Zugriff auf Daten mit ADOBE® FLEX® 4.6

Rechtliche Hinweise

Rechtliche Hinweise finden Sie unter http://help.adobe.com/de_DE/legalnotices/index.html.

Letzte Aktualisierung 6.12.2011

Inhalt

Kapitel 1: Überblick über den Zugriff auf Datendienste

Datenzugriff in Flex im Vergleich zu anderen Technologien

Flex verarbeitet Datenquellen und Daten anders als Anwendungen, deren Benutzeroberfläche mit HTML erstellt wird.

Clientseitige und serverseitige Verarbeitung

Anders als bei HTML-Vorlagen, die mithilfe von JSPs und Servlets, ASP, PHP oder CFML erstellt wurden, wird bei Flex der Clientcode vom Servercode getrennt. Die Benutzeroberfläche der Anwendung wird zu einer binären SWF-Datei kompiliert, die an den Client gesendet wird.

Wenn die Anwendung eine Anforderung an einen Datendienst stellt, wird die SWF-Datei nicht neu kompiliert und es ist keine Seitenaktualisierung erforderlich. Der Remotedienst gibt nur Daten zurück. Flex bindet die zurückgegebenen Daten an Benutzeroberflächenkomponenten in der Clientanwendung.

Beispiel: Wenn in Flex ein Benutzer auf ein Button-Steuerelement einer Anwendung klickt, ruft der clientseitige Code einen Webservice auf. Die Ergebnisdaten des Webservice werden ohne Seitenaktualisierung an die binäre SWF-Datei zurückgegeben. Die Ergebnisdaten können daher als dynamischer Inhalt in der Anwendung verwendet werden.

```
<?xm l version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
    xmlns:s="library://ns.adobe.com/flex/spark"
    xmlns:mx="library://ns.adobe.com/flex/halo" minWidth="1024" minHeight="768"
xmlns:employeesservice="services.employeesservice.*" xmlns:valueObjects="valueObjects.*">

    <fx:Declarations>
        <s:WebService
            id="RestaurantSvc"
        wsdl="http://examples.adobe.com/flex3app/restaurant_ws/RestaurantWS.xml?wsdl" />
        <s:CallResponder id="getRestaurantsResult"
            result="restaurants = getRestaurantsResult.lastResult as Restaurant"/>
    </fx:Declarations>

    <fx:Script>
        <![CDATA[
            import mx.controls.Alert;

            protected function b1_clickHandler(event:MouseEvent):void {
                getRestaurantsResult.token = RestaurantWS.getRestaurantss();
            }
        ]]>
    </fx:Script>
. . .
    <s:Button id="b1" label="GetRestaurants" click="button_clickHandler(event)"/>
```

Vergleichen Sie dieses Flex-Beispiel mit dem folgenden Beispiel, das JSP-Code zum Aufrufen eines Webservice mithilfe eines benutzerdefinierten JSP-Tags zeigt. Wenn ein Benutzer JSP anfordert, erfolgt die Webserviceanforderung auf dem Server statt auf dem Client. Das Ergebnis wird zum Generieren von Inhalt auf der HTML-Seite verwendet. Der Anwendungsserver generiert die gesamte HTML-Seite erneut, bevor sie an den Webbrowser des Benutzers zurückgesendet wird.

```
<%@ taglib prefix="web" uri="webservicetag" %>

<% String str1="BRL";
String str2="USD";%>

<!-- Call the web service. -->
<web:invoke
    url="http://www.itfinity.net:8008/soap/exrates/default.asp"
    namespace="http://www.itfinity.net/soap/exrates/exrates.xsd"
    operation="GetRate"
    resulttype="double"
    result="myresult">
    <web:param name="fromCurr" value="<%=str1%>"/>
    <web:param name="ToCurr" value="<%=str2%>"/>
</web:invoke>

<!-- Display the web service result. -->
<%= pageContext.getAttribute("myresult") %>
```

Datenquellenzugriff

Ein weiterer Unterschied zwischen Flex und anderen Webanwendungstechnologien liegt darin, dass Sie in Flex niemals direkt mit einer Datenquelle kommunizieren. Sie verwenden eine Datenzugriffskomponente, um eine Verbindung zu einem Remotedienst herzustellen und mit der serverseitigen Datenquelle zu interagieren.

Das folgende Beispiel zeigt eine ColdFusion-Seite, die direkt auf eine Datenquelle zugreift:

```
...
<CFQUERY DATASOURCE="Dsn"
    NAME="myQuery">
    SELECT * FROM table
</CFQUERY>
...
```

Um eine ähnliche Funktionalität in Flex zu erhalten, rufen Sie mithilfe einer HTTPService-, Webservice- oder RemoteObject-Komponente ein serverseitiges Objekt auf, das Ergebnisse aus einer Datenquelle zurückgibt.

Ereignisse, Dienstaufrufe und Datenbindung

Flex ist eine ereignisgesteuerte Technologie. Eine Benutzeraktion oder ein Programmereignis können den Zugriff auf einen Dienst auslösen. Wenn beispielsweise ein Benutzer auf eine Schaltfläche klickt, ist dies ein Benutzeraktionsereignis, das zum Auslösen eines Serviceaufrufs verwendet werden kann. Um ein Programmereignis handelt es sich beispielsweise, wenn die Anwendung die Erstellung einer Benutzeroberflächenkomponente, wie z. B. eines DataGrid, abschließt. Das creationComplete-Ereignis für das DataGrid kann für das Aufrufen eines Remotedienstes zum Füllen des DataGrid verwendet werden.

Dienstaufrufe in Flex sind asynchron. Die Clientanwendung muss nicht auf zurückgegebene Daten warten. Asynchrone Dienstaufrufe sind beim Abrufen oder Aktualisieren großer Datenmengen hilfreich. Die Clientanwendung wird nicht durch das Warten auf die abgerufenen oder aktualisierten Daten blockiert.

Die von einem Dienstaufruf zurückgegebenen Daten werden in einer CallResponder-Eigenschaft gespeichert, die mit dem Dienstaufruf verknüpft ist. Benutzeroberflächenkomponenten rufen anschließend mithilfe der Datenbindung die zurückgegebenen Daten vom CallResponder ab.

Mit der Datenbindung in Flex können Sie eine Benutzeroberflächenkomponente dynamisch mit einer Datenquelle aktualisieren. Beispielsweise kann eine Flex-Komponente ihr Textattribut mit dem lastResult-Attribut eines CallResponder verknüpfen. Wenn sich die Daten im CallResponder ändern, wird die Flex-Komponente automatisch aktualisiert.

Flex implementiert auch die bidirektionale Datenbindung. Wenn sich bei der bidirektionalen Datenbindung Daten in der Flex-Komponente oder der Datenquelle ändern, wird die entsprechende Datenquelle bzw. Flex-Komponente automatisch aktualisiert. Die bidirektionale Datenbindung ist nützlich beim Aktualisieren von Remotedaten aus Benutzereingaben in einer Formular- oder einer Flex-Datenkomponente.

Verwandte Themen

„Erstellen datenorientierter Anwendungen mit Flash Builder“ auf Seite 8

Verwenden von Flash Builder für den Zugriff auf einen Datendienst

In Flex Builder 3 implementieren Sie Remoteprozeduraufrufe für Datendienste mithilfe von Flex-Datenzugriffskomponenten. Dieser Vorgang wird in Flash Builder jedoch vereinfacht.

Flash Builder stellt Assistenten und andere Werkzeuge für folgende Vorgänge zur Verfügung:

- Gewähren des Zugriffs auf Datendienste
- Konfigurieren der vom Datendienst zurückgegebenen Daten
- Unterstützung für das Paging der vom Datendienst zurückgegebenen Daten
- Unterstützung der Datenverwaltungsfunktionalität, die mehrere Aktualisierungen der Serverdaten synchronisiert
- Generieren von Clientcode für den Datendienstzugriff
- Binden der vom Dienst zurückgegebenen Daten an Benutzeroberflächenkomponenten

Flash Builder-Arbeitsablauf für den Zugriff auf Dienste

Halten Sie den folgenden Arbeitsablauf ein, wenn Sie mithilfe von Flash Builder eine Anwendung erstellen, die auf Datendienste zugreift.

1 Je nach den jeweiligen Umständen beginnen Sie mit dem Herstellen einer Verbindung zu einem Datendienst oder mit dem Erstellen der Benutzeroberfläche.

Verbindung zum Remotedienst herstellen. Wenn Sie mit dem Herstellen der Verbindung zum Remotedienst beginnen, erstellen Sie die Benutzeroberfläche im Anschluss.

Benutzeroberfläche erstellen. Wenn Sie mit dem Erstellen der Benutzeroberfläche beginnen, stellen Sie die Verbindung zum Remotedienst im Anschluss her.

Hinweis: *Wo Sie beginnen, bleibt Ihnen überlassen. Beispiel: Wenn Sie bereits ein Design für eine Benutzeroberfläche geplant haben, können Sie die Benutzeroberfläche zuerst erstellen. Umgekehrt können Sie zuerst eine Verbindung zu den Daten herstellen und dann mithilfe von Flash Builder Anwendungskomponenten generieren.*

2 Binden Sie Datenmethoden an Anwendungskomponenten.

Flash Builder bietet mehrere Möglichkeiten, um Sie beim Binden von Datenmethoden an Anwendungskomponenten zu unterstützen. Mithilfe von Flash Builder können Sie Folgendes tun:

- Verschiedene Formulare für die von Dienstmethoden zurückgegebenen Daten generieren
- Dienstmethoden auswählen, die an die Benutzeroberflächenkomponenten gebunden werden
- Ein Formular generieren, das die von einem Dienst zurückgegebenen komplexen Daten darstellt

3 (Optional) Verwalten Sie das Abrufen und das Aktualisieren von Daten.

Mit Flash Builder-Werkzeugen können Sie das Paging zurückgegebener Daten implementieren und das Aktualisieren von Datengruppen koordinieren.

Beim Zurückgeben großer Mengen von Datensätzen implementieren Sie im Allgemeinen das Paging, um eine Gruppe von Datensätzen „nach Bedarf“ abzurufen.

Für Anwendungen, die mehrere Datensätze aktualisieren, können Sie Datenverwaltungsfunktionen implementieren. Die Datenverwaltung umfasst folgende Funktionen:

- Funktion für das gleichzeitige Aktualisieren geänderter Datensätze
- Mechanismus zum Zurücksetzen von Änderungen, bevor sie auf den Server geschrieben werden
- Codegenerierung, die die Benutzeroberfläche beim Hinzufügen, Löschen bzw. Ändern von Datensätzen automatisch aktualisiert

4 Führen Sie die Anwendung aus und überwachen Sie den Datenfluss.

Wenn die Anwendung fertig ist, führen Sie sie aus, damit Sie sie im laufenden Betrieb sehen. Zeigen Sie mithilfe der Flash Builder-Netzwerküberwachung die zwischen der Anwendung und dem Dienst übergebenen Daten an. Die Netzwerküberwachung ist nützlich, um Fehler zu diagnostizieren und die Leistung zu analysieren.

Flash Builder bietet auch stabile Debugging- und Profiling-Umgebungen. In Flash Builder Premium sind Network Monitor und Flash Profiler enthalten.

Verwandte Themen

„Erstellen datenorientierter Anwendungen mit Flash Builder“ auf Seite 8

Erweitern der von Flash Builder unterstützten Dienste

Flash Builder-Assistenten und -Werkzeuge unterstützen den Zugriff auf den folgenden Typ von Dienstimplementierungen:

- PHP-Dienste
- ColdFusion-Dienste
- BlazeDS
- ADEP Data Services (früher LiveCycle Data Services genannt)
- HTTP (REST-Style)-Dienste
- Webservices (SOAP)
- Statische XML-Dateien

Wenn Sie Werkzeuge für zusätzliche Diensttypen wie etwa Ruby on Rails benötigen, können Sie die Flash Builder-Implementierung erweitern. Siehe Flash Builder-Erweiterungsreferenz.

Datenzugriffskomponenten

Datenzugriffskomponenten ermöglichen es Clientanwendungen, Methoden und Dienste in Netzwerken aufzurufen. Dabei verwenden die Datenzugriffskomponenten Remoteprozeduraufrufe für die Interaktion mit Serverumgebungen. Es gibt drei Datenzugriffskomponenten: RemoteObject-, HTTPService- und WebService-Komponenten.

Datenzugriffskomponenten sind für Clientanwendungen konzipiert, bei denen ein Aufruf- und Antwortmodell eine geeignete Methode für den Zugriff auf externe Daten darstellt. Mithilfe dieser Komponenten kann der Client asynchrone Anforderungen an Remotedienste richten, die die Ereignisse verarbeiten und anschließend Daten an Ihre Flex-Anwendung zurückgeben.

Eine Datenzugriffskomponente ruft einen Remotedienst auf. Dann speichert sie die Antwortdaten des Diensts in einem ActionScript-Objekt oder einem anderen Format, das der Dienst zurückgibt. Datenzugriffskomponenten in der Clientanwendung werden für das Arbeiten mit drei Diensttypen verwendet:

- Remoteobjektdienste (RemoteObject)
- SOAP-basierte Webservices (WebService)
- HTTP-Dienste einschließlich REST-basierter Webservices (HTTPService)

Adobe® Flash® Builder™ stellt Assistenten und Werkzeuge für die Implementierung von Datenzugriffskomponenten innerhalb eines Dienst-Wrappers zur Verfügung. Der Dienst-Wrapper kapselt die Funktionen der Datenzugriffskomponente und erspart Ihnen den Großteil der Grundimplementierung. Dadurch können Sie sich auf die Implementierung der Dienste und das Erstellen der Clientanwendungen für den Zugriff auf die Dienste konzentrieren. Weitere Informationen dazu, wie mit Flash Builder auf Datendienste zugegriffen werden kann, finden Sie unter „Erstellen datenorientierter Anwendungen mit Flash Builder“ auf Seite 8.

Bereitstellen des Zugriffs auf Dienste

Standardmäßig blockiert Adobe Flash Player den Zugriff auf jeden Host, der nicht exakt mit demjenigen identisch ist, der zum Laden einer Anwendung verwendet wird. Wenn Sie keine serverseitige Anwendung wie ADEP Data Services oder BlazeDS für die Proxyverarbeitung von Anforderungen verwenden, muss sich der HTTP-Dienst oder Webservice entweder auf dem Hostserver Ihrer Anwendung befinden oder der Remoteserver, der den HTTP- oder Webservice hostet, muss eine `crossdomain.xml`-Datei definieren. Eine `crossdomain.xml`-Datei bietet dem Server die Möglichkeit, anzugeben, dass seine Daten und Dokumente für SWF-Dateien aus bestimmten oder allen anderen Domänen zugänglich sind. Die `crossdomain.xml`-Datei muss sich im Webstamm des Servers, den die Anwendung kontaktiert, befinden.

HTTPService-Komponenten

Verwenden Sie HTTPService-Komponenten, um HTTP GET- oder POST-Anforderungen zu senden und Daten aus der HTTP-Antwort in die Clientanwendung zu übernehmen. Wenn Sie Flex zur Erzeugung von Desktopanwendungen (die in Adobe AIR® ausgeführt werden) verwenden, werden HTTP PUT und DELETE unterstützt.

Wenn Sie ADEP Data Services oder BlazeDS verwenden, können Sie einen HTTPProxyService verwenden, der den Einsatz weiterer HTTP-Methoden ermöglicht. Mit einem HTTPProxyService können Sie Anforderungen der Typen GET, POST, HEAD, OPTIONS, PUT, TRACE und DELETE senden.

Bei einem HTTP-Dienst kann es sich um jeden HTTP-URI handeln, der HTTP-Anforderungen akzeptiert und Antworten sendet. Dieser Diensttyp wird häufig auch als REST-Style-Webservice bezeichnet. REST steht für Representational State Transfer, eine Architektur für verteilte Hypermediasysteme.

HTTPService-Komponenten sind eine gute Option, wenn der von einem SOAP-Webservice oder einem Remoteobjektdienst gebotene Funktionsumfang nicht zur Verfügung steht. Beispiel: Sie können HTTPService-Komponenten für die Interaktion mit JavaServer Pages (JSPs), Servlets und ASP-Seiten verwenden, die nicht als Webservices oder Remoting Service-Ziele zur Verfügung stehen.

Wenn Sie die `send()`-Methode des HTTPService-Objekts aufrufen, wird eine HTTP-Anforderung an den angegebenen URI gesendet und eine HTTP-Antwort zurückgegeben. Optional können Sie dem angegebenen URI auch Argumente übergeben.

Flash Builder stellt Workflows bereit, mit deren Hilfe Sie die Verbindung zu HTTP-Diensten interaktiv herstellen können. Weitere Informationen hierzu finden Sie unter „Zugriff auf HTTP-Dienste“ auf Seite 13.

Verwandte Themen

„Zugriff auf HTTP-Dienste“ auf Seite 13

Dissertation: Representational State Transfer (REST) von Roy Thomas Fielding

WebService-Komponenten

Mithilfe von WebService-Komponenten können Sie auf SOAP-Webservices zugreifen, bei denen es sich um Softwaremodule mit Methoden handelt. Webservicemethoden werden meist als *operations* bezeichnet. Webserviceschnittstellen werden in der Web Services Description Language (WSDL) definiert. Webservices stellen eine standardkonforme Methode für die Interaktion zwischen Softwaremodulen, die auf verschiedenen Plattformen ausgeführt werden, bereit. Weitere Informationen über Webservices finden Sie im Abschnitt über Webservices der World Wide Web Consortium-Website unter der folgenden Adresse: www.w3.org/2002/ws/.

Clientanwendungen können mit Webservices interagieren, die ihre Schnittstellen in einem Web Services Description Language (WSDL)-Dokument definieren, das als URL zur Verfügung steht. WSDL ist ein Standardformat, mit dem Folgendes beschrieben wird: Meldungen, die ein Webservice verstehen kann, das Format seiner Antworten auf diese Meldungen, das Protokoll, das der Webservice unterstützt, und schließlich das Ziel, an das die Meldungen gesendet werden.

Flex unterstützt WSDL 1.1. Eine Beschreibung ist unter folgender Adresse erhältlich: www.w3.org/TR/wsdl. Flex unterstützt RPC-encoded- und Document-literal-Webservices.

Flex unterstützt Webserviceanforderungen und -ergebnisse, die als SOAP-Meldungen formatiert sind und über HTTP transportiert werden. SOAP stellt die Definition des auf XML basierenden Formats bereit, das Sie für den Austausch von strukturierten und typisierten Daten zwischen einem Webserviceclient (beispielsweise einer in Flex erstellten Anwendung) und einem Webservice verwenden können.

Sie können mit einer WebService-Komponente eine Verbindung zu einem SOAP-kompatiblen Webservice herstellen, wenn Webservices in Ihrer Umgebung standardmäßig verwendet werden. WebService-Komponenten sind auch nützlich bei Objekten, die innerhalb einer Unternehmensumgebung, jedoch nicht zwangsläufig im Quellpfad der Webanwendung verfügbar sind.

Flash Builder stellt Workflows bereit, mit deren Hilfe Sie interaktiv eine Verbindung zu Webservices herstellen können. Weitere Informationen finden Sie unter „Zugriff auf Webservices“ auf Seite 16.

RemoteObject-Komponenten

Mithilfe von Remoteobjektdiensten können Sie direkt auf die Unternehmenslogik im systemeigenen Format zugreifen, anstatt sie als XML zu formatieren, wie beispielsweise bei REST-Style-Diensten oder Webservices. Dadurch sparen Sie die Zeit, die für die Darstellung einer vorhandenen Logik als XML erforderlich ist. Ein weiterer Vorteil von Remote-Objektdiensten ist die Kommunikationsgeschwindigkeit im Netz. Der Datenaustausch erfolgt weiterhin über HTTP oder https, aber die Daten selbst werden in eine binäre Darstellung serialisiert. Die Verwendung von RemoteObject-Komponenten führt zu einem geringeren Datenfluss im Netz, einer reduzierten clientseitigen Speichernutzung und einer verringerten Verarbeitungszeit.

ColdFusion, PHP, BlazeDS und ADEP Data Services können mithilfe serverseitiger Typisierung auf Serverdaten zugreifen. Die Clientanwendung greift auf ein Java-Objekt, eine ColdFusion-Komponente (die intern ein Java-Objekt ist) oder eine PHP-Klasse direkt zu. Dies erfolgt durch Remoteaufruf einer Methode für ein festgelegtes Objekt. Das Objekt auf dem Server verwendet seine eigenen nativen Datentypen als Argumente, fragt anhand dieser eine Datenbank ab und gibt Werte in seinen nativen Datentypen zurück.

Für den Fall, dass die serverseitige Typisierung nicht zur Verfügung steht, hat Flash Builder Werkzeuge zur Implementierung der clientseitigen Typisierung. Konfigurieren und definieren Sie mithilfe von Flash Builder die vom Dienst zurückzugebenden Datentypen. Durch die clientseitige Typisierung kann die Clientanwendung eine Datenbank abfragen und korrekt typisierte Daten abrufen Die clientseitige Typisierung ist für alle Dienste erforderlich, die den vom Dienst zurückgegebenen Datentyp nicht definieren.

Flash Builder stellt Workflows bereit, anhand derer Sie eine Verbindung zu Remoteobjektdiensten interaktiv herstellen können. Weitere Informationen finden Sie unter „Verbindung zu Datendiensten herstellen“ auf Seite 9.

Kapitel 2: Erstellen datenorientierter Anwendungen mit Flash Builder

Die Werkzeuge in Flash Builder unterstützen Sie dabei, Anwendungen zu erstellen, die auf Datendienste zugreifen. Als ersten Schritt erstellen Sie ein Flex-Projekt für Ihre Anwendungen. Dann stellen Sie eine Verbindung zum Datendienst her, konfigurieren den Datenzugriff im Dienst und erstellen eine Benutzeroberfläche für eine Anwendung. In manchen Fällen erfolgt das Erstellen der Benutzeroberfläche zuerst und erst danach greifen Sie auf den Datendienst zu.

Erstellen eines Flex-Projekts für den Zugriff auf einen Datendienst

Flex greift auf Datendienste entweder als Remoteobjekt, HTTP-Dienst (REST-Stil) oder Webservice (SOAP) zu.

Auf die folgenden Arten von Datendiensten greifen Sie mithilfe eines Remoteobjekts zu:

- ColdFusion-Dienste
- AMF-basierte PHP-Dienste
- BlazeDS
- ADEP Data Services (früher LiveCycle Data Services genannt)

Informationen zur Verwendung des LiveCycle Service Discovery-Assistenten finden Sie unter Verwenden von LiveCycle Discovery.

Erstellen Sie für jeden Dienst, auf den als Remoteobjekt zugegriffen wird, ein Flex-Projekt, das für den zutreffenden Anwendungsservertyp konfiguriert ist. Der Assistent „Neues Flex-Projekt“ führt Sie durch die Konfiguration eines Projekts für die nachstehend aufgeführten Anwendungsservertypen:

Servertyp	**Unterstützte Remoteobjektdienste**
PHP	• AMF-basierte PHP-Dienste
ColdFusion	• ColdFusion Flash Remoting • BlazeDS • ADEP Data Services
J2EE	• BlazeDS • ADEP Data Services

Sie können aus jeder beliebigen Flex-Projektkonfiguration, einschließlich Projekten, in denen keine Servertechnologie angegeben ist, auf HTTP-Dienste (REST-Stil) und Webservices zugreifen.

Ein Projekt, das für den Zugriff auf ein Remoteobjekt konfiguriert ist, kann nur auf jenen Remoteobjektdienst zugreifen, für den es konfiguriert ist. So können Sie beispielsweise aus einem Projekt, das für ColdFusion konfiguriert ist, nicht auf einen AMF-basierten PHP-Dienst zugreifen. Es ist jedoch möglich, aus diesem Projekt auf den PHP-Dienst zuzugreifen, wenn der Zugriff als Webservice oder HTTP-Dienst erfolgt.

Verwandte Themen

„Überblick über den Zugriff auf Datendienste“ auf Seite 1

Ändern des Servertyps eines Projekts

Flash Builder warnt Sie, wenn Sie auf einen Dienst zuzugreifen versuchen, für den das Flex-Projekt nicht konfiguriert ist. Wenn im Flex-Projekt nicht die korrekte Serverkonfiguration angegeben ist, stellt Flash Builder eine Verknüpfung zum Dialogfeld „Projekteigenschaften“ bereit. Im Dialogfeld „Projekteigenschaften“ können Sie den Zugriff des Projekts auf den Datendienst konfigurieren. So warnt Sie Flash Builder beispielsweise, wenn Sie versuchen, aus einem Projekt, in dem keine Serverkonfiguration angegeben ist, auf einen AMF-basierten PHP-Dienst zuzugreifen.

Wenn das Flex-Projekt bereits für den Zugriff auf einen anderen Diensttyp konfiguriert wurde, konfigurieren Sie entweder ein neues Flex-Projekt oder ändern die Konfiguration im vorhandenen Projekt. Wenn Sie die Serverkonfiguration eines Projekts ändern, können Sie nicht mehr auf früher konfigurierte Dienste zugreifen. Beispiel: Wenn Sie die Konfiguration eines Projekts von ColdFusion auf PHP ändern, ist der Zugriff auf die im Projekt konfigurierten ColdFusion-Dienste nicht mehr möglich.

Wenn Sie aus demselben Projekt auf verschiedene Diensttypen zugreifen möchten, können Sie auf die Dienste entweder als HTTP-Dienste oder Webservices zugreifen.

Cross-Domain-Richtliniendatei

Um von der SWF-Datei der Anwendung aus auf Dienste in einer anderen Domäne zugreifen zu können, ist eine Cross-Domain-Richtliniendatei erforderlich. AMF-basierte Dienste benötigen üblicherweise keine Cross-Domain-Richtliniendatei, weil sich diese Dienste in derselben Domäne wie die Anwendung befinden.

Verbindung zu Datendiensten herstellen

Stellen Sie mit dem Flash Builder-Dienstassistenten eine Verbindung zu einem Datendienst her.

Für Remoteobjektdienste erstellen Sie in der Regel ein Flex-Projekt mit dem entsprechenden Anwendungsservertyp. Flash Builder introspektiert den Dienst und kann die vom Dienst zurückgegebenen Datentypen konfigurieren.

Remoteobjektdienste beinhalten Datendienste, die in ColdFusion, PHP, BlazeDS und ADEP Data Services (früher LiveCycle Data Services genannt) implementiert sind.

Informationen zur Verwendung des LiveCycle Service Discovery-Assistenten finden Sie unter Verwenden von LiveCycle Discovery.

Verwandte Themen

„Erstellen eines Flex-Projekts für den Zugriff auf einen Datendienst“ auf Seite 8

Zugriff auf ColdFusion-Dienste

Greifen Sie auf einen ColdFusion-Datendienst, der als ColdFusion-Komponente (CFC) implementiert wurde, mithilfe des Flash Builder Service-Assistenten zu. Flex greift auf diese Dienste als Remoteobjekte zu.

Verwenden Sie ein Flex-Projekt mit ColdFusion als Anwendungsservertyp. Legen Sie beim Erstellen eines Flex-Projekts „Remote Object Access Service verwenden“ fest und verwenden Sie ColdFusion Flash Remoting.

Herstellen einer Verbindung zu ColdFusion-Datendiensten

Bei dieser Prozedur wird davon ausgegangen, dass Sie einen ColdFusion-Dienst implementiert und ein Flex-Projekt für den Zugriff auf ColdFusion-Dienste erstellt haben.

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit ColdFusion verbinden“, um den Dienstassistenten zu öffnen.

2 Navigieren Sie im Dialogfeld „ColdFusion-Dienst konfigurieren“ zum Speicherort der ColdFusion-Komponente, die den Dienst implementiert.

Hinweis: *Wenn Sie keinen ColdFusion-Dienst implementiert haben, kann Flash Builder aus einer einzelnen Datenbanktabelle einen Beispieldienst generieren. Verwenden Sie das generierte Beispiel als Muster dafür, wie Sie auf Datendienste zugreifen können. Siehe „Generieren eines ColdFusion-Beispieldiensts aus einer Datenbanktabelle.“ auf Seite 10.*

3 (Optional) Ändern Sie die Details des Diensts.

Dienstname	Geben Sie einen Namen für den Dienst ein. Basierend auf dem Dateinamen des Diensts generiert Flash Builder einen Dienstnamen. Die Namen, die Sie für einen Dienst verwenden können, unterliegen bestimmten Einschränkungen. Siehe „Benennen von Datendiensten“ auf Seite 21.
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Datentyppaket	Legen Sie einen Namen für das Paket mit den generierten ActionScript-Dateien, in denen die vom Dienst abzurufenden Datentypen definiert sind, fest. Standardmäßig erstellt Flash Builder das `valueObjects`-Paket.

4 (Optional) Klicken Sie auf „Weiter“, um die Dienstmethoden anzuzeigen.

5 Klicken Sie auf „Fertig stellen“, um ActionScript-Dateien zu generieren, die auf den Dienst zugreifen.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Nächster Schritt: „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

Generieren eines ColdFusion-Beispieldiensts aus einer Datenbanktabelle.

Flash Builder kann einen ColdFusion-Beispieldienst generieren, den Sie als Prototyp für Ihre eigenen Dienste verwenden können. Der Beispieldienst greift auf eine einzelne Datenbanktabelle zu und verfügt über Methoden zum Erstellen, Lesen, Aktualisieren und Löschen.

Flash Builder konfiguriert Rückgabedatentypen für die generierten Dienste und ermöglicht Datenzugriffsfunktionen wie Paging und Datenverwaltung.

Wichtig: *Verwenden Sie den generierten Dienst nur in einer sicheren Entwicklungsumgebung. Der generierte Code ermöglicht allen Personen mit Netzwerkzugriff auf Ihren Server, auf Daten der Datenbanktabelle zuzugreifen, sowie diese zu ändern und zu löschen. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben von sicheren Diensten finden Sie unter About User Security.*

Bei der folgenden Prozedur wird davon ausgegangen, dass Sie ein Flex-Projekt für den Zugriff auf ColdFusion-Dienste erstellt und ColdFusion-Datenquellen zur Verfügung haben.

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit ColdFusion verbinden“, um den Dienstassistenten zu öffnen.

2 Klicken Sie im Dialogfeld „ColdFusion-Dienst konfigurieren“ auf den Hyperlink zum Generieren eines Beispieldiensts.

3 Wählen Sie „Aus RDS-Datenquelle generieren“ und geben Sie eine ColdFusion-Datenquelle und Tabelle an.

   Wenn in der Tabelle kein Primärschlüssel definiert ist, wählen Sie einen Primärschlüssel für die Tabelle.

   Hinweis: *Wenn Sie keine ColdFusion-Datenquelle zur Verfügung haben, wählen Sie „Aus Vorlage generieren“. Flash Builder schreibt eine ColdFusion-Beispielkomponente (CFC) mit typischen Dienstmethoden. Heben Sie die Auskommentierung bestimmter Funktionen in der CFC auf und ändern Sie die Methoden, damit Sie einen Beispieldienst erhalten, der Ihnen als Prototyp dienen kann.*

4 Verwenden Sie den standardmäßigen oder einen neuen Speicherort. Klicken Sie auf „OK“.

   Flash Builder generiert den Beispieldienst. Ändern Sie den Dienstnamen und die Speicherorte für die Pakete, um die Standardwerte zu überschreiben.

5 (Optional) Klicken Sie auf „Weiter“, um die Methoden des Diensts anzuzeigen.

6 Klicken Sie auf „Fertig stellen“.

   Flash Builder generiert ActionScript-Dateien, die auf den Beispieldienst zugreifen. Des Weiteren öffnet Flash Builder den Beispieldienst in dem auf Ihrem System für die Bearbeitung von ColdFusion-CFC-Dateien festgelegten Editor.

Zugriff auf PHP-Dienste

Stellen Sie mit dem Flash Builder-Dienstassistenten eine Verbindung zu einem in PHP implementierten Datendienst her. Flex verwendet Action Message Format (AMF), um Daten zwischen der Clientanwendung und dem Datendienst zu serialisieren. Flash Builder installiert das Zend AMF Framework, um Zugriff auf in PHP implementierte Dienste zu gewähren. Siehe „Installieren des Zend Framework“ auf Seite 21.

Greifen Sie auf PHP-Datendienste aus einem Flex-Projekt zu, in dem als Anwendungsservertyp PHP festgelegt ist. Der Datendienst muss unter dem Webstamm verfügbar sein, den Sie beim Konfigurieren des Projekts für PHP angegeben haben. Platzieren Sie den Dienst wie unten beschrieben in einem Dienstverzeichnis:

```
<webroot>/MyServiceFolder/services
```

Verwandte Themen

„Erstellen eines Flex-Projekts für den Zugriff auf einen Datendienst“ auf Seite 8

Herstellen einer Verbindung zu PHP-Datendiensten

Bei dieser Prozedur wird davon ausgegangen, dass Sie einen PHP-Dienst implementiert und ein Flex-Projekt für den Zugriff auf PHP-Dienste erstellt haben.

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit PHP verbinden“, um den Dienstassistenten zu öffnen.

2 Navigieren Sie im Dialogfeld „PHP-Dienst konfigurieren“ zur PHP-Datei, die den Dienst implementiert.

Hinweis: *Wenn Sie keinen PHP-Dienst implementiert haben, kann Flash Builder aus einer einzelnen Datenbanktabelle einen Beispieldienst generieren. Verwenden Sie das generierte Beispiel als Muster dafür, wie Sie auf Datendienste zugreifen können. Siehe „Generieren eines PHP-Beispieldiensts aus einer Datenbanktabelle“ auf Seite 12.*

3 (Optional) Ändern Sie die Details des Diensts.

Dienstname	Geben Sie einen Namen für den Dienst ein. Basierend auf dem Dateinamen des Diensts generiert Flash Builder einen Dienstnamen. Die Namen, die Sie für einen Dienst verwenden können, unterliegen bestimmten Einschränkungen. Siehe „Benennen von Datendiensten“ auf Seite 21.
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Datentyppaket	Legen Sie einen Namen für das Paket mit den generierten ActionScript-Dateien, in denen die vom Dienst abzurufenden Datentypen definiert sind, fest. Standardmäßig erstellt Flash Builder das `valueObjects`-Paket.

4 Klicken Sie auf „Weiter“, um die Dienstmethoden anzuzeigen.

Wenn Sie nicht über die unterstützte Version des Zend Framework für den Zugriff auf PHP-Dienste verfügen, fordert Flash Builder Sie auf, die Minimalversion des Zend Framework zu installieren. Siehe „Installieren des Zend Framework“ auf Seite 21.

5 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Dienst zugreifen.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Nächster Schritt: „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

Generieren eines PHP-Beispieldiensts aus einer Datenbanktabelle

Flash Builder kann einen PHP-Beispieldienst generieren, den Sie als Prototyp für Ihre eigenen Dienste verwenden können. Der Beispieldienst greift auf eine einzelne MySQL-Datenbanktabelle zu und verfügt über Methoden zum Erstellen, Lesen, Aktualisieren und Löschen.

Flash Builder konfiguriert Rückgabedatentypen für die generierten Dienste und ermöglicht Datenzugriffsfunktionen wie Paging und Datenverwaltung.

Wichtig: *Verwenden Sie den generierten Dienst nur in einer sicheren Entwicklungsumgebung. Der generierte Code ermöglicht allen Personen mit Netzwerkzugriff auf Ihren Server, auf Daten der Datenbanktabelle zuzugreifen, sowie diese zu ändern und zu löschen. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben von sicheren Diensten finden Sie unter About User Security.*

Bei der folgenden Prozedur wird davon ausgegangen, dass Sie ein Flex-Projekt für den Zugriff auf PHP-Dienste erstellt und eine MySQL-Datenquelle zur Verfügung haben.

