타내는 file:// URL 문자열에 대한 포인터입니다.

반환값

부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

예제

```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3
    /Configuration/Extensions.txt";
MM_CreateConfigFolder(dwConfig);
```

JSBool MM_RemoveConfigFolder()

지원 버전

Dreamweaver MX

설명
이 함수는 지정된 폴더와 해당 폴더의 파일 및 하위 폴더를 제거합니다. 지정된 폴더가 Dreamweaver Configuration 폴더의 하위 폴더인 경우에는 mm_deleted_files.xml 파일에서 해당 폴더를 삭제된 것으로 표시합니다.

인수
char *fileURL

- **char *fileURL** 인수는 제거할 폴더의 이름을 나타내는 문자열에 대한 포인터로, file:// URL 형식으로 지정됩니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

예제
```
char *dwConfig = "file:///c|/Program Files/Adobe/Adobe Dreamweaver CS3
    /Configuration/Objects";
MM_RemoveConfigFolder(dwConfig);
```

JSBool MM_DeleteConfigFile()

지원 버전
Dreamweaver MX

설명
이 함수는 지정된 파일이 존재하면 해당 파일을 삭제합니다. 지정된 파일이 Dreamweaver Configuration 폴더에 존재하면 mm_deleted_files.xml 파일에서 해당 파일을 삭제된 것으로 표시합니다.

fileURL 인수에 Dreamweaver Configuration 폴더 내의 폴더를 지정하지 않으면 이 함수는 지정된 파일을 삭제합니다.

인수
char *fileURL

- **char *fileURL** 인수는 제거할 구성 폴더의 이름을 나타내는 문자열에 대한 포인터로, file:// URL 형식으로 지정됩니다.

반환값
부울 값, JS_TRUE는 성공을 나타내고 JS_FALSE는 실패를 나타냅니다.

예제
```
char dwConfig = "file:///c:|Program Files/Adobe/Adobe Dreamweaver CS3
    /Configuration/Objects/insertbar.xml";
MM_DeleteConfigFile(dwConfig);
```

JavaScript에서 C 함수 호출

Dreamweaver에서 C 레벨 확장성이 작동하는 방식과 특정 데이터 유형 및 함수에 대한 종속성을 이해한 후에는 라이브러리 작성 및 함수 호출 방법을 익히는 것이 좋습니다.

다음 예제에는 Dreamweaver 응용 프로그램 폴더/Tutorial_assets/Extending 폴더에 있는 다음 다섯 개의 파일이 필요합니다. 이러한 파일은 Macintosh 플랫폼과 Windows 플랫폼용으로 제공됩니다.

- mm_jsapi.h 헤더 파일: 325페이지의 "C 레벨 확장성 및 JavaScript 인터프리터"에 설명된 데이터 유형과 함수에 대한 정의가 포함되어 있습니다.
- mm_jsapi_environment.h 파일: MM_Environment.h 구조를 정의합니다.
- MMInfo.h 파일: 디자인 노트 API에 액세스하는 데 사용합니다.
- Sample.c 예제 파일: computeSum() 함수를 정의합니다.
- Sample.mak make 파일을 사용하여 Microsoft Visual C++로 Sample.c 소스 파일을 DLL로 빌드할 수 있습니다. Sample.mcp는 Metrowerks CodeWarrior를 사용하여 Mach-O 번들을 빌드하는 파일이고, Sample.xcode는 Apple Xcode에 해당하는 파일입니다. 다른 도구를 사용할 경우 메이크파일을 직접 만들 수 있습니다.

Windows에서 VS.Net 2003을 사용하여 DLL 빌드

1. [파일] > [열기]를 선택한 다음 [파일 형식]을 [모든 파일(*.*)]로 설정한 상태에서 Sample.mak를 선택합니다. VS.Net 2003에서는 MAK 파일을 바로 열 수 없으므로 프로젝트를 새 형식으로 변환할지 묻는 메시지가 표시됩니다.
2. [빌드] > [솔루션 다시 빌드]를 선택합니다.

빌드 작업이 완료되면 Sample.dll이라는 파일이 Sample.mak가 포함된 폴더 또는 이 폴더의 하위 폴더 중 하나에 나타납니다.

Windows에서 Microsoft Visual C++를 사용하여 DLL 빌드

1. Microsoft Visual C++에서 [파일] > [작업 영역 열기]를 선택하고 Sample.mak를 선택합니다.
2. [빌드] > [모두 다시 빌드]를 선택합니다.

빌드 작업이 완료되면 Sample.dll이라는 파일이 Sample.mak가 포함된 폴더 또는 이 폴더의 하위 폴더 중 하나에 나타납니다.

Macintosh에서 Metrowerks CodeWarrior 9 이상을 사용하여 공유 라이브러리 빌드

1. Sample.mcp를 엽니다.
2. [프로젝트] > [만들기]를 선택하여 프로젝트를 빌드하고 Mach-O 번들을 생성합니다.

빌드 작업이 완료되면 Sample.bundle이라는 파일이 Sample.mcp가 포함된 폴더에 나타납니다.

참고: 생성된 Mach-O 번들은 Dreamweaver 8 이상에서만 사용할 수 있습니다. 그 이전 버전의 Dreamweaver에서는 이 번들을 인식하지 못합니다.

Macintosh에서 Apple Xcode 1.5 이상을 사용하여 공유 라이브러리 빌드

1. Sample.xcode를 엽니다.
2. [빌드] > [빌드]를 선택하여 프로젝트를 빌드하고 Mach-O 번들을 생성합니다.

빌드 작업이 완료되면 Sample.bundle이라는 파일이 Sample.xcode 파일 옆에 있는 build 폴더에 나타납니다.

참고: 생성된 Mach-O 번들은 Dreamweaver 8 이상에서만 사용할 수 있습니다. 그 이전 버전의 Dreamweaver에서는 이 번들을 인식하지 못합니다.

Insert Horizontal Rule 객체에서 computeSum() 함수 호출

1. Dreamweaver 응용 프로그램 폴더 내의 Configuration 폴더에 JSExtensions라는 폴더를 만듭니다.
2. Sample.dll(Windows) 또는 Sample.bundle(Macintosh)을 JSExtensions 폴더에 복사합니다.
3. 텍스트 편집기에서 Configuration/Objects/Common 폴더에 있는 HR.htm 파일을 엽니다.
4. 다음 예제와 같이 alert(Sample.computeSum(2,2)); 행을 objectTag() 함수에 추가합니다.

```
function objectTag() {
    // Return the html tag that should be inserted
    alert(Sample.computeSum(2,2));
    return "<HR>";
}
```

5 파일을 저장하고 Dreamweaver를 다시 시작합니다.

computeSum() 함수를 실행하려면 [삽입] > [HTML] > [수평선]을 선택합니다.

2 더하기 2의 계산 결과인 숫자 4가 표시된 대화 상자가 나타납니다.

24장: Shared 폴더

Shared 폴더는 다른 모든 Extension에서 공통적으로 사용하는 유틸리티 함수, 클래스 및 이미지를 모아 놓은 중앙 저장소입니다. 모든 Extension은 Shared 폴더의 하위 폴더에 있는 파일을 참조할 수 있습니다. Adobe Dreamweaver에서 이미 제공하는 유틸리티에 사용자 정의 공용 유틸리티를 추가할 수 있습니다. Windows XP, Windows 2000 및 Mac OS X 사용자를 위해 설치된 다중 사용자 Configuration 폴더에도 개별 사용자 정의에 사용할 수 있는 Shared 폴더가 포함되어 있습니다. 예를 들어 Adobe Exchange에서 Extension을 설치할 때 새 Extension이 Dreamweaver 응용 프로그램의 Configuration/Shared 폴더가 아니라 사용자의 Configuration/Shared 폴더에 내용을 추가할 수도 있습니다. 다중 사용자 컴퓨터의 Dreamweaver Configuration 폴더에 대한 자세한 내용은 72페이지의 "다중 사용자 Configuration 폴더"를 참조하십시오.

Shared 폴더 내용

Shared 폴더에는 사용자의 폴더 시스템 탐색, 트리 컨트롤 삽입, 편집 가능한 격자 작성 및 기타 기능에 대한 함수를 비롯하여 다양한 Extension에서 공유되는 파일이 들어 있는 하위 폴더가 있습니다.