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit PHP verbinden“, um den Dienstassistenten zu öffnen.

2 Klicken Sie im Dialogfeld „PHP-Dienst konfigurieren“ auf den Hyperlink zum Generieren eines Beispieldiensts.

3 Klicken Sie auf „Aus Datenbank generieren“ und legen Sie die Informationen für die Verbindung zu einer Datenbank fest. Klicken Sie auf „Datenbankverbindung herstellen“.

Hinweis: *Wenn Sie keine PHP-Datenquelle zur Verfügung haben, wählen Sie „Aus Vorlage generieren“. Flash Builder schreibt ein Beispielprojekt mit typischen Dienstmethoden. Heben Sie die Auskommentierung bestimmter Bereich des Projekts auf und ändern Sie die Methoden, damit Sie einen Beispieldienst erhalten, der Ihnen als Prototyp dienen kann.*

4 Wählen Sie eine Tabelle in der Datenbank aus und geben Sie den Primärschlüssel an.

5 Verwenden Sie den standardmäßigen oder einen neuen Speicherort. Klicken Sie auf „OK“.

Wenn Sie nicht über die unterstützte Version des Zend Framework für den Zugriff auf PHP-Dienste verfügen, fordert Flash Builder Sie auf, die Minimalversion des Zend Framework zu installieren. Siehe „Installieren des Zend Framework“ auf Seite 21.

Flash Builder generiert den Beispieldienst. Ändern Sie den Dienstnamen und die Speicherorte für die Pakete, um die Standardwerte zu überschreiben.

6 (Optional) Klicken Sie auf „Weiter“, um die Methoden des Diensts anzuzeigen.

7 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Beispieldienst zugreifen. Des Weiteren öffnet Flash Builder den Beispieldienst im Editor auf Ihrem System, der für die Bearbeitung von PHP-Dateien festgelegt ist.

Zugriff auf HTTP-Dienste

Stellen Sie mit dem Flash Builder-Dienstassistenten eine Verbindung zu einem REST-basierten HTTP-Dienst her. Sie können über jedes Flex-Projekt eine Verbindung zu HTTP-Diensten herstellen. Sie brauchen keine Servertechnologie für das Projekt anzugeben.

Um von der SWF-Datei der Clientanwendung aus auf Dienste in einer anderen Domäne zugreifen zu können, ist eine Cross-Domain-Richtliniendatei erforderlich. Weitere Informationen finden Sie unter Verwenden von domänenübergreifenden Richtliniendateien.

Konfigurieren von HTTP-Diensten

Beim Zugreifen auf REST-basierte HTTP-Dienste gibt es verschiedene Möglichkeiten, den Zugriff auf HTTP-Dienste zu konfigurieren. Der Assistent „HTTP-Dienst konfigurieren“ unterstützt Folgendes:

- Basis-URL als Präfix

  Die Basis-URL als Präfix ist praktisch, wenn Sie mit einem einzigen Dienst auf mehrere Methoden zugreifen. Wenn Sie eine Basis-URL für den Dienst angeben, brauchen Sie für jede angegebene Methode nur den relativen Pfad zu den HTTP-Methoden anzugeben.

  Wenn Sie auf mehrere Dienste zugreifen möchten, können Sie keine Basis-URL verwenden.

- URLs mit Abfrageparametern

  Wenn Sie eine URL für eine Methode angeben, können Sie die Abfrageparameter für die Dienstmethoden einschließen. Der Assistent „HTTP-Dienst konfigurieren“ füllt die Parametertabelle mit den in der Methoden-URL eingeschlossenen Parametern.

- RESTful-Dienste mit durch Trennzeichen getrennten Parametern

  Flash Builder unterstützt den Zugriff auf RESTful-Dienste, die durch Trennzeichen getrennte Parameter anstelle von GET-Abfrageparametern verwenden. Beispiel: Sie verwenden die folgende URL für den Zugriff auf einen RESTful-Dienst:

`http://restfulService/items/itemID`

Geben Sie in der Methoden-URL die Parameter in geschweiften Klammern an: {}. Beispiel:

`http://restfulService/{items}/{itemID}`

Dann füllt der Assistent „HTTP-Dienst konfigurieren“ die Parametertabelle:

Name	Datentyp	Parametertyp
items	String	URL
itemID	String	URL

Bei der Angabe von Parametern für RESTful-Dienste wird der Datentyp immer als String und der Parametertyp immer als URL konfiguriert.

Hinweis: *Sie können die Parameter von RESTful-Diensten und Abfrageparameter kombinieren, wenn Sie die URL für eine Methode angeben.*

- Pfad zu einer lokalen Datei für eine Methoden-URL

  Bei einer Methoden-URL können Sie einen Pfad zu einer lokalen Datei, die HTTP-Dienste implementiert, angeben. Geben Sie zum Beispiel für eine Methoden-URL Folgendes an:

  `c:/MyHttpServices/MyHttpService.xml`

- Hinzufügen von GET- und POST-Methoden

  Beim Konfigurieren eines HTTP-Diensts können Sie weitere Methoden hinzufügen. Klicken Sie dazu in der Methodentabelle auf „Hinzufügen“.

  Geben Sie als Methode GET oder POST an.

- Hinzufügen von Parametern zu einer Methode

  In der Methodentabelle können Sie ausgewählten Methoden Parameter hinzufügen. Wählen Sie die Methode in der Methodentabelle aus und klicken Sie in der Parametertabelle auf „Hinzufügen“.

  Geben Sie einen Namen und einen Datentyp für den hinzugefügten Parameter an. Der Parametertyp (GET oder POST) entspricht der Methode.

- Inhaltstyp von POST-Methoden

  Bei POST-Methoden können Sie den Inhaltstyp angeben. Der Inhaltstyp kann entweder `application/x-www-form-urlencoded` oder `application/xml` sein.

  Wenn Sie den Inhaltstyp `application/xml` wählen, generiert Flash Builder einen Abfrageparameter, der nicht bearbeitet werden kann. strXML ist der Standardname. Den tatsächlichen Parameter geben Sie zur Laufzeit an.

Name	Datentyp	Parametertyp
strXML	String	POST

Es ist nicht möglich, zusätzliche Parameter für den Inhaltstyp `application/xml` anzugeben.

Herstellen einer Verbindung zu HTTP-Diensten

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit HTTP verbinden“, um den Dienstassistenten zu öffnen.

2 (Optional) Geben Sie eine Basis-URL an, die als Präfix für alle Methoden dient.

3 Legen Sie unter „Methoden“ für jede Methode, auf die Sie zugreifen möchten, die folgenden Informationen fest:

- die Methode (GET oder POST)
- die URL zur Dienstmethode

  Schließen Sie alle Parameter für die Methode in der URL ein. Verwenden Sie geschweifte Klammern (`{}`), um die REST-Stil-Dienstparameter anzugeben.

  Flash Builder unterstützt den Zugriff auf die folgenden Protokolle:

  `http://`

  `https://`

  absolute Standardpfade, wie z. B. `C:/` oder `/Anwendungen/`
- einen Namen für den Vorgang.

4 Geben Sie für jeden Methodenparameter in einer ausgewählten URL den Namen und Datentyp an.

5 (Optional) Klicken Sie auf „Hinzufügen“ oder „Löschen“, um der ausgewählten Methode Parameter hinzuzufügen oder sie daraus zu löschen.

6 (Optional) Ändern Sie die Details des Diensts.

Dienstname	Geben Sie einen Namen für den Dienst ein. Basierend auf dem Dateinamen des Diensts generiert Flash Builder einen Dienstnamen. Die Namen, die Sie für einen Dienst verwenden können, unterliegen bestimmten Einschränkungen. Siehe „Benennen von Datendiensten“ auf Seite 21
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Datentyppaket	Legen Sie einen Namen für das Paket mit den generierten ActionScript-Dateien, in denen die vom Dienst abzurufenden Datentypen definiert sind, fest. Standardmäßig erstellt Flash Builder das `valueObjects`-Paket.

7 (Optional) Ändern Sie den generierten Paketnamen für den Dienst.

8 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Dienst zugreifen.

Nach dem Herstellen der Verbindung zum HTTP-Dienst konfigurieren Sie die Rückgabetypen für die Dienstmethoden. Beim Konfigurieren der Rückgabetypen konfigurieren Sie auch die Parametertypen der Methode. Siehe „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Nächster Schritt: „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

Zugreifen auf XML-Dateien, die HTTP-Dienste implementieren

Sie können auf statische XML-Dateien, die einen HTTP-Dienst implementieren, zugreifen. Die statische XML-Datei kann als URL oder lokal verfügbar sein.

Der Dienst verwendet eine GET-Methode, die eine XML-Antwort zurückgibt. Diese Funktion ist nützlich, um sich mit HTTP-Diensten in Flex vertraut zu machen und um bei Client-Anwendungen Prototypensimulationen durchzuführen.

Wenn Sie auf den Dienst zugreifen, geben Sie den Knoten an, der die XML-Antwort zurückgibt. Flash Builder verwendet diesen Knoten, um automatisch einen Rückgabetyp für die Daten zu erstellen. Nach Herstellen der Verbindung können Sie Methoden an den Dienst für Benutzeroberflächenkomponenten binden.

Herstellen der Verbindung zu einer XML-Dienstdatei

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „HTTP“, um den Dienstassistenten zu öffnen.

2 Geben Sie „Lokale Datei“ oder „URL“ an und navigieren Sie zur Datei.

3 Wählen Sie einen Knoten in der Datei mit der gewünschten Antwort.

Geben Sie an, ob die Antwort ein Array sein soll.

Flash Builder konfiguriert einen Rückgabetyp für den ausgewählten Knoten.

4 Ändern Sie die Details des Diensts.

Dienstname	Geben Sie einen Namen für den Dienst ein. Basierend auf dem Dateinamen des Diensts generiert Flash Builder einen Dienstnamen. Die Namen, die Sie für einen Dienst verwenden können, unterliegen bestimmten Einschränkungen. Siehe „Benennen von Datendiensten“ auf Seite 21
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Datentyppaket	Legen Sie einen Namen für das Paket mit den generierten ActionScript-Dateien, in denen die vom Dienst abzurufenden Datentypen definiert sind, fest. Standardmäßig erstellt Flash Builder das `valueObjects`-Paket.

5 (Optional) Ändern Sie den generierten Paketnamen für den Dienst.

6 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Dienst zugreifen.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Zugriff auf Webservices

Stellen Sie mit dem Flash Builder-Dienstassistenten eine Verbindung zu (SOAP) her. Sie können über jedes beliebige Flex-Projekt eine Verbindung zu Webservices herstellen. Sie brauchen keine Servertechnologie für das Projekt anzugeben.

Um von der SWF-Datei der Clientanwendung aus auf Dienste in einer anderen Domäne zugreifen zu können, ist eine Cross-Domain-Richtliniendatei erforderlich.

Verwandte Themen

Using cross-domain policy files

Herstellen einer Verbindung zu Webservices

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Webservice“, um den Dienstassistenten zu öffnen.

2 (ADEP Data Services/BlazeDS) Wenn Sie ADEP Data Services oder BlazeDS installiert haben, können Sie über einen Proxy auf den Webservice zugreifen.

Wählen Sie „Über ein LCDS/BlazeDS-Proxy-Ziel“.

Geben Sie ein Ziel an. Klicken Sie auf „Weiter“ und fahren Sie mit Schritt 5 fort.

Hinweis: *Der Zugriff auf Webservices über einen Proxy ist nur aktiviert, wenn in Ihrem Flex-Projekt J2EE als Anwendungsservertyp festgelegt ist.*

3 Geben Sie eine URI zum SOAP-Dienst ein.

4 (Optional) Ändern Sie die Details des Diensts.

Dienstname	Geben Sie einen Namen für den Dienst ein. Basierend auf dem WSDL-URI generiert Flash Builder einen Dienstnamen. Die Namen, die Sie für einen Dienst verwenden können, unterliegen bestimmten Einschränkungen. Siehe „Benennen von Datendiensten“ auf Seite 21
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Datentyppaket	Legen Sie einen Namen für das Paket mit den generierten ActionScript-Dateien, in denen die vom Dienst abzurufenden Datentypen definiert sind, fest. Standardmäßig erstellt Flash Builder das `dataValues`-Paket.

5 (Optional) Konfigurieren Sie die Codegenerierung für den Dienst:

Dienst **Port**	Wählen Sie einen der verfügbaren Dienste aus. Basierend auf dem WSDL-URI generiert Flash Builder einen Dienstnamen.
Methodenliste	Wählen Sie die Methoden des Diensts, auf die Sie in Ihrer Clientanwendung zugreifen möchten.

6 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Dienst zugreifen.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Nach dem Herstellen der Verbindung zum Webservice konfigurieren Sie die Rückgabetypen für Dienstmethoden. Ausführliche Informationen finden Sie unter „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

Zugriff auf BlazeDS

Sie können auf BlazeDS-Dienste nur zugreifen, wenn Sie Adobe ® BlazeDS installiert und einen Remote Development Services-(RDS-)Server konfiguriert haben. Informationen zur Installation und Konfiguration von BlazeDS finden Sie in der ADEP Data Services ES-Dokumentation.

Üblicherweise greifen Sie auf BlazeDS-Datendienste aus einem Flex-Projekt zu, in dem als Anwendungsservertyp J2EE festgelegt ist.

Verwandte Themen

„Erstellen eines Flex-Projekts für den Zugriff auf einen Datendienst“ auf Seite 8

Herstellen einer Verbindung zu BlazeDS-Diensten

Bei dieser Prozedur wird davon ausgegangen, dass Sie BlazeDS installiert, einen Remote Development Server konfiguriert und ein Flex-Projekt für den Zugriff auf BlazeDS-Dienste erstellt haben.

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit BlazeDS verbinden“, um den Dienstassistenten zu öffnen.

2 Wählen Sie einen Speicherort aus, von dem importiert werden soll.

3 (Optional) Ändern Sie die Details des Diensts.

Dienstname	Geben Sie einen Namen für den Dienst ein. Basierend auf dem Ziel generiert Flash Builder einen Dienstnamen. Die Namen, die Sie für einen Dienst verwenden können, unterliegen bestimmten Einschränkungen. Siehe „Benennen von Datendiensten“ auf Seite 21
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Datentyppaket	Legen Sie einen Namen für das Paket mit den generierten ActionScript-Dateien, in denen die vom Dienst abzurufenden Datentypen definiert sind, fest. Standardmäßig erstellt Flash Builder das `valueObjects`-Paket.

4 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Dienst zugreifen.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Auf ADEP Data Services zugreifen

Sie können nur auf Dienste zugreifen, die von ADEP Data Services (früher LiveCycle Data Services genannt) verfügbar sind, wenn ADEP Data Services installiert sind und ein Remote Development Services(RDS)-Server konfiguriert ist. Weitere Informationen finden Sie in der Adobe ADEP Data Services-Dokumentation.

Sie können auf ADEP Data Services-Dienste aus einem Flex-Projekt zugreifen, in dem als Anwendungsservertyp J2EE oder ColdFusion festgelegt ist.

Diensttypen für ADEP Data Services

Wenn Sie eine Verbindung zu ADEP Data Services herstellen, sind die folgenden Datendiensttypen als Ziele verfügbar:

- Remoting Service

  Remoting Services werden mithilfe von AMF-Typisierung implementiert. Diese Dienste stellen keine serverseitige Datenverwaltung bereit. Sie können Flash Builder-Werkzeuge verwenden, um clientseitige Datenverwaltung zu konfigurieren. Siehe „Aktivieren der Datenverwaltung“ auf Seite 35.

- Datendienste

Datendienste sind Dienste, die serverseitige Datenverwaltung implementieren. Weitere Informationen finden Sie in der ADEP Data Services-Dokumentation.

- Webservice

  Webservices sind über einen ADEP Data Service-Proxy, der als ADEP Data Services-Ziel konfiguriert ist, verfügbar. Die serverseitige Typisierung wird üblicherweise bei der Verbindung zu einem Webservice nicht bereitgestellt.

Datentypkonfiguration und Datenverwaltung

Flash Builder stellt Werkzeuge für die clientseitige Datenkonfiguration und -verwaltung bereit. Welche Flash Builder-Werkzeuge zur Verfügung stehen, hängt vom Typ des ADEP Data Services-Ziels ab:

- Remoting Service

  Remoting-Dienste implementieren AMF-Typisierung des Diensts. Für Remoting Service-Ziele werden keine Rückgabedatentypen konfiguriert.

  Sie können jedoch Flash Builder verwenden, um Code für die clientseitige Datenverwaltung zu generieren. Siehe „Aktivieren der Datenverwaltung“ auf Seite 35.

- Datendienste

  Datendienste implementieren die serverseitige Datenverwaltung. Für Datendienstziele werden keine Rückgabedatentypen konfiguriert.

  Datendienstziele stellen auch serverseitige Datenverwaltung bereit. Clientseitige Datenverwaltung wird bei Datendienstzielen nicht verwendet.

- Webservice

  Webserviceziele, die über einen ADEP Data Service-Proxy verfügbar sind, implementieren keine serverseitige Typisierung. Sie können Flash Builder-Werkzeuge verwenden, um Rückgabetypen für Webservicemethoden zu konfigurieren. Siehe „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

  Sie können Flash Builder verwenden, um Code für die clientseitige Datenverwaltung zu generieren. Siehe „Aktivieren der Datenverwaltung“ auf Seite 35.

Verbindungen zu ADEP Data Service-Zielen herstellen (Datendienst- und Remoting Service-Ziele)

Bei dieser Prozedur wird davon ausgegangen, dass Sie ADEP Data Services installiert, einen Remote Development Server konfiguriert und ein Flex-Projekt für den Zugriff auf LCDS-Dienste erstellt haben.

1. Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit Daten/Dienst verbinden“, um den Dienstassistenten zu öffnen.
2. Wählen Sie im Dialogfeld „Diensttyp auswählen“ die Option „LCDS“. Klicken Sie auf „Weiter“.
3. Geben Sie bei Bedarf die Anmeldedaten ein.
4. (Optional) Ändern Sie die Details des Diensts.

Dienstname	Sie geben keinen Dienstnamen an. Der Dienstname wird von Flash Builder generiert. Basierend auf dem Ziel generiert Flash Builder einen Dienstnamen.
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Ziele	Legen Sie ein oder mehrere Ziele, die im ADEP Data Services-Server verfügbar sind, fest.
Datentyppaket	Geben Sie den Namen für das Datentyppaket an. Dieses Paket enthält generierte ActionScript-Klassendateien, in denen die vom Dienst abzurufenden Datentypen definiert sind. Standardmäßig erstellt Flash Builder das `valueObjects`-Paket.

5 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Dienst zugreifen.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Verbindungen zu ADEP Data Service-Zielen herstellen (Webserviceziele)

Bei dieser Prozedur wird davon ausgegangen, dass Sie ADEP Data Services installiert, einen Remote Development Server konfiguriert und ein Flex-Projekt für den Zugriff auf DS-Dienste erstellt haben.

1 Wählen Sie im Flash Builder-Menü „Daten“ die Option „Mit Daten/Dienst verbinden“, um den Dienstassistenten zu öffnen.

2 Wählen Sie im Dialogfeld „Diensttyp auswählen“ die Option „Webservice“ aus. Klicken Sie auf „Weiter“.

3 Wählen Sie „Über ein LCDS/BlazeDS-Proxy-Ziel“.

4 Geben Sie bei Bedarf die Anmeldedaten ein.

5 Wählen Sie ein Ziel.

6 (Optional) Ändern Sie die Details des Diensts. Klicken Sie auf „Weiter“.

Dienstname	Geben Sie einen Namen für den Dienst ein. Basierend auf dem Namen des Ziels generiert Flash Builder einen Dienstnamen. Die Namen, die Sie für einen Dienst verwenden können, unterliegen bestimmten Einschränkungen. Siehe „Benennen von Datendiensten“ auf Seite 21
Dienstpaket	Legen Sie einen Namen für das Paket fest, das generierte ActionScript-Dateien enthält, die auf den Dienst zugreifen. Basierend auf dem Dienstnamen generiert Flash Builder ein Paket und legt es in einem `services`-Paket ab.
Datentyppaket	Legen Sie einen Namen für das Paket mit den generierten ActionScript-Dateien, in denen die vom Dienst abzurufenden Datentypen definiert sind, fest. Standardmäßig erstellt Flash Builder das `dataValues`-Paket.

7 (Optional) Konfigurieren Sie die Codegenerierung für den Dienst:

Dienst **Port**	Treffen Sie aus den verfügbaren Diensten und Ports eine Auswahl.
Methodenliste	Wählen Sie die Methoden des Diensts, auf die Sie in Ihrer Clientanwendung zugreifen möchten.

8 Klicken Sie auf „Fertig stellen“.

Flash Builder generiert ActionScript-Dateien, die auf den Dienst zugreifen.

Hinweis: *Nach dem Herstellen der Verbindung zum Dienst können Sie die Diensteigenschaften ändern. Wählen Sie in der Ansicht „Daten/Dienste“ den Dienst aus. Wählen Sie im Kontextmenü „Eigenschaften“ aus.*

Verwandte Themen

„Erstellen eines Flex-Projekts für den Zugriff auf einen Datendienst“ auf Seite 8

Benennen von Datendiensten

Die Namen für einen Dienst, auf den Sie von Flash Builder aus zugreifen, unterliegen bestimmten Einschränkungen. Manche dieser Einschränkungen treten erst beim Kompilieren der Anwendung zutage.

Nachstehend sind die Benennungsrichtlinien für Dienste aufgeführt:

- Das erste Zeichen des Diensts darf keine Ziffer sein.
- Dienstnamen dürfen keine ActionScript-Schlüsselwörter sein.
- Verwenden Sie keine ActionScript-Klassennamen, einschließlich Namen benutzerdefinierter Klassen, als Dienstnamen.
- (Nur PHP) Wenn ein Dienstname einen Unterstrich enthält, kann Flash Builder den Dienst nicht importieren.

Hinweis: *Es wird empfohlen, Dienstnamen zu verwenden, die sich von den Namen Ihrer MXML-Dateien unterscheiden.*

Installieren des Zend Framework

Beim ersten Zugriff auf PHP-Dienste ermittelt Flash Builder, ob die unterstützte Version des Zend Framework installiert ist. Wenn die unterstützte Version des Zend Framework nicht gefunden wird, werden Sie von Flash Builder aufgefordert, die Installation des Zend Framework zu bestätigen. Wenn Sie dies akzeptieren, installiert Flash Builder eine Minimalversion des Zend Framework. Lehnen Sie dies ab, müssen Sie das Zend Framework manuell installieren, wenn Sie auf PHP-Dienste zugreifen möchten.

Flash Builder-Standardinstallation

Flash Builder installiert das Zend Framework im Stammordner Ihres Webservers im Ordner `ZendFramework`.

```
<web root>/ZendFramework/
```

Bei Flex-Projekten, die auf PHP-Dienste zugreifen, erstellt Flash Builder die folgenden Konfigurationsdateien im Projektausgabeordner:

- `amf_config.ini`
- `gateway.php`

Produktionsserver

Adobe empfiehlt, bei Produktionsservern den `ZendFramework`-Ordner nach außerhalb des Webstamms zu verschieben. Aktualisieren Sie die `zend_path`-Variable, die in `amf_config.ini` definiert ist.

Wenn die `zend_path`-Variable auskommentiert ist, heben Sie die Auskommentierung der `zend_path`-Variablen auf. Geben Sie den Speicherort Ihrer Zend Framework-Installation an.

Manuelle Installation des Zend Framework

Sie haben die Möglichkeit, das Zend Framework manuell zu installieren.

1 Laden Sie die neueste Version des Zend Framework herunter.

Sie können entweder die minimale oder die Vollversion installieren. Flash Builder installiert die Minimalversion.

2 Entpacken Sie die heruntergeladene Version an einem Speicherort auf Ihrem System.

3 Aktualisieren Sie im Flex-Projektordner für den Zugriff auf PHP-Dienste die `zend_path`-Variable, die in `amf_config.ini` definiert ist.

Wenn die `zend_path`-Variable auskommentiert ist, heben Sie die Auskommentierung der `zend_path`-Variablen auf. Geben Sie den absoluten Pfad zum Speicherort Ihrer Zend Framework-Installation an.

Fehlerbehebung bei einer Zend Framework-Installation

Im Folgenden einige Tipps für den Fall, dass beim Herstellen einer Verbindung zum Zend Framework ein Fehler auftritt.

Manuelle Installation des Zend Framework

Wenn das Zend Framework manuell installiert wurde, prüfen Sie die zend_path-Variable in der Datei `amf_config.ini`.

Die `amf_config.ini`-Datei befindet sich im Projektausgabeordner.

Prüfen Sie, ob Folgendes zutrifft:

- Die Auskommentierung von `zend_amf` ist aufgehoben.
- Der angegebene Pfad zur Zend Framework-Installation ist korrekt:
  - Es handelt sich um einen absoluten Pfad zu einem Ziel auf dem lokalen Dateisystem. Ein Pfad zu einer zugeordneten Netzwerkressource ist nicht zulässig.
  - Der Pfad geht zum Bibliotheksordner Ihrer Zend Framework-Installation. Normalerweise befindet sich der Bibliotheksordner unter folgendem Pfad:

    (Windows) `C:\apache\PHPFrameworks/ZendFramework/library`

    (Mac OS) `/user/apache/PHP/frameworks/ZendFramework/library`

Flash Builder-Installation des Zend Framework

Wenn das Zend Framework von Flash Builder installiert wurde, überprüfen Sie Folgendes:

- Pfad des Webstammordners

  Flash Builder installiert das Zend Framework im Webstammordner des Projekts. Prüfen Sie den Pfad des Webstammordners Wählen Sie „Projekt“ > „Eigenschaften“ > „Flex-Server“.

- Stellen Sie sicher, dass für den Webserver die Verwendung von PHP konfiguriert ist.
- Prüfen Sie die zend_path-Variable in der Datei `amf_config.ini`.

  Die `amf_config.ini`-Datei befindet sich im Projektausgabeordner.

  Prüfen Sie, ob Folgendes zutrifft:

  - Die Auskommentierung von `zend_path` ist aufgehoben.
  - Der angegebene Pfad zeigt auf die Zend Framework-Installation im Webstamm des Projekts

- Es handelt sich um einen absoluten Pfad zu einem Ziel auf dem lokalen Dateisystem. Ein Pfad zu einer zugeordneten Netzwerkressource ist nicht zulässig.

Verwenden einer einzelnen Serverinstanz

Wenn die Verbindung zu einem Datendienst hergestellt ist, kann jede Anwendung in einem Projekt auf den Dienst zugreifen. Standardmäßig erstellt jede Anwendung ihre eigene Dienstinstanz, wenn sie auf den Server zugreift.

Dieses Verhalten können Sie ändern, sodass je Projekt nur eine einzelne Dienstinstanz vorhanden ist. Alle Anwendungen im Projekt greifen jetzt auf dieselbe Dienstinstanz zu. Üblicherweise erstellen Sie eine einzelne Serverinstanz, wenn Sie den Datenzugriff aus mehreren Anwendungen koordinieren möchten.

Sie können den Zugriff auf eine einzelne Dienstinstanz entweder auf Projektbasis oder als Voreinstellung für alle Projekte festlegen.

Auf eine einzelne Serverinstanz für ein Projekt zugreifen

1 Wählen Sie „Projekt“ > „Eigenschaften“ > „Daten/Dienste“.

2 Aktivieren Sie das Kontrollkästchen für die Verwendung einer einzelnen Serverinstanz. Klicken Sie auf „OK“.

Einzelne Serverinstanz als Präferenz festlegen

1 Öffnen Sie das Dialogfeld „Voreinstellungen“.

2 Wählen Sie „Flash Builder“ > „Daten/Dienste“.

3 Aktivieren Sie das Kontrollkästchen für die Verwendung einer einzelnen Serverinstanz. Klicken Sie auf „OK“.

Erstellen der Clientanwendung

Erstellen Sie mithilfe des MXML-Editors eine Benutzeroberfläche. Sie können den Editor entweder im Design- oder im Quellmodus verwenden.

Nachdem Sie die Komponenten für die Anwendung angeordnet haben, binden Sie die vom Dienst zurückgegebenen Daten an Komponenten der Benutzeroberfläche. Generieren Sie die für die Interaktion der Benutzer mit der Anwendung benötigten Ereignisprozeduren.

Über Dienstmethoden in der Ansicht „Daten/Dienste“ können Sie auch ein Formular generieren.

Verwenden des Designmodus zum Erstellen einer Anwendung

Flex stellt einen umfangreichen Satz an Containern und Steuerelementen zur Erstellung von Benutzeroberflächen bereit. Ein Container stellt eine hierarchische Struktur für die Anordnung und das Layout der Benutzeroberfläche bereit. Innerhalb eines Containers ordnen Sie weitere Container, Navigatoren, Steuerelemente und andere Komponenten an.

Zu den grundlegenden Steuerelementen zählen Oberflächenelemente wie „Button“, „TextArea“ und „CheckBox“. Datenorientierte Steuerelemente wie z. B. „DataGrid“- und „List“-Komponenten eignen sich ideal für das Anzeigen der von Diensten abgerufenen Daten. Navigatoren sind Container, die die Navigation der Benutzer durch untergeordnete Container, wie etwa einen Satz Registerkarten, steuern.

Ziehen Sie im Designmodus des MXML-Editors Container und Steuerelemente per Drag & Drop aus der Komponentenansicht in den Designbereich. Ordnen Sie die Komponenten an und konfigurieren Sie ihre Eigenschaften. Nachdem Sie das Anwendungslayout entworfen haben, binden Sie vom Datendienst zurückgegebene Daten an die Komponenten.

Binden von Dienstmethoden an Steuerelemente

Es gibt mehrere Möglichkeiten, Dienstmethoden an eine Benutzeroberflächenkomponente zu binden. Sie können eine Dienstmethode aus der Ansicht „Daten/Dienste“ auf eine Komponente im Designbereich ziehen. Sie können auch das Dialogfeld „An Daten binden“ öffnen, um eine Methode zum Binden an eine Komponente auszuwählen.

Das Dialogfeld „An Daten binden“ wird in der Ansicht „Daten/Dienste“ über die Symbolleiste geöffnet. Sie erreichen es auch im Designmodus des MXML-Editors, indem Sie eine Komponente, die Daten annimmt (z. B. DataGrid), auswählen. Wenn Sie die Komponente ausgewählt haben, öffnen Sie im Kontextmenü der Komponente das Dialogfeld „An Daten binden“. Das Dialogfeld „An Daten binden“ ist auch in der Ansicht „Eigenschaften“ über das Feld „Datenquelle“ verfügbar.

Wenn Sie eine Dienstmethode an eine Komponente binden, generiert Flash Builder MXML- und ActionScript-Code, um die Dienstmethode über die Clientanwendung aufzurufen.

Rückgabetypen für Dienstmethoden

Wenn Sie eine Dienstmethode an ein Steuerelement binden, verwendet Flash Builder den Datentyp der von der Methode zurückgegebenen Daten. Es kommt häufig vor, dass Sie den Rückgabetyp für eine Dienstmethode konfigurieren, bevor Sie sie an eine Komponente binden.

Wurde der Rückgabetyp für eine Dienstmethode noch nicht konfiguriert, werden Sie im Dialogfeld „An Daten binden“ zur Ausführung dieses Schritts aufgefordert.

Siehe „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

Dienstmethode an ein DataGrid-Steuerelement binden (Drag & Drop)

Bei dieser Prozedur wird angenommen, dass Sie eine Verbindung zu einem Datendienst hergestellt haben.

1 Ziehen Sie im Designmodus des MXML-Editors eine DataGrid-Komponente aus der Komponentenansicht in den Bearbeitungsbereich.

2 Ziehen Sie eine Methode aus der Ansicht „Daten/Dienste“ auf die DataGrid-Komponente.

Wenn der Rückgabetyp für die Methode bereits konfiguriert wurde, bindet Flash Builder die Methode an die DataGrid-Komponente. Die DataGrid-Komponente ändert sich entsprechend und zeigt die von der Datenbank abgerufenen Felder an.

Wenn der Rückgabetyp für die Methode noch nicht konfiguriert wurde, öffnet Flash Builder das Dialogfeld „An Daten binden“. Siehe „DataGrid-Steuerelement an eine Dienstmethode binden (Dialogfeld „An Daten binden“)“ auf Seite 25.

3 Passen Sie die Anzeige der DataGrid-Komponente an.

Siehe DataGrid- und AdvancedDataGrid-Komponenten konfigurieren.

4 Speichern Sie die Anwendung und führen Sie sie aus.

DataGrid-Steuerelement an eine Dienstmethode binden (Dialogfeld „An Daten binden“)

Bei dieser Prozedur wird angenommen, dass Sie eine Verbindung zu einem Datendienst hergestellt haben.

1 Ziehen Sie im Designmodus des MXML-Editors eine DataGrid-Komponente aus der Komponentenansicht in den Bearbeitungsbereich.

2 Wenn Sie die DataGrid-Komponente ausgewählt haben, öffnen Sie das Dialogfeld „An Daten binden“ auf eine der folgenden Arten:

- Wählen Sie im Flash Builder-Menü „Daten“ oder im DataGrid-Kontextmenü oder in der Ansicht „Daten/Dienste“ in der Symbolleiste die Option „An Daten binden“.
- Wählen Sie in der Ansicht „Eigenschaften“ die Schaltfläche „An Daten binden“ (in der Nähe des Felds „Datenquelle“).

3 Wählen Sie „Neuer Dienstaufruf“ und anschließend „Dienst“ und „Methode“ aus.

Wenn Sie früher bereits eine Dienstmethode an eine Komponente gebunden haben, können Sie diese Ergebnisse verwenden. Geben Sie in diesem Fall „Vorhandenes Aufrufergebnis“ an und wählen Sie die Methode, die verwendet werden soll.

4 (Optional) Wählen Sie „Rückgabetyp ändern“ aus:

Wählen Sie „Rückgabetyp ändern“ aus, wenn Sie den Rückgabetyp für die Dienstmethode erneut konfigurieren möchten.

Wenn der Rückgabetyp für die Methode nicht bereits konfiguriert wurde, wählen Sie „Rückgabetyp konfigurieren“ aus.

Siehe „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

5 Klicken Sie auf „OK“.

Die DataGrid-Komponente ändert sich entsprechend und zeigt die von der Datenbank abgerufenen Felder an.

Siehe DataGrid- und AdvancedDataGrid-Komponenten konfigurieren.

6 Speichern Sie die Anwendung und führen Sie sie aus.

Generieren eines Dienstaufrufs für eine Methode

Flash Builder kann eine ActionScript-Methode generieren, die eine Methode eines Diensts aufruft. Die Methode ist nicht an eine Benutzeroberflächenkomponente gebunden, sondern steht zur Verwendung im Anwendungscode bereit

Zusätzlich zum Generieren der ActionScript-Methode erstellt Flash Builder einen CallResponder, der Zugriff auf die vom Dienst zurückgegebenen Daten gewährt. Siehe „Call Responder“ auf Seite 40.

Dienstaufruf für eine Methode generieren

Bei dieser Prozedur wird angenommen, dass Sie eine Verbindung zu einem Datendienst hergestellt haben.

1 Wählen Sie in der Ansicht „Daten/Dienste“ eine Methode.

2 Wählen Sie im Kontextmenü der Methode die Option „Dienstaufruf generieren“.

Flash Builder generiert eine Methode für das Aufrufen der Methode und zeigt die generierte Methode im Quellmodus des MXML-Editors an. Flash Builder erstellt einen CallResponder, in dem die Ergebnisse des Dienstaufrufs enthalten sind.