참고: Shared 폴더에 있는 JavaScript 파일은 코드 내에 주석을 포함하고 있습니다. 이 주석을 통하여 JavaScript 파일에 들어 있는 함수에 대해 자세히 알 수 있습니다.

Shared 폴더에서 JavaScript 파일을 찾는 것뿐만 아니라, 이러한 JavaScript 파일이 들어 있는 HTML 파일을 Configuration 폴더에서 찾아보면 JavaScript 파일이 어떻게 사용되는지 알 수 있습니다.

일반적으로 Common 및 MM 폴더에 있는 함수와 리소스를 사용하거나 새 Extension에 사용하기 위해 Common 폴더에 리소스를 추가합니다. 항상 Shared/Common/Scripts 폴더에서 먼저 유틸리티 및 함수를 검색하는 것이 좋습니다. 이러한 함수 및 유틸리티는 가장 최근에 작성된 것이며 Shared 폴더에 대한 외형적인 인터페이스를 구성합니다. Shared 폴더가 아닌 다른 폴더의 파일은 오래전에 만들어진 것일 수도 있으므로 일은 사용하지 않는 것이 좋습니다.

특히 Shared 폴더에는 다음과 같은 유용한 폴더가 들어 있습니다.

Common 폴더

Common 폴더에는 타사의 Extension에 사용할 수 있는 공유 스크립트 및 클래스가 들어 있습니다.

파일	설명
CodeBehindMgr.js	이 파일에는 코드 숨김 문서를 작성하는 데 사용하는 함수가 들어 있습니다. 코드 숨김 문서는 UI(사용자 인터페이스) 논리에 사용하는 코드와 UI 디자인에 사용하는 코드를 분리하는 별도의 페이지를 만들 수 있도록 해 줍니다. 이 파일에 정의된 JSCodeBehindMgr의 메서드는 새 코드 숨김 문서를 만들고 문서 디자인을 위한 링크를 관리할 수 있습니다.
ColumnValueNodeClass.js	이 파일에는 데이터베이스 열을 값에 매핑하는 함수가 들어 있습니다. 이 파일에 정의된 ColumnValueNode의 메서드를 사용하면 데이터베이스 열의 다양한 값과 속성을 가져오거나 설정할 수 있습니다. Dreamweaver에서는 편집 작업 객체(레코드 삽입 및 업데이트 객체)를 적용하고 관리하는 경우와 SQLStatement 클래스로 작업하는 경우에 이 저장 클래스를 사용합니다.
CompilerClass.js	이 파일에는 CompilerASPNetCSharp 및 CompilerASPNVBNet에 의해 사용되는 기본 클래스에 대한 함수가 들어 있지만 다른 컴파일러를 지원하도록 이를 확장할 수도 있습니다.
DataSourceClass.js	이 파일에는 280페이지의 "findDynamicSources()"의 반환 구조를 정의하는 함수가 들어 있습니다.