Diese Option ist auch über die Symbolleiste „Daten/Dienste“ verfügbar.

Generieren eines Formulars für eine Anwendung

Formulare zählen zu den in Webanwendungen am häufigsten verwendeten Methoden zum Erfassen von Benutzerinformationen. Flash Builder kann Formulare für durch Dienstaufrufe abgerufene Daten oder für benutzerdefinierte Datentypen, die für den Zugriff auf Remotedaten verwendet werden, generieren.

Beim Generieren eines Formulars erstellt Flash Builder einen Formularlayoutcontainer und fügt Komponenten zum Anzeigen bzw. Bearbeiten der jeweiligen vom Dienst abgerufenen Daten hinzu.

Flash Builder generiert die folgenden Formulartypen.

Formular	Beschreibung
Datentyp	Enthält Komponenten, die die Felder des Datentyps darstellen.
Master-Detail-Formular	Die Masterkomponente ist üblicherweise ein Datensteuerelement, in dem von einem Dienst abgerufene Daten aufgelistet werden. Das Detailformular stellt individuelle, in der Masterkomponente ausgewählte Elemente dar.
Dienstaufruf	Erstellen Sie zwei Formulare. Das eine Formular gibt die Eingaben für eine Methode an. Das andere Formular zeigt die zurückgegebenen Daten an.

Legen Sie beim Generieren eines Formulars die Felder, die Sie verwenden möchten, und die Typen für die Benutzeroberflächensteuerelemente, die die einzelnen Felder darstellen sollen, sowie die Bearbeitbarkeit des Formulars fest.

Generieren von Formularen

Diese Prozedur zeigt, wie ein Formular für einen Dienstaufruf generiert wird. Die Prozedur zum Generieren anderer Formulartypen ist ähnlich. Diese Prozedur setzt voraus, dass Sie sich im Designmodus des MXML-Editors befinden.

1. Es gibt mehrere Möglichkeiten, den Assistenten „Formular generieren“ auszuführen. Wählen Sie in der Ansicht „Daten/Dienste“ eine Methode:
   - Wählen Sie im Kontextmenü der Methode die Option „Formular generieren“.
   - Wählen Sie im Flash Builder-Menü „Daten“ die Option „Formular generieren“ aus.
   - Ziehen Sie die Methode aus der Ansicht „Daten / Dienste“ auf eine Formularkomponente im Designbereich.
2. Wählen Sie im Assistenten „Formular generieren“ die Option „Formular für Dienstaufruf generieren“ aus.
3. Wählen Sie „Neuer Dienstaufruf“ oder „Vorhandenes Aufrufergebnis“.

   Geben Sie „Vorhandenes Aufrufergebnis“ an, um den früher für einen Dienstaufruf generierten Code zu verwenden.

   Geben Sie andernfalls „Neuer Dienstaufruf“ an und wählen Sie einen Dienst und eine Methode für das Formular.
4. (Optional) Je nach Methode stehen Ihnen mehrere Möglichkeiten zum Generieren des Formulars zur Verfügung.

   Wenn die Methode Parameter akzeptiert, können Sie ein Formular für die Parameter einschließen.

   Wenn die Methode einen Wert zurückgibt, können Sie ein Formular für den Wert einschließen.

   Dieses Formular kann bearbeitbar oder nur leseberechtigt sein.
5. (Optional) Konfigurieren Sie die Eingabe- bzw. Rückgabetypen

   Wenn die gewählte Methode über Eingabeparameter verfügt oder einen Wert zurückgibt, können Sie den Eingabetyp bzw. die Rückgabetypen konfigurieren.

Konfigurieren Sie die Eingabe- und Rückgabetypen der Methode, damit Sie das Formular konfigurieren können. Wenn Sie diese Typen bereits früher konfiguriert haben, können Sie sie hier optional erneut konfigurieren.

Siehe „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

6 Klicken Sie auf „Weiter“. Wählen Sie im Dialogfeld „Zuweisung von Steuerelementen für Eigenschaften“, welche Felder im Formular enthalten sein sollen und welcher Steuerelementtyp die Daten darstellen soll.

7 Klicken Sie auf „Fertig stellen“.

8 Ordnen Sie die generierten Formulare im Designbereich an.

Wenn Flash Builder Formulare generiert, werden diese möglicherweise übereinander platziert. Wählen Sie ein Formular und ziehen Sie es an die gewünschte Position.

Wechseln Sie in den Quellmodus, um sicherzustellen, dass Sie ein Formular ausgewählt und verschoben haben und nicht etwa eine Komponente des Formulars. Wenn Flash Builder ein Formular über einem anderen platziert, kann dies das Auswählen des Formulars erschweren. Wählen Sie im Quellmodus das Tag eines der Formulare. Im Designmodus ist dieses Formular ausgewählt.

Generieren eines Master-Detail-Formulars

Beim Erstellen eines Master-Detail-Formulars fügen Sie der Anwendung zuerst eine Datensteuerkomponente hinzu und binden dann die Ergebnisse einer Methode an das Steuerelement.

Beispiel: Fügen Sie eine DataGrid-Komponente hinzu und binden Sie die Ergebnisse einer Methode wie z. B. `getItems_paged()` an die DataGrid-Komponente.

1 Wählen Sie im Designmodus das Datensteuerelement, z. B. ein DataGrid-Steuerelement.

2 Wählen Sie im Menü „Daten“die Option „Detailformular generieren“ aus.

3 Fahren Sie mit dem Generieren des Formulars fort wie unter „Generieren von Formularen“ beschrieben.

Generieren eines Formulars für einen Datentyp

Konfigurieren Sie zum Generieren eines Formulars mit Komponenten, die die Felder eines benutzerdefinierten Datentyps darstellen, zuerst den Datentyp. Siehe „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

1 Wählen Sie in der Ansicht „Daten/Dienste“ einen benutzerdefinierten Datentyp.

2 Wählen Sie im Kontextmenü „Formular generieren“ aus.

3 Vergewissern Sie sich, dass unter „Formular generieren für“ die Option „Datentyp“ ausgewählt ist und wählen Sie einen Datentyp.

4 (Optional) Sie können festlegen, ob das Formular bearbeitbar sein soll.

5 Klicken Sie auf „Fertig stellen“.

Generieren von Ereignisprozeduren für das Abrufen von Remotedaten

Wenn Sie eine Datendienstmethode an eine Komponente binden, generiert Flash Builder eine Ereignisprozedur, die Daten vom Dienst abruft, um sie in die Komponente einzufügen.

Wenn Sie zum Beispiel eine Methode wie `getAllItems()` an eine DataGrid-Komponente binden, generiert Flash Builder eine `creationComplete`-Ereignisprozedur. Die DataGrid-Komponente verweist auf die generierte Ereignisprozedur. Die Ergebnisse des Aufrufs werden zur Datenquelle für die DataGrid-Komponente.

```
. . .
protected function dataGrid_creationCompleteHandler(event:FlexEvent):void
{
    getAllItemsResult.token = productService.getAllItems();
}
. . .
<mx:DataGrid creationComplete="dataGrid_creationCompleteHandler(event)"
dataProvider="{getAllItemsResult.lastResult}">
. . .
</mx:DataGrid>
. . .
```

Beim Ausführen der Anwendung füllt die Ereignisprozedur die erstellte DataGrid-Komponente mit den vom Dienst abgerufenen Daten.

Beim Generieren von Ereignisprozeduren können Sie die generierten Prozeduren akzeptieren oder bei Bedarf durch andere ersetzen. Beispielsweise können Sie die `creationComplete`-Ereignisprozedur in der DataGrid-Komponente durch eine `creationComplete`-Prozedur in der Anwendung ersetzen.

Sie können auch für Steuerelemente wie Buttons oder Text, die Benutzereingaben akzeptieren, Ereignisprozeduren generieren oder erstellen.

Generieren einer Ereignisprozedur für eine Benutzeroberflächenkomponente

1. Erstellen Sie eine Anwendung, die eine Benutzeroberflächenkomponente, wie z. B. ein DataGrid oder einen Button, enthält.
2. Wechseln Sie zum Designmodus des MXML-Editors.

   Führen Sie einen der folgenden Schritte aus:

   - Ziehen Sie in der Ansicht „Daten/Dienste“ eine Methode auf die Benutzeroberflächenkomponente.
   - Wählen Sie die Benutzeroberflächenkomponente aus und klicken Sie in der Ansicht „Eigenschaften“ auf das Symbol zum Generieren von Ereignisprozeduren. Wählen Sie „Ereignisprozedur generieren“ aus.

     Flash Builder generiert eine Ereignisprozedur für das Standardereignis der Komponente. Bei einem Button wäre die Ereignisprozedur das Click-Ereignis.

     Flash Builder öffnet die Quellansicht des Editors und markiert den generierten Codeabschnitt für die Ereignisprozedur.

     Geben Sie den restlichen Code für die Ereignisprozedur ein. Flash Builder bietet eine Inhaltshilfe, um Ihnen beim Kodieren der Ereignisprozedur zu helfen.

Konfigurieren von Datentypen für Dienstmethoden

Beim Herstellen einer Verbindung zu einem Datendienst muss Flash Builder den Datentyp für von einer Dienstmethode zurückgegebene Daten kennen. Bei den unterstützten Datentypen handelt es sich um die von AMF erkannten Typen für den Austausch von Daten mit einem Daten- oder Remotedienst.

Viele Datendienste definieren den Typ zurückgegebener Daten auf dem Server (serverseitige Typisierung). Wenn der Server den Typ jedoch nicht definiert, muss die Clientanwendung den Typ der zurückgegebenen Daten konfigurieren (clientseitige Typisierung).

Dienstmethoden, die Parameter festlegen, müssen ebenfalls einen Typ festlegen, der den Daten entspricht, auf die im Dienst zugegriffen wird. Bei der clientseitigen Typisierung konfigurieren Sie den Typ für Eingabeparameter.

Wenn Sie Typen für die clientseitige Typisierung konfigurieren, erkennt Flash Builder nur AMF-Datentypen. Beim Typ kann es sich um einen benutzerdefinierten Datentyp handeln, der komplexe Daten darstellt, oder um „void“, um anzugeben, dass die Methode keine Daten zurückgibt.

Sie können benutzerdefinierte Typen für Dienstmethoden, die komplexe Daten zurückgeben, definieren. Wenn Sie zum Beispiel Datensätze aus einer Personaldatenbank abrufen, würden Sie den komplexen Rückgabetyp „Employee“ definieren. In diesem Fall würde der benutzerdefinierte Datentyp für „Employee“ Einträge für alle Felder im Datensatz der Datenbank enthalten.

Datentypen für clientseitige Typisierung

Datentyp	**Beschreibung**
ActionScript-Typen	Boolean Boolean[] ByteArray ByteArray[] Date Date[] int int[] Number Number[] Object Object[] String String[]
No data returned (keine Daten zurückgegeben)	void
User-defined type (benutzerdefinierter Typ)	*CustomType* *CustomType*[]

Benutzerdefinierter Typ (Employee)

Feld	**Datentyp**
emp_no	Number
first_name	String
last_name	String
hire_date	Date
birth_date	Date

Authentifizieren des Zugriffs auf Dienste

Üblicherweise erfordern Datendienste die Authentifizierung der Benutzer, um den Zugriff auf die Dienste zu gewähren. PHP, BlazeDS und ColdFusion-Dienste, die Zugriff über das HTTP-Protokoll gewähren, können eine zusätzliche Authentifizierung erfordern. In manchen Fällen erfordern diese Typen sowohl HTTP- als auch Remoteauthentifizierung.

Flash Builder stellt für folgende Fälle eine Option für die Dienstauthentifizierung bereit:

- Konfigurieren eines Rückgabetyps für eine Methode

  Siehe „Konfigurieren des Rückgabetyps für Daten einer Methode“ auf Seite 31.

- Verwenden der Schnittstelle „Methode testen“

  Siehe „Testen von Dienstmethoden“ auf Seite 33.

Wenn Sie „Authentifizierung erforderlich“ festlegen, öffnet Flash Builder das Dialogfeld „Dienstauthentifizierung“. Je nach Diensttyp, auf den Sie zugreifen, können Sie entweder „Basisauthentifizierung“ oder „Remote-Authentifizierung“ festlegen.

Basisauthentifizierung

Die Basisauthentifizierung bietet Zugriff auf HTTP- und Webservices. Geben Sie den Benutzernamen und das Kennwort für den Zugriff auf diese Dienste an.

Legen Sie „Benutzernamen und Kennwort speichern“ fest, wenn Flash Builder diese Anmeldedaten während der gesamten Sitzung verwenden soll.

Remote-Authentifizierung

Die Remote-Authentifizierung bietet Zugriff auf Remoteobjektdienste. Remoteobjektdienste sind Dienste, die in ColdFusion, PHP, BlazeDS oder ADEP Data Services (früher LiveCycle Data Services genannt) als Remoteobjekte implementiert wurden.

Flash Builder stellt für Projekte, die keine Remoteobjektdienste implementieren, keine Anmeldeschnittstelle für die Remote-Authentifizierung bereit.

Geben Sie den Benutzernamen und das Kennwort für den Zugriff auf Remoteobjektdienste an.

Legen Sie „Benutzernamen und Kennwort speichern“ fest, wenn Flash Builder diese Anmeldedaten während der gesamten Sitzung verwenden soll.

Konfigurieren von Eingabeparametern einer Methode

Für die clientseitige Typisierung konfigurieren Sie die Eingabeparameter einer Methode, die im Datendienst zur Verfügung stehen.

Bei der folgenden Prozedur wird davon ausgegangen, dass Sie in Flash Builder die Verbindung zu einem Datendienst hergestellt haben und der Datendienst über Methoden verfügt, die konfigurierbare Eingabeparameter erfordern.

1. Wählen Sie in der Ansicht „Daten/Dienste“ eine Methode, die konfigurierbare Eingabeparameter enthält. Wählen Sie im Kontextmenü der Methode „Eingabetypen konfigurieren“ aus.
2. Wählen Sie im Dialogfeld „Eingabetypen konfigurieren“ für jedes Argument der Methode den Datentyp in der Liste verfügbarer Typen. Klicken Sie auf „OK“.

   Wenn Sie früher bereits benutzerdefinierte Rückgabedatentypen für den Dienst konfiguriert haben, stehen diese Typen zur Auswahl.

Bei der serverseitigen Typisierung legt der Dienst den Datentyp für die Eingabeparameter fest.

Konfigurieren des Rückgabetyps für Daten einer Methode

Ein Dienst, der von Methoden zurückgegebene Datentypen definiert, stellt serverseitige Typisierung bereit. Wenn ein Dienst den durch eine Methode zurückgegebenen Datentyp nicht definiert, verwendet Flash Builder clientseitige Typisierung, um den zurückgegebenen Datentyp zu definieren.

Flash Builder führt eine Introspektion der durch eine Dienstmethode zurückgegebenen Daten aus, um den Datentyp zu bestimmen. Beim Konfigurieren des Rückgabetyps einer Methode stehen zwei Optionen zur Auswahl:

- Rückgabetyp automatisch aus Beispieldaten ermitteln

  Wenn der Dienst serverseitige Typisierung implementiert, erkennt Flash Builder den vom Dienst definierten Datentyp.

  Wenn der Dienst keine serverseitige Typisierung implementiert, erstellt Flash Builder einen benutzerdefinierten Typ für die Clientanwendung. Geben Sie bei der clientseitigen Typisierung den Namen für den benutzerdefinierten Typ an. Üblicherweise beschreibt der Name die zurückgegebenen Daten. Wenn die Methode etwa ein Array von Büchern aus einer Datenbanktabelle zurückgibt, nennen Sie den Typ „Buch“.

- Vorhandenen Typ verwenden

  Ein vorhandener Typ kann ein durch den Dienst definierter Typ sein, ein ActionScript-Typ oder ein früher definierter benutzerdefinierter Typ.

Die von Flash Builder für die Introspektion verwendeten Prozeduren weisen je nach Datendiensttyp geringfügige Unterschiede auf. Die Prozedur für die Introspektion und Konfiguration des Rückgabetyps für einen HTTP-Dienst ist beispielsweise anders als die Prozedur für PHP- oder ColdFusion-Dienste.

Zusammenführen und Ändern von Datentypen

Während der Introspektion von Serverdaten können Sie Felder von anderen Datentypen zusammenführen oder basierend auf einem vorhandenen Datentyp einen neuen erstellen. Nachstehend sind einige der Vorgehensweisen zum Modifizieren benutzerdefinierter Datentypen aufgeführt:

- Neuen Namen für vorhandenen Datentyp verwenden

  Verwenden Sie einen neuen Namen, wenn Sie die zurückgegebenen Daten in der Clientanwendung auf verschiedene Arten verwenden möchten.

  Beispiel: Abgerufene Personaldaten können in der Clientanwendung in Tabellen für die Mitarbeiterübersicht oder für Mitarbeiterdetails verwendet werden.

- Felder zusammenführen

  Einem vorhandenen Datentyp können zurückgegebene Felder hinzugefügt werden. Das Hinzufügen von Feldern ist bei der Zusammenführung von Daten aus mehreren Quellen sinnvoll. Ein Beispiel dafür ist eine JOIN-Methode, die Daten aus mehreren Datenbanktabellen zurückgibt.

  Ein weiteres Beispiel sind Daten, die von unterschiedlichen Diensten abgerufen werden. Ein Beispiel dafür ist das Zusammenführen der von einem HTTP- und einem ColdFusion-Dienst empfangenen Buchdaten.

Konfigurieren eines benutzerdefinierten Datentyps (PHP- oder ColdFusion-Dienst)

Bei dieser Prozedur wird angenommen, dass Sie eine Verbindung zu einem Datendienst hergestellt haben, der in PHP oder ColdFusion implementiert wurde.

1 Wählen Sie in der Ansicht „Daten/Dienste“ im Kontextmenü einer Methode die Option „Rückgabetyp konfigurieren“ aus.

2 Geben Sie Argumentwerte ein, wenn die Methode über Argumente verfügt. Legen Sie den richtigen Datentyp für die Argumente fest.

3 (Neuer oder geänderter benutzerdefinierter Typ) Wählen Sie die Option zum automatischen Erkennen des von dieser Methode zurückgegebenen Datentyps.

Wenn der Dienst eine Authentifizierung erfordert, wählen Sie „Authentifizierung erforderlich“ und geben Sie die Anmeldedaten ein. Siehe „Authentifizieren des Zugriffs auf Dienste“ auf Seite 30.

Flash Builder führt eine Introspektion der Methode durch und erzeugt einen benutzerdefinierten Datentyp.

Legen Sie einen Namen für den benutzerdefinierten Datentyp fest.

Wenn Sie bereits früher einen benutzerdefinierten Datentyp definiert haben, können Sie die zurückgegebenen Felder der Definition des vorhandenen Typs hinzufügen.

4 (Vorhandenen Typ verwenden) Verwenden Sie diese Option, um einen ActionScript-Typ oder einen früher konfigurierten Typ anzugeben.

5 Klicken Sie auf „Fertig stellen“.

Konfigurieren eines benutzerdefinierten Datentyps (HTTP-Dienst)

Bei dieser Prozedur wird angenommen, dass Sie eine Verbindung zu einem HTTP-Dienst hergestellt haben.

1 Wählen Sie in der Ansicht „Daten/Dienste“ im Kontextmenü einer Methode die Option „Rückgabetyp konfigurieren“ aus.

2 (Neuer benutzerdefinierter Typ) Wählen Sie die Option zum automatischen Erkennen des von dieser Methode zurückgegebenen Datentyps.

Wenn der Dienst eine Authentifizierung erfordert, wählen Sie „Authentifizierung erforderlich“ und geben Sie die Anmeldedaten ein.

Flash Builder führt eine Introspektion der Methode aus und erzeugt einen benutzerdefinierten Datentyp. Wählen Sie eine Methode, mit der Flash Builder Parameterwerte für die Methode übergibt, und klicken Sie auf „Weiter“

- (Parameterwerte eingeben) Geben Sie einen Wert für jeden Parameter ein.

  Sie können auch den Datentyp für die Parameter festlegen. Flash Builder wählt automatisch den Standarddatentyp.

- (Dienst-URL eingeben) Geben Sie die URL zum HTML-Dienst, einschließlich Parameter und Werte in der URL, ein. Beispiel:

  `http://httpserviceaddress/service_operation?param1=94105`

- (XML/JSON-Antwort eingeben) Kopieren Sie die XML/JSON-Antwort in das Textfeld.

  Verwenden Sie diese Option, wenn Sie offline sind oder der HTTP-Dienst noch entwickelt wird, Sie aber die Serverantwort kennen.

3 (Neuer benutzerdefinierter Typ, Fortsetzung) Legen Sie einen Namen für den benutzerdefinierten Datentyp fest oder wählen Sie einen Knoten in den zurückgegebenen Daten.

Wenn Sie einen Knoten für die zurückgegebenen Daten wählen, erstellt Flash Builder einen benutzerdefinierten Datentyp für die für diesen Knoten zurückgegebenen Daten.

Geben Sie an, ob die Daten als Array zurückgegeben werden.

Gibt der Dienst eine XML-Datei zurück, ist die Dropdownliste „Wählen Sie root“ aktiviert. Wählen Sie einen Knoten der XML-Datei, um einen Datentyp anzugeben.

4 (Vorhandenen Typ verwenden) Verwenden Sie diese Option, um einen ActionScript-Typ oder einen früher konfigurierten Typ anzugeben.

5 Klicken Sie auf „Fertig stellen“.

Testen von Dienstmethoden

Sie können Flash Builder zum Testen von Dienstmethoden und Anzeigen der durch eine Methode zurückgegebenen Daten verwenden. Mit dieser Funktion überprüfen Sie das Verhalten von Diensten.

Wichtig: *Manche Methoden, z. B. Aktualisieren und Löschen, ändern Daten auf dem Server.*

Dienstmethode testen

Bei dieser Prozedur wird angenommen, dass Sie eine Verbindung zu einem Datendienst hergestellt haben.

1 Wählen Sie in der Ansicht „Daten/Dienste“ eine Methode in einem Dienst. Wählen Sie im Kontextmenü „Methode testen“ aus.

Die Ansicht „Methode testen“ wird geöffnet und die ausgewählte Methode wird angezeigt. Wenn die Methode Eingabeparameter benötigt, werden die Parameter in der Ansicht „Methode testen“ aufgelistet.

2 Klicken Sie für jeden erforderlichen Parameter auf das Feld „Wert eingeben“ und legen Sie den Parameterwert fest.

Wenn der Parameter einen komplexen Typ erfordert, klicken Sie im Feld auf die Schaltfläche mit der Ellipse, um den Eingabeargumenteditor zu öffnen. Legen Sie im Editor den Wert fest.

Der Eingabeargumenteditor akzeptiert die JSON-Notation für das Darstellen komplexer Typen in ActionScript.

3 Wenn eine Authentifizierung für den Server erforderlich ist, wählen Sie „Authentifizierung erforderlich“. Klicken Sie auf „Test“.

Geben Sie die für die Authentifizierung erforderlichen Anmeldedaten ein. Siehe „Authentifizieren des Zugriffs auf Dienste“ auf Seite 30.

Flash Builder zeigt die durch den Dienst zurückgegebenen Daten an.

4 (Optional) Wählen Sie in der Ansicht „Methode testen“ weitere Dienste und Methoden, die getestet werden können.

Verwalten des Datenzugriffs vom Server

Paging Paging ist das inkrementelle Abrufen großer Datenmengen von einem Remotedienst.

Beispiel: Sie möchten auf eine Datenbank zugreifen, die 10.000 Datensätze enthält, und dann die Daten in einem DataGrid anzeigen, das 20 Reihen hat. Dafür können Sie eine Paging-Methode implementieren, die jeweils 20 Reihen

abruft. Wenn der Anwender weitere Daten anfordert (indem er einen Bildlauf im DataGrid durchführt), werden die nächsten Datensätze abgerufen und auf einer Seite angezeigt.

Datenverwaltung In Flash Builder bedeutet „Datenverwaltung“ die Synchronisierung von Aktualisierungen der Serverdaten über die Clientanwendung. Mithilfe der Datenverwaltung können Sie ein oder mehrere Elemente in einer Clientanwendung ändern, ohne Aktualisierungen auf dem Server vorzunehmen. Sie übernehmen anschließend mit einem einzigen Vorgang alle Änderungen in den Server. Sie können die Änderungen wieder aufheben, ohne Daten zu aktualisieren.

Die Datenverwaltung umfasst die Koordinierung mehrerer Methoden (create, get, update, delete), um auf Ereignisse in der Clientanwendung (z. B. die Aktualisierung eines Personaldatensatzes) zu reagieren.

Wenn Sie die Datenverwaltung durch Flash Builder aktivieren, generiert Flash Builder auch Code, der Benutzeroberflächenkomponenten automatisch aktualisiert. So generiert Flash Builder etwa Code, durch den DataGrids mit Daten auf dem Server synchronisiert bleiben.

Aktivieren der Paging-Funktion

Mithilfe der folgenden Signatur aktivieren Sie Paging für einen Datendienst, der die Pagingfunktion implementiert:

```
getItems_paged(startIndex:Number, numItems:Number): myDataType
```

Funktionsname	Sie können jeden gültigen Namen für die Funktion verwenden.
startIndex	Die erste Datenreihe, die abgerufen werden soll. Definieren Sie den Datentyp für „startIndex“ in der Clientmethode als „Number“.
numItems	Die Anzahl der Datenreihen, die für jede Seite abgerufen werden sollen. Definieren Sie den Datentyp für „numItems“ in der Clientmethode als „Number“.
myDataType	Der Datentyp, der vom Datendienst zurückgegeben wird.

Beim Implementieren von Paging für einen Dienst können Sie auch die `count()`-Methode implementieren. Eine `count()`-Methode gibt die Anzahl der durch den Dienst zurückgegebenen Elemente zurück. Flash Builder erfordert, dass die `count()`-Methode die folgende Signatur implementiert:

```
count(): Number
```

Funktionsname	Sie können jeden gültigen Namen für die Funktion verwenden.
Number	Die Anzahl der von der Methode abgerufenen Datensätze.

Flex verwendet die count-Methode, um Benutzeroberflächenkomponenten, die große Datenmengen abrufen, korrekt anzuzeigen. Beispielsweise hilft die `count()`-Methode dabei, die Größe des Bildlauffelds in der Bildlaufleiste einer DataGrid-Komponente zu bestimmen.

Einige Remotedienste stellen keine `count()`-Methode bereit. Paging funktioniert trotzdem, jedoch stellt das Steuerelement, das die Daten im Pagingformat anzeigt, die Größe der Datenmenge nicht richtig dar.

Paging-Methoden für gefilterte Abfragen

Sie können Paging für Abfragen aktivieren, die Ergebnisse aus der Datenbank filtern. Beim Filtern von Ergebnissen in der Abfrage verwenden Sie diese Signaturen für die Paging- und Zählfunktionen.

```
getItems_pagedFiltered(filterParam1:String, filterParam2:String, startIndex:Number,
numItems:Number): myDataType

countFiltered(filterParam1:String, filterParam2:String)
```

filterParam1	Optionaler Filterparameter. Dieser Parameter ist in „getItems_PagedFiltered()“ und „countFiltered()“ gleich.
filterParam2	Optionaler Filterparameter. Dieser Parameter ist in „getItems_PagedFiltered()“ und „countFiltered()“ gleich.

Es folgt das Codefragment einer `getItems_pagedFiltered()`-Funktion, die in PHP implementiert wird, um auf eine MySQL-Datenbank zuzugreifen. Dieses Codefragment zeigt, wie der optionale Filterparameter zu verwenden ist.

```
get_Items_paged($expression, $startIndex, $numItems) {
    . . .
    SELECT * from employees where name LIKE $expression LIMIT $startIndex, $numItems;
    . . .
}
```

Aktivieren der Paging-Funktion für eine Methode

Bei der im Folgenden beschriebenen Prozedur wird angenommen, dass Sie in Ihrem Remotedienst sowohl die `getItems_paged()`- als auch die `count()`-Methode kodiert haben. Außerdem wird angenommen, dass Sie den Rückgabetyp für die Methode wie in „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28 erläutert konfiguriert haben.

1. Wählen Sie in der Ansicht „Daten/Dienste“ aus dem Kontextmenü für die `getItems_paged`-Methode die Option „Paging aktivieren“ aus.
2. Wenn Sie für Ihren Datentyp noch keinen eindeutigen Schlüssel festgelegt haben, geben Sie die Attribute an, die eine Instanz dieses Datentyps eindeutig identifizieren. Klicken Sie auf „Weiter“.

   Im Allgemeinen ist dieses Attribut der Primärschlüssel.
3. (Optional) Legen Sie die Anzahl der abzurufenden Elemente fest, um eine benutzerdefinierte Seitengröße festzulegen.

   Wenn Sie keine benutzerdefinierte Seitengröße angeben, wird auf Dienstebene eine Standardseitengröße festgelegt. Das Standardseitenformat enthält 20 Datensätze pro Seite.
4. (Optional) Legen Sie die `count()`-Methode fest. Klicken Sie auf „Fertig stellen“.

   Die `count()`-Methode ermöglicht es Flash Builder, die Größe von Benutzeroberflächenelementen, z. B. eines Bildlauffelds in der Bildlaufleiste, richtig anzuzeigen.

   Die Paging-Funktion ist jetzt für diese Methode aktiviert.

   In der Ansicht „Daten/Dienste“ enthält die Signatur der Funktion, die das Paging implementiert, die startIndex- und numItems-Parameter nicht mehr. Flash Builder fügt diese Werte nun dynamisch hinzu. Flash Builder bestimmt diese Werte anhand des von Ihnen festgelegten benutzerdefinierten Seitenformats oder des Standardseitenformats von 20 Datensätzen pro Seite.

Aktivieren der Datenverwaltung

Für die Aktivierung der Datenverwaltung für einen Dienst implementiert der Dienst eine oder mehrere der nachfolgend aufgeführten Funktionen. Die Datenverwaltungsfunktion verwendet diese Funktionen für das Synchronisieren von Updates der Daten auf dem Remoteserver.

- Add (createItem)

  ```
  createItem(item: myDatatype):int
  createItem(item: myDatatype):String
  createItem(item: myDatatype):myDataType
  ```

  Der Rückgabetyp für createItem() hat den Typ des Primärschlüssels der Datenbank.

- Get All Properties (getItem)

```
getItem(itemID:Number): myDatatype
```

- Update (updateItem)

```
updateItem((item: myDataType):void
updateItem((item: myDataType, originalItem: myDataType):void
updateItem((item: myDataType, originalItem: myDataType, changes: String[]):void
```

- Delete (deleteItem)

```
deleteItem(itemID:Number):void
```

Flash Builder erfordert, dass diese Funktionen die folgenden Signaturen aufweisen:

Funktionsname	Sie können jeden gültigen Namen für die Funktion verwenden.
item **originalItem**	Ein Element des Datentyps, das vom Datendienst zurückgegeben wird.
itemID	Eine eindeutige ID für das Element (in der Regel der Primärschlüssel in der Datenbank).
changes[]	Ein Array, der Feldern in einem angegebenen Item entspricht. Dieses Argument wird nur in einer Version von `updateItem()` verwendet.
myDataType	Der Datentyp des Elements, der vom Datendienst verfügbar ist. Im Allgemeinen definieren Sie beim Abrufen der Daten von einem Dienst einen benutzerdefinierten Datentyp.

autoCommit-Flag

Die Datenverwaltung erlaubt Ihnen die Synchronisierung von Datenaktualisierungen auf dem Server. Änderungen der Daten in der Clientanwendung werden auf dem Server erst aktualisiert, wenn Sie die *service*`.commit()`-Methode aufrufen.

Wenn Sie diese Funktion jedoch deaktivieren möchten, setzen Sie das autoCommit-Flag auf „true“. Wenn autoCommit auf „true“ gesetzt ist, werden die Aktualisierungen der Serverdaten nicht zwischengespeichert, sondern sofort wirksam. Siehe „Aktivieren der Datenverwaltung für einen Dienst“ auf Seite 42.

deleteItemOnRemoveFromFill-Flag

Ist die Datenverwaltung aktiviert und Sie löschen Elemente, verwenden Sie das Flag „deleteItemOnRemoveFromFill“ . Standardmäßig ist dieses Flag auf „true“ gesetzt. Wenn Sie ein Element löschen, wird es sofort aus der Datenbank entfernt.

Setzen Sie deleteItemOnRemoveFromFill auf „false“, um das Löschen bis zum Aufruf der commit()-Methode zu verzögern. Das folgende Beispiel zeigt den Code für eine creationComplete-Ereignisprozedur für ein DataGrid. Löscht der Benutzer ein ausgewähltes Employee-Element im DataGrid, so wird dieses erst aus der Datenbank entfernt, wenn die commit()-Methode aufgerufen wird.

```
protected function dg_creationCompleteHandler(event:FlexEvent):void
{
    employeeService.     getDataManager(employeeService.DATA_MANAGER_EMPLOYEE).autoCommit=false;
    employeeService.getDataManager(  e m p l
oyeeService.DATA_MANAGER_EMPLOYEE).deleteItemOnRemoveFromFill=      true;
    getAllEmployeesResult.token = employeeService.getAllEmployees();
}
```

Aktivieren der Datenverwaltung für eine Methode

Bei den im Folgenden beschriebenen Schritten wird angenommen, dass Sie die erforderlichen Methoden in Ihrem Remotedienst implementiert haben. Außerdem wird angenommen, dass Sie den Rückgabedatentyp für die Methoden konfiguriert haben, die einen benutzerdefinierten Datentyp verwenden. Siehe „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

1 Erweitern Sie in der Ansicht „Daten/Dienste“ den Knoten „Datentypen“.

2 Wählen Sie aus dem Kontextmenü für einen Datentyp die Option „Datenverwaltung aktivieren“ aus.

3 Wenn Sie für Ihren Datentyp noch keinen eindeutigen Schlüssel festgelegt haben, geben Sie die Attribute an, die eine Instanz dieses Datentyps eindeutig identifizieren. Klicken Sie auf „Weiter“.

   Im Allgemeinen ist dieses Attribut der Primärschlüssel.

4 Geben Sie die „Add“-, „Get All Properties“-, „Update“- und „Delete“-Methoden an, die Sie implementiert haben. Klicken Sie auf „Fertig stellen“.

   Hinweis: *Sie brauchen nicht alle diese Funktionen zu implementieren. Implementieren Sie nur die Funktionen, die für Ihre Anwendung erforderlich sind.*

Jetzt ist die Datenverwaltung für diese Methode aktiviert.

Generieren des Flash Builder-Codes für Clientanwendungen

Flash Builder generiert Clientcode, der Zugriff auf Remotedienstmethoden bietet. Flash Builder generiert Code, wenn Folgendes erfolgt:

- Herstellen einer Verbindung zu einem Datendienst
- Aktualisieren des Datendienstes in der Ansicht „Daten/Dienste“
- Konfigurieren eines Rückgabetyps für eine Methode
- Dienstmethode an ein Steuerelement der Benutzeroberfläche binden
- Aktivieren von Paging für eine Dienstmethode
- Aktivieren der Datenverwaltung für einen Dienst
- Generieren einer Ereignisprozedur oder eines Dienstaufrufs

Dienstklassen

Stellen Sie mit dem Dienstassistenten eine Verbindung zu einem Datendienst her. Beim Herstellen einer Verbindung zu einem Dienst generiert Flash Builder eine ActionScript-Klassendatei, die Zugriff auf die Dienstmethoden bietet.

Bei Diensten, die auf ein Remoteobjekt zugreifen, erweitert die generierte Klasse die RemoteObjectServiceWrapper-Klasse. Der Zugriff auf ein Remoteobjekt erfolgt üblicherweise durch Dienste, die in PHP, ColdFusion, BlazeDS oder ADEP Data Services implementiert wurden.

Bei HTTP-Diensten erweitert die generierte Klasse die HTTPServiceWrapper-Klasse.

Bei Webservices erweitert die generierte Klasse die WebServiceWrapper-Klasse.

Flash Builder legt dem Namen der generierten Klassendatei den Namen, den Sie im Dienstassistenten für den Dienst angegeben haben, zugrunde. Standardmäßig platziert Flash Builder diese Klasse im Hauptquellordner eines Projekts. Dieser Ordner heißt üblicherweise `src`. Der Name des Pakets basiert auf dem Dienstnamen. Beispielsweise generiert Flash Builder die folgenden ActionScript-Klassen für eine EmployeeService-Klasse.

```
- project
    |
    - src
        |
        + (default package)
        |
        + services
        |   |
        |   +employeeservice
        |       |
        |       + _Super_EmployeeService.as
        |       |
        |       + EmployeeService.as
```

Die Superklasse enthält die Implementierung für die EmployeeService-Klasse.

Bearbeiten Sie niemals die Superklasse. Bei ihr handelt es sich um eine generierte Klasse. Änderungen an der Superklasse können überschrieben werden. Änderungen, die Sie an der Implementierung vornehmen, können nicht definiertes Verhalten zur Folge haben.

Verwenden Sie im unten stehenden Beispiel `EmployeeService.as` zum Erweitern der generierten Superklasse und zum Hinzufügen zu Ihrer Implementierung.

Verwandte Themen

„Verbindung zu Datendiensten herstellen“ auf Seite 9

Klassen für benutzerdefinierte Datentypen

Viele Remotedatendienste stellen serverseitige Typisierung zur Verfügung. Diese Dienste geben komplexe Daten als benutzerdefinierten Datentyp zurück.

Für Remotedatendienste, die keine typisierten Daten zurückgeben, stellt Flash Builder clientseitige Typisierung zur Verfügung. Bei clientseitiger Typisierung wird mit dem Flash Builder-Verbindungsassistenten der Datentyp für komplexe, vom Dienst zurückgegebene Daten konfiguriert. Beispiel: Für einen Dienst, der Mitarbeiterdatensätze zurückgibt, definieren und konfigurieren Sie einen Employee-Datentyp.

Flash Builder generiert eine ActionScript-Klasse für die Implementierung des jeweiligen, vom Dienst zurückgegebenen benutzerdefinierten Datentyps. Flash Builder erstellt mithilfe dieser Klasse Wertobjekte, mit deren Hilfe anschließend über den Remotedienst auf Daten zugegriffen wird.

Beispielsweise generiert Flash Builder die folgenden ActionScript-Klassen für eine EmployeeService-Klasse, die einen Employee-Datentyp enthält:

```
- project
    |
    - src
        |
        + (default package)
        |
        + services
        |   |
        |   +employeeservice
        |       |
        |       + _Super_EmployeeService.as
        |       |
        |       + EmployeeService.as
        |
        + valueObjects
            |
            + _Super_Employee.as
            |
            + Employee.as
```

Die jeweiligen Superklassen enthalten die Implementierung für den EmployeeService- und den Employee-Datentyp.

Bearbeiten Sie niemals eine generierte Superklasse. Änderungen an der Superklasse können überschrieben werden. Änderungen, die Sie an der Implementierung vornehmen, können nicht definiertes Verhalten zur Folge haben.

Verwenden Sie in diesem Beispiel `EmployeeService.as` und `Employee.as` zum Erweitern der generierten Superklasse und fügen Sie Ihre Implementierung hinzu.

Dienstmethode an ein Steuerelement der Benutzeroberfläche binden

„Binden von Dienstmethoden an Steuerelemente“ auf Seite 24 zeigt, wie Sie von Dienstmethoden zurückgegebene Daten an ein Steuerelement der Benutzeroberfläche binden können. Wenn Sie eine Dienstmethode an ein Steuerelement binden, generiert Flash Builder den folgenden Code:

- Declarations-Tag, das ein CallResponder- und Dienst-Tag enthält
- Ereignisprozedur für das Aufrufen des Dienstaufrufs
- Datenbindung zwischen dem Steuerelement und den von einer Methode zurückgegebenen Daten

Declarations-Tag

Ein Declarations-Tag ist ein MXML-Element, das nicht standardmäßige, nicht visuelle Eigenschaften der aktuellen Klasse deklariert. Bei der Bindung einer Dienstmethode an eine Benutzeroberfläche generiert Flash Builder ein Declarations-Tag, das ein CallResponder- und ein Dienst-Tag enthält. Die CallResponder- und generierte Dienstklasse sind Eigenschaften des Containerelements, das im Allgemeinen ein Application-Tag ist.

Das folgende Beispiel zeigt ein Declarations-Tag, das Zugriff auf einen Remote-EmployeeService bietet:

```
<fx:Declarations>
    <s:CallResponder id="getAllEmployeesResult"/>
    <employeesservice:EmployeesService id="employeesService"
        fault="Alert.show(event.fault.faultString + '\n'
        + event.fault.faultDetail)" showBusyCursor="true"/>
</fx:Declarations>
```

Call Responder

Ein CallResponder verwaltet Ergebnisse für Aufrufe eines Dienstes. Er enthält eine Token-Eigenschaft, die auf den von einem Dienstaufruf zurückgegebenen Async-Token festgelegt wird. Der CallResponder enthält auch eine lastResult-Eigenschaft, die auf das letzte erfolgreiche Ergebnis des Dienstaufrufs festgelegt wird. Sie fügen dem CallResponder Ereignisprozeduren hinzu, um Zugriff auf die über die lastResult-Eigenschaft zurückgegebenen Daten zu bieten.

Wenn Flash Builder einen CallResponder generiert, wird eine ID-Eigenschaft basierend auf dem Namen der Dienstmethode erstellt, an die er gebunden ist. Das folgende Codebeispiel zeigt CallResponder für zwei Methoden eines EmployeeService. Die `getAllItems`-Methode wird an die creationComplete-Ereignisprozedur für eine DataGrid-Komponente gebunden. Die Delete-Methode wird an das ausgewählte Element im DataGrid gebunden. Das DataGrid zeigt die vom `getAllItems`-Dienstaufruf unmittelbar nach seiner Erstellung abgerufenen Elemente an. Das Button-Steuerelement zum Löschen von Elementen entfernt den ausgewählten Datensatz im DataGrid aus der Datenbank.

```
<fx:Script>
    <![CDATA[
        import mx.events.FlexEvent;
        import mx.controls.Alert;

        protected function dg_creationCompleteHandler(event:FlexEvent):void
        {
            getAllItemsResult.token = employeesService.getAllItems();
        }

        protected function button_clickHandler(event:MouseEvent):void
        {
            deleteItemResult.token =
                employeesService.deleteItem(dg.selectedItem.emp_no);
        }
    ]]>
</fx:Script>

<fx:Declarations>
    <s:CallResponder id="getAllItemsResult"/>
    <employeesservice:EmployeesService id="employeesService"
        fault="Alert.show(event.fault.faultString + '\n'
        + event.fault.faultDetail)" showBusyCursor="true"/>
    <s:CallResponder id="deleteItemResult"/>
</fx:Declarations>
<mx:DataGrid id="dg" editable="true"

creationComplete="dg_creationCompleteHandler(event)"dataProvider="{getAllItemsResult.lastRes
ult}">
    <mx:columns>
        <mx:DataGridColumn headerText="emp_no" dataField="emp_no"/>
        <mx:DataGridColumn headerText="last_name" dataField="last_name"/>
        <mx:DataGridColumn headerText="hire_date" dataField="hire_date"/>
    </mx:columns>
</mx:DataGrid>
<s:Button label="Delete Item" id="button" click="button_clickHandler(event)"/>
```

Ereignisprozeduren

Wenn Sie eine Dienstmethode an eine Komponente der Benutzeroberfläche binden, generiert Flash Builder eine Ereignisprozedur für den CallResponder. Die Ereignisprozedur verwaltet die Ergebnisse der Methode. Sie können auch eine Ereignisprozedur in einem ActionScript-Codeblock erstellen und auf diese Ereignisprozedur über eine Eigenschaft einer Benutzeroberflächenkomponente verweisen.

Im Allgemeinen fügen Sie von einem Dienst zurückgegebene Daten in Steuerelemente wie Listen und DataGrids ein. Flash Builder generiert standardmäßig eine creationComplete-Ereignisprozedur für das Steuerelement, das unmittelbar nach Erstellen des Steuerelements ausgelöst wird. Für andere Steuerelemente generiert Flash Builder eine Prozedur für das Standardereignis des Steuerelements. Bei einem Button generiert Flash Builder beispielsweise ein Ereignis für das Button-Klickereignis.

Die Ereigniseigenschaft des Steuerelements wird auf die generierte Ereignisprozedur gesetzt. Das folgende Beispiel zeigt die generierte creationComplete-Ereignisprozedur für ein DataGrid:

```
<fx:Script>
    <![CDATA[
        import mx.events.FlexEvent;
        import mx.controls.Alert;

        protected function dg_creationCompleteHandler(event:FlexEvent):void
        {
            getAllItemsResult.token = employeesService.getAllItems();
        }
    ]]>
</fx:Script>
. . .

<mx:DataGrid id="dg" editable="true"
        creationComplete="dg_creationCompleteHandler(event)"
        dataProvider="{getAllItemsResult.lastResult}">
    <mx:columns>
        <mx:DataGridColumn headerText="emp_no" dataField="emp_no"/>
        <mx:DataGridColumn headerText="last_name" dataField="last_name"/>
        <mx:DataGridColumn headerText="hire_date" dataField="hire_date"/>
    </mx:columns>
</mx:DataGrid>
```

Sie können Ereignisprozeduren für Steuerelemente generieren, die auf Benutzerereignisse reagieren, beispielsweise Buttons. Das folgende Beispiel zeigt eine generierte Ereignisprozedur für einen Button, der ein DataGrid ausfüllt:

```
<fx:Script>
    <![CDATA[
        import mx.events.FlexEvent;
        import mx.controls.Alert;

        protected function button_clickHandler(event:MouseEvent):void
        {
            deleteItemResult.token =
                employeesService.deleteItem(dg.selectedItem.emp_no);
        }
    ]]>
</fx:Script>
. . .

<s:Button label="Delete Item" id="button" click="button_clickHandler(event)"/>
```

Datenbindung

Beim Erzeugen einer Benutzeroberfläche binden Sie Dienstmethoden an Steuerelemente. Siehe „Binden von Dienstmethoden an Steuerelemente“ auf Seite 24.

Flash Builder generiert Code, der die von einer Dienstmethode zurückgegebenen Daten an das Steuerelement der Benutzeroberfläche, das die Daten anzeigt, bindet.

Im folgenden Beispiel wird Code gezeigt, den Flash Builder für das Füllen des DataGrid-Steuerelements mit Daten generiert. Die `getAllItems()`-Methode gibt Mitarbeiterdatensätze für den benutzerdefinierten Datentyp „Employee“ zurück.

Die dataProvider-Eigenschaft des DataGrid ist an die im CallResponder (getAllItemsResult) gespeicherten Ergebnisse gebunden. Flash Builder aktualisiert das DataGrid automatisch mit DataGridColumns, die jedem für einen Employee-Datensatz zurückgegebenen Feld entsprechen. Die headerText- und dataField-Eigenschaften jeder Spalte werden entsprechend den in einem Employee-Datensatz zurückgegebenen Daten festgelegt.

```
<mx:DataGrid creationComplete="datagrid1_creationCompleteHandler(event)"
    dataProvider="{getAllItemsResult.lastResult}" editable="true">
        <mx:columns>
            <mx:DataGridColumn headerText="gender" dataField="gender"/>
            <mx:DataGridColumn headerText="emp_no" dataField="emp_no"/>
            <mx:DataGridColumn headerText="birth_date" dataField="birth_date"/>
            <mx:DataGridColumn headerText="last_name" dataField="last_name"/>
            <mx:DataGridColumn headerText="hire_date" dataField="hire_date"/>
            <mx:DataGridColumn headerText="first_name" dataField="first_name"/>
        </mx:columns>
    </mx:DataGrid>
```

Aktivieren von Paging für eine Dienstmethode

Wenn Sie Paging aktivieren, ändert Flash Builder die Implementierung des generierten Dienstes. Wenn Sie in ein Datensteuerelement (beispielsweise ein DataGrid oder eine Liste) Daten im Paging-Format einfügen, legt Flash Builder die Anzahl der im Datensteuerelement sichtbaren Datensätze und die Gesamtzahl der Datensätze in der Datenbank fest. Flash Builder stellt diese Werte als Argumente für die Dienstmethode bereit, die Sie zum Implementieren des Paging verwendet haben.

Nachdem das Paging aktiviert wurde, brauchen Sie keinen Clientanwendungscode zu ändern.

Weitere Informationen finden Sie unter „Aktivieren der Paging-Funktion“ auf Seite 34.

Aktivieren der Datenverwaltung für einen Dienst

In Flash Builder wird unter Datenverwaltung die Synchronisierung einer Gruppe von Datenaktualisierungen auf dem Server verstanden. Sie können die Datenverwaltung für vom Dienst zurückgegebene, benutzerdefinierte Datentypen aktivieren. Bei aktivierter Datenverwaltung können Sie ein oder mehrere Elemente in einer Clientanwendung ändern, ohne hierzu Aktualisierungen auf dem Server vornehmen zu müssen. Sie übernehmen anschließend mit einem einzigen Vorgang alle Änderungen in den Server. Sie können die Änderungen auch wieder aufheben, ohne Daten auf dem Server zu aktualisieren. „Aktivieren der Datenverwaltung“ auf Seite 35 zeigt, wie diese Funktion implementiert wird.

Wenn Sie die Datenverwaltung aktivieren, ändert Flash Builder die Implementierung der generierten Dienstklasse und die generierte Klasse für benutzerdefinierte Datentypen. Flash Builder erstellt einen DataManager zur Implementierung dieser Funktionen.

Synchronisieren von Aktualisierungen der Serverdaten

Wenn Sie Dienstmethoden für einen verwalteten Datentyp aufrufen, werden die Änderungen in der Clientanwendung übernommen. Sie können jedoch festlegen, dass Daten auf dem Server erst aktualisiert werden, wenn Sie die `commit()`-Methode des DataManagers aufrufen.

Wenn für einen Dienst die Datenverwaltung aktiviert ist, verfügt er über ein autoCommit-Flag. Standardmäßig hat autoCommit den Wert „false“.

Das autoCommit-Flag bestimmt, ob Änderungen sofort übernommen werden oder darauf gewartet wird, dass *`service`*`.commit()` aufgerufen wird.

Wenn autoCommit auf „false“ gesetzt ist, werden alle Aktualisierungen des Diensts in der Clientanwendung zwischengespeichert, bis *`service`*`.commit()` aufgerufen wird. Zum Aufheben von Änderungen rufen Sie die `revertChanges()`-Methode eines Diensts auf.

Wenn autoCommit auf „true“ gesetzt ist, werden die Aktualisierungen sofort an den Server gesendet. `revertChanges()` kann nicht zum Verwerfen von Änderungen aufgerufen werden.

Das deleteItemOnRemoveFromFill-Flag bestimmt, ob gelöschte Elemente sofort aus der Datenbank entfernt werden. Lautet die Einstellung „true“, wird das Element erst beim Aufruf von *`service`*`.commit()` gelöscht.

Der folgende Code deaktiviert in der Datenverwaltung die Synchronisierung von Serverdatenaktualisierungen. Änderungen an Daten des verwalteten Typs werden direkt auf dem Server aktualisiert.

```
bookService.getDataManager(bookService.DATA_MANAGER__BOOK).autoCommit = true;
```

Der folgende Code ermöglicht der Datenverwaltung die Synchronisierung von Aktualisierungen der Serverdaten. Änderungen an Daten des verwalteten Typs werden erst aktualisiert, wenn für den Dienst `commit()` aufgerufen wird. Weiterhin werden gelöschte Elemente erst aus der Datenbank entfernt, wenn `commit()` aufgerufen wird.

```
bookService.getDataManager(bookService.DATA_MANAGER__BOOK).autoCommit = false;
bookService.getDataManager(bookService.DATA_MANAGER__BOOK).deleteItemOnRemoveFromFill= true;
```

Aufheben von Änderungen

Der DataManager stellt die `revertChanges()`-Methode bereit. Die `revertChanges`-Methode setzt die in der Clientanwendung angezeigten Daten auf die Werte zurück, die vor dem letzten Übernahmeaufruf vom Server abgerufen wurden.

Rufen Sie `revertChanges()` auf, bevor Sie mithilfe von `commit()` die Änderungen eines verwalteten Datentyps in der Clientanwendung aufheben:

```
bookService.getDataManager (bookService.DATA_MANAGER_BOOK).revertChanges();
```

Rufen Sie die `commit()`-Methode auf, um die Änderungen am verwalteten Datentyp festzuschreiben.

```
bookService.getDataManager (employeeService.DATA_MANAGER_EMPLOYEE).commit();
```

Sie können die `commit()`-Methode auch direkt über die bookService-Instanz aufrufen. Durch Aufrufen der Commit-Methode direkt über die Dienstinstanz werden alle Änderungen für alle verwalteten Datentypen übernommen.

```
bookService.commit();
```

Hinweis: *Sie können `revertChanges()` nicht direkt aus der Dienstinstanz aufrufen, um Änderungen aller verwalteten Datentypen aufzuheben. Die Methode kann nur für einen bestimmten verwalteten Datentyp aufgerufen werden.*

Wenn Sie für die Datenverwaltung das Standardverhalten überschreiben und die Möglichkeit, Änderungen wiederherzustellen deaktivieren möchten, setzen Sie autoCommit auf „true“. Wenn Sie beispielsweise eine Instanz von „bookService“ und die Datenverwaltung für den Datentyp „Book“ aktiviert haben, setzen Sie autoCommit auf „true“:

```
bookService.getDataManager(bookService.DATA_MANAGER__BOOK).autoCommit = true;
```

Jetzt werden Änderungen an Daten des verwalteten Typs direkt auf dem Server aktualisiert.

Bereitstellen von Anwendungen, die auf Datendienste zugreifen

Wenn Sie Ihre Anwendung von der Entwicklungsumgebung in eine Bereitstellungsumgebung übertragen, sind mehrere Faktoren zu berücksichtigen. Der Prozess der Bereitstellung einer Anwendung ist von der Anwendung, den Anforderungen der Anwendung und der Bereitstellungsumgebung abhängig.

Beispiel: Der Prozess der Bereitstellung einer Anwendung auf einer internen Website, auf die nur die Mitarbeiter des Unternehmens zugreifen können, unterscheidet sich vom Prozess der Bereitstellung derselben Anwendung auf einer öffentlichen Website.

Unter Deploying applications erhalten Sie einen Überblick über die Faktoren, die zu beachten sind, sowie eine Deployment checklist. In der Prüfliste werden einige allgemeine Systemkonfigurationsprobleme diskutiert, auf die Kunden bei der Bereitstellung von Anwendungen für die Produktion gestoßen sind. Die Dokumentation enthält auch Tipps zur Diagnose häufiger Bereitstellungsprobleme und zur Fehlerbehebung.

Empfohlene Vorgehensweisen für das Kodieren des Zugangs zu Diensten

Mithilfe von Flash Builder-Werkzeugen können Sie Clientcode für den Zugriff auf Daten in Datenbanken generieren. Diese Funktion steht für PHP und ColdFusion zur Verfügung. Dieser Code eignet sich jedoch nur für die Erzeugung von Prototypen. Verwenden Sie diesen Code nicht als Vorlage für das Schreiben sicherer Anwendungen.

Standardmäßig ermöglicht der generierte Code allen Personen mit Netzwerkzugriff auf Ihren Server, Daten der Datenbanktabelle auszuwählen, zu aktualisieren, zu löschen sowie Daten einzufügen. Nachfolgend sind empfohlene Vorgehensweisen für das Ändern des generierten Codes oder das Schreiben von Code, der auf Dienste zugreift, aufgeführt. Weitere Informationen finden Sie unter Securing Data Services.

Entfernen Sie nicht verwendete Funktionen

Löschen Sie Funktionen, die in Ihrer Anwendung voraussichtlich nicht verwendet werden, oder kommentieren Sie sie aus.

Fügen Sie eine Authentifizierungsfunktion hinzu

Fügen Sie Benutzerauthentifizierung hinzu, um sicherzustellen, dass nur vertrauenswürdige Benutzer auf die Datenbankinformationen zugreifen können.

Fügen Sie Autorisierungsprüfungen hinzu

Falls die Autorisierung nötig ist, fügen Sie Autorisierungsprüfungen hinzu. Selbst wenn Sie die Authentifizierung von Benutzern für Ihre Anwendung erforderlich gemacht haben, sollten Sie dennoch sicherstellen, dass die Benutzer auch zur Ausführung bestimmter Abfragen berechtigt sind.

Beispiel: Sie erlauben allen Benutzern die Auswahl, beschränken aber die Berechtigung für das Löschen.

Ein weiteres Beispiel ist die Autorisierung von Benutzer A, seine eigenen Informationen durch eine bestimmte Abfrage abzurufen. Benutzer A ist aber nicht berechtigt, mithilfe einer bestimmten Abfrage die Informationen von Benutzer B abzurufen.

Daten validieren

Sie sollten unbedingt Datenvalidierung hinzufügen. Validieren Sie beispielsweise Daten, die an Einfügeanweisungen übergeben werden, um sicherzustellen, dass keine beschädigten oder bösartigen Daten in die Datenbank gelangen.

Die clientseitige Validierung kann Sie nicht davor schützen, dass jemand eine manuelle Anfrage an Ihren Webserver sendet. Die Validierung von Daten schützt Sie vor Hackern und stellt die Qualität der gespeicherten Informationen sicher.

Beschränken Sie die abgerufene Datenmenge

Select-Methoden ermöglichen die Auswahl aller Elemente in einer Tabelle. In manchen Fällen führt dies dazu, dass zu viele Informationen über das Netzwerk übertragen werden. Rufen Sie nur die benötigten Daten ab.

Wenn Sie etwa `SELECT *` auf eine Benutzertabelle anwenden, können Sie Benutzernamen und Kennwort über das Netzwerk abrufen.

Ziehen Sie bei sensiblen Daten die Verwendung von SSL in Betracht

Die Zuhilfenahme eines Sicherheitsprotokolls garantiert die Datensicherheit beim Senden.

Exportieren von Quelldateien mit der Release-Version einer Anwendung

Beim Exportieren des Releasebuild einer Anwendung bietet Flash Builder die Option „Quelle anzeigen aktivieren“. Diese Option lässt es zu, dass die Benutzer die Quelldateien anzeigen können, die die Anwendung implementieren. Bei Serverprojekten enthalten die Quelldateien den `services`-Ordner mit den Dateien, die Zugriff auf Ihre Dienstimplementierung gewähren.

Wichtig: *Seien Sie vorsichtig, wenn Sie Dienstdateien mit der Option „Quelle anzeigen aktivieren“ einschließen. Die Dienstdateien enthalten Informationen für den Zugriff auf Ihre Datenbank und möglicherweise auch vertrauliche Daten wie Benutzernamen und Kennwörter. Wenn bei der Option „Quelle anzeigen“ Dienste eingeschlossen werden, haben alle Benutzer mit Zugriff auf die gestartete Anwendung auch Zugriff auf vertrauliche Daten.*

Verwandte Themen

Flex Security

Schreiben sicherer Dienste

Die Beispiele der Adobe-Dokumentation, einschließlich der Übungen und der mit Flash Builder-Codegenerierung erstellten Anwendungen, sollen als Modelle für verschiedene Vorgehensweisen dienen. Sie demonstrieren, wie Sie aus einer Clientanwendung auf Datendienste zugreifen. Allerdings sind sie im Hinblick auf gute Verständlichkeit geschrieben, nicht als empfohlene Vorgehensweisen für den sicheren Datenzugriff.

Die Flash Builder-Dokumentation enthält Beispiele u. a. für die Erstellung von Anwendungen aus generiertem Code. Diese Beispiele sollten in sicheren Entwicklungsumgebungen bereitgestellt werden. Eine sichere Entwicklungsumgebung ist ein lokaler Rechner oder ein Speicherort innerhalb einer Firewall. Wenn keine weiteren Sicherheitsmaßnahmen vorhanden sind, kann jede Person mit Netzwerkzugriff auch auf Ihre Datenbank zugreifen.

Im Folgenden werden einige empfohlene Vorgehensweisen zum Schreiben von Diensten genannt:

- Authentifizieren Sie Benutzer, bevor Methoden für einen Dienst aufgerufen werden.
- Verwenden Sie Dienstauthentifizierung, um bestimmte Aktionen nur bestimmten Benutzern zu erlauben.

  Nehmen Sie beispielsweise an, Sie verfügten über eine Anwendung, die die Änderung von Mitarbeiterdaten über einen RemoteObject-Aufruf ermöglicht. Verwenden Sie in diesem Fall die RemoteObject-Authentifizierung, um sicherzustellen, dass nur Manager die Personaldaten ändern können.

- Verwenden Sie programmgesteuerte Sicherheitsmaßnahmen, um den Zugriff auf Dienste zu beschränken.
- Wenden Sie deklarative Sicherheitsbeschränkungen auf ganze Dienste an.
- Wenn Sie auf einen Webservice (`<mx:WebService>`) oder HTTP-Dienst (`<mx:HTTPService>`) zugreifen, muss eine der folgenden Voraussetzungen erfüllt sein:
  - Die Dienstimplementierung befindet sich in derselben Domäne wie die aufrufende Anwendung.
  - Das Hostsystem für den Dienst verfügt über eine `crossdomain.xml`-Datei, die den Zugriff auf die Anwendungsdomäne explizit erlaubt.

Verwandte Themen

Flex Security

Securing Data Services

Writing secure applications

Adobe® Flash® Player führt in Flash erstellte Anwendungen aus. Inhalte werden als Abfolgen von Anweisungen im Binärformat über Webprotokolle in einem genau beschriebenen SWF-Dateiformat an Flash Player übermittelt. Die SWF-Dateien selbst werden üblicherweise auf einem Server gehostet und auf Abfrage auf den Clientcomputer heruntergeladen und dort angezeigt. Die meisten Inhalte bestehen aus binären ActionScript-Anweisungen. ActionScript ist die von Flash verwendete, auf dem ECMA-Standard basierende Skripterstellungssprache. ActionScript verfügt über APIs, die die Erstellung und Bearbeitung clientseitiger Benutzeroberflächenelemente und das Arbeiten mit Daten ermöglichen.

Das Sicherheitsmodell für Flex schützt sowohl den Client als auch den Server. Berücksichtigen Sie die beiden folgenden allgemeinen Sicherheitsaspekte:

- Autorisierung und Authentifizierung von Benutzern, die auf Serverressourcen zugreifen
- Ausführung von Flash Player in einer Sandbox auf dem Client

Flex unterstützt das Arbeiten mit der Webanwendungssicherheit eines beliebigen J2EE-Anwendungsservers. Darüber hinaus können in Flex vorkompilierte Anwendungen mit der darunter liegenden Servertechnologie für die Authentifizierung und Autorisierung integriert werden, um den Zugriff von Benutzern auf Ihre Anwendungen zu verhindern. Im Flex-Framework sind auch Sicherheitsfunktionen integriert, die Ihnen die Zugriffssteuerung auf Webservices, HTTP-Dienste und serverbasierte Ressourcen wie EJBs ermöglichen.

Flash Player wird innerhalb einer Sicherheitssandbox ausgeführt, die verhindert, dass sich bösartiger Anwendungscode des Clients bemächtigt.

Hinweis: *Bei SWF-Inhalten, die in Adobe® AIR® ausgeführt werden, gelten andere Sicherheitsregeln als bei Inhalten, die im Browser ausgeführt werden. Weitere Informationen finden Sie im Thema „AIR-Sicherheit“ in der AIR-Dokumentation.*

Hyperlinks zu verschiedenen Sicherheitsthemen finden Sie im Security Topic Center der Adobe Developer Connection.

Verwandte Themen

Flex Security

Kapitel 3: Implementieren von Diensten für datenorientierte Anwendungen

Action Message Format (AMF)

Mithilfe von Remoteobjektdiensten und AMF greift Flex auf in ColdFusion, PHP, BlazeDS und ADEP Data Services implementierte Dienste zu. AMF stellt das Messaging für den Austausch von Daten zwischen einer Datenbank und der Clientanwendung bereit.

ColdFusion, BlazeDS und ADEP Data Services stellen jeweils ein AMF-Framework für Remoteobjektdienste zur Verfügung. Für in PHP implementierte Dienste verwendet Flash Builder das Zend AMF-Framework.

ColdFusion- und PHP-Dienste können serverseitige Typisierung zur Verfügung stellen. Bei der serverseitigen Typisierung definiert der Dienst den Rückgabedatentyp. Wenn die Dienstimplementierung den Rückgabedatentyp jedoch nicht definiert, stellt Flash Builder clientseitige Typisierung zur Verfügung. Flash Builder fragt Daten des Dienstes ab, sodass Sie den Rückgabetyp in der Clientanwendung konfigurieren können.

Clientseitige und serverseitige Typisierung

Bei Flex verwendet eine Clientanwendung den Datentyp der aufgrund eines Dienstaufrufs zurückgegeben Daten in Methoden, die auf diese Daten zugreifen und sie darstellen.

Die unten aufgeführten Dienstbeispiele geben jedoch typenlose Daten zurück.

- „Implementieren von ColdFusion-Diensten“ auf Seite 47
- „Implementieren von PHP-Diensten“ auf Seite 54
- „Beispiel für das Implementieren von Diensten aus mehreren Quellen“ auf Seite 69

Beispiel: Bei der `getAllEmployees()`-Methode gibt der Dienst ein Array typenloser Objekte zurück, die Datensätze aus der Datenbank darstellen. Flash Builder stellt Werkzeuge für clientseitige Typisierung zur Verfügung. Mithilfe von Flash Builder-Werkzeugen prüfen Sie die zurückgegebenen Daten und definieren dafür einen benutzerdefinierten Datentyp.

Für das zurückgegebene Objekt der Mitarbeiterdatensätze definieren Sie den benutzerdefinierten Datentyp „Employee“. Aus jeder Spalte des Datensatzes wird eine Eigenschaft des Employee-Datentyps.

Mithilfe des benutzerdefinierten Employee-Datentyps kann die Clientanwendung auf die zurückgegebenen Daten zugreifen und sie korrekt in der Clientanwendung anzeigen.

Flash Builder kann auch auf Dienste zugreifen, die serverseitige Typisierung implementieren. Ein Beispiel zur serverseitigen Typisierung finden Sie unter Flash Builder-Beispiele zur serverseitigen Typisierung.

Implementieren von ColdFusion-Diensten

Implementieren Sie Dienste in ColdFusion als ColdFusion-Komponenten-(CFC-)Dateien. Jede CFC-Datei enthält Funktionen, die die Dienstmethoden bereitstellen.

Sie können ColdFusion-Dienste in beliebigen IDEs wie z. B. Adobe ColdFusion® Builder™ erstellen. Flash Builder stellt keinen Editor zur Verfügung, der zum Bearbeiten von ColdFusion-Dateien optimiert ist. Wenn Sie jedoch eine ColdFusion-Datei in Flash Builder öffnen, startet Flash Builder jene Anwendung im System, die mit ColdFusion-Dateien verknüpft ist.

Beispiel für ColdFusion-Dienste

Sie können einen einfachen ColdFusion-Dienst implementieren, indem Sie eine ColdFusion-Komponenten-(CFC-)Datei erstellen, die Funktionen für die Dienstmethoden enthält. Das folgende Beispiel, `employeeService.cfc`, stellt den EmployeeService-Dienst dar, der zwei Funktionen implementiert. Die Funktion `getAllIEmployees()` ruft alle Mitarbeiterdatensätze aus der Datenbank ab. Die Funktion `getEmployees()` gibt einen einzelnen Mitarbeiterdatensatz für das Argument `emp_no` der Funktion zurück.

In diesem Beispiel wird die clientseitige Typisierung dargestellt. Der Dienst gibt typenlose Daten zurück. Flash Builder verwendet die clientseitige Typisierung, um die zurückgegebenen Daten zu introspektieren und den Datentyp zu definieren.

In den nachstehenden Beispielen wird gezeigt, wie Dienste für Paging und Datenverwaltung implementiert werden.

Sie können Flash Builder auch für den Zugriff auf Dienste verwenden, die serverseitige Typisierung implementieren. Siehe „Clientseitige und serverseitige Typisierung“ auf Seite 47.

Zum Zeitpunkt der Fertigstellung dieses Dokuments waren keine Beispiele für die serverseitige Typisierung verfügbar. Beispiele zur serverseitigen Typisierung finden Sie unter Flash Builder-Beispiele zur serverseitigen Typisierung.

ColdFusion-Beispiel für die Implementierung eines einfachen Diensts

Dieses Beispiel zeigt, wie ein einfacher Dienst in ColdFusion implementiert wird. Das Beispiel beruht auf Code, den Flash Builder beim Zugreifen auf eine Datenbanktabelle generiert. Siehe „Generieren eines ColdFusion-Beispieldiensts aus einer Datenbanktabelle.“ auf Seite 10.

In diesem Beispiel wird die clientseitige Typisierung implementiert. Siehe „Clientseitige und serverseitige Typisierung“ auf Seite 47.

Beispiele zur serverseitigen Typisierung finden Sie unter Flash Builder server-side type examples.

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer ColdFusion-Dienste finden Sie in der ColdFusion-Dokumentation About User Security.*

```
<cfcomponent output="false">

<!---
    This sample service contains functions that illustrate typical service operations.
    This code is for prototyping only.

    Authenticate the user prior to allowing them to call these methods. You can find more
    information at http://www.adobe.com/go/cf9_usersecurity

--->
    <cffunction name="getAllemployees" output="false" access="remote" returntype="any" >

        <!--- Retrieve set of records and return them as a query or array.
                Add authorization or any logical checks for secure access to your data --->

        <cfset var qAllItems="">
        <cfquery name="qAllItems" datasource="employees">
            SELECT * FROM employees
        </cfquery>
        <cfreturn qAllItems>

    </cffunction>

    <cffunction name="getemployees" output="false" access="remote"  returntype="any" >
        <cfargument name="emp_no" type = "numeric" required="true" />

        <!--- Retrieve a single record and return it as a query or array.
                Add authorization or any logical checks for secure access to your data --->

        <cfset var qItem="">
        <cfquery name="qItem" datasource="employees">
            SELECT *
            FROM employees
            WHERE emp_no = <CFQUERYPARAM CFSQLTYPE="CF_SQL_INTEGER"
VALUE="#ARGUMENTS.emp_no#">
        </cfquery>

        <cfreturn qItem>

    </cffunction>

</cfcomponent>
```

Wichtigste Informationen zu EmployeeService:

- Stellt eine Verbindung zur Personaldatenbank her und greift auf die Mitarbeitertabelle in der Datenbank zu.
- Gibt ein Array an Objekten zurück.

  Beim Programmieren mit dem Flex-Framework geben die Dienste nur Daten zurück. Die Clientanwendung verarbeitet die Formatierung und Darstellung der Daten. Dieses Modell unterscheidet sich von traditionellen ColdFusion-CFM-Anwendungen, die in einer HTML-Vorlage formatierte Daten zurückgeben.

  Flex verarbeitet zurückgegebene Datensatzgruppen als Objektarray. Jede Zeile steht für einen abgerufenen Datensatz aus der Datenbank. Jede Spalte des Datensatzes wird eine Eigenschaft des zurückgegebenen Objekts. Die Clientanwendung kann nun auf die zurückgegebenen Daten als Objekte mit einer Gruppe von Eigenschaften zugreifen.