파일	설명
DBTreeControlClass.js	이 파일에는 데이터베이스 트리 컨트롤을 만드는 함수가 들어 있습니다. 이 클래스는 데이터베이스 트리 컨트롤을 만들고 이 트리 컨트롤과 상호 작용하는 데 사용됩니다. 고급 레코드세트 서버 비헤이비어에 있는 컨트롤과 같은 데이터베이스 트리 컨트롤을 만들려면 HTML 파일에 type="mmdatabasetree"로 지정된 특별한 <select> 목록을 만듭니다. <select> 목록 이름을 클래스 생성자에 전달하여 해당 HTML 컨트롤에 CBTreeControl 클래스를 첨부합니다. 그런 다음 DBTreeControl 함수를 사용하여 해당 컨트롤을 조작합니다.
dotNetUtils.js	이 파일에는 객체 속성 관리자 작업과 변환되는 ASP.NET 양식 컨트롤에 대한 서버 비헤이비어 작업을 쉽게 해 주는 함수가 들어 있습니다.
dwscripts.js	모든 Dreamweaver Extension에 유용한 함수가 들어 있는 기본 파일로, 문자열, 파일, 디자인 노트 등을 사용하는 작업에 필요한 함수가 들어 있습니다.
dwscriptsExtData.js	이 파일은 dwscripts.js 파일의 Extension으로, 서버 비헤이비어, 특히 서버 비헤이비어 EDML 파일을 사용하는 작업을 쉽게 해 줍니다. 이 파일은 Dreamweaver에서 서버 비헤이비어를 구현할 때 광범위하게 사용됩니다.
dwscriptsServer.js	이 파일은 dwscripts.js 파일의 Extension으로, 서버 모델에만 사용되는 함수가 들어 있습니다. 이러한 함수 중 다수는 서버 비헤이비어를 사용하여 작업할 때 사용됩니다.
GridControlClass.js	이 클래스는 편집 가능한 격자를 만들고 조작하는 데 사용됩니다. HTML에 특별한 선택 목록을 추가하고 JavaScript에서 해당 목록에 이 클래스를 첨부하여 격자를 조작합니다.
ImageButtonClass.js	이 클래스를 사용하면 버튼의 Pressed/Mouse-over-while-pressed/Mouse-over/Disabled-while-pressed 모양을 더욱 쉽게 제어할 수 있습니다.
ListControlClass.js	이 파일에는 목록 컨트롤이라고도 하는 <select> 태그를 관리하는 함수가 들어 있습니다. 이 파일에 있는 ListControl 객체의 메서드는 SELECT 컨트롤의 값을 가져오고 설정하고 변경합니다.
PageSettingsASPNet.js	이 파일에는 ASP.NET 문서의 속성을 설정하는 함수가 들어 있습니다.
RadioGroupClass.js	이 파일에는 라디오 버튼 그룹을 정의하고 관리하는 함수가 들어 있습니다. 이 파일에 있는 RadioGroup 객체의 메서드는 라디오 버튼 그룹의 값 및 비헤이비어를 설정하고 가져옵니다. HTML에서 이 클래스를 라디오 버튼에 첨부하여 버튼의 비헤이비어를 제어할 수 있습니다.
SBDatabaseCallClass.js	ServerBehavior 클래스의 하위 클래스입니다. 이 클래스는 예를 들어 저장 프로시저를 호출하거나 SQL을 사용하여 레코드세트를 반환하는 등의 데이터베이스 호출 작업과 관련된 기능을 제공합니다. 이 클래스는 추상 기본 클래스이므로 단독으로 작성하거나 사용할 수 없습니다. 이 클래스를 사용하려면 SBDatabaseCall()의 하위 클래스를 만들어 자리 표시자 함수를 구현해야 합니다. Dreamweaver에서는 이 클래스를 사용하여 레코드세트 및 저장 프로시저 서버 비헤이비어를 구현합니다.
ServerBehaviorClass.js	이 파일에는 서버 비헤이비어에 대한 정보를 Dreamweaver에 알려 주는 함수가 들어 있습니다. 사용자 고유의 서버 비헤이비어를 구현할 때 이 클래스의 하위 클래스를 만들 수 있습니다.
ServerSettingsASPNet.js	이 파일에는 ASP.NET 서버의 속성을 저장하는 함수가 들어 있습니다.
SQLStatementClass.js	이 파일에는 SELECT, INSERT, UPDATE 및 DELETE 등의 SQL 문과 저장 프로시저 명령문을 만들고 편집할 수 있는 함수가 들어 있습니다.
tagDialogsCmn.js	이 파일에는 사용자 정의 태그 대화 상자를 개발하는 데 유용한 함수가 들어 있습니다. 이 파일에 정의된 tagDialog 객체의 메서드는 특정 태그의 속성 및 값을 수정합니다.
TagEditClass.js	이 파일에는 현재 페이지의 DOM을 변경하지 않고 태그를 편집하는 함수가 들어 있습니다. 이 파일에 정의된 TagEdit 객체의 메서드는 태그의 값, 속성 및 자식을 가져오고 설정합니다.
TreeControlClass.js	이 파일에는 Dreamweaver 내에서 트리 컨트롤을 관리하는 함수가 들어 있습니다. 이 파일에 정의된 TreeControl 객체의 메서드는 트리의 값을 가져오고 설정하고 정렬합니다. HTML에서 이 클래스를 특수한 MM:TREECONTROL 태그에 첨부하여 트리 컨트롤 기능을 관리할 수 있습니다.
XMLPropSheetClass.js	이 클래스에는 XML 속성 시트의 위치와 값을 관리하는 함수가 들어 있습니다.