Konfigurieren Sie den Datentyp für das zurückgegebene Objekt. Siehe „Clientseitige und serverseitige Typisierung“ auf Seite 47.

- Der ColdFusion-Server stellt die Fehlerverarbeitung bereit.

  Die Fehlerverarbeitung durch ColdFusion ist beim Debuggen des Diensts hilfreich. Ändern Sie in ColdFusion Administrator die Debugging- und Protokollierungseinstellungen, um solide Informationen für das Debuggen bereitzustellen.

  In der Schnittstelle „Methode testen“ von Flash Builder werden die vom ColdFusion-Server zurückgegebenen Informationen angezeigt.

  Weitere Informationen zum Testen von Diensten finden Sie unter „Debuggen von Remotediensten“ auf Seite 66.

- Verwendet `cfqueryparam` zum Erstellen von Datenbankabfragen.

  `cfqueryparam` schützt vor SQL-Einschleusungsbefehlen in Aufrufen an den Server. Weitere Informationen finden Sie in der ColdFusion-Dokumentation unter Enhancing security with cfqueryparam.

- Authentifizieren Sie Benutzer, bevor Sie ihnen Zugriff auf die Funktionen des Diensts gewähren.

  Der Beispielcode stellt nicht dar, wie Benutzer authentifiziert werden. Siehe ColdFusion-Dokumentation, About User Security.

Verwandte Themen

„Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28

„Zugriff auf ColdFusion-Dienste“ auf Seite 9

„Generieren eines ColdFusion-Beispieldiensts aus einer Datenbanktabelle.“ auf Seite 10

ColdFusion-Beispiel für die Implementierung von Paging

Mit Flash Builder-Werkzeugen können Sie das Paging von über einen Remotedienst abgerufenen Daten implementieren. Paging steht für das inkrementelle Abrufen großer Datenmengen.

Für das Implementieren von Paging benötigt Flash Builder spezifische Funktionssignaturen. Das folgende Codebeispiel zeigt eine Möglichkeit, einen ColdFusion-Dienst für das Paging von Daten zu implementieren.

Das EmployeeServicePaged-Beispiel beruht auf Flash Builder-Code, der beim Zugreifen auf eine Datenbanktabelle generiert wird. Siehe „Generieren eines ColdFusion-Beispieldiensts aus einer Datenbanktabelle.“ auf Seite 10.

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Das Beispiel ermöglicht allen Personen mit Netzwerkzugriff auf Ihren Server, auf Daten der Datenbanktabelle zuzugreifen, sowie diese zu ändern und zu löschen. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer ColdFusion-Dienste finden Sie in der ColdFusion-Dokumentation About User Security.*

```
<cfcomponent output="false">

<!---
    This sample service contains functions that illustrate typical service operations.
    This code is for prototyping only.

    Authenticate the user prior to allowing them to call these methods. You can find more
    information at http://www.adobe.com/go/cf9_usersecurity
--->
    <cffunction name="count" output="false" access="remote" returntype="numeric" >

        <!---   Return the number of items in your table.
                Add authorization or any logical checks for secure access to your data --->
        <cfquery name="qread" datasource="employees">
            SELECT COUNT(emp_no) AS itemCount FROM employees
        </cfquery>

        <cfreturn qread.itemCount>

    </cffunction>

    <cffunction name="getemployees_paged" output="false" access="remote" returntype="any" >
        <cfargument name="startIndex" type="numeric" required="true" />
        <cfargument name="numItems" type="numeric" required="true" />

    <!---Return a page of numRows number of records as an array or
        query from the database for this startRow.
        Add authorization or any logical checks for secure access to your data --->
    <!---The LIMIT keyword is valid for mysql database only.
        Modify it for your database --->

        <cfset var qRead="">
        <cfquery name="qRead" datasource="employees">
            SELECT * FROM employees LIMIT #startIndex#, #numItems#
        </cfquery>
        <cfreturn qRead>

    </cffunction>
</cfcomponent>
```

Der EmployeeServicePaged-Dienst gibt typenlose Daten zurück. Verwenden Sie die Flash Builder-Werkzeuge zum Konfigurieren des Rückgabetyps für `getEmployees_Paged()`. Aktivieren Sie nach der Konfiguration des Rückgabetyps das Paging für die `getEmployees_Paged()`-Methode.

Verwandte Themen

„Beispiel für ColdFusion-Dienste“ auf Seite 48

„Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28

„Verwalten des Datenzugriffs vom Server“ auf Seite 33

ColdFusion-Beispiel für das Implementieren von Datenverwaltungsvorgängen

Mit Flash Builder-Werkzeugen können Sie Datenverwaltungsfunktionen für Remotedienste implementieren. „Datenverwaltung“ bedeutet das Synchronisieren von Aktualisierungen der auf dem Server befindlichen Daten über die Clientanwendung.

Flash Builder benötigt eine Kombination spezifischer Funktionssignaturen, um die Datenverwaltung zu implementieren. Das folgende Codebeispiel zeigt eine Möglichkeit, einen ColdFusion-Dienst für die Datenverwaltung zu implementieren.

Das EmployeeServiceDM-Beispiel beruht auf Flash Builder-Code, der beim Zugreifen auf eine Datenbanktabelle generiert wird. Siehe „Generieren eines ColdFusion-Beispieldiensts aus einer Datenbanktabelle.“ auf Seite 10.

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer ColdFusion-Dienste finden Sie in der ColdFusion-Dokumentation About User Security.*

```
<cfcomponent output="false">

<!---
    This sample service contains functions that illustrate typical service operations.
    This code is for prototyping only.

    Authenticate the user prior to allowing them to call these methods. You can find more
    information at http://www.adobe.com/go/cf9_usersecurity
--->
    <cffunction name="getAllemployees" output="false" access="remote" returntype="any" >

        <!--- Auto-generated method
                Retrieve set of records and return them as a query or array.
                Add authorization or any logical checks for secure access to your data --->

        <cfset var qAllItems="">
        <cfquery name="qAllItems" datasource="employees">
            SELECT * FROM employees
        </cfquery>
        <cfreturn qAllItems>

    </cffunction>

    <cffunction name="getemployees" output="false" access="remote"  returntype="any" >
        <cfargument name="emp_no" type = "numeric" required="true" />

        <!---
                Retrieve a single record and return it as a query or array.
                Add authorization or any logical checks for secure access to your data --->

        <cfset var qItem="">
        <cfquery name="qItem" datasource="employees">
            SELECT *
            FROM employees
            WHERE emp_no = <CFQUERYPARAM CFSQLTYPE="CF_SQL_INTEGER"
VALUE="#ARGUMENTS.emp_no#">
        </cfquery>

        <cfreturn qItem>

    </cffunction>

    <cffunction name="createemployees" output="false" access="remote" returntype="any" >
        <cfargument name="item" required="true" />
```

```
        <!---   Insert a new record in the database.
            Add authorization or any logical checks for secure access to your data --->

        <cfquery name="createItem" datasource="employees" result="result">
            INSERT INTO employees (birth_date, first_name, last_name, gender, hire_date)
            VALUES (<CFQUERYPARAM cfsqltype="CF_SQL_DATE" VALUE="#item.birth_date#">,
                            <CFQUERYPARAM cfsqltype="CF_SQL_VARCHAR"
VALUE="#item.first_name#">,
                            <CFQUERYPARAM cfsqltype="CF_SQL_VARCHAR"
VALUE="#item.last_name#">,
                            <CFQUERYPARAM cfsqltype="CF_SQL_CHAR" VALUE="#item.gender#">,
                            <CFQUERYPARAM cfsqltype="CF_SQL_DATE" VALUE="#item.hire_date#">)

        </cfquery>

        <!--- The GENERATED_KEY is valid for mysql database only, you can modify it for your
database --->
        <cfreturn result.GENERATED_KEY/>

    </cffunction>

    <cffunction name="updateemployees" output="false" access="remote" returntype="void" >
        <cfargument name="item" required="true" />

        <!--- Update an existing record in the database.
              Add authorization or any logical checks for secure access to your data --->

         <cfquery name="updateItem" datasource="employees">
            UPDATE employees SET birth_date = <CFQUERYPARAM cfsqltype=CF_SQL_DATE
VALUE="#item.birth_date#">,
                             first_name = <CFQUERYPARAM cfsqltype=CF_SQL_VARCHAR
VALUE="#item.first_name#">,
                             last_name = <CFQUERYPARAM cfsqltype=CF_SQL_VARCHAR
VALUE="#item.last_name#">,
                             gender = <CFQUERYPARAM cfsqltype=CF_SQL_CHAR
VALUE="#item.gender#">,
                             hire_date = <CFQUERYPARAM cfsqltype=CF_SQL_DATE
VALUE="#item.hire_date#">
```

```
            WHERE emp_no = <CFQUERYPARAM CFSQLTYPE="CF_SQL_INTEGER" VALUE="#item.emp_no#">
        </cfquery>

    </cffunction>

    <cffunction name="deleteemployees" output="false" access="remote" returntype="void" >
        <cfargument name="emp_no" type="numeric" required="true" />

        <!--- Delete a record in the database.
              Add authorization or any logical checks for secure access to your data --->

        <cfquery name="delete" datasource="employees">
            DELETE FROM employees
            WHERE emp_no = <CFQUERYPARAM CFSQLTYPE="CF_SQL_INTEGER"
VALUE="#ARGUMENTS.emp_no#">
        </cfquery>

    </cffunction>
</cfcomponent>
```

Der EmployeeServiceDM-Dienst gibt typenlose Daten zurück. Verwenden Sie die Flash Builder-Werkzeuge zum Konfigurieren des Rückgabetyps für `getAllEmployees()` und `getEmployees()`. Verwenden Sie „Employee“ für den von diesen Methoden zurückgegebenen benutzerdefinierten Datentyp.

Aktivieren Sie nach der Konfiguration des Rückgabetyps die Datenverwaltung für den Employee-Datentyp.

Verwandte Themen

„Beispiel für ColdFusion-Dienste“ auf Seite 48

„Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28

„Verwalten des Datenzugriffs vom Server“ auf Seite 33

Erstellen von CFCs mit Adobe ColdFusion Builder

Adobe® ColdFusion® Builder™ enthält den Adobe CFC Generator. Erstellen Sie ORM CFC oder eine traditionelle CFC aus einer Reihe von Datenbanktabellen. Die von ColdFusion Builder generierte CFC kann dann als Datendienst für Flash Builder verwendet werden. Der Adobe CFC-Generator erstellt Dienste, die serverseitige Typisierung implementieren.

Weitere Informationen finden Sie unter Using Adobe CFC Generator.

Hinweis: *ColdFusion Object Relational Mapping (ORM) definiert eine Zuordnungsstrategie für das Speichern und Abrufen von Daten aus einer relationalen Datenbank mithilfe eines Objektmodells. Siehe ColdFusion ORM.*

Implementieren von PHP-Diensten

Beim Implementieren von Diensten in PHP implementieren Sie die Dienste im Allgemeinen als PHP-Klassen. Die Klassen in PHP müssen nicht zwangsläufig objektorientiert sein. Jede Klasse kann eine Bibliothek von Funktionen sein, die die Dienstmethoden bereitstellen.

Sie können PHP-Dienste in einer beliebigen Bearbeitungsumgebung erstellen, beispielsweise Dreamweaver oder Zend Studio. Flash Builder stellt keinen Editor zur Verfügung, der zum Bearbeiten von PHP-Dateien optimiert ist. Wenn Sie jedoch eine PHP-Datei in Flash Builder öffnen, startet Flash Builder die Anwendung im System, die mit den PHP-Dateien verknüpft ist. Flash Builder stellt auch einen Texteditor bereit, den Sie für die Bearbeitung von PHP-Dateien verwenden können.

Verwenden von AMF für den Zugriff auf in PHP implementierte Dienste

PHP-Datendienste sind über AMF (Action Message Format) verfügbar. AMF bietet Messaging zwischen einem Flash-Client und einem Webserver. In Flash Builder werden AMF-Messaging-Funktionen für PHP-Datendienste über das Zend AMF-Framework implementiert.

Informationen zu Zend AMF finden Sie unter Zend Framework-Referenz für Programmierer.

Informationen zum Installieren des Zend Framework finden Sie unter „Installieren des Zend Framework“ auf Seite 21.

Informationen zur Verwendung von Zend mit Flash Builder für PHP finden Sie auf der Zend-Website.

Hinweis: *Flash Builder nutzt zwar das Zend AMF-Framework, Sie sind beim Erstellen von PHP-Diensten aber nicht gezwungen, Zend-Komponenten zu verwenden. Zend-Komponenten funktionieren zwar gut mit Flash Builder, aber Sie können zum Erstellen von Diensten auch jede andere PHP-Entwicklungsumgebung verwenden.*

Beispiel für PHP-Dienste

Sie können einen einfachen PHP-Dienst implementieren, indem Sie eine PHP-Klassendatei erstellen, die Funktionen für die Dienstmethoden enthält. Das folgende Beispiel stellt den EmployeeService-Dienst dar, der zwei Funktionen implementiert:

- `getAllIEmployees()`

  Ruft alle Mitarbeiterdatensätze aus der Datenbank ab.

- `getEmployeeByID($itemID)`

  Ruft einen einzelnen Mitarbeiterdatensatz ab.

In diesem Beispiel wird die clientseitige Typisierung dargestellt. Der Dienst gibt typenlose Daten zurück. Flash Builder verwendet die clientseitige Typisierung, um die zurückgegebenen Daten zu introspektieren und den Datentyp zu definieren.

In den nachstehenden Beispielen wird gezeigt, wie Dienste für Paging und Datenverwaltung implementiert werden.

Sie können Flash Builder auch für den Zugriff auf Dienste verwenden, die serverseitige Typisierung implementieren. Siehe „Clientseitige und serverseitige Typisierung“ auf Seite 47.

Zum Zeitpunkt der Fertigstellung dieses Dokuments waren keine Beispiele für die serverseitige Typisierung verfügbar. Beispiele zur serverseitigen Typisierung finden Sie unter Flash Builder-Beispiele zur serverseitigen Typisierung.

Beispiel: Einfacher PHP-Dienst

Dieses Beispiel zeigt, wie ein einfacher Dienst in PHP implementiert wird. Das Beispiel beruht auf Code, den Flash Builder beim Zugreifen auf eine Datenbanktabelle generiert. Siehe „Generieren eines PHP-Beispieldiensts aus einer Datenbanktabelle“ auf Seite 12.

In diesem Beispiel wird die clientseitige Typisierung dargestellt. Siehe „Clientseitige und serverseitige Typisierung“ auf Seite 47.

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer Dienste finden Sie unter „Bereitstellen von Anwendungen, die auf Datendienste zugreifen“ auf Seite 44.*

```
<?php
/**
 * This sample service contains functions that illustrate typical service operations.
 * This code is for prototyping only.
 *
 *  Authenticate users before allowing them to call these methods.
 */
class EmployeeService {

    var $username = "root";
    var $password = "root";
    var $server = "localhost";
    var $port = "3306";
    var $databasename = "employees";
    var $tablename = "employees";

    var $connection;

    /**
     * The constructor initializes the connection to database. Everytime a request is
     * received by Zend AMF, an instance of the service class is created and then the
     * requested method is called.
     */
    public function __construct() {
        $this->connection = mysqli_connect(
                                $this->server,
                                $this->username,
                                $this->password,
                                $this->databasename,
                                $this->port
                            );

        $this->throwExceptionOnError($this->connection);
    }

    /**
     * Returns all the rows from the table.
     *
     * Add authroization or any logical checks for secure access to your data
     *
     * @return array
     */
    public function getAllEmployees() {

        $stmt = mysqli_prepare($this->connection, "SELECT * FROM $this->tablename");
        $this->throwExceptionOnError();

        mysqli_stmt_execute($stmt);
        $this->throwExceptionOnError();

        $rows = array();
```

```
    mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
        $row->first_name, $row->last_name, $row->gender, $row->hire_date);

    while (mysqli_stmt_fetch($stmt)) {
      $rows[] = $row;
      $row = new stdClass();
      mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
            $row->first_name, $row->last_name, $row->gender, $row->hire_date);
    }

    mysqli_stmt_free_result($stmt);
    mysqli_close($this->connection);

    return $rows;
}

/**
 * Returns the item corresponding to the value specified for the primary key.
 *
 * Add authroization or any logical checks for secure access to your data
 *
 *
 * @return stdClass
 */
public function getEmployeesByID($itemID) {

    $stmt = mysqli_prepare($this->connection, "SELECT * FROM $this->tablename where emp_no=?");
    $this->throwExceptionOnError();

    mysqli_bind_param($stmt, 'i', $itemID);
    $this->throwExceptionOnError();

    mysqli_stmt_execute($stmt);
    $this->throwExceptionOnError();

    mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
            $row->first_name, $row->last_name, $row->gender, $row->hire_date);

    if(mysqli_stmt_fetch($stmt)) {
        return $row;
    } else {
        return null;
```

```
        }
    }

    /**
     * Utitity function to throw an exception if an error occurs
     * while running a mysql command.
     */
    private function throwExceptionOnError($link = null) {
        if($link == null) {
            $link = $this->connection;
        }
        if(mysqli_error($link)) {
            $msg = mysqli_errno($link) . ": " . mysqli_error($link);
            throw new Exception('MySQL Error - '. $msg);
        }
    }
}


?>
```

Wichtigste Informationen zu EmployeeService:

- Stellt über Port 3306 von localhost eine Verbindung zur Personaldatenbank her. Greift auf die Mitarbeitertabelle in der Datenbank zu.
- Stellt Klassenvariablen für die Verbindung zum Dienst und den Zugriff auf die Datenbanktabellen bereit.

  Diese Variablen können für Funktionen der Klasse verwendet werden.

  Ersetzen Sie die Werte dieser Variablen durch die Werte Ihres Systems.
- Gibt das Array von Objekten an die Clientanwendung zurück.

  Beim Programmieren mit dem Flex-Framework geben die Dienste nur Daten zurück. Die Clientanwendung verarbeitet die Formatierung und Darstellung der Daten.

  Dieses Modell unterscheidet sich von traditionellen PHP-Diensten, die in einer HTML-Vorlage formatierte Daten zurückgeben.
- Die Funktion `getEmployeesByID($itemID)` bindet die Eingabeparameter an die Datentypen.

  Die Anzahl der Variablen und die Länge der Zeichenfolgetypen muss mit den Parametern in der Anweisung übereinstimmen. Das „`?`“ in der Prepare-Anweisung dient als Platzhalter für den Parameter.

  mysqli erkennt die folgenden Typen:

  - integer (`i`)
  - double (`d`)
  - string (`s`)
  - blob (`b`)
- Bindet die Ergebnisse, indem ein Objektarray erstellt wird (`$row[]`).

  Flex verarbeitet Datensatzgruppen als Objektarray. Jedes Objekt steht für einen aus der Datenbank abgerufenen Datensatz. Jede Spalte des Datensatzes wird eine Eigenschaft des zurückgegebenen Objekts. Die Clientanwendung kann nun auf die zurückgegebenen Daten als Objekte mit einer Gruppe von Eigenschaften zugreifen.

  Da der Server für die zurückgegebenen Daten keinen Datentyp definiert, konfigurieren Sie den Datentyp für das zurückgegebene Objekt. Siehe „Clientseitige und serverseitige Typisierung“ auf Seite 47.

- Stellt eine Konstruktorfunktion für das Initialisieren der Verbindung zur Datenbank bereit.
- Verwendet mysqli-Prepare-Anweisungen für das Erstellen von Datenbankabfragen.

  Prepare-Anweisungen schützen vor SQL-Einschleusungsbefehlen in Aufrufen an den Server. Die Anweisung wird erst nach ihrer Vorbereitung auf dem Server ausgeführt.
- Authentifizieren Sie Benutzer, bevor Sie ihnen Zugriff auf die Funktionen des Diensts gewähren.

  Der Beispielcode stellt nicht dar, wie Benutzer authentifiziert werden. Siehe ColdFusion-Dokumentation, About User Security. Die Sicherheitsprinzipien für die Benutzerauthentifizierung und -autorisierung in dieser ColdFusion-Dokumentation gelten für PHP-Dienste.
- Gibt bei Fehlern eine Ausnahme aus.

  Die in Ausnahmen bereitgestellten Informationen sind beim Debuggen der Dienstimplementierung hilfreich. In der Schnittstelle „Methode testen“ von Flash Builder werden die bei Ausnahmen zurückgegebenen Informationen angezeigt.

  Weitere Informationen zum Testen von Diensten finden Sie unter „Debuggen von Remotediensten“ auf Seite 66.
- Der Dateiname `EmployeeService.php` entspricht dem PHP-Klassennamen für den Dienst.

  Wenn Dateiname und Klassenname nicht übereinstimmen, sind Fehler beim Zugriff auf den Dienst die Folge.

Verwandte Themen

„Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28

„Zugriff auf PHP-Dienste“ auf Seite 11

„Generieren eines PHP-Beispieldiensts aus einer Datenbanktabelle“ auf Seite 12

PHP-Beispiel für die Implementierung von Paging

Mit Flash Builder-Werkzeugen können Sie das Paging von über einen Remotedienst abgerufenen Daten implementieren. Paging steht für das inkrementelle Abrufen großer Datenmengen.

Für das Implementieren von Paging benötigt Flash Builder spezifische Funktionssignaturen. Das folgende Codebeispiel zeigt eine Möglichkeit, einen PHP-Dienst für das Paging von Daten zu implementieren.

Dieses Beispiel beruht auf Flash Builder-Code, der beim Zugreifen auf eine Datenbanktabelle generiert wird. Siehe „Generieren eines PHP-Beispieldiensts aus einer Datenbanktabelle“ auf Seite 12.

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer Dienste finden Sie unter „Bereitstellen von Anwendungen, die auf Datendienste zugreifen“ auf Seite 44.*

```
<?php

/**
 *  This sample service contains functions that illustrate typical service operations.
 *  This code is for prototyping only.
 *
 *  Authenticate the user prior to allowing them to call these methods.
 *
 */
class EmployeeServicePaged {

    var $username = "root";
    var $password = "root";
    var $server = "localhost";
    var $port = "3306";
    var $databasename = "employees";
    var $tablename = "employees";

    var $connection;

    /**
     * The constructor initializes the connection to database. Everytime a request is
     * received by Zend AMF, an instance of the service class is created and then the
     * requested method is invoked.
     */
    public function __construct() {
        $this->connection = mysqli_connect(
                                $this->server,
                                $this->username,
                                $this->password,
                                $this->databasename,
                                $this->port
                            );

        $this->throwExceptionOnError($this->connection);
    }


    /**
     * Returns the number of rows in the table.
     *
     * Add authroization or any logical checks for secure access to your data
     *
     *
     */
    public function count() {
        $stmt = mysqli_prepare($this->connection, "SELECT COUNT(*) AS COUNT
                                                    FROM $this->tablename");
        $this->throwExceptionOnError();

        mysqli_stmt_execute($stmt);
        $this->throwExceptionOnError();

        mysqli_stmt_bind_result($stmt, $rec_count);
        $this->throwExceptionOnError();

        mysqli_stmt_fetch($stmt);
```

```
        $this->throwExceptionOnError();

        mysqli_stmt_free_result($stmt);
        mysqli_close($this->connection);

        return $rec_count;
    }


    /**
     * Returns $numItems rows starting from the $startIndex row from the
     * table.
     *
     * Add authroization or any logical checks for secure access to your data
     *
     * @return array
     */
    public function getEmployees_paged($startIndex, $numItems) {

        $stmt = mysqli_prepare($this->connection, "SELECT * FROM
                                    $this->tablename LIMIT ?, ?");
        $this->throwExceptionOnError();

        mysqli_bind_param($stmt, 'ii', $startIndex, $numItems);
        mysqli_stmt_execute($stmt);
        $this->throwExceptionOnError();

        $rows = array();

        mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
                                $row->first_name, $row->last_name,
                                $row->gender, $row->hire_date);

        while (mysqli_stmt_fetch($stmt)) {
          $rows[] = $row;
          $row = new stdClass();
          mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
                                    $row->first_name, $row->last_name,
                                    $row->gender, $row->hire_date);
        }

        mysqli_stmt_free_result($stmt);
        mysqli_close($this->connection);
```

```
        return $rows;
    }


    /**
     * Utitity function to throw an exception if an error occurs
     * while running a mysql command.
     */
    private function throwExceptionOnError($link = null) {
        if($link == null) {
            $link = $this->connection;
        }
        if(mysqli_error($link)) {
            $msg = mysqli_errno($link) . ": " . mysqli_error($link);
            throw new Exception('MySQL Error - '. $msg);
        }
    }
}
?>
```

Der EmployeeServicePaged-Dienst gibt typenlose Daten zurück. Verwenden Sie die Flash Builder-Werkzeuge zum Konfigurieren des Rückgabetyps für `getEmployees_Paged()`. Aktivieren Sie nach der Konfiguration des Rückgabetyps das Paging für die `getEmployees_Paged()`-Methode.

Verwandte Themen

„Beispiel für PHP-Dienste“ auf Seite 55

„Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28

„Verwalten des Datenzugriffs vom Server“ auf Seite 33

PHP-Beispiel für das Implementieren der Datenverwaltung

Mit Flash Builder-Werkzeugen können Sie Datenverwaltungsfunktionen für Remotedienste implementieren. „Datenverwaltung“ bedeutet das Synchronisieren von Aktualisierungen der auf dem Server befindlichen Daten über die Clientanwendung.

Flash Builder benötigt eine Kombination spezifischer Funktionssignaturen, um die Datenverwaltung zu implementieren. Das folgende Codebeispiel zeigt eine Möglichkeit, einen PHP-Dienst für die Datenverwaltung zu implementieren.

Dieses Beispiel beruht auf Flash Builder-Code, der beim Zugreifen auf eine Datenbanktabelle generiert wird. Siehe „Generieren eines PHP-Beispieldiensts aus einer Datenbanktabelle“ auf Seite 12.

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer Dienste finden Sie unter „Bereitstellen von Anwendungen, die auf Datendienste zugreifen“ auf Seite 44.*

```
<?php

/**
 *  This sample service contains functions that illustrate typical service operations.
 *  This code is for prototyping only.
 *
 *  Authenticate the user prior to allowing them to call these methods.
 */
class EmployeeServiceDM {

    var $username = "root";
    var $password = "root";
    var $server = "localhost";
    var $port = "3306";
    var $databasename = "employees";
    var $tablename = "employees";

    var $connection;

    /**
     * The constructor initializes the connection to database. Everytime a request is
     * received by Zend AMF, an instance of the service class is created and then the
     * requested method is invoked.
     */
    public function __construct() {
        $this->connection = mysqli_connect(
                                $this->server,
                                $this->username,
                                $this->password,
                                $this->databasename,
                                $this->port
                            );

        $this->throwExceptionOnError($this->connection);
    }

    /**
     * Returns all the rows from the table.
     *
     * Add authroization or any logical checks for secure access to your data
     *
     * @return array
     */
    public function getAllEmployees() {

        $stmt = mysqli_prepare($this->connection, "SELECT * FROM $this->tablename");
        $this->throwExceptionOnError();

        mysqli_stmt_execute($stmt);
        $this->throwExceptionOnError();

        $rows = array();

        mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
                                $row->first_name, $row->last_name,
                                $row->gender, $row->hire_date);
```

```
        while (mysqli_stmt_fetch($stmt)) {
          $rows[] = $row;
          $row = new stdClass();
          mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
                                  $row->first_name, $row->last_name,
                                  $row->gender, $row->hire_date);
        }

        mysqli_stmt_free_result($stmt);
        mysqli_close($this->connection);

        return $rows;
    }

    /**
     * Returns the item corresponding to the value specified for the primary key.
     *
     * Add authroization or any logical checks for secure access to your data
     *
     *
     * @return stdClass
     */
    public function getEmployeesByID($itemID) {

        $stmt = mysqli_prepare($this->connection, "SELECT * FROM
                                        $this->tablename where emp_no=?");
        $this->throwExceptionOnError();

        mysqli_bind_param($stmt, 'i', $itemID);
        $this->throwExceptionOnError();

        mysqli_stmt_execute($stmt);
        $this->throwExceptionOnError();

        mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
                                $row->first_name, $row->last_name,
                                $row->gender, $row->hire_date);

        if(mysqli_stmt_fetch($stmt)) {
            return $row;
        } else {
            return null;
        }
    }

    /**
     * Returns the item corresponding to the value specified for the primary key.
     *
     * Add authroization or any logical checks for secure access to your data
     *
     *
     * @return stdClass
     */
    public function createEmployees($item) {

        $stmt = mysqli_prepare($this->connection, "INSERT INTO $this->tablename
                                (emp_no, birth_date, first_name, last_name,
```

```
                            gender, hire_date) VALUES (?, ?, ?, ?, ?, ?)");
    $this->throwExceptionOnError();

    mysqli_bind_param($stmt, 'isssss', $item->emp_no, $item->birth_date
                              $item->first_name, $item->last_name,
                              $item->gender, $item->hire_date);
    $this->throwExceptionOnError();

    mysqli_stmt_execute($stmt);
    $this->throwExceptionOnError();

    $autoid = mysqli_stmt_insert_id($stmt);

    mysqli_stmt_free_result($stmt);
    mysqli_close($this->connection);

    return $autoid;
}

/**
 * Updates the passed item in the table.
 *
 * Add authroization or any logical checks for secure access to your data
 *
 * @param stdClass $item
 * @return void
 */
public function updateEmployees($item) {

    $stmt = mysqli_prepare($this->connection, "UPDATE $this->tablename
                            SET emp_no=?, birth_date=?, first_name=?,
                            last_name=?, gender=?, hire_date=?
                            WHERE emp_no=?");
    $this->throwExceptionOnError();

    mysqli_bind_param($stmt, 'isssssi', $item->emp_no, $item->birth_date,
                        $item->first_name, $item->last_name, $item->gender,
                        $item->hire_date, $item->emp_no);
    $this->throwExceptionOnError();

    mysqli_stmt_execute($stmt);
    $this->throwExceptionOnError();

    mysqli_stmt_free_result($stmt);
    mysqli_close($this->connection);
}

/**
 * Deletes the item corresponding to the passed primary key value from
 * the table.
 *
 * Add authroization or any logical checks for secure access to your data
 *
 *
 * @return void
 */
public function deleteEmployees($itemID) {
```

```
        $stmt = mysqli_prepare($this->connection, "DELETE FROM $this->tablename
                                                    WHERE emp_no = ?");
        $this->throwExceptionOnError();

        mysqli_bind_param($stmt, 'i', $itemID);
        mysqli_stmt_execute($stmt);
        $this->throwExceptionOnError();

        mysqli_stmt_free_result($stmt);
        mysqli_close($this->connection);
    }

    /**
     * Utitity function to throw an exception if an error occurs
     * while running a mysql command.
     */
    private function throwExceptionOnError($link = null) {
        if($link == null) {
            $link = $this->connection;
        }
        if(mysqli_error($link)) {
            $msg = mysqli_errno($link) . ": " . mysqli_error($link);
            throw new Exception('MySQL Error - '. $msg);
        }
    }
}
?>
```

Der EmployeeServiceDM-Dienst gibt typenlose Daten zurück. Verwenden Sie die Flash Builder-Werkzeuge zum Konfigurieren des Rückgabetyps für `getAllEmployees()` und `getEmployeesByID()`. Verwenden Sie „Employee“ für den von diesen Methoden zurückgegebenen benutzerdefinierten Datentyp.

Aktivieren Sie nach der Konfiguration des Rückgabetyps die Datenverwaltung für den Employee-Datentyp.

Verwandte Themen

„Beispiel für PHP-Dienste“ auf Seite 55

„Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28

„Verwalten des Datenzugriffs vom Server“ auf Seite 33

Debuggen von Remotediensten

Anwendungen, die auf Remotedienste zugreifen, können auf mehrere Weisen debuggt werden.

- Flash Builder-Ansicht „Methode testen“

  Die Flash Builder-Ansicht „Methode testen“ ermöglicht Ihnen das Aufrufen von Dienstmethoden und das Anzeigen der zurückgegebenen Daten. In der Ansicht „Methode testen“ werden durch den Dienst angezeigte Fehlermeldungen dargestellt.

- Serverseitige Skripten

Für das weitere Debuggen von Diensten können Sie Skripten zum Testen von Servercode und Schreiben der Ausgabedatenstrominformationen in Protokolldateien schreiben.

- Flash Builder-Netzwerküberwachung

  Verwenden Sie die Netzwerküberwachung, nachdem Sie in Flash Builder eine Anwendung für den Zugriff auf einen Dienst erstellt haben. Mithilfe der Netzwerküberwachung werden die zwischen dem Server und dem Client gesendeten Daten angezeigt.

Flash Builder-Ansicht „Methode testen“

Mithilfe der Flash Builder-Ansicht „Methode testen“ rufen Sie Dienstmethoden auf und zeigen die zurückgegebenen Daten an. Die Ergebnisse enthalten alle durch den Dienst zurückgegebenen Fehlermeldungen.

Mithilfe der Ansicht „Methode testen“ zeigen Sie Daten an, die von Methoden für Dienste, die Sie entweder selbst geschrieben haben oder die über HTTP- oder Webservices verfügbar sind, zurückgegeben werden.

Dienstmethode testen

Bei dieser Prozedur wird davon ausgegangen, dass Sie einen von Ihnen geschriebenen Dienst testen möchten oder Zugriff auf einen HTTP- oder Webservice haben.

1. Navigieren Sie in der Flash Builder-Ansicht „Daten/Dienste“ zur Dienstmethode, die Sie testen möchten.
2. Wählen Sie im Kontextmenü der Dienstmethode „Methode testen“ aus.
3. (Optional) Wählen Sie in der Ansicht „Methode testen“ die Option „Authentifizierung erforderlich“ und geben Sie die Anmeldeinformationen für den Dienst an.
4. Wenn die Methode Parameter verwendet, klicken Sie auf das Feld „Wert eingeben“ und geben Sie den Wert für den Parameter ein.

   Wenn der Parameter einen komplexen Typ erfordert, klicken Sie im Feld „Wert eingeben“ auf die Schaltfläche mit der Ellipse, um einen Eingabeargumenteditor zu öffnen, der die JSON-Notation akzeptiert. Geben Sie den Wert für die Parameter in der JSON-Notation ein.
5. Klicken Sie zum Anzeigen der Ergebnisse der Methode auf „Test“.

Skripten zum Testen des Servercodes

Verwenden Sie Testskripten, um Servercode anzuzeigen und zu debuggen, bevor Sie in Flash Builder eine Verbindung zum Server herzustellen versuchen. Testskripten bieten die folgenden Vorteile:

- Sie können Testergebnisse in einem Webbrowser anzeigen.

  Wenn Sie den Code ändern, brauchen Sie nur die Browserseite zu aktualisieren, um die Ergebnisse zu sehen.
- Sie können Ergebnisse an den Ausgabestrom weiterleiten bzw. ausdrucken; dies ist von AMF aus nicht direkt möglich.
- Die Fehleranzeige ist übersichtlicher formatiert und normalerweise umfassender als bei Fehlern, die über AMF erfasst wurden.

ColdFusion-Skripten

Erstellen Sie mit dem folgenden Skript (`tester.cfm`) einen Dump für einen Funktionsaufruf.

```
<!--- tester.cfm --->
<cfobject component="EmployeeService" name="o"/>
<cfdump var="#o.getAllItems()#">
```

In `tester2.cfm` geben Sie die Methode und die Argumente an, die in der URL aufgerufen werden sollen.

```
<!--- tester2.cfm --->
<cfdump var="#url#">

<cfinvoke component="#url.cfc#" method="#url.method#" argumentCollection="#url#"
returnVariable="r">

<p>Result:

<cfif isDefined("r")>
    <cfdump var="#r#">
<cfelse>
    (no result)
</cfif>
```

Rufen Sie beispielsweise die `getItemID()`-Methode in EmployeeService mit der folgenden URL auf:

```
http://localhost/tester2.cfm?EmployeeService&method=getItemId&id=12
```

`tester3.cfm` schreibt ein Protokoll, in dem Aufrufe der Methode protokolliert und die Eingabeargumente mithilfe von `cfdump` abgelegt werden.

```
<!--- tester3.cfm --->
<cfsavecontent variable="d"><cfdump var="#arguments#"></cfsavecontent>

<cffile action="append"
file="#getDirectoryFromPath(getCurrentTemplatePath())#MyServiceLog.htm"
output="<p>#now()# operationName #d#">
```

PHP-Skripten

Erstellen Sie mit dem folgenden Skript (`tester.php`) einen Dump für einen Funktionsaufruf.

```
<pre>
<?php
include('MyService.php');
    $o = new MyService();
    var_dump($o->getAllItems());
?>
</pre>
```

Fügen Sie den folgenden Code zum PHP-Dienst hinzu, um Meldungen während der Ausführung des Codes zu protokollieren.

```
$message = 'updateItem: '.$item["id"];
$log_file = '/Users/me/Desktop/myservice.log';
error_log(date('d/m/Y H:i:s').' '.$message.PHP_EOL, 3, $log_file);
```

Fügen Sie den folgenden Code zum PHP-Dienst hinzu, um einen Dump in einer Logdatei zu erstellen:

```
ob_start();
var_dump($item);
$result = ob_get_contents();
ob_end_clean();

$message = 'updateItem: '.$result;
$log_file = '/Users/me/Desktop/myservice.log';
error_log(date('d/m/Y H:i:s').' '.$message.PHP_EOL, 3, $log_file);
```

Netzwerküberwachung

Die Netzwerküberwachung ist in Flash Builder von der Flex-Debugperspektive aus verfügbar. Aktivieren Sie die Überwachung, um damit Daten zu überwachen. Details zur Aktivierung und Verwendung der Netzwerküberwachung finden Sie unter Überwachen von Anwendungen, die auf Datendienste zugreifen.

Beispiel für das Implementieren von Diensten aus mehreren Quellen

Üblicherweise greifen Anwendungen auf Daten aus verschiedenen Quellen zu und zeigen die Ergebnisse der zusammengeführten Daten in einer Anwendung an. In diesem Beispiel wird gezeigt, wie Daten aus drei Tabellen einer Personaldatenbank zusammengeführt werden:

- Departments

  Jeder Datensatz enthält die Felder Abteilungsnummer und Abteilungsname.

- Dept_emp

  Jeder Datensatz enthält folgende Felder: emp_no, dept_no, from_date, to_date

- Employees

  Jeder Datensatz enthält folgende Felder: emp_no, birth_date, first_name, last_name, gender, hire_date

Die Beispielanwendung verfügt über zwei DataGrids: eines für Abteilungen („Departments“) und eines für Mitarbeiter („Employees“).

In „Departments“ sind alle Abteilungen aufgelistet. Wenn Sie eine Abteilung auswählen, werden im DataGrid „Employees“ alle Mitarbeiter der Abteilung aufgelistet.

Wenn Sie im DataGrid „Employees“ einen Mitarbeiter auswählen, wird ein Formular gefüllt, mit dem Sie den Datensatz für den Mitarbeiter aktualisieren können.

Erstellen der Dienste

Erstellen Sie für dieses Beispiel einen einzelnen Dienst. Der Dienst enthält die folgenden Methoden:

- getAllDepartments()
- getEmployeesByDept()
- `getEmployeeByID()`
- `updateEmployee()`

EmployeeService (PHP)

`EmployeeService.php` implementiert einen Dienst, der eine einzelne Funktion enthält. `GetEmployeesByID()` akzeptiert als Argument die Abteilungs-ID und gibt alle Mitarbeiter der Abteilung zurück. Die Funktion gibt auch das Ein- und Austrittsdatum des Mitarbeiters in Bezug auf die Abteilung zurück. `GetEmployeesByDept()` führt die folgende SQL-Abfrage aus:

```
SELECT
    employees.emp_no,
    employees.birth_date,
    employees.first_name,
    employees.last_name,
    employees.gender,
    employees.hire_date,
    dept_emp.from_date,
    dept_emp.to_date
FROM employees, dept_emp
WHERE dept_emp.emp_no = employees.emp_no and
    dept_emp.dept_no = departments.dept_no
```

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer Dienste finden Sie unter „Bereitstellen von Anwendungen, die auf Datendienste zugreifen“ auf Seite 44.*

```
<?php

/**
 * EmployeeService.php
 *
 * This sample service contains functions that illustrate typical service operations.
 * Use these functions as a starting point for creating your own service implementation.
 *
 *  This code is for prototyping only.
 *
 *  Authenticate the user before allowing them to call these methods.
 */
class EmployeeService {

    var $username = "admin2";
    var $password = "Cosmo49";
    var $server = "localhost";
    var $port = "3306";
    var $databasename = "employees";
    var $tablename = "employees";

    var $connection;

    /**
     * The constructor initializes the connection to database. Everytime a request is
     * received by Zend AMF, an instance of the service class is created and then the
     * requested method is called.
     */
    public function __construct() {
        $this->connection = mysqli_connect(
                                $this->server,
                                $this->username,
                                $this->password,
                                $this->databasename,
                                $this->port
                            );

        $this->throwExceptionOnError($this->connection);
    }
```

```
/**
 * Returns all the rows from the table.
 *
 * Add authroization or any logical checks for secure access to your data
 *
 * @return array
 */
public function getAllDepartments() {

    $stmt = mysqli_prepare($this->connection, "SELECT * FROM departments");
    $this->throwExceptionOnError();

    mysqli_stmt_execute($stmt);
    $this->throwExceptionOnError();

    $rows = array();

    mysqli_stmt_bind_result($stmt, $row->dept_no, $row->dept_name);

    while (mysqli_stmt_fetch($stmt)) {
      $rows[] = $row;
      $row = new stdClass();
      mysqli_stmt_bind_result($stmt, $row->dept_no, $row->dept_name);
    }

    mysqli_stmt_free_result($stmt);
    mysqli_close($this->connection);

    return $rows;
}

public function getEmployeesByDept($deptId) {
    $stmt = mysqli_prepare($this->connection, "select employees.emp_no,
                                            employees.first_name,
                                            employees.last_name,
                                            employees.gender,
                                            dept_emp.dept_no
                                    from employees, dept_emp
                            where dept_emp.emp_no = employees.emp_no
                                and dept_emp.dept_no = ?
                            limit 0,30;");
    $this->throwExceptionOnError();

    mysqli_bind_param($stmt, 's', $deptId);
    $this->throwExceptionOnError();

    mysqli_stmt_execute($stmt);
    $this->throwExceptionOnError();

    $rows = array();

    mysqli_stmt_bind_result($stmt, $row->emp_no, $row->first_name,
                        $row->last_name, $row->gender, $row->dept_no);

    while (mysqli_stmt_fetch($stmt)) {
```

```
        $rows[] = $row;
        $row = new stdClass();

        mysqli_stmt_bind_result($stmt, $row->emp_no, $row->first_name,
                        $row->last_name, $row->gender, $row->dept_no);
    }

    mysqli_stmt_free_result($stmt);
    mysqli_close($this->connection);

    return $rows;
}

/**
 * Returns the item corresponding to the value specified for the primary key.
 *
 * Add authroization or any logical checks for secure access to your data
 *
 *
 * @return stdClass
 */
public function getEmployeesByID($itemID) {

    $stmt = mysqli_prepare($this->connection, "SELECT * FROM employees
                                                    where emp_no=?");
    $this->throwExceptionOnError();

    mysqli_bind_param($stmt, 'i', $itemID);
    $this->throwExceptionOnError();

    mysqli_stmt_execute($stmt);
    $this->throwExceptionOnError();

    mysqli_stmt_bind_result($stmt, $row->emp_no, $row->birth_date,
                            $row->first_name, $row->last_name,
                            $row->gender, $row->hire_date);

    if(mysqli_stmt_fetch($stmt)) {
        return $row;
    } else {
        return null;
    }
}


/**
 * Updates the passed item in the table.
 *
 * Add authroization or any logical checks for secure access to your data
 *
 * @param stdClass $item
 * @return void
 */
public function updateEmployees($item) {

    $stmt = mysqli_prepare($this->connection, "UPDATE employees
                                SET emp_no=?, birth_date=?, first_name=?,
```

```
                                    last_name=?, gender=?, hire_date=?
                                WHERE emp_no=?");
        $this->throwExceptionOnError();

        mysqli_bind_param($stmt, 'issssi', $item->emp_no, $item->birth_date,
                          $item->first_name, $item->last_name, $item->gender,
                          $item->hire_date, $item->emp_no);
        $this->throwExceptionOnError();

        mysqli_stmt_execute($stmt);
        $this->throwExceptionOnError();

        mysqli_stmt_free_result($stmt);
        mysqli_close($this->connection);
    }


    /**
     * Utitity function to throw an exception if an error occurs
     * while running a mysql command.
     */
    private function throwExceptionOnError($link = null) {
        if($link == null) {
            $link = $this->connection;
        }
        if(mysqli_error($link)) {
            $msg = mysqli_errno($link) . ": " . mysqli_error($link);
            throw new Exception('MySQL Error - '. $msg);
        }
    }
}
>?>
```

EmployeeService (ColdFusion)

`EmployeeService.cfc` implementiert einen Dienst, der eine einzelne Funktion enthält. `GetEmployeesByID()` akzeptiert als Argument die Abteilungs-ID und gibt alle Mitarbeiter der Abteilung zurück. Die Funktion gibt auch das Ein- und Austrittsdatum des Mitarbeiters in Bezug auf die Abteilung zurück. `GetEmployeesByDept()` führt die folgende SQL-Abfrage aus:

```
SELECT
    employees.emp_no,
    employees.birth_date,
    employees.first_name,
    employees.last_name,
    employees.gender,
    employees.hire_date,
    dept_emp.from_date,
    dept_emp.to_date
FROM employees, dept_emp
WHERE dept_emp.emp_no = employees.emp_no and dept_emp.dept_no = departments.dept_no
```

Wichtig: *Die Beispieldienste dienen nur als Prototypen. Verwenden Sie den Beispieldienst nur in einer sicheren Entwicklungsumgebung. Bevor Sie diesen Dienst bereitstellen, sollten Sie unbedingt die Sicherheit erhöhen und den Zugriff entsprechend einschränken. Informationen zum Schreiben sicherer ColdFusion-Dienste finden Sie in der ColdFusion-Dokumentation About User Security.*

```
<cfcomponent output="false">

<!---
    This sample service contains functions that illustrate typical service operations.
    Use these functions as a starting point for creating your own service implementation.

    This code is for prototyping only.

    Authenticate the user before allowing them to call these methods. You can find more
    information at http://www.adobe.com/go/cf9_usersecurity
--->
    <cffunction name="getEmployeesByDept" output="false" access="remote"  returntype="any" >
        <cfargument name="dept_no" type="string" required="true" />

        <cfset var qItem="">
        <cfquery name="qItem" datasource="employees">
        SELECT employees.emp_no,
            employees.birth_date,
            employees.first_name,
            employees.last_name,
            employees.gender,
            employees.hire_date,
            dept_emp.from_date,
            dept_emp.to_date
        FROM employees, dept_emp
        WHERE dept_emp.emp_no = employees.emp_no and
        dept_emp.dept_no = <CFQUERYPARAM CFSQLTYPE="CF_SQL_VARCHAR"
VALUE="#ARGUMENTS.dept_no#">
        </cfquery>

        <cfreturn qItem>

    </cffunction>

</cfcomponent>?>
```

Importieren der Dienste in ein Serverprojekt

1 Erstellen Sie in Flash Builder ein Flex-Projekt namens „Associations“:

(PHP) Legen Sie beim Erstellen des Projekts PHP als Anwendungsservertyp fest.

(PHP) Nach dem Erstellen des Projekts erstellt Flash Builder einen Ausgabeordner, der dem Webstamm Ihrer PHP-Konfiguration entspricht. Der Standardname für das Projekt „PHP_Associations“ ist `PHP_Associations-debug`.

(ColdFusion) Legen Sie bei der Projekterstellung ColdFusion als Anwendungsservertyp fest. Legen Sie anschließend ColdFusion Flash Remoting fest.

2 (PHP) Erstellen Sie innerhalb von `PHP_Associations-debug` einen Ordner namens `services`. Kopieren Sie `EmployeeService.php` in den Ordner `services`.

3 (ColdFusion) Erstellen Sie einen Ordner namens `Associations` im Webstamm Ihrer ColdFusion-Konfiguration. Kopieren Sie `EmployeeService.chc` in den Ordner `Associations`.

4 Importieren Sie EmployeeService in das Projekt:

Vergewissern Sie sich, dass PHP_Associations in Flash Builder das aktive Projekt ist.

Wählen Sie „Daten“ > „Mit PHP verbinden“. Navigieren Sie zum Festlegen der PHP-Klasse zum Ordner `services` und wählen Sie `EmployeeService.php`. Klicken Sie auf „Fertig stellen“.

Weitere Informationen finden Sie unter „Herstellen einer Verbindung zu PHP-Datendiensten“ auf Seite 11.

5 Konfigurieren Sie in EmployeeService den Rückgabetyp für die Methode.

- DepartmentService

  Wählen Sie im Kontextmenü für die Methode `getAllDepartments()` „Rückgabetyp konfigurieren“ aus.

  Klicken Sie auf „Weiter“, um den Rückgabetyp automatisch zu erkennen.

  Legen Sie **Department** als benutzerdefinierten Rückgabetyp fest. Klicken Sie auf „Fertig stellen“.

- EmployeeService

  Wählen Sie für die Methode `getEmployeesByDept()` „Rückgabetyp konfigurieren“ aus.

  Klicken Sie auf „Weiter“, um den Rückgabetyp automatisch zu erkennen.

  Legen Sie als Parameterwert **d007** fest. Klicken Sie auf „Weiter“.

  Legen Sie **Employee** als benutzerdefinierten Rückgabetyp fest. Klicken Sie auf „Fertig stellen“.

Weitere Informationen finden Sie unter „Konfigurieren von Datentypen für Dienstmethoden“ auf Seite 28.

Erzeugen der Benutzeroberfläche und Binden der zurückgegebenen Daten an die DataGrid-Komponenten

1 Fügen Sie im Designmodus des MXML-Editors zwei DataGrid-Komponenten in den Bearbeitungsbereich ein.

Die DataGrid-Komponente ist in der Komponentenansicht im Ordner „Steuerelemente“ verfügbar. Ziehen Sie die DataGrid-Komponente in den Bearbeitungsbereich.

Geben Sie für das DataGrid „Departments“ **deptDG** an. Legen Sie für das DataGrid „Employees“ **empDeptDG** als ID fest.

2 Ziehen Sie in der Ansicht „Daten/Dienste“die getEmployeesByDept()-Methode auf das DataGrid für die Mitarbeiter.

Der Editor wechselt in den Quellmodus und der Parameter für die getEmployeesByDept()-Methode ist markiert. Löschen Sie jedoch die generierte Ereignisprozedur.

Navigieren Sie zum DataGrid „Employees“. Löschen Sie den Verweis auf das Attribut für die creationComplete-Ereignisprozedur für das empDeptDG-DataGrid.

Nachdem Sie den Verweis auf die Ereignisprozedur gelöscht haben, sieht die erste Codezeile des DataGrid etwa so aus:

```
<mx:DataGrid x="361" y="27" id="empDeptDG"
dataProvider="{getEmployeesByDeptResult.lastResult}">
```

Hinweis: *Die Ereignisprozedur „creationComplete“ wird für das Employees-DataGrid nicht benötigt. Das Employees-DataGrid wird gefüllt, wenn im Departments-DataGrid eine Abteilung ausgewählt wird.*

3 Wechseln Sie zum Designmodus des Editors. Ziehen Sie in der Ansicht „Daten/Dienste“ die `getAllDepartments()`-Methode auf das DataGrid für die Abteilungen.

4 Generieren Sie eine Ereignisprozedur für Änderungen am Departments-DataGrid:

Vergewissern Sie sich, dass das Departments-DataGrid ausgewählt ist. Wählen Sie in der Ansicht „Eigenschaften“ das Symbol „Bei Änderung“ und „Ereignisprozedur generieren“.

Der Editor wechselt in den Quellmodus und der Text der Ereignisprozedur ist markiert. Legen Sie Folgendes für die Ereignisprozedur fest:

```
protected function deptDG_changeHandler(event:ListEvent):void {
    getEmployeesByDeptResult.token =
employeeService.getEmployeesByDept(deptDG.selectedItem.dept_no);
}
```

5 Speichern Sie die Anwendung und führen Sie sie aus.

Wenn Sie im Departments-DataGrid auf eine Abteilung klicken, werden im DataGrid „Employees“ alle Mitarbeiter der Abteilung aufgelistet.

Schließen Sie die Anwendung.

6 Wählen Sie im Designmodus des MXML-Editors das Employees-DataGrid aus. Wählen Sie im Kontextmenü „Detailformular generieren“ und führen Sie folgende Schritte aus:

a Wählen Sie für den Detailaufruf „Dienst aufrufen...“.

b Wählen Sie als Dienst „EmployeeService“.

c Wählen Sie als Methode `getEmployeesByID()`.

d Klicken Sie auf „Fertig stellen“.

7 Legen Sie in der generierten Ereignisprozedur das folgende Argument für `getEmployeesById()` fest:

```
protected function empDeptDG_changeHandler(event:ListEvent):void
{
    getEmployeesByIDResult.token =
employeeService.getEmployeesByID(empDeptDG.selectedItem.emp_no);
}
```

8 Ziehen Sie im Designmodus das Formular unter die DataGrids.

9 Fügen Sie neben dem Formular eine Schaltfläche ein und ändern Sie in der Ansicht „Eigenschaften“ Folgendes:

- Geben Sie bei der ID „updateButton“ an.
- Legen Sie bei der Beschriftung **Update Employee** fest.
- Klicken Sie auf das Symbol „Beim Klicken“ und legen Sie „Dienstaufruf generieren“ fest:

  Wählen Sie als Dienst „EmployeeService“. Wählen Sie als Methode `updateEmployees()`. Der Editor wechselt zum Quellmodus.

- Ändern Sie den Dienstaufruf für „updateEmployees()“ wie folgt:

```
protected function updateButton_clickHandler(event:MouseEvent):void
{
    var e:Employee = new Employee();
    e.birth_date = birth_dateTextInput.text;
    e.first_name = first_nameTextInput.text;
    e.last_name = last_nameTextInput.text;
    e.hire_date = hire_dateTextInput.text;
    e.gender = genderTextInput.text;
    e.emp_no = employee.emp_no;

    updateEmployeesResult.token = employeeService.updateEmployees(e);
    getEmployeesByDeptResult.token =
        employeeService.getEmployeesByDept(deptDG.selectedItem.dept_no);
}
```

Kapitel 4: Zugriff auf serverseitige Daten

Datenzugriffskomponenten von Adobe® Flex® verwenden Remoteprozeduraufrufe für die Interaktion mit Serverumgebungen, beispielsweise PHP, Adobe ColdFusion und Microsoft ASP.NET. Diese Komponenten stellen für Clientanwendungen, die mit dem Adobe Flex Framework erstellt wurden, Daten bereit und senden Daten an Back-End-Datenquellen. Eine Einführung in Datenzugriffskomponenten finden Sie im Abschnitt „Datenzugriffskomponenten“ auf Seite 5.

Arbeiten mit HTTPService-Komponenten

HTTPService-Komponenten können für alle Arten serverseitiger Technologien verwendet werden, einschließlich PHP-Seiten, ColdFusion-Seiten, JavaServer Pages (JSPs), Java-Servlets, Ruby on Rails und Microsoft ASP-Seiten. Darüber hinaus können Sie mit HTTPService auf REST-basierte Webservices zugreifen.

API-Referenzinformationen über die HTTPService-Komponente finden Sie unter mx.rpc.http.mxml.HTTPService.

Arbeiten mit PHP- und SQL-Daten

Sie können eine HTTPService-Komponente zusammen mit PHP und einem SQL-Datenbankverwaltungssystem verwenden, um die Ergebnisse einer Datenbankabfrage in einer Anwendung anzuzeigen. Mithilfe der Komponente können Sie auch Daten in die Datenbank einfügen sowie Datenbankdaten aktualisieren oder löschen. Sie können eine PHP-Seite mit GET oder POST aufrufen, um eine Datenbankabfrage durchzuführen. Anschließend können Sie die Abfrageergebnisdaten in einer XML-Struktur formatieren und die XML-Struktur der Anwendung in der HTTP-Antwort zurückgeben. Wenn das Ergebnis an die Anwendung zurückgegeben wurde, können Sie es in einem oder mehreren Steuerelementen der Benutzeroberfläche anzeigen.

MXML-Code

Die Anwendung im folgenden Beispiel ruft eine PHP-Seite mithilfe der POST-Methode auf. Die PHP-Seite fragt die MySQL-Datenbanktabelle „users“ ab. Sie formatiert die Abfrageergebnisse im XML-Format und gibt dieses XML an die Anwendung zurück. Dort wird es an die `dataProvider`-Eigenschaft eines DataGrid-Steuerelements gebunden und im DataGrid-Steuerelement angezeigt. Die Anwendung sendet auch Benutzernamen und E-Mail-Adressen neuer Benutzer an die PHP-Seite, die das Einfügen in die Benutzerdatenbanktabelle durchführt.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark"
        xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" minHeight="600"
        creationComplete="send_data()">
    <fx:Declarations>
        <s:HTTPService id="userRequest" url="http://myserver/myproj/request_post2.php"
                                                    useProxy="false" method="POST">
            <mx:request xmlns="">
                <username>{username.text}</username>
                <emailaddress>{emailaddress.text}</emailaddress>
            </mx:request>
        </s:HTTPService>
    </fx:Declarations>
    <fx:Script>
        <![CDATA[
        private function send_data():void {
        userRequest.send();
        }
        ]]>
    </fx:Script>
    <mx:Form x="20" y="10" width="300">
        <mx:FormItem>
            <s:Label text="Username" />
            <s:TextInput id="username"/>
        </mx:FormItem>
        <mx:FormItem>
            <s:Label text="Email Address" />
            <s:TextInput id="emailaddress"/>
        </mx:FormItem>
        <s:Button label="Submit" click="send_data()"/>
    </mx:Form>
    <mx:DataGrid id="dgUserRequest" x="20" y="160"
        dataProvider="{userRequest.lastResult.users.user}">
        <mx:columns>
            <mx:DataGridColumn headerText="User ID" dataField="userid"/>
            <mx:DataGridColumn headerText="User Name" dataField="username"/>
        </mx:columns>
    </mx:DataGrid>
    <s:TextInput x="20" y="340" id="selectedemailaddress"
text="{dgUserRequest.selectedItem.emailaddress}"/>

</s:Application>
```

Die `send()`-Methode des HTTPService ruft die PHP-Seite auf. Dieser Aufruf erfolgt über die `send_data()`-Methode im Skriptblock der MXML-Datei.

Die `resultFormat`-Eigenschaft der HTTPService-Komponente wird auf `object` gesetzt. Dadurch werden die Daten als Diagramm für ActionScript-Objekte wieder an die Anwendung gesendet. Dies ist der Standardwert für die `resultFormat`-Eigenschaft. Alternativ können Sie mit dem `resultFormate4x` Daten als XMLList-Objekt zurückgeben, auf das Sie Methoden gemäß ECMAScript for XML (E4X)-anwenden können. Das Ändern der `resultFormat`-Eigenschaft in `e4x` erfordert die folgenden kleineren Änderungen am MXML-Code.

Hinweis: *Wenn das Ergebnisformat `e4x` lautet, schließen Sie beim Binden an DataGrid den Stammknoten der XML-Struktur nicht in die Punktnotation ein.*

Das in diesem Beispiel zurückgegebene XML enthält keine Namespace-Informationen. Informationen zum Arbeiten mit XML, das Namespaces enthält, finden Sie unter „Verarbeiten von Ergebnissen als XML mit dem E4X-Ergebnisformat“ auf Seite 130.

```
...
<s:HTTPService id="userRequest" url="http://myserver/myproj/request_post2.php"
                            useProxy="false" method="POST" resultFormat="e4x">
...
<mx:DataGrid id="dgUserRequest" x="22" y="150"
    dataProvider="{userRequest.lastResult.user}">
...
```

Bei Verwendung des `e4x`-Ergebnisformats können Sie optional die `lastResult`-Eigenschaft an ein XMLListCollection-Objekt binden und anschließend dieses Objekt an die `DataGrid.dataProvider`-Eigenschaft binden (siehe folgendes Codefragment):

```
<fx:Declarations>
...
    <mx:XMLListCollection id="xc"
         source="{userRequest.lastResult.user}"/>
...
</fx:Declarations>
...
    <mx:DataGrid id="dgUserRequest" x="22" y="128" dataProvider="{xc}">
...
```

MySQL-Datenbankskript

Der PHP-Code für diese Anwendung verwendet die Datenbanktabelle „users“ in der MySQL-Datenbank „sample“. Das folgende MySQL-Skript erstellt die Tabelle:

```
CREATE TABLE `users` (
`userid` int(10) unsigned NOT NULL auto_increment,
`username` varchar(255) collate latin1_general_ci NOT NULL,
`emailaddress` varchar(255) collate latin1_general_ci NOT NULL,
PRIMARY KEY (`userid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci AUTO_INCREMENT=3 ;
```

PHP-Code

Diese Anwendung ruft die folgende PHP-Seite auf. Dieser PHP-Code führt Einfüge- und Abfragemethoden der SQL-Datenbank durch und gibt Abfrageergebnisse in einer XML-Struktur an die Anwendung zurück.

```
<?php
define( "DATABASE_SERVER", "servername" );
define( "DATABASE_USERNAME", "username" );
define( "DATABASE_PASSWORD", "password" );
define( "DATABASE_NAME", "sample" );

//connect to the database.
$mysql = mysql_connect(DATABASE_SERVER, DATABASE_USERNAME, DATABASE_PASSWORD);

mysql_select_db( DATABASE_NAME );

// Quote variable to make safe
function quote_smart($value)
{
    // Stripslashes
    if (get_magic_quotes_gpc()) {
    $value = stripslashes($value);
    }
    // Quote if not integer
    if (!is_numeric($value)) {
    $value = "'" . mysql_real_escape_string($value) . "'";
    }
    return $value;
}

if( $_POST["emailaddress"] AND $_POST["username"])
{
//add the user
$Query = sprintf("INSERT INTO users VALUES ('', %s, %s)",
        quote_smart($_POST['username']), quote_smart($_POST['emailaddress']));

$Result = mysql_query( $Query );
}

//return a list of all the users
$Query = "SELECT * from users";
$Result = mysql_query( $Query );

$Return = "<users>";

while ( $User = mysql_fetch_object( $Result ) )
{
$Return .= "<user><userid>".$User->userid."</userid><username>".
            $User->username."</username><emailaddress>".
            $User->emailaddress."</emailaddress></user>";
}
$Return .= "</users>";
mysql_free_result( $Result );
print ($Return)
?>
```

Arbeiten mit ColdFusion- und SQL-Daten

Sie können eine HTTPService-Komponente zusammen mit einer ColdFusion-Seite und einem SQL-Datenbankverwaltungssystem verwenden, um die Ergebnisse einer Datenbankabfrage in einer Anwendung anzuzeigen. Mithilfe der Komponente können Sie auch Daten in die Datenbank einfügen sowie Datenbankdaten aktualisieren oder löschen. Sie rufen eine ColdFusion-Seite mit GET oder POST auf, um eine Datenbankabfrage durchzuführen. Anschließend formatieren Sie die Abfrageergebnisdaten in einer XML-Struktur und geben die XML-Struktur in der HTTP-Antwort an die Anwendung zurück. Wenn das Ergebnis an die Anwendung zurückgegeben wurde, zeigen Sie es in einem oder mehreren Steuerelementen der Benutzeroberfläche an.

MXML-Code

Die Anwendung im folgenden Beispiel ruft eine ColdFusion-Seite mithilfe der POST-Methode auf. Die ColdFusion-Seite fragt die MySQL-Datenbanktabelle „users“ ab. Sie formatiert die Abfrageergebnisse im XML-Format und gibt dieses XML an die Anwendung zurück. Dort wird es an die `dataProvider`-Eigenschaft eines DataGrid-Steuerelements gebunden und im DataGrid-Steuerelement angezeigt. Die Anwendung sendet auch Benutzernamen und E-Mail-Adressen neuer Benutzer an die ColdFusion-Seite, die das Einfügen in die Benutzerdatenbanktabelle durchführt.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark"
        xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" minHeight="600"
        creationComplete="userRequest.send()">

    <fx:Declarations>
    <s:HTTPService id="userRequest" url="http://server:8500/flexapp/returncfxml.cfm"
        useProxy="false" method="POST">
        <mx:request xmlns="">
            <username>{username.text}</username>
            <emailaddress>{emailaddress.text}</emailaddress>
        </mx:request>
    </s:HTTPService>
    </fx:Declarations>
    <mx:Form x="22" y="10" width="300">
        <mx:FormItem>
            <s:Label text="Username" />
            <s:TextInput id="username"/>
        </mx:FormItem>
        <mx:FormItem>
            <s:Label text="Email Address" />
            <s:TextInput id="emailaddress"/>
        </mx:FormItem>
        <s:Button label="Submit" click="userRequest.send()"/>
    </mx:Form>
    <mx:DataGrid id="dgUserRequest" x="22" y="128"
        dataProvider="{userRequest.lastResult.users.user}">
        <mx:columns>
            <mx:DataGridColumn headerText="User ID" dataField="userid"/>
            <mx:DataGridColumn headerText="User Name" dataField="username"/>
        </mx:columns>
    </mx:DataGrid>
    <s:TextInput x="22" y="300" id="selectedemailaddress"
text="{dgUserRequest.selectedItem.emailaddress}"/>
</s:Application>
```

Die `send()`-Methode des HTTPService ruft die ColdFusion-Seite auf. Dieser Aufruf erfolgt über die `send_data()`-Methode im Skriptblock der MXML-Datei.

Die `resultFormat`-Eigenschaft der HTTPService-Komponente wird auf `object` gesetzt. Dadurch werden die Daten als Diagramm für ActionScript-Objekte wieder an die Anwendung gesendet. Dies ist der Standardwert für die `resultFormat`-Eigenschaft. Alternativ können Sie mit dem Ergebnisformat `e4x` Daten als XMLList-Objekt zurückgeben, auf das Sie Methoden gemäß ECMAScript for XML (E4X) anwenden können. Das Ändern der `resultFormat`-Eigenschaft in `e4x` erfordert die folgenden kleineren Änderungen am MXML-Code.

Hinweis: *Wenn das Ergebnisformat `e4x` lautet, schließen Sie beim Binden an DataGrid den Stammknoten der XML-Struktur nicht in die Punktnotation ein.*

Das in diesem Beispiel zurückgegebene XML enthält keine Namespace-Informationen. Informationen zum Arbeiten mit XML, das Namespaces enthält, finden Sie unter „Verarbeiten von Ergebnissen als XML mit dem E4X-Ergebnisformat“ auf Seite 130.

```
...
<s:HTTPService id="userRequest" url="http://myserver:8500/flexapp/returncfxml.cfm"
                        useProxy="false" method="POST" resultFormat="e4x">
...
<mx:DataGrid id="dgUserRequest" x="22" y="128"
    dataProvider="{userRequest.lastResult.user}">
...
```

Bei Verwendung des `e4x`-Ergebnisformats können Sie optional die `lastResult`-Eigenschaft an ein XMLListCollection-Objekt binden und anschließend dieses Objekt an die DataGrid-`dataProvider`-Eigenschaft binden (siehe folgendes Codefragment):

```
<fx:Declarations>
...
    <mx:XMLListCollection id="xc"
         source="{userRequest.lastResult.user}"/>
...
</fx:Declarations>
...
    <mx:DataGrid id="dgUserRequest" x="22" y="128" dataProvider="{xc}">
...
```

SQL-Skript

Der ColdFusion-Code für diese Anwendung verwendet die Datenbanktabelle „users“ in der MySQL-Datenbank „sample“. Das folgende MySQL-Skript erstellt die Tabelle:

```
CREATE TABLE `users` (
`userid` int(10) unsigned NOT NULL auto_increment,
`username` varchar(255) collate latin1_general_ci NOT NULL,
`emailaddress` varchar(255) collate latin1_general_ci NOT NULL,
PRIMARY KEY (`userid`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1 COLLATE=latin1_general_ci AUTO_INCREMENT=3 ;
```

ColdFusion-Code

Die unter „Arbeiten mit ColdFusion- und SQL-Daten“ genannte Anwendung ruft die ColdFusion-Anwendung `returncfxml.cfm` auf. Dieser ColdFusion-Code führt Einfüge- und Abfragemethoden der SQL-Datenbank durch und gibt Abfrageergebnisse an die Anwendung zurück. Die ColdFusion-Seite verwendet das `cfquery`-Tag, um Daten in die Datenbank einzufügen und die Datenbank abzufragen, sowie das `cfxml`-Tag, um die Abfrageergebnisse in einer XML-Struktur zu formatieren.

```
<!--- returncfxml.cfm --->

<cfprocessingdirective pageencoding = "utf-8" suppressWhiteSpace = "Yes">
<cfif isDefined("username") and isDefined("emailaddress") and username NEQ "">
    <cfquery name="addempinfo" datasource="sample">
    INSERT INTO users (username, emailaddress) VALUES (
    <cfqueryparam value="#username#" cfsqltype="CF_SQL_VARCHAR" maxlength="255">,
        <cfqueryparam value="#emailaddress#" cfsqltype="CF_SQL_VARCHAR" maxlength="255"> )
</cfquery>
</cfif>
<cfquery name="alluserinfo" datasource="sample">
    SELECT userid, username, emailaddress FROM users
</cfquery>
<cfxml variable="userXML">
    <users>
        <cfloop query="alluserinfo">
            <cfoutput>
                <user>
                    <userid>#toString(userid)#</userid>
                    <username>#username#</username>
                    <emailaddress>#emailaddress#</emailaddress>
                </user>
        </cfoutput>
        </cfloop>
    </users>
</cfxml>
<cfoutput>#userXML#</cfoutput>
</cfprocessingdirective>
```

Arbeiten mit JavaServer Pages

Sie können eine Flex-HTTPService-Komponente zusammen mit einer JSP-Seite und einem SQL-Datenbankverwaltungssystem verwenden, um die Ergebnisse einer Datenbankabfrage in einer Anwendung anzuzeigen. Mithilfe der Komponente können Sie auch Daten in die Datenbank einfügen sowie Datenbankdaten aktualisieren oder löschen. Rufen Sie eine JSP-Seite mit GET oder POST auf, um eine Datenbankabfrage durchzuführen. Anschließend formatieren Sie die Abfrageergebnisdaten in einer XML-Struktur und geben die XML-Struktur in der HTTP-Antwort an die Anwendung zurück. Wenn das Ergebnis an die Anwendung zurückgegeben wurde, zeigen Sie es in einem oder mehreren Steuerelementen der Benutzeroberfläche an.