MM 폴더

MM 폴더에는 Dreamweaver와 함께 제공되는 Extension에 사용되는 공유 스크립트, 이미지 및 클래스가 들어 있습니다. 여기에는 내비게이션 막대 구성, 미리 로드 호출 지정 및 단축키 정의에 사용하는 스크립트가 포함됩니다.

Scripts 하위 폴더

Scripts 하위 폴더에는 다음 유틸리티 함수가 들어 있습니다.

파일	설명
CFCutilities.js	이 파일에는 Adobe ColdFusion 구성 요소와 관련된 유틸리티 함수가 들어 있습니다. 이러한 함수는 특정 노드의 열기 태그 내에서 속성을 파싱하고, CFC 트리를 파싱하고, 현재 URL DOM을 가져오고, CFC DOM을 가져오는 등의 작업을 수행합니다.
event.js	이 파일에는 이벤트를 등록하고, menus.xml 파일에서 이벤트 관련 요소에 알림을 보내고, menus.xml 파일에 이벤트 알림 기능을 추가하는 함수가 들어 있습니다.
FlashObjects.js	이 파일에는 색상 선택기 업데이트, 16진수 색상 확인, 절대 링크 확인, 파일 이름에 확장명 추가, 오류 메시지 생성, Flash 속성 설정, Flash 객체의 링크 확인 등의 작업을 수행하는 함수가 들어 있습니다.
insertFireworksHTML.js	이 파일에는 Adobe Fireworks CS3 HTML 코드를 Dreamweaver 문서에 삽입하는 함수가 들어 있습니다. 이러한 함수는 현재 문서가 Fireworks 문서인지 확인하고, Fireworks HTML을 삽입점에 삽입하고, Fireworks 스타일 블록을 Dreamweaver로 업데이트하는 등의 작업을 수행합니다. 이 파일에는 관련 유틸리티 함수도 들어 있습니다.
jumpMenuUI.js	이 파일에는 점프 메뉴 객체 및 점프 메뉴 비헤이비어와 함께 사용할 함수가 들어 있습니다. 이러한 함수는 메뉴 옵션을 채우고, 옵션 레이블을 만들고, 옵션을 추가 또는 삭제하는 등의 작업을 수행합니다.
keyCodes.js	이 파일에는 키보드 키 코드의 배열이 들어 있습니다.
navBar.js	이 파일에는 내비게이션 막대 및 내비게이션 막대 요소를 사용하는 작업에 사용할 수 있는 클래스 및 함수가 들어 있습니다. 또한 이 파일에는 내비게이션 막대 요소를 추가, 제거 및 조작하는 함수도 포함되어 있습니다.
NBInit.js	이 파일에는 내비게이션 막대 이미지 비헤이비어와 관련된 함수가 들어 있습니다.
pageEncodings.js	이 파일은 다양한 언어 코드를 정의합니다.
preload.js	이 파일에는 BODY/onLoad MM_preloadImages 핸들러에 대한 미리 로드된 이미지 호출을 추가하고 삭제하는 함수가 들어 있습니다.
RecordsetDialogClass.js	이 파일에는 레코드세트 서버 비헤이비어 UI를 표시하는 정적 클래스 및 함수가 들어 있습니다. 이러한 함수는 표시할 인터페이스를 단순하게 할지 정교하게 할지를 결정합니다. 또한 여러 UI를 구현할 때 공유될 기능을 저장하고 UI 사이에서 무리없이 사용되도록 중재합니다.
sbUtils.js	이 파일에는 Adobe 서버 비헤이비어에서 사용하는 공유 함수가 들어 있습니다. Configuration/Shared/Common/Scripts 폴더의 dwscripts 클래스에는 더 많은 범용 유틸리티가 들어 있습니다.
setText.js	이 파일에는 표현식 문자열을 이스케이프 처리하거나 이스케이프 처리를 해제하거나 추출하는 함수가 들어 있습니다.
sortTable.js	이 파일에는 테이블을 초기화하고 정렬하는 함수뿐 아니라 배열을 정렬하고, 마우스 포인터를 손모양 아이콘이나 포인터로 설정하고, 브라우저의 형식과 버전을 확인하는 함수도 들어 있습니다.