MXML-Code

Die Anwendung im folgenden Beispiel ruft eine JSP-Seite auf, die Daten von einer SQL-Datenbank abruft. Sie formatiert Datenbankabfrageergebnisse im XML-Format und gibt dieses XML an die Anwendung zurück. Dort wird es an die `dataProvider`-Eigenschaft eines DataGrid-Steuerelements gebunden und im DataGrid-Steuerelement angezeigt.

```
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark"
        xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" minHeight="600">

    <fx:Declarations>
        <s:HTTPService id="srv" url="catalog.jsp"/>
    </fx:Declarations>

    <mx:DataGrid dataProvider="{srv.lastResult.catalog.product}"
                                    width="100%" height="100%"/>

    <s:Button label="Get Data" click="srv.send()"/>

</mx:Application>
```

Die `send()`-Methode des HTTPService ruft die JSP-Seite auf. Dieser Aufruf erfolgt im `click`-Ereignis des Button-Steuerelements in der MXML-Datei.

Die `resultFormat`-Eigenschaft der HTTPService-Komponente wird auf `object` gesetzt. Dadurch werden die Daten als Diagramm für ActionScript-Objekte wieder an die Anwendung gesendet. Dies ist der Standardwert für die `resultFormat`-Eigenschaft. Alternativ können Sie mit dem Ergebnisformat `e4x` Daten als XMLList-Objekt zurückgeben, auf das Sie Methoden gemäß ECMAScript for XML (E4X) anwenden können. Das Ändern der `resultFormat`-Eigenschaft in `e4x` erfordert die folgenden kleineren Änderungen am MXML-Code.

Hinweis: *Wenn das Ergebnisformat `e4x` lautet, schließen Sie beim Binden an DataGrid den Stammknoten der XML-Struktur nicht in die Punktnotation ein.*

Das in diesem Beispiel zurückgegebene XML enthält keine Namespace-Informationen. Informationen zum Arbeiten mit XML, das Namespaces enthält, finden Sie unter „Verarbeiten von Ergebnissen als XML mit dem E4X-Ergebnisformat“ auf Seite 130.

```
...
    <s:HTTPService id="srv" url="catalog.jsp" resultFormat="e4x"/>
...
    <mx:DataGrid dataProvider="{srv.lastResult.product}" width="100%" height="100%"/>
```

Bei Verwendung des e4x-Ergebnisformats können Sie optional die `lastResult`-Eigenschaft an ein XMLListCollection-Objekt binden und anschließend dieses Objekt an die DataGrid.dataProvider-Eigenschaft binden:

```
<fx:Declarations>
...
    <mx:XMLListCollection id="xc"
         source="{userRequest.lastResult.user}"/>
...
</fx:Declarations>
...
    <mx:DataGrid id="dgUserRequest" x="22" y="128" dataProvider="{xc}">
...
```

JSP-Code

Im folgenden Beispiel wird die in dieser Anwendung verwendete JSP-Seite gezeigt. Diese JSP-Seite ruft eine Datenbank nicht direkt auf. Sie ruft Daten über die Java-Klasse „ProductService“ ab, die wiederum die Java-Klasse „Product“ verwendet, um einzelne Produkte darzustellen.

```
<%@page import="flex.samples.product.ProductService,
                flex.samples.product.Product,
                java.util.List"%>
<?xml version="1.0" encoding="utf-8"?>
<catalog>
<%
    ProductService srv = new ProductService();
    List list = null;
    list = srv.getProducts();
    Product product;
    for (int i=0; i<list.size(); i++)
    {
        product = (Product) list.get(i);
%>
<product productId="<%= product.getProductId()%>">
<name><%= product.getName() %></name>
<description><%= product.getDescription() %></description>
<price><%= product.getPrice() %></price>
<image><%= product.getImage() %></image>
<category><%= product.getCategory() %></category>
<qtyInStock><%= product.getQtyInStock() %></qtyInStock>
</product>
<%
    }
%>
</catalog>
```

Aufrufen von HTTP-Diensten in ActionScript

Im folgenden Beispiel wird ein HTTP-Dienstaufruf in einem ActionScript-Skriptblock gezeigt. Durch Aufrufen der `useHTTPService()`-Methode wird der Dienst deklariert und das Ziel festgelegt, es werden result-Ereignis- und fault-Ereignis-Listener eingerichtet und die `send()`-Methode des Dienstes aufgerufen.

```
<?xml version="1.0"?>
<!-- fds\rpc\HttpServiceInAS.mxml. Compiles -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" verticalGap="10">
    <mx:Script>
        <![CDATA[
            import mx.controls.Alert;
            import mx.rpc.http.HTTPService;
            import mx.rpc.events.ResultEvent;
            import mx.rpc.events.FaultEvent;

            private var service:HTTPService
            public function useHttpService(parameters:Object):void {
                service = new HTTPService();
                service.url = "catalog.jsp";
                service.method = "POST";
                service.addEventListener("result", httpResult);
                service.addEventListener("fault", httpFault);
                service.send(parameters);
            }

            public function httpResult(event:ResultEvent):void {
                var result:Object = event.result;
            //Do something with the result.
            }

            public function httpFault(event:FaultEvent):void {
                var faultstring:String = event.fault.faultString;
                Alert.show(faultstring);
            }
        ]]>
    </mx:Script>
</mx:Application>
```

Arbeiten mit WebService-Komponenten

Mit dem Flex-Framework erstellte Anwendungen können mit SOAP-basierten Webservices interagieren, die ihre Schnittstellen in einem Web Services Description Language 1.1-(WSDL 1.1-)Dokument definieren, das über eine URL zur Verfügung steht. WSDL ist ein Standardformat, mit dem Folgendes beschrieben wird: Meldungen, die ein Webservice verstehen kann, das Format seiner Antworten auf diese Meldungen, das Protokoll, das der Webservice unterstützt, und schließlich das Ziel, an das die Meldungen gesendet werden. Die Flex Web Service API unterstützt im Allgemeinen Simple Object Access Protocol (SOAP) 1.1, XML Schema 1.0 (Versionen 1999, 2000 und 2001) sowie WSDL 1.1 RPC-encoded, RPC-literal und Document-literal (Stilparameter „bare“ und „wrapped“). Die beiden häufigsten Typen von Webservices verwenden Remoteprozeduraufruf´-(RPC encoded-) oder Document-literal-SOAP-Bindungen; die Begriffe *encoded* und *literal* weisen auf den Typ der WSDL-to-SOAP-Zuordnung hin, die ein Dienst verwendet.

Flex unterstützt Webserviceanforderungen und -ergebnisse, die als SOAP-Meldungen formatiert sind. SOAP stellt die Definition des auf XML basierenden Formats bereit, das Sie für den Austausch von strukturierten und typisierten Daten zwischen einem Webserviceclient (beispielsweise einer in Flex erstellten Anwendung) und einem Webservice verwenden können.

Adobe® Flash® Player arbeitet innerhalb einer Sicherheitssandbox, die einschränkt, welche mit Flex erstellten Anwendungen und sonstigen mit Flash Player erstellte Anwendungen über HTTP zugreifen können. Für mit Flash erstellte Anwendungen ist der HTTP-Zugriff auf Ressourcen nur in derselben Domäne und über dasselbe Protokoll zulässig, über das sie Daten erhalten. Dies stellt ein Problem für Webservices dar, weil sie normalerweise von Remotestandorten aufgerufen werden. Der Proxydienst, der in ADEP Data Services und BlazeDS zur Verfügung steht, fängt Anforderungen an Remotewebservices ab, leitet die Anforderungen um und gibt anschließend die Antworten an den Client zurück.

Wenn Sie ADEP Data Services oder BlazeDS nicht verwenden, können Sie auf Webservices in derselben Domäne wie Ihre Anwendung zugreifen oder die crossdomain.xml-Datei (Cross-Domain-Richtlinie), die den Zugriff über die Domäne ihrer Anwendung ermöglicht, muss auf dem Webserver installiert sein, auf dem der RPC-Dienst gehostet wird.

API-Referenzinformationen über die WebService-Komponente finden Sie unter mx.rpc.soap.mxml.WebService.

WebService-Beispielanwendung

Der folgende Beispielcode ist für eine Anwendung, die eine WebService-Komponente zum Aufrufen von Webservicemethoden verwendet.

MXML-Code

Die Anwendung im folgenden Beispiel ruft einen Webservice auf, der die SQL-Datenbanktabelle „users“ abfragt und Daten an die Anwendung zurückgibt. Dort werden die Daten an die `dataProvider`-Eigenschaft eines DataGrid-Steuerelements gebunden und im DataGrid-Steuerelement angezeigt. Die Anwendung sendet auch Benutzernamen und E-Mail-Adresse neuer Benutzer an den Webservice, der das Einfügen in die Benutzerdatenbanktabelle durchführt. Die Back-End-Implementierung des Webservice ist eine ColdFusion-Komponente; dieselbe ColdFusion-Komponente wird als Remoteobjekt in „Arbeiten mit RemoteObject-Komponenten“ auf Seite 105 aufgerufen.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark"
        xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" minHeight="600">
    <fx:Declarations>
        <s:WebService
            id="userRequest"
            wsdl="http://localhost:8500/flexapp/returnusers.cfc?wsdl">

            <mx:operation name="returnRecords" resultFormat="object"
                          fault="mx.controls.Alert.show(event.fault.faultString)"
                          result="remotingCFCHandler(event)"/>

            <mx:operation name="insertRecord" result="insertCFCHandler()"
                          fault="mx.controls.Alert.show(event.fault.faultString)"/>
        </s:WebService>
    </fx:Declarations>
    <fx:Script>
        <![CDATA[
            import mx.rpc.events.ResultEvent;

            private function remotingCFCHandler(e:ResultEvent):void
            {
                dgUserRequest.dataProvider = e.result;
            }

            private function insertCFCHandler():void
```

```
            {
                userRequest.returnRecords();
            }
            private function clickHandler():void
            {
                userRequest.insertRecord(username.text, emailaddress.text);
            }
        ]]>
    </fx:Script>

    <mx:Form x="22" y="10" width="300">
        <mx:FormItem>
            <s:Label text="Username" />
            <s:TextInput id="username"/>
        </mx:FormItem>
        <mx:FormItem>
            <s:Label text="Email Address" />
            <s:TextInput id="emailaddress"/>
        </mx:FormItem>
        <s:Button label="Submit" click="clickHandler()"/>
    </mx:Form>

    <mx:DataGrid id="dgUserRequest" x="22" y="160">
        <mx:columns>
            <mx:DataGridColumn headerText="User ID" dataField="USERID"/>
            <mx:DataGridColumn headerText="User Name" dataField="USERNAME"/>
        </mx:columns>
    </mx:DataGrid>

    <s:TextInput x="22" y="320" id="selectedemailaddress"
text="{dgUserRequest.selectedItem.emailaddress}"/>
</s:Application>
```

WSDL-Dokument

Im folgenden Beispiel wird das WSDL-Dokument gezeigt, das die API des Webservice definiert:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://flexapp"
    xmlns:apachesoap="http://xml.apache.org/xml-soap"
    xmlns:impl="http://flexapp" xmlns:intf="http://flexapp"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tns1="http://rpc.xml.coldfusion"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by ColdFusion version 8,0,0,171651-->
    <wsdl:types>
<schema targetNamespace="http://rpc.xml.coldfusion"
xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://flexapp"/>
    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="CFCInvocationException">
<sequence/>
    </complexType>

    <complexType name="QueryBean">
```

```
<sequence>
    <element name="columnList" nillable="true" type="impl:ArrayOf_xsd_string"/>
    <element name="data" nillable="true" type="impl:ArrayOfArrayOf_xsd_anyType"/>
</sequence>
    </complexType>
</schema>
<schema targetNamespace="http://flexapp" xmlns="http://www.w3.org/2001/XMLSchema">
    <import namespace="http://rpc.xml.coldfusion"/>

    <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
    <complexType name="ArrayOf_xsd_string">
<complexContent>
    <restriction base="soapenc:Array">
<attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:string[]"/>
    </restriction>
</complexContent>
    </complexType>
    <complexType name="ArrayOfArrayOf_xsd_anyType">

<complexContent>
    <restriction base="soapenc:Array">
<attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:anyType[][]"/>
    </restriction>
</complexContent>
    </complexType>
</schema>
    </wsdl:types>

    <wsdl:message name="CFCInvocationException">

<wsdl:part name="fault" type="tns1:CFCInvocationException"/>
</wsdl:message>
<wsdl:message name="returnRecordsRequest">
</wsdl:message>
<wsdl:message name="insertRecordResponse">
</wsdl:message>
<wsdl:message name="returnRecordsResponse">
<wsdl:part name="returnRecordsReturn" type="tns1:QueryBean"/>
</wsdl:message>
<wsdl:message name="insertRecordRequest">
<wsdl:part name="username" type="xsd:string"/>
<wsdl:part name="emailaddress" type="xsd:string"/>
</wsdl:message>
<wsdl:portType name="returncfxml">
<wsdl:operation name="insertRecord" parameterOrder="username emailaddress">
<wsdl:input message="impl:insertRecordRequest" name="insertRecordRequest"/>
<wsdl:output message="impl:insertRecordResponse" name="insertRecordResponse"/>
<wsdl:fault message="impl:CFCInvocationException" name="CFCInvocationException"/>
</wsdl:operation>
<wsdl:operation name="returnRecords">
<wsdl:input message="impl:returnRecordsRequest" name="returnRecordsRequest"/>
<wsdl:output message="impl:returnRecordsResponse" name="returnRecordsResponse"/>
<wsdl:fault message="impl:CFCInvocationException" name="CFCInvocationException"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="returncfxml.cfcSoapBinding" type="impl:returncfxml">
<wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
```

```
<wsdl:operation name="insertRecord">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="insertRecordRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://flexapp" use="encoded"/>
</wsdl:input>
<wsdl:output name="insertRecordResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://flexapp" use="encoded"/>
</wsdl:output>
<wsdl:fault name="CFCInvocationException">
<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://flexapp" use="encoded"/>
</wsdl:fault>
</wsdl:operation>
<wsdl:operation name="returnRecords">
<wsdlsoap:operation soapAction=""/>
<wsdl:input name="returnRecordsRequest">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://flexapp" use="encoded"/>
</wsdl:input>
<wsdl:output name="returnRecordsResponse">
<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://flexapp" use="encoded"/>
</wsdl:output>
<wsdl:fault name="CFCInvocationException">
<wsdlsoap:fault encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
name="CFCInvocationException" namespace="http://flexapp" use="encoded"/>
</wsdl:fault>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="returncfxmlService">
<wsdl:port binding="impl:returncfxml.cfcSoapBinding" name="returncfxml.cfc">
<wsdlsoap:address location="http://localhost:8500/flexapp/returnusers.cfc"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Aufrufen von Webservices in ActionScript

Im folgenden Beispiel wird ein Webserviceaufruf in einem ActionScript-Skriptblock gezeigt. Durch Aufrufen der `useWebService()`-Methode wird der Dienst deklariert, das Ziel festgelegt, das WSDL-Dokument abgerufen und die `echoArgs()`-Methode des Dienstes aufgerufen.

Hinweis: *Beim Deklarieren einer WebService-Komponente in ActionScript rufen Sie die* `WebService.loadWSDL()`*-Methode auf.*

```
<?xml version="1.0"?>
<!-- fds\rpc\WebServiceInAS.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:Script>
        <![CDATA[
        import mx.rpc.soap.WebService;
        import mx.rpc.events.ResultEvent;
        import mx.rpc.events.FaultEvent;
        private var ws:WebService;
        public function useWebService(intArg:int, strArg:String):void {
            ws = new WebService();
            ws.wsdl="http://myserver:8500/flexapp/app1.cfc?wsdl";
            ws.echoArgs.addEventListener("result", echoResultHandler);
            ws.addEventListener("fault", faultHandler);
            ws.loadWSDL();
            ws.echoArgs(intArg, strArg);
        }

        public function echoResultHandler(event:ResultEvent):void {
            var retStr:String = event.result.echoStr;
            var retInt:int = event.result.echoInt;
         //Do something.
        }

        public function faultHandler(event:FaultEvent):void {
      //deal with event.fault.faultString, etc
        }
        ]]>
    </mx:Script>
</mx:Application>
```

Namen für reservierte Methoden

WebService-Methoden können aufgerufen werden, indem sie nach einer Dienstvariablen benannt werden. Namenskonflikte können jedoch auftreten, wenn ein Methodenname mit einer definierten Methode für den Dienst übereinstimmt. Mithilfe der folgenden Methode in ActionScript für eine WebService-Komponente können Sie die Methode des gegebenen Namens zurückgeben:

```
public function getOperation(name:String):Operation
```

Lesen von WSDL-Dokumenten

Sie können ein WSDL-Dokument in einem Webbrowser, einem einfachen Webeditor, einem XML-Editor oder einer Entwicklungsumgebung wie Adobe Dreamweaver anzeigen, das ein integriertes Dienstprogramm zum Anzeigen von WSDL-Dokumenten in einem einfach zu lesenden Format enthält.

Ein WSDL-Dokument enthält die in der folgenden Tabelle beschriebenen Tags.

Tag	Beschreibung
`<binding>`	Legt das Protokoll fest, das Clients wie z. B. mit Flex erstellte Anwendungen für die Kommunikation mit einem Webservice verwenden. Bindungen sind für SOAP, HTTP GET, HTTP POST und MIME vorhanden. Flex unterstützt nur die SOAP-Bindung.
`<fault>`	Gibt einen Fehlerwert an, der aufgrund eines Problems beim Verarbeiten der Meldung zurückgegeben wird.
`<input>`	Gibt eine Meldung an, die ein Client, z. B. eine mit Flex erstellte Anwendung, an einen Webservice sendet.

Tag	Beschreibung
`<message>`	Definiert die Daten, die eine WebService-Methode überträgt.
`<operation>`	Definiert eine Kombination aus den `<input>`-, `<output>`- und `<fault>`-Tags.
`<output>`	Gibt eine Meldung an, die der Webservice an einen Webserviceclient, z. B. eine mit Flex erstellte Anwendung, sendet.
`<port>`	Gibt einen Webserviceendpunkt an, der eine Zuordnung zwischen einer Bindung und einer Netzwerkadresse angibt.
`<portType>`	Definiert eine oder mehrere Methoden, die ein Webservice bereitstellt.
`<service>`	Definiert eine Sammlung von `<port>`-Tags. Jeder Dienst wird einem `<portType>`-Tag zugeordnet und gibt verschiedene Möglichkeiten für den Zugriff auf Methoden in diesem `<portType>`-Tag an.
`<types>`	Definiert Datentypen, die von den Meldungen eines Webservice verwendet werden.

RPC-orientierte und dokumentorientierte Methoden

Eine WSDL-Datei kann entweder RPC-orientierte oder dokumentorientierte (Document-literal) Methoden angeben. Flex unterstützt beide Methodenstile.

Beim Aufrufen einer RPC-basierten Methode sendet eine Anwendung eine SOAP-Meldung, die eine Methode und zugehörige Parameter angibt. Beim Aufrufen einer dokumentbasierten Methode sendet eine Anwendung eine SOAP-Meldung, die ein XML-Dokument enthält.

In einem WSDL-Dokument hat jedes `<port>`-Tag eine `binding`-Eigenschaft, die den Namen eines bestimmten `<soap:binding>`-Tags angibt (siehe folgendes Beispiel):

```
<binding name="InstantMessageAlertSoap" type="s0:InstantMessageAlertSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
        style="document"/>
```

Die `style`-Eigenschaft des verknüpften `<soap:binding>`-Tags legt den Methodenstil fest. In diesem Beispiel handelt es sich bei dem Stil um `document`.

Jede Methode in einem Dienst kann denselben Stil angeben oder den Stil überschreiben, der für den mit dem Dienst verknüpften Port angegeben wird (siehe folgendes Beispiel):

```
<operation name="SendMSN">
    <soap:operation soapAction="http://www.bindingpoint.com/ws/imalert/
        SendMSN"style="document"/>
```

Stateful-Webservices

Flex verwendet Java-Serversitzungen, um den Status von Webserviceendpunkten, die Cookies zur Speicherung von Sitzungsinformationen verwenden, aufrechtzuerhalten. Diese Funktion fungiert als Vermittler zwischen Anwendungen und Webservices. Sie fügt den von einem Endpunkt an eine Anwendung übertragenen Inhalten die ID des Endpunkts hinzu. Wenn der Endpunkt Sitzungsinformationen sendet, werden sie von der Anwendung empfangen. Diese Funktion erfordert keine Konfiguration; sie wird nicht für Ziele unterstützt, die bei der Verwendung des Proxydiensts den RMTP-Kanal verwenden.

Arbeiten mit SOAP-Headern

Ein SOAP-Header ist ein optionales Tag in einem SOAP-Envelope. Er enthält normalerweise anwendungsspezifische Informationen wie beispielsweise Authentifizierungsdaten.

Hinzufügen von SOAP-Headern zu Webserviceanforderungen

Einige Webservices erfordern es, dass Sie beim Aufrufen einer Methode einen SOAP-Header weitergeben.

Sie können einen SOAP-Header allen Webservicemethoden oder einzelnen Methoden hinzufügen, indem Sie die `addHeader()`- oder `addSimpleHeader()`-Methoden eines WebService- oder Operation-Objekts in einer Ereignis-Listener-Funktion aufrufen.

Wenn Sie die `addHeader()`-Methode verwenden, müssen Sie zuerst SOAPHeader- und QName-Objekte separat erstellen. Die `addHeader()`-Methode hat folgende Signatur:

```
addHeader(header:mx.rpc.soap.SOAPHeader):void
```

Verwenden Sie zum Erstellen eines SOAPHeader-Objekts den folgenden Konstruktor:

```
SOAPHeader(qname:QName, content:Object)
```

Verwenden Sie zum Erstellen des QName-Objekts im ersten Parameter der `SOAPHeader()`-Methode den folgenden Konstruktor:

```
QName(uri:String, localName:String)
```

Der `content`-Parameter des `SOAPHeader()`-Konstruktors ist eine Gruppe von Name/Wert-Paaren auf der Basis des folgenden Formats:

```
{name1:value1, name2:value2}
```

Die `addSimpleHeader()`-Methode ist ein Kurzbefehl für einen einzelnen Name/Wert-SOAP-Header. Wenn Sie die `addSimpleHeader()`-Methode verwenden, erstellen Sie SOAPHeader- und QName-Objekte in Parametern der Methode. Die `addSimpleHeader()`-Methode hat folgende Signatur:

```
addSimpleHeader(qnameLocal:String, qnameNamespace:String, headerName:String,
    headerValue:Object):void
```

Die `addSimpleHeader()`-Methode weist die folgenden Parameter auf:

- *`qnameLocal`* ist der lokale Name für den Header „QName“.
- *`qnameNamespace`* ist der Namespace für den Header „QName“.
- *`headerName`* ist der Name des Headers.
- *`headerValue`* ist der Wert des Headers. Dies kann ein String sein, wenn es ein einfacher Wert ist, ein Objekt, das eine einfache XML-Kodierung erfährt oder XML, wenn Sie den Header XML selbst angeben wollen.

  Der Code im folgenden Beispiel zeigt, wie die `addHeader()`- und `addSimpleHeader()`-Methoden zum Hinzufügen eines SOAP-Headers verwendet werden sollen. Die Methoden werden in der Ereignis-Listener-Funktion `headers` aufgerufen und der Ereignis-Listener wird in der `load`-Eigenschaft eines `<mx:WebService>`-Tags zugewiesen:

```
<?xml version="1.0"?>
<!-- fds\rpc\WebServiceAddHeader.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" height="600">
    <mx:WebService id="ws" wsdl="http://myserver:8500/flexapp/app1.cfc?wsdl"
load="headers();"/>
    <mx:Script>
        <![CDATA[
            import mx.rpc.soap.SOAPHeader;
            private var header1:SOAPHeader;
            private var header2:SOAPHeader;
            public function headers():void {

            // Create QName and SOAPHeader objects.
            var q1:QName=new QName("http://soapinterop.org/xsd", "Header1");
            header1=new SOAPHeader(q1, {string:"bologna",int:"123"});
            header2=new SOAPHeader(q1, {string:"salami",int:"321"});


            // Add the header1 SOAP Header to all web service requests.
            ws.addHeader(header1);

            // Add the header2 SOAP Header to the getSomething operation.
            ws.getSomething.addHeader(header2);

            // Within the addSimpleHeader method,
            // which adds a SOAP header to web
            //service requests, create SOAPHeader and QName objects.
            ws.addSimpleHeader
                ("header3", "http://soapinterop.org/xsd", "foo","bar");
            }
        ]]>
    </mx:Script>
</mx:Application>
```

Löschen von SOAP-Headern

Mithilfe der `clearHeaders()`-Methode des WebService- oder Methodenobjekts entfernen Sie SOAP-Header, die Sie dem Objekt hinzugefügt haben (siehe folgendes Beispiel für ein WebService-Objekt). Sie müssen `clearHeaders()` auf der Ebene (WebService oder Methode) aufrufen, auf der der Header hinzugefügt wurde.

```
<?xml version="1.0"?>
<!-- fds\rpc\WebServiceClearHeader.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" height="600" >

    <!-- The value of the destination property is for demonstration only and is not a real
destination. -->
    <mx:WebService id="ws" wsdl="http://myserver:8500/flexapp/app1.cfc?wsdl"
load="headers();"/>

    <mx:Script>
        <![CDATA[
            import mx.rpc.*;
            import mx.rpc.soap.SOAPHeader;

            private function headers():void {
                // Create QName and SOAPHeader objects.
                var q1:QName=new QName("Header1", "http://soapinterop.org/xsd");
                var header1:SOAPHeader=new SOAPHeader(q1, {string:"bologna",int:"123"});
                var header2:SOAPHeader=new SOAPHeader(q1, {string:"salami",int:"321"});
                    // Add the header1 SOAP Header to all web service request.
                ws.addHeader(header1);
                // Add the header2 SOAP Header to the getSomething operation.
                ws.getSomething.addHeader(header2);

                // Within the addSimpleHeader method, which adds a SOAP header to all
                // web service requests, create SOAPHeader and QName objects.
                ws.addSimpleHeader("header3","http://soapinterop.org/xsd", "foo", "bar");
            }

            // Clear SOAP headers added at the WebService and Operation levels.
            private function clear():void {
                ws.clearHeaders();
                ws.getSomething.clearHeaders();
            }
        ]]>
    </mx:Script>

    <mx:HBox>
            <mx:Button label="Clear headers and run again" click="clear();"/>
    </mx:HBox>

</mx:Application>
```

Umleiten eines Webservice an eine andere URL

Einige Webservices erfordern es, dass Sie nach der WSDL-Verarbeitung und dem ersten Aufruf des Webservice zu einer anderen Endpunkt-URL wechseln. Beispiel: Sie möchten einen Webservice verwenden, für den Sie Sicherheitsanmeldedaten übergeben müssen. Wenn Sie den Webservice zum Senden der Anmeldedaten aufrufen, übernimmt er die Anmeldedaten und gibt die aktuelle Endpunkt-URL zurück, für die die Unternehmensmethoden des Dienstes verwendet werden sollen. Vor dem Aufrufen der Unternehmensmethoden müssen Sie die `endpointURI`-Eigenschaft Ihrer WebService-Komponente ändern.

Im folgenden Beispiel wird ein result-Ereignis-Listener gezeigt, der die Endpunkt-URL speichert, die ein Webservice in einer Variablen zurückgibt, und anschließend diese Variable an eine Funktion übergibt, um die Endpunkt-URL für nachfolgende Anforderungen zu ändern:

```
...
public function onLoginResult(event:ResultEvent):void {

//Extract the new service endpoint from the login result.
var newServiceURL = event.result.serverUrl;

// Redirect all service operations to the URL received in the login result.
    serviceName.endpointURI=newServiceURL;

}
...
```

Ein Webservice, für den Sie Sicherheitsanmeldedaten übergeben, kann unter Umständen auch einen Bezeichner zurückgeben, den Sie in einem SOAP-Header für nachfolgende Anforderungen anfügen müssen. Weitere Informationen hierzu finden Sie unter „Arbeiten mit SOAP-Headern“ auf Seite 92.

Serialisieren von Webservicedaten

Kodieren von ActionScript-Daten

Die folgende Tabelle zeigt die Kodierungszuordnungen von ActionScript 3.0-Typen zu komplexen XML-Schematypen.

Definition des XML-Schemas	Unterstützte ActionScript 3.0-Typen	Hinweis
Elemente der obersten Ebene		
xsd:element nillable == true	Object	Wenn der Eingabewert `null` ist, wird die kodierte Ausgabe mit dem `xsi:nil`-Attribut festgelegt.
xsd:element fixed != null	Object	Der Eingabewert wird ignoriert und stattdessen wird der vorgegebene Wert verwendet.
xsd:element default != null	Object	Wenn der Eingabewert `null` ist, wird stattdessen der Standardwert verwendet.
Lokale Elemente		
xsd:element maxOccurs == 0	Object	Der Eingabewert wird ignoriert und von der kodierten Ausgabe weggelassen.
xsd:element maxOccurs == 1	Object	Der Eingabewert wird als einzelne Entität verarbeitet. Wenn der verknüpfte Typ ein SOAP-kodiertes Array ist, werden Arrays und `mx.collection.IList`-Implementierungen intakt übergeben und vom SOAP-Kodierer für diesen Typ als Spezialfall verarbeitet.
xsd:element maxOccurs > 1	Object	Der Eingabewert sollte iterabel sein (beispielsweise ein Array oder eine `mx.collections.IList`-Implementierung), obwohl nicht iterable Werte vor der Verarbeitung eingeschlossen werden. Einzelne Elemente werden als separate Entitäten entsprechend der Definition kodiert.
xsd:element minOccurs == 0	Object	Wenn der Eingabewert undefiniert oder `null` ist, wird die kodierte Ausgabe weggelassen.

Die folgende Tabelle zeigt die Kodierungszuordnungen von ActionScript 3.0-Typen zu integrierten XML-Schematypen.

XML-Schematyp	**Unterstützte ActionScript 3.0-Typen**	**Hinweis**
`xsd:anyType` `xsd:anySimpleType`	Object	Boolean -> `xsd:boolean` ByteArray -> `xsd:base64Binary` Date -> `xsd:dateTime` int -> `xsd:int` Number -> `xsd:double` String -> `xsd:string` uint -> `xsd:unsignedInt`
`xsd:base64Binary`	`flash.utils.ByteArray`	`mx.utils.Base64Encoder` wird verwendet (ohne Zeilenumbruch).
`xsd:boolean`	Boolean Number Object	Immer als `true` oder `false` kodiert. `Number == 1` dann `true`, ansonsten `false`. `Object.toString()` == "true" oder "1" dann `true`, ansonsten `false`.
`xsd:byte` `xsd:unsignedByte`	Number String	String zuerst in „Number“ konvertiert.
`xsd:date`	Date Number String	Date UTC-Zugriffsmethoden werden verwendet. „Number“ wird zum Festlegen von „Date.time“ verwendet. Bei String wird angenommen, dass er entsprechend vorformatiert und kodiert wird.
`xsd:dateTime`	Date Number String	Date UTC-Zugriffsmethoden werden verwendet. „Number“ wird zum Festlegen von „Date.time“ verwendet. Bei String wird angenommen, dass er entsprechend vorformatiert und kodiert wird.
`xsd:decimal`	Number String	`Number.toString()` wird verwendet. Infinity, -Infinity und NaN sind für diesen Typ ungültig. String zuerst in „Number“ konvertiert.
`xsd:double`	Number String	Begrenzt auf Bereich von „Number“. String zuerst in „Number“ konvertiert.
`xsd:duration`	Object	`Object.toString()` wird aufgerufen.
`xsd:float`	Number String	Begrenzt auf Bereich von „Number“. String zuerst in „Number“ konvertiert.
`xsd:gDay`	Date Number String	`Date.getUTCDate()` wird verwendet. „Number“ direkt für Tag verwendet. String als „Number“ für Tag analysiert.

XML-Schematyp	Unterstützte ActionScript 3.0-Typen	Hinweis
`xsd:gMonth`	Date Number String	`Date.getUTCMonth()` wird verwendet. „Number“ direkt für Monat verwendet. String als „Number“ für Monat analysiert.
`xsd:gMonthDay`	Date String	`Date.getUTCMonth()` und `Date.getUTCDate()` werden verwendet. String für Monat und Tag analysiert.
`xsd:gYear`	Date Number String	`Date.getUTCFullYear()` wird verwendet. „Number“ direkt für Jahr verwendet. String als „Number“ für Jahr analysiert.
`xsd:gYearMonth`	Date String	`Date.getUTCFullYear()` und `Date.getUTCMonth()` werden verwendet. String für Jahr und Monat analysiert.
`xsd:hexBinary`	flash.utils.ByteArray	mx.utils.HexEncoder wird verwendet.

`xsd:integer`
`und Derivate:`
`xsd:negativeInteger`
`xsd:nonNegativeInteger`
`xsd:positiveInteger`
`xsd:nonPositiveInteger`
	Number
	String
		Begrenzt auf Bereich von „Number“.
		String zuerst in „Number“ konvertiert.

`xsd:int` `xsd:unsignedInt`	Number String	String zuerst in „Number“ konvertiert.
`xsd:long` `xsd:unsignedLong`	Number String	String zuerst in „Number“ konvertiert.
`xsd:short` `xsd:unsignedShort`	Number String	String zuerst in „Number“ konvertiert.

XML-Schematyp	Unterstützte ActionScript 3.0-Typen	Hinweis

`xsd:string`
`und Derivate:`
`xsd:ID`
`xsd:IDREF`
`xsd:IDREFS`
`xsd:ENTITY`
`xsd:ENTITIES xsd:language`
`xsd:Name`
`xsd:NCName`
`xsd:NMTOKEN`
`xsd:NMTOKENS`
`xsd:normalizedString`
`xsd:token`
	Object	`Object.toString()` wird aufgerufen.

`xsd:time`	Date Number String	Date UTC-Zugriffsmethoden werden verwendet. „Number“ wird zum Festlegen von „Date.time“ verwendet. Bei String wird angenommen, dass er entsprechend vorformatiert und kodiert wird.
`xsi:nil`	null	Wenn die entsprechende Elementdefinition für das XML-Schema „minOccurs > 0“ aufweist, wird der Wert `null` mithilfe von „xsi:nil“ kodiert; andernfalls wird das Element ganz weggelassen.

Die folgende Tabelle zeigt die Zuordnung von ActionScript 3.0-Typen zu SOAP-kodierten Typen.

SOAPENC-Typ	Unterstützte ActionScript 3.0-Typen	Hinweis
soapenc:Array	Array mx.collections.IList	SOAP-kodierte Arrays werden als Spezialfälle behandelt und werden nur bei RPC-kodierten Webservices unterstützt.
soapenc:base64	flash.utils.ByteArray	Genauso kodiert wie `xsd:base64Binary`.
soapenc:*	Object	Jeder andere SOAP-kodierte Typ wird verarbeitet, als wäre er im auf dem `localName` des QName des Typs basierenden XSD-Namespace.

Dekodieren des XML-Schemas und von SOAP in ActionScript 3.0

Die folgende Tabelle zeigt die Dekodierungszuordnungen von integrierten Typen des XML-Schemas zu ActionScript 3.0-Typen.

XML-Schematyp	Dekodierte ActionScript 3.0-Typen	Hinweis
`xsd:anyType` `xsd:anySimpleType`	String Boolean Number	Wenn Inhalt leer ist -> `xsd:string`. Wenn Inhalt in „Number“ konvertiert wurde und Wert „NaN“ ist; oder wenn Inhalt mit „0" oder „-0" beginnt oder wenn Inhalt mit „E" endet: wenn Inhalt „true" oder „false" ist -> `xsd:boolean` ansonsten -> `xsd:string`. Ansonsten ist Inhalt eine gültige Zahl und somit -> `xsd:double`.
`xsd:base64Binary`	`flash.utils.ByteArray`	`mx.utils.Base64Decoder` wird verwendet.
`xsd:boolean`	Boolean	Wenn Inhalt „true" oder „1" ist, dann `true`, ansonsten `false`.
`xsd:date`	Date	Wenn keine Zeitzoneninformationen vorhanden sind, wird von lokaler Zeit ausgegangen.
`xsd:dateTime`	Date	Wenn keine Zeitzoneninformationen vorhanden sind, wird von lokaler Zeit ausgegangen.
`xsd:decimal`	Number	Inhalt wird über `Number(content)` erstellt und ist somit auf den Bereich von „Number“ begrenzt.
`xsd:double`	Number	Inhalt wird über `Number(content)` erstellt und ist somit auf den Bereich von „Number“ begrenzt.
`xsd:duration`	String	Inhalt wird mit ausgeblendetem Whitespace zurückgegeben.
`xsd:float`	Number	Inhalt wird über `Number(content)` konvertiert und ist somit auf den Bereich von „Number“ begrenzt.
`xsd:gDay`	uint	Inhalt wird über `uint(content)` konvertiert.
`xsd:gMonth`	uint	Inhalt wird über `uint(content)` konvertiert.
`xsd:gMonthDay`	String	Inhalt wird mit ausgeblendetem Whitespace zurückgegeben.
`xsd:gYear`	uint	Inhalt wird über `uint(content)` konvertiert.
`xsd:gYearMonth`	String	Inhalt wird mit ausgeblendetem Whitespace zurückgegeben.
`xsd:hexBinary`	`flash.utils.ByteArray`	`mx.utils.HexDecoder` wird verwendet.

XML-Schematyp	Dekodierte ActionScript 3.0-Typen	Hinweis

`xsd:integer`
`und Derivate:`
`xsd:byte`
`xsd:int`
`xsd:long`
`xsd:negativeInteger`
`xsd:nonNegativeInteger`
`xsd:nonPositiveInteger`
`xsd:positiveInteger`
`xsd:short`
`xsd:unsignedByte`
`xsd:unsignedInt`
`xsd:unsignedLong`
`xsd:unsignedShort`
	Number	Inhalt wird über `parseInt()` dekodiert.

`xsd:string`
und Derivate:
`xsd:ID`
`xsd:IDREF`
`xsd:IDREFS`
`xsd:ENTITY`
`xsd:ENTITIES xsd:language`
`xsd:Name`
`xsd:NCName`
`xsd:NMTOKEN`
`xsd:NMTOKENS`
`xsd:normalizedString`
`xsd:token`
	String	Der Quellinhalt wird einfach als String zurückgegeben.

`xsd:time`	Date	Wenn keine Zeitzoneninformationen vorhanden sind, wird von lokaler Zeit ausgegangen.
`xsi:nil`	null	

Die folgende Tabelle zeigt die Dekodierungszuordnungen von SOAP-kodierten Typen zu ActionScript 3.0-Typen.

SOAPENC-Typ	Dekodierter ActionScript-Typ	Hinweis
`soapenc:Array`	Array `mx.collections.ArrayCollection`	SOAP-kodierte Arrays werden als Spezialfälle behandelt. Wenn `makeObjectsBindable` den Wert `true` hat, wird das Ergebnis in eine ArrayCollection eingeschlossen; andernfalls wird ein einfaches Array zurückgegeben.
`soapenc:base64`	`flash.utils.ByteArray`	Genauso dekodiert wie `xsd:base64Binary`.
`soapenc:*`	Object	Jeder andere SOAP-kodierte Typ wird verarbeitet, als wäre er im auf dem `localName` des QName des Typs basierenden XSD-Namespace.

Die folgende Tabelle zeigt die Dekodierungszuordnungen von benutzerdefinierten Datentypen zu ActionScript 3.0-Datentypen.

Benutzerdefinierter Typ	Dekodierter ActionScript 3.0-Typ	Hinweis
Apache Map `http://xml.apache.org/xml-soap:Map`	Object	Die SOAP-Darstellung von `java.util.Map. Keys` muss in Form von Strings erfolgen können.
Apache Rowset `http://xml.apache.org/xml-soap:Rowset`	Array von Objekten	
ColdFusion QueryBean `http://rpc.xml.coldfusion:QueryBean`	Array von Objekten `mx.collections.ArrayCollection` von Objekten	Wenn `makeObjectsBindable` den Wert „true“ hat, wird das resultierende Array in eine ArrayCollection eingeschlossen.

Unterstützung für XML-Schemaelement

Die folgenden Strukturen oder Strukturattribute für das XML-Schema sind in Flex 4 nur teilweise implementiert:

```
<choice>
<all>
<union
```

Die folgenden Strukturen oder Strukturattribute für das XML-Schema werden ignoriert und in Flex 4 nicht unterstützt:

```
<attribute use="required"/>

<element
    substitutionGroup="..."
    unique="..."
    key="..."
    keyref="..."
    field="..."
    selector="..."/>

<simpleType>
    <restriction>
        <minExclusive>
        <minInclusive>
        <maxExclusiv>
        <maxInclusive>
        <totalDigits>
        <fractionDigits>
        <length>
        <minLength>
        <maxLength>
        <enumeration>
        <whiteSpace>
        <pattern>
    </restriction>
</simpleType>

<complexType
    final="..."
    block="..."
    mixed="..."
    abstract="..."/>

<any
processContents="..."/>

<annotation>
```

Anpassen der Typzuordnung des Webservice

Bei der Verarbeitung von Daten aus einem Webserviceaufruf erstellt Flex üblicherweise typenlose anonyme ActionScript-Objekte, die die XML-Struktur im Text der SOAP-Meldung imitieren. Wenn Flex eine Instanz einer spezifischen Klasse erstellen soll, können Sie ein mx.rpc.xml.SchemaTypeRegistry-Objekt verwenden und ein QName-Objekt mit einer entsprechenden ActionScript-Klasse registrieren.

Beispiel: Sie haben die folgende Klassendefinition in der Datei „User.as“:

```
package
{
     public class User
     {
       public function User() {}

       public var firstName:String;
       public var lastName:String;
     }
}
```

Als Nächstes möchten Sie eine `getUser`-Methode für einen Webservice aufrufen, die folgende XML zurückgibt:

```
<tns:getUserResponse xmlns:tns="http://example.uri">
     <tns:firstName>Ivan</tns:firstName>
     <tns:lastName>Petrov</tns:lastName>
</tns:getUserResponse>
```

Um sicherzustellen, dass Sie beim Aufrufen der getUser-Methode eine Instanz der Benutzerklasse statt eines generischen Objekts abrufen, benötigen Sie den folgenden ActionScript-Code innerhalb einer Methode der Anwendung:

```
SchemaTypeRegistry.getInstance().registerClass(new QName("http://example.uri",
"getUserResponse"), User);
```

`SchemaTypeRegistry.getInstance()` ist eine statische Methode, die die Standardinstanz der Typregistrierung zurückgibt. In den meisten Fällen ist dies alles, was Sie benötigen. Dadurch wird jedoch ein gegebener QName mit derselben ActionScript-Klasse über alle Webservicemethoden hinweg in Ihrer Anwendung registriert. Wenn Sie verschiedene Klassen für unterschiedliche Methoden registrieren möchten, benötigen Sie den folgenden Code in einer Methode Ihrer Anwendung:

```
var qn:QName = new QName("http://the.same", "qname");
var typeReg1:SchemaTypeRegistry = new SchemaTypeRegistry();
var typeReg2:SchemaTypeRegistry = new SchemaTypeRegistry();
typeReg1.registerClass(qn, someClass);
myWS.someOperation.decoder.typeRegistry = typeReg1;

typeReg2.registerClass(qn, anotherClass);
myWS.anotherOperation.decoder.typeRegistry = typeReg2;
```

Arbeiten mit der benutzerdefinierten Webserviceserialisierung

Es gibt zwei Ansätze, um vollständige Kontrolle darüber zu übernehmen, wie ActionScript-Objekte in XML serialisiert werden und wie XML-Antwortmeldungen deserialisiert werden. Es wird empfohlen, direkt mit E4X zu arbeiten.

Wenn Sie eine Instanz von XML als den einzigen Parameter für eine Webservicemethode übergeben, wird sie unverändert als untergeordnetes Element des `<SOAP:Body>`-Knotens in der serialisierten Anforderung übergeben. Verwenden Sie diese Strategie, wenn Sie vollständige Kontrolle über die SOAP-Meldung benötigen. Analog dazu können Sie beim Deserialisieren einer Webserviceantwort die `resultFormat`-Eigenschaft der Methode auf e4x festlegen. Dadurch wird ein XMLList-Objekt mit dem untergeordneten Element des `<SOAP:Body>`-Knotens in der Antwortmeldung zurückgegeben. Davon ausgehend können Sie die erforderliche benutzerdefinierte Logik implementieren, um die entsprechenden ActionScript-Objekte zu erstellen.

Der zweite und mühsamere Ansatz besteht darin, Ihre eigenen Implementierungen von „mx.rpc.soap.ISOAPDecoder“ und „mx.rpc.soap.ISOAPEncoder“ bereitzustellen. Beispiel: Wenn Sie die Klasse „MyDecoder“ geschrieben haben, die ISOAPDecoder implementiert, könnte eine Methode der Anwendung Folgendes enthalten:

```
myWS.someOperation.decoder = new MyDecoder();
```

Beim Aufrufen von `someOperation` ruft Flex die `decodeResponse()`-Methode der MyDecoder-Klasse auf. Ab diesem Punkt liegt es an der benutzerdefinierten Implementierung, die vollständige SOAP-Meldung zu verarbeiten und die erwarteten ActionScript-Objekte zu erzeugen.

Arbeiten mit RemoteObject-Komponenten

Sie können mit einer Flex-RemoteObject-Komponente die Methoden für eine ColdFusion-Komponente oder Java-Klasse aufrufen.

Sie können auch RemoteObject-Komponenten mit PHP- und .NET-Objekten zusammen mit Drittanbieter-Software (beispielsweise die Open Source-Projekte AMFPHP und SabreAMF) sowie Midnight Coders WebORB verwenden. Weitere Informationen finden Sie auf folgenden Websites:

- Zend Framework http://framework.zend.com/
- AMFPHP http://amfphp.sourceforge.net/
- SabreAMF http://www.osflash.org/sabreamf
- Midnight Coders WebORB http://www.themidnightcoders.com/

RemoteObject-Komponenten verwenden zum Senden und Empfangen von Daten das AMF-Protokoll, dagegen verwenden WebService- und HTTPService-Komponenten das HTTP-Protokoll. AMF ist wesentlich schneller als HTTP. Die serverseitige Kodierung und Konfiguration sind in der Regel jedoch komplexer.

Flash Builder for PHP ist ein in Zusammenarbeit mit Zend Technologies erstelltes Entwicklungstool mit integrierter Kopie von Zend Studio. Weitere Informationen finden Sie auf der Adobe-Website.

Wie auch bei HTTPService- und WebService-Komponenten können Sie eine RemoteObject-Komponente zum Anzeigen der Ergebnisse einer Datenbankabfrage in einer Anwendung verwenden. Mithilfe der Komponente können Sie auch Daten in die Datenbank einfügen sowie Datenbankdaten aktualisieren oder löschen. Wenn das Abfrageergebnis an die Anwendung zurückgegeben wurde, können Sie es in einem oder mehreren Steuerelementen der Benutzeroberfläche anzeigen.

API-Referenzinformationen über die RemoteObject-Komponente finden Sie unter mx.rpc.remoting.mxml.RemoteObject.

RemoteObject-Beispielanwendung

MXML-Code

Die Anwendung ruft im folgenden Beispiel mithilfe einer RemoteObject-Komponente eine ColdFusion-Komponente auf. Die ColdFusion-Komponente fragt die MySQL-Datenbanktabelle „users“ ab. Sie gibt das Abfrageergebnis an die Anwendung zurück. Dort wird es an die `dataProvider`-Eigenschaft eines DataGrid-Steuerelements gebunden und im DataGrid-Steuerelement angezeigt. Die Anwendung sendet auch Benutzernamen und E-Mail-Adressen neuer Benutzer an die ColdFusion-Komponente, die das Einfügen in die Benutzerdatenbanktabelle durchführt.

```
<?xml version="1.0" encoding="utf-8"?>
<s:Application xmlns:fx="http://ns.adobe.com/mxml/2009"
        xmlns:s="library://ns.adobe.com/flex/spark"
        xmlns:mx="library://ns.adobe.com/flex/mx" minWidth="955" minHeight="600">
    <fx:Declarations>
        <mx:RemoteObject
            id="userRequest"
            destination="ColdFusion"
            source="flexapp.returnusers">

            <mx:method name="returnRecords" result="returnHandler(event)"
                       fault="mx.controls.Alert.show(event.fault.faultString)"/>
            <mx:method name="insertRecord" result="insertHandler()"
                       fault="mx.controls.Alert.show(event.fault.faultString)"/>
        </mx:RemoteObject>
    </fx:Declarations>

    <fx:Script>
        <![CDATA[
            import mx.rpc.events.ResultEvent;

            private function returnHandler(e:ResultEvent):void
            {
                dgUserRequest.dataProvider = e.result;
            }
            private function insertHandler():void
            {
                userRequest.returnRecords();
            }
            private function clickHandler():void
            {
                userRequest.insertRecord(username.text, emailaddress.text);
            }
        ]]>
    </fx:Script>

    <mx:Form x="22" y="10" width="300">
        <mx:FormItem>
            <s:Label text="Username" />
            <s:TextInput id="username"/>
        </mx:FormItem>
        <mx:FormItem>
            <s:Label text="Email Address" />
            <s:TextInput id="emailaddress"/>
        </mx:FormItem>
        <s:Button label="Submit" click="clickHandler()"/>
    </mx:Form>

    <mx:DataGrid id="dgUserRequest" x="22" y="200">
        <mx:columns>
            <mx:DataGridColumn headerText="User ID" dataField="userid"/>
            <mx:DataGridColumn headerText="User Name" dataField="username"/>
        </mx:columns>
    </mx:DataGrid>
</s:Application>
```

In dieser Anwendung wird die `destination`-Eigenschaft der RemoteObject-Komponente auf `Coldfusion` und die `source`-Eigenschaft auf den vollständig qualifizierten Namen der ColdFusion-Komponente festgelegt.

Dagegen geben Sie beim Arbeiten mit ADEP Data Services oder BlazeDS einen vollständig qualifizierten Klassennamen in der `source`-Eigenschaft eines Remotedienstziels in einer Konfigurationsdatei an, bei der es sich standardmäßig um die remoting-config.xml-Datei handelt. Sie geben den Namen des Ziels in der `destination`-Eigenschaft der RemoteObject-Komponente an. Die Zielklasse muss auch einen Konstruktor ohne Argumente aufweisen. Sie können beim Arbeiten mit ColdFusion auf diese Weise optional ein Ziel konfigurieren, anstatt die `source`-Eigenschaft für die RemoteObject-Komponente zu verwenden.

ColdFusion-Komponente

Die Anwendung ruft die folgende ColdFusion-Komponente auf. Dieser ColdFusion-Code führt Einfüge- und Abfragemethoden der SQL-Datenbank durch und gibt Abfrageergebnisse an die Anwendung zurück. Die ColdFusion-Seite verwendet das `cfquery`-Tag, um Daten in die Datenbank einzufügen und die Datenbank abzufragen, sowie das `cfreturn`-Tag, um die Abfrageergebnisse als ColdFusion-Abfrageobjekt zu formatieren.

```
<cfcomponent name="returnusers">
    <cffunction name="returnRecords" access="remote" returnType="query">

        <cfquery name="alluserinfo" datasource="flexcf">
                SELECT userid, username, emailaddress FROM users
        </cfquery>
        <cfreturn alluserinfo>
    </cffunction>
    <cffunction name="insertRecord" access="remote" returnType="void">

        <cfargument name="username" required="true" type="string">
        <cfargument name="emailaddress" required="true" type="string">
        <cfquery name="addempinfo" datasource="flexcf">
            INSERT INTO users (username, emailaddress) VALUES (
            <cfqueryparam value="#arguments.username#" cfsqltype="CF_SQL_VARCHAR"
maxlength="255">,
            <cfqueryparam value="#arguments.emailaddress#" cfsqltype="CF_SQL_VARCHAR"
maxlength="255"> )
        </cfquery>
        <cfreturn>
    </cffunction>
</cfcomponent>
```

Aufrufen von RemoteObject-Komponenten in ActionScript

Durch Aufrufen der `useRemoteObject()()`-Methode im folgenden ActionScript-Beispiel wird der Dienst deklariert, das Ziel festgelegt, werden result-Ereignis- und fault-Ereignis-Listener eingerichtet und die `getList()`-Methode des Dienstes aufgerufen.

```
<?xml version="1.0"?>
<!-- fds\rpc\ROInAS.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:Script>
        <![CDATA[
            import mx.controls.Alert;
            import mx.rpc.remoting.RemoteObject;
            import mx.rpc.events.ResultEvent;
            import mx.rpc.events.FaultEvent;

            [Bindable]
            public var empList:Object;
            public var employeeRO:RemoteObject;

            public function useRemoteObject(intArg:int, strArg:String):void {
                employeeRO = new RemoteObject();
                employeeRO.destination = "SalaryManager";
                employeeRO.getList.addEventListener("result", getListResultHandler);
                employeeRO.addEventListener("fault", faultHandler);
                employeeRO.getList(deptComboBox.selectedItem.data);
            }

            public function getListResultHandler(event:ResultEvent):void {
                 // Do something
                empList=event.result;
            }

            public function faultHandler (event:FaultEvent):void {
             // Deal with event.fault.faultString, etc.
                Alert.show(event.fault.faultString, 'Error');
            }
        ]]>
    </mx:Script>
    <mx:ComboBox id="deptComboBox"/>
</mx:Application>
```

Java-Objekten im Quellpfad aufrufen

Mithilfe der RemoteObject-Komponente können Sie auf stateless- und stateful-Java-Objekte zugreifen, die sich in ADEP Data Services, BlazeDS oder im Quellpfad der ColdFusion-Webanwendung befinden. Sie können eigenständige Klassendateien im WEB-INF/classes-Verzeichnis der Webanwendung platzieren, um sie dem Quellpfad hinzuzufügen. Sie können in Java Archive (JAR) enthaltene Dateien im WEB-INF/lib-Verzeichnis der Webanwendung platzieren, um sie dem Quellpfad hinzuzufügen. Sie geben den vollständig qualifizierten Klassennamen in der `source`-Eigenschaft eines Remotedienstziels in der ADEP Data Services-, BlazeDS- oder ColdFusion-Datei „services-config.xml“ oder einer Datei an, die ihn/sie als Referenz einschließt, beispielsweise die remoting-config.xml-Datei. Die Klasse muss auch einen Konstruktor ohne Argumente aufweisen. Für ColdFusion können Sie optional die `destination`-Eigenschaft der RemoteObject-Komponente auf `Coldfusion` und die `source`-Eigenschaft auf den vollständig qualifizierten Namen einer ColdFusion-Komponente oder Java-Klasse festlegen.

Wenn Sie ein Remotedienstziel für den Zugriff auf stateless-Objekte (Anforderungsbereich) konfigurieren, erstellt Flex ein unterschiedliches Objekt für jeden Methodenaufruf, anstatt Methoden für dasselbe Objekt aufzurufen. Sie können für den Bereich eines Objekts den Anforderungsbereich (Standardwert), den Anwendungsbereich oder den Sitzungsbereich festlegen. Objekte im Anwendungsbereich stehen der Webanwendung zur Verfügung, die das Objekt enthält. Objekte im Sitzungsbereich stehen der gesamten Clientsitzung zur Verfügung.

Wenn Sie ein Remoteobjektziel für den Zugriff auf stateful-Objekte konfigurieren, erstellt Flex das Objekt einmal auf dem Server und behält den Status zwischen Methodenaufrufen bei. Wenn das Speichern des Objekts im Anwendungs- oder Sitzungsbereich Speicherprobleme verursacht, verwenden Sie den Anforderungsbereich.

EJBs und anderen Objekten in JNDI aufrufen

Sie können Enterprise JavaBeans (EJBs) und andere in JNDI (Java Naming and Directory Interface) gespeicherte Objekte aufrufen, indem Sie Methoden für ein Ziel aufrufen, das eine facade-Dienstklasse ist, die ein Objekt in JNDI sucht und seine Methoden aufruft.

Sie können mithilfe von stateless- oder stateful-Objekten die Methoden von Enterprise JavaBeans und anderen Objekten aufrufen, die JNDI verwenden. Für ein EJB können Sie eine facade-Dienstklasse aufrufen, die das EJB-Objekt von JNDI zurückgibt und eine Methode für das EJB aufruft.

In Ihrer Java-Klasse verwenden Sie das Java-Standardkodierungsmuster, in dem Sie einen Anfangskontext erstellen und eine JNDI-Suche durchführen. Für ein EJB verwenden Sie auch Standardkodierungsmuster, in denen Ihre Klasse Methoden enthält, die die `create()`-Methode des EJB-Home-Objekts und die sich daraus ergebenden EJB-Unternehmensmethoden aufrufen.

Im folgenden Beispiel wird die Methode `getHelloData()` für ein facade-Klassenziel verwendet:

```
<mx:RemoteObject id="Hello" destination="roDest">
    <mx:method name="getHelloData"/>
</mx:RemoteObject>
```

Auf der Java-Seite könnte die `getHelloData()`-Methode alles beinhalten, was zum Aufrufen einer Unternehmensmethode für ein EJB erforderlich ist. Die Java-Methode im folgenden Beispiel führt die folgenden Aktionen durch:

- Erstellt neuen Anfangskontext zum Aufrufen der EJB
- Führt eine JNDI-Suche durch, die ein EJB-Home-Objekt abruft
- Ruft die `create()`-Methode des EJB-Home-Objekts auf
- Ruft die `sayHello()`-Methode des EJB auf

```
...
public void getHelloData() {
    try{
        InitialContext ctx = new InitialContext();
        Object obj = ctx.lookup("/Hello");
        HelloHome ejbHome = (HelloHome)
        PortableRemoteObject.narrow(obj, HelloHome.class);
        HelloObject ejbObject = ejbHome.create();
        String message = ejbObject.sayHello();
        }
    catch (Exception e);
    }
...
```

Reservierte Methodennamen

Die Flex-Remotebibliothek nutzt die folgenden Methodennamen; verwenden Sie diese nicht als eigene Methodennamen:

```
addHeader()
addProperty()
deleteHeader()
hasOwnProperty()
isPropertyEnumerable()
isPrototypeOf()
registerClass()
toLocaleString()
toString()
unwatch()
valueOf()
watch()
```

Daher sollten Sie Methodennamen nicht mit einem Unterstrich (_) beginnen.

RemoteObject-Methoden (Operationen) können aufgerufen werden, indem sie nach einer Dienstvariablen benannt werden. Namenskonflikte können jedoch auftreten, wenn ein Methodenname mit einer definierten Methode für den Dienst übereinstimmt. Mithilfe der folgenden Methode in ActionScript für eine RemoteObject-Komponente können Sie die Methode des gegebenen Namens zurückgeben:

```
public function getOperation(name:String):Operation
```

Serialisieren zwischen ActionScript und Java

ADEP Data Services und BlazeDS serialisieren Daten zwischen ActionScript(AMF 3)- und Java- und ColdFusion-Datentypen in beide Richtungen. Informationen zu den ColdFusion-Datentypen finden Sie in der ColdFusion-Dokumentation.

Konvertieren von ActionScript-Daten in Java-Daten

Wenn Methodenparameter Daten von einer Anwendung an ein Java-Objekt senden, werden die Daten automatisch von einem ActionScript- in einen Java-Datentyp konvertiert. Wenn ADEP Data Services oder BlazeDS nach einer geeigneten Methode für das Java-Objekt suchen, verwenden sie weitere, weniger strenge Konvertierungen, um eine Übereinstimmung zu finden.

Einfache Datentypen des Clients (beispielsweise Boolean- und String-Werte) stimmen im Allgemeinen exakt mit einer Remote-API überein. Bei der Suche nach einer geeigneten Methode für ein Java-Objekt versucht Flex jedoch einige einfache Konvertierungen.

Ein ActionScript-Array kann Einträge auf zwei Arten indexieren. Bei einem *Strict-Array* sind alle Indizes Zahlen. Bei einem *assoziativen Array* basiert mindestens ein Index auf einem String. Es ist wichtig zu wissen, welchen Arraytyp Sie an den Server senden, weil dabei der Datentyp der Parameter, die zum Aufrufen einer Methode für ein Java-Objekt verwendet werden, geändert wird. Bei einem *dichten Array* folgen alle numerischen Indizes ohne Lücke aufeinander, beginnend mit 0 (null). Bei einem *spärlich besetzten Array* gibt es Lücken zwischen den numerischen Indizes; das Array wird wie ein Objekt behandelt und aus den numerischen Indizes werden Eigenschaften, die in ein java.util.Map-Objekt deserialisiert werden, um das Senden vieler Nulleinträge zu vermeiden.

Die folgende Tabelle listet die unterstützten Konvertierungen für einfache Datentypen von ActionScript (AMF 3) in Java auf.

ActionScript-Typ (AMF 3)	Deserialisierung in Java	Unterstützte Java-Typbindung
Array (dicht)	java.util.List	java.util.Collection, *Object*[ ] (systemeigenes Array) Wenn es sich bei dem Typ um eine Schnittstelle handelt, wird er den folgenden Schnittstellenimplementierungen zugeordnet: • List wird zu ArrayList • SortedSet wird zu TreeSet • Set wird zu HashSet • Collection wird zu ArrayList Eine neue Instanz einer benutzerdefinierten Collection-Implementierung ist an diesen Typ gebunden.
Array (sehr dicht)	java.util.Map	java.util.Map
Boolean String von `true` oder `false`	java.lang.Boolean	Boolean, boolean, String
flash.utils.ByteArray	byte []	
flash.utils.IExternalizable	java.io.Externalizable	
Date	java.util.Date (formatiert für Coordinated Universal Time (UTC))	java.util.Date, java.util.Calendar, java.sql.Timestamp, java.sql.Time, java.sql.Date
int/uint	java.lang.Integer	java.lang.Double, java.lang.Long, java.lang.Float, java.lang.Integer, java.lang.Short, java.lang.Byte, java.math.BigDecimal, java.math.BigInteger, String, primitive Typen von double, long, float, int, short, byte
null	null	Primitive Typen
Number	java.lang.Double	java.lang.Double, java.lang.Long, java.lang.Float, java.lang.Integer, java.lang.Short, java.lang.Byte, java.math.BigDecimal, java.math.BigInteger, String, 0 (null) Wenn `null` gesendet wird, primitive Typen von „double“, „long“, „float“, „int“, „short“, „byte“
Object (generisch)	java.util.Map	Wenn eine Map-Schnittstelle angegeben ist, wird eine java.util.HashMap für java.util.Map und eine neue java.util.TreeMap für java.util.SortedMap erstellt.
String	java.lang.String	java.lang.String, java.lang.Boolean, java.lang.Number, java.math.BigInteger, java.math.BigDecimal, char[], alle primitiven number-Typen
typed Object	typed Object Bei Verwendung des `[RemoteClass]`-Metadaten-Tags, das den Remoteklassennamen angibt. Bean-Typ muss einen public-Konstruktor ohne Argumente haben.	typed Object

ActionScript-Typ (AMF 3)	Deserialisierung in Java	Unterstützte Java-Typbindung
undefined (nicht definiert)	null	`null` für „object“, Standardwerte für primitive Typen
XML	org.w3c.dom.Document	org.w3c.dom.Document
XMLDocument (legacy-XML-Typ)	org.w3c.dom.Document	org.w3c.dom.Document Sie können Unterstützung für legacy-XML für den XMLDocument-Typ für jeden in der services-config.xml-Datei definierten Kanal aktivieren. Diese Einstellung ist nur für das Senden von Daten vom Server zurück zum Client wichtig. Sie steuert, wie org.w3c.dom.Document-Instanzen an ActionScript gesendet werden. Weitere Informationen finden Sie unter Konfigurieren der AMF-Serialisierung für einen Kanal.

Primitive Werte können in Java nicht auf `null` gesetzt werden. Wenn Boolean- und Number-Werte vom Client an ein Java-Objekt übergeben werden, interpretiert Flex `null`-Werte als Standardwerte für primitive Typen; beispielsweise 0 für „double“, „float“, „long“, „int“, „short“, „byte“, „\u0000“ für „char“ und `false` für „Boolean“. Nur primitive Java-Typen rufen Standardwerte ab.

ADEP Data Services und BlazeDS verarbeiten java.lang.Throwable-Objekte wie jedes andere typisierte Objekt. Sie werden mit Regeln verarbeitet, die nach public-Feldern und bean-Eigenschaften suchen, und typisierte Objekte werden an den Client zurückgegeben. Die Regeln entsprechen normalen Bean-Regeln, mit Ausnahme davon, dass sie nach Gettern für schreibgeschützte Eigenschaften suchen. Dadurch erhalten Sie weitere Informationen von einer Java-Ausnahme. Wenn Sie legacy-Verhalten für Throwable-Objekte benötigen, können Sie die `legacy-throwable`-Eigenschaft für einen Kanal auf `true` festlegen; weitere Informationen finden Sie unter Konfigurieren der AMF-Serialisierung für einen Kanal.

Sie können Strict-Arrays als Parameter an Methoden übergeben, die eine Implementierung der java.util.Collection oder der systemeigenen Java-Array-APIs erwarten.

Eine Java-Collection kann eine beliebige Zahl von Objekttypen enthalten. Dagegen erfordert ein Java-Array, dass Einträge denselben Typ aufweisen (beispielsweise java.lang.Object[ ] und int[ ]).

ADEP Data Services und BlazeDS konvertieren auch ActionScript-Strict-Arrays in entsprechende Implementierungen für allgemeine Collection-API-Schnittstellen. Beispiel: Wenn ein ActionScript-Strict-Array an die Java-Objektmethode `public void addProducts(java.util.Set products)` gesendet wird, konvertieren ADEP Data Services ES und BlazeDS, bevor es als Parameter übergeben wird, in eine java.util.HashSet-Instanz, da HashSet eine geeignete Implementierung der java.util.Set-Schnittstelle ist. Analog dazu übergeben ADEP Data Services und BlazeDS eine Instanz von java.util.TreeSet an Parameter, die mit der java.util.SortedSet-Schnittstelle typisiert wurden.

ADEP Data Services und BlazeDS übergeben eine Instanz von java.util.ArrayList an Parameter, die mit der java.util.List-Schnittstelle und jeder anderen Schnittstelle, die java.util.Collection erweitert, typisiert wurden. Anschließend werden diese Typen als mx.collections.ArrayCollection-Instanzen an den Client zurückgesendet. Wenn normale ActionScript-Arrays an den Client zurückgesendet werden sollen, müssen Sie beim `legacy-collection`-Element im `serialization`-Abschnitt für Eigenschaften von „channel-definition“ den Wert `true` festlegen. Weitere Informationen finden Sie unter Konfigurieren der AMF-Serialisierung für einen Kanal.

Explizites Zuordnen von ActionScript- und Java-Objekten

Bei Java-Objekten, die ADEP Data Services und BlazeDS nicht implizit verarbeiten, werden in public-bean-Eigenschaften mit get/set-Methoden gefundene Werte und public-Variablen als Eigenschaften für ein „Object“ an den Client gesendet. Private Eigenschaften, Konstanten, statische Eigenschaften, schreibgeschützte Eigenschaften usw. werden nicht serialisiert. Bei ActionScript-Objekten werden mit den Zugriffsmethoden get/set definierte public-Eigenschaften und public-Variablen an den Server gesendet.

ADEP Data Services und BlazeDS verwenden die Java-Standardklasse und java.beans.Introspector, um Eigenschaftendeskriptoren für eine JavaBean-Klasse abzurufen. Es verwendet auch Reflection, um public-Felder für eine Klasse zu sammeln. Es verwendet vorzugsweise bean-Eigenschaften statt Felder. Die Namen für Java- und ActionScript-Eigenschaften sollten übereinstimmen. Der systemeigene Flash Player-Code legt fest, wie ActionScript-Klassen auf dem Client introspektiert werden.

In der ActionScript-Klasse erstellen Sie mit dem `[RemoteClass(alias=" ")]`-Metadaten-Tag ein ActionScript-Objekt, das direkt dem Java-Objekt zugeordnet wird. Die ActionScript-Klasse, in die Daten konvertiert werden, muss verwendet oder in der MXML-Datei referenziert werden, damit sie mit der SWF-Datei verknüpft werden kann und zur Laufzeit verfügbar ist. Eine gute Möglichkeit dazu besteht darin, das Ergebnisobjekt dem folgenden Beispiel entsprechend zu konvertieren:

```
var result:MyClass = MyClass(event.result);
```

Die Klasse selbst sollte stark typisierte Verweise enthalten, damit ihre Abhängigkeiten auch verknüpft sind.

Die folgenden Beispiele zeigen den Quellcode für eine ActionScript-Klasse, die das `[RemoteClass(alias=" ")]`-Metadaten-Tag verwendet:

```
package samples.contact {
    [Bindable]
    [RemoteClass(alias="samples.contact.Contact")]
    public class Contact {
        public var contactId:int;

        public var firstName:String;

        public var lastName:String;

        public var address:String;

        public var city:String;

        public var state:String;

        public var zip:String;
    }
}
```

Sie können das `[RemoteClass]`-Metadaten-Tag ohne Alias verwenden, wenn Sie keine Zuordnung zu einem Java-Objekt auf dem Server vornehmen, aber Ihre Objektart vom Server zurück senden. Ihr ActionScript-Objekt wird mit einem speziellen Map-Objekt serialisiert, wenn es an den Server gesendet wird, aber das vom Server zu den Clients zurückgesendete Objekt ist Ihr ursprünglicher ActionScript-Typ.

Um zu verhindern, dass eine spezifische Eigenschaft von einer ActionScript-Klasse an den Server gesendet wird, verwenden Sie das `[Transient]`-Metadaten-Tag über der Deklaration dieser Eigenschaft in der ActionScript-Klasse.

Konvertieren von Java-Daten in ActionScript-Daten

Ein von einer Java-Methode zurückgegebenes Objekt wird von Java in ActionScript konvertiert. ADEP Data Services und BlazeDS verarbeiten auch innerhalb von Objekten gefundene Objekte. ADEP Data Services verarbeitet implizit die Java-Datentypen in der folgenden Tabelle.

Java-Typ	**ActionScript-Typ (AMF 3)**
java.lang.String	String
java.lang.Boolean, boolean	Boolean
java.lang.Integer, int	int Wenn der Wert „< 0xF0000000 || value > 0x0FFFFFFF“ ist, wird der Wert aufgrund der AMF-Kodierungsanforderungen auf „Number“ hochgestuft.
java.lang.Short, short	int Wenn i „< 0xF0000000 || i > 0x0FFFFFFF“ ist, wird der Wert auf „Number“ hochgestuft.
java.lang.Byte, byte[]	int Wenn i „< 0xF0000000 || i > 0x0FFFFFFF“ ist, wird der Wert auf „Number“ hochgestuft.
java.lang.Byte[]	flash.utils.ByteArray
java.lang.Double, double	Number
java.lang.Long, long	Number
java.lang.Float, float	Number
java.lang.Character, char	String
java.lang.Character[], char[]	String
java. math.BigInteger	String
java.math.BigDecimal	String
java.util.Calendar	Date Datumsangaben werden in der Zeitzone für Coordinated Universal Time (UTC) gesendet. Clients und Server müssen die Zeit dementsprechend für Zeitzonen anpassen.
java.util.Date	Date Datumsangaben werden in der UTC-Zeitzone gesendet