Scripts 폴더에는 두 개의 하위 폴더인 Class와 CMN이 포함되어 있습니다.

Class 폴더

Class 폴더에는 다음 유틸리티 함수가 들어 있습니다.

파일	설명
classCheckbox.js	이 파일은 HTML Extension에서 체크 상자 컨트롤을 조작하는 데 유용합니다.
FileClass.js	이 파일에는 파일 시스템의 파일을 나타내는 클래스가 들어 있습니다. 경로는 플랫폼간 호환을 위해 URL 형식으로 표시됩니다. 포함된 메서드에는 toString(), getName(), getSimpleName(), getExtension(), getPath(), setPath(), isAbsolute(), getAbsolutePath(), getParent(), getAbsoluteParent(), exists(), getAttributes(), canRead(), canWrite(), isFile(), isFolder(), listFolder(), createFolder(), getContents(), setContents(), copyTo() 및 remove()가 있습니다.
GridClass.js	이 파일에는 MM:TREECONTROL을 관리하는 클래스가 들어 있습니다.
GridControlClass.js	Common 폴더에 있는 GridControlClass의 이전 버전입니다. 자세한 내용은 Shared/Common/Scripts 폴더의 GridControlClass.js 파일을 참조하십시오.
ImageButtonClass.js	Common 폴더에 있는 ImageButtonClass의 이전 버전입니다. 자세한 내용은 Shared/Common/Scripts 폴더의 ImageButtonClass.js 파일을 참조하십시오.
ListControlClass.js	Common 폴더에 있는 ListControlClass의 이전 버전입니다. 자세한 내용은 Shared/Common/Scripts 폴더의 ListControlClass.js 파일을 참조하십시오.
NameValuePairClass.js	이 파일은 이름/값 쌍의 목록을 만들고 관리합니다. 이름에는 모든 문자가 포함될 수 있습니다. 값은 비어 있을 수 있지만 값 삭제와 동일한 의미를 가지는 null로 설정될 수 없습니다.
PageControlClass.js	이 파일은 TabControl 클래스와 함께 사용될 페이지 클래스의 예입니다. TabControlClass.js 설명을 참조하십시오.
PreferencesClass.js	이 파일에는 명령에 대한 모든 환경 설정 정보가 포함되어 있는 객체 및 메서드가 들어 있습니다.
RadioGroupClass.js	Common 폴더에 있는 RadioGroupClass의 이전 버전입니다. 자세한 내용은 Shared/Common/Scripts 폴더의 RadioGroupClass.js 파일을 참조하십시오.
TabControlClass.js	이 파일은 다중 탭 뷰가 있는 Extension인 page.lastUnload()를 작성하는 데 유용합니다.

CMN 폴더

CMN 폴더에는 다음 유틸리티 함수가 들어 있습니다.

파일	설명
dateID.js	이 파일에는 createDateID()와 decipherDateID()라는 두 개의 함수가 들어 있습니다. 세 가지 문자열 "dayFormat", "dateFormat" 및 "timeFormat"이 제공되면 createDateID()는 이 문자열로 ID를 만듭니다. 날짜 배열이 제공되면 decipherDateID()는 dayFormat, dateFormat 및 timeFormat 항목이 들어 있는 배열을 반환합니다.
displayHelp.js	이 파일에는 지정된 도움말 문서를 표시하는 함수가 하나 들어 있습니다.
docInfo.js	이 파일에는 사용자의 문서에 대한 정보를 제공하는 함수가 들어 있습니다. 이 함수가 수행하는 작업에는 지정된 브라우저 유형 및 태그에 대한 객체 참조의 배열 반환, 지정된 태그 이름의 모든 인스턴스 반환 및 현재 선택을 둘러싸는 태그 검색 등이 있습니다.
DOM.js	이 파일에는 Dreamweaver DOM을 사용하는 작업에 필요한 일반 보조 함수가 들어 있습니다. 활성 문서의 루트 노드를 가져오는 함수, 지정된 이름의 태그를 찾는 함수, 지정된 시작 노드로부터 노드 목록을 만드는 함수, 지정된 태그가 다른 태그에 포함되어 있는지 여부를 확인하는 함수, 비헤이비어 함수에 대해 다양한 작업을 수행하는 함수 등과 같은 여러 함수가 포함되어 있습니다.
enableControl.js	이 파일에는 SetEnabled()라는 하나의 함수가 들어 있습니다. 이 함수는 수신되는 인수에 따라 특정 컨트롤을 사용하거나 사용할 수 없게 설정합니다. 이미 사용 가능한 컨트롤을 사용할 수 있도록 설정하거나 이미 사용 불가능한 컨트롤을 사용할 수 없게 설정해도 됩니다.
errmsg.js	이 파일에는 대화 상자에 나타나는 로그 페이지의 배열에 추적 출력을 누적하는 데 사용되는 로그 작성 함수가 들어 있습니다.

파일	설명
file.js	이 파일에는 파일 작업과 관련된 함수가 들어 있습니다. 이러한 함수를 사용하면 사용자가 로컬 파일 이름을 찾아보고, 상대 경로를 파일 URL 경로로 변환하고, 현재 문서의 파일 이름을 반환하고, 지정된 문서가 현재 사이트에 저장되었는지 확인하여 문서의 상대 경로를 반환하고, 지정된 파일이 현재 열려 있는지 확인하는 등의 작업을 수행할 수 있습니다.
form.js	이 파일에는 현재 문서 또는 AP 요소에 양식이 없는 경우 지정된 텍스트 문자열 주위에 양식을 추가하는 함수가 들어 있습니다. 여기에는 객체가 AP 요소인지 확인하고 삽입 포인터가 양식 내부에 있는지 확인하는 함수도 포함되어 있습니다.
handler.js	이 파일에는 이벤트 핸들러에 사용하는 함수를 가져오고, 이벤트 핸들러에 함수를 추가하거나 이벤트 핸들러에서 함수를 삭제하는 함수가 들어 있습니다.
helper.js	이 파일에는 인코딩을 교체하고, 인용 부호(")의 이스케이프 처리를 해제하고, 노드가 선택 범위 내에 존재하는지 검사하고, 객체 이름이 중복되는지 확인하는 유용한 함수가 들어 있습니다.
insertion.js	이 파일에는 현재 삽입점에서 텍스트 문자열을 문서에 삽입하는 insertIntoDocument() 함수가 들어 있습니다. 또한 두 개의 지원 함수인 getHigherBlockTag() 및 arrContains()도 들어 있습니다. getHigherBlockTag() 함수는 blockTags 배열에 정의되어 있는 다음으로 가장 높은 blockTag를 가져오며 arrContains() 함수는 배열에서 지정된 항목을 찾습니다.
localText.js	이 파일에는 예약된 변수가 들어 있습니다. 이 변수들은 범용으로 사용할 수 없습니다. 범용 변수로 사용하려면 이 파일 대신 Startup/mminit.htm을 사용하거나 Dreamweaver Configuration/Strings/*.xml 파일에 있는 문자열을 사용합니다.
menuItem.js	이 파일에는 메뉴 항목 목록의 값을 추가하거나 제거하는 함수가 들어 있습니다.
niceName.js	이 파일에는 객체 참조 배열을 보다 단순한 이름 배열로 변환하는 함수가 들어 있습니다.
quickString.js	이 파일에는 메모리 할당 작업을 매번 수행하지 않고 더 작은 문자열을 수집하는 함수가 들어 있습니다.
string.js	이 파일에는 텍스트 문자열을 조작하고 파싱하는 데 사용되는 일련의 일반 함수가 들어 있습니다. 이러한 함수로는 extractArgs(), escQuotes(), unescQuotes(), quoteMeta(), errMsg(), badChars(), getParam(), quote(), stripSpaces(), StripChars(), AllInRange(), reformat(), trim(), createDisplayString(), entityNameEncode(), entityNameDecode(), stripAccelerator() 및 SprintF()가 있습니다.
TemplateUtils.js	이 파일에는 Dreamweaver 템플릿에 사용되는 유틸리티 함수가 들어 있습니다. 이러한 함수는 문서에 편집 가능한 영역이나 반복 영역을 삽입하고, 문서에서 지정된 편집 가능 영역을 검색하는 등의 작업을 수행합니다.
UI.js	이 파일에는 UI를 제어하는 일반 함수가 들어 있습니다. 이러한 함수는 현재 문서에서 특정 객체를 찾고, 지역화된 문자열로 선택 목록 옵션을 로드하고, 선택된 옵션에 대한 속성 값을 반환하고, 경고에 사용할 텍스트 메시지를 줄 바꿈하는 등의 작업을 수행합니다.

기타 폴더

다음 목록에서는 Shared 폴더에서 알아두면 유용한 다른 폴더에 대해 설명합니다.

Controls Controls 폴더에는 서버 비헤이비어를 만드는 데 사용되는 요소가 들어 있습니다. 이러한 컨트롤에는 텍스트 및 레코드세트 메뉴의 인터페이스가 포함됩니다.

참고: 이러한 컨트롤은 Dreamweaver 서버 비헤이비어 구성 관리자와 여러 Dreamweaver 서버 비헤이비어에 사용되지만 일부는 Extension에서 컨트롤을 관리하는 데에도 유용합니다.

Fireworks Fireworks 폴더에는 Fireworks를 통합하는 데 필요한 지원 파일이 들어 있습니다.

UltraDev Dreamweaver에서는 기본적으로 이전 버전과의 호환성을 위해 이 폴더를 유지 관리하며, 새 Extension에는 이 폴더를 사용하면 안 되고 대신 이 폴더에 있는 대다수 함수가 들어 있는 Dreamweaver Configuration/Shared/Common 폴더를 사용해야 합니다. 자세한 내용은 342페이지의 “Common 폴더”를 참조하십시오.

Shared 폴더 사용

유용한 Extension 코드가 필요하면 먼저 Dreamweaver의 Configuration/Shared/Common 폴더를 찾아보십시오. 이 폴더에는 가장 일반적으로 사용되는 최신 기능이 포함되어 있습니다.

Shared 폴더의 리소스를 이용하여 Extension의 고유한 기능을 만들 수 있습니다. 객체, 명령 또는 기타 Extension은 script 태그에서 Shared 폴더에 있는 JavaScript 파일 중 하나를 소스 파일로 지정한 다음, 해당 파일의 본문이나 포함된 또 다른 JavaScript 파일의 함수를 사용할 수 있습니다. 객체와 명령은 여러 개의 JavaScript 파일을 함께 연결할 수 있으며 이러한 JavaScript 파일은 Shared 폴더 리소스를 이용할 수 있습니다.

예를 들어 응용 프로그램 폴더 Configuration/Objects/Common에서 하이퍼텍스트 객체 파일(Hyperlink.htm)을 엽니다. 이 파일의 head 태그에는 다음과 같은 행이 포함되어 있습니다.

```
<script language="javascript" src="../../Shared/Common/Scripts/ListControlClass.js"></script>
<script language="javascript" src="Hyperlink.js"></script>
```

또한 관련된 Hyperlink.js 파일을 열면 다음과 같은 행을 볼 수 있습니다.

```
LIST_LINKS = new ListControl('linkPath');
```

및

```
LIST_TARGETS = new ListControl('linkTarget');
```

Hyperlink.js에서는 new listControl을 선언하여 두 개의 새로운 ListControl 객체를 정의합니다. 그런 다음 Hyperlink.htm 파일의 코드가 다음과 같이 두 객체를 양식의 SELECT 컨트롤에 첨부합니다.

```
<td  align="left"> <input name="linkText" type="text" class="basicTextField" value="">
```

및

```
<td align="left" nowrap><select name="linkPath" class="basicTextField" editable="true">
```

이제 Hyperlink.js 스크립트는 LIST_LINKS 또는 LIST_TARGETS 객체에 대한 메서드를 호출하거나 속성을 가져와서 양식의 SELECT 컨트롤과 상호 작용할 수 있습니다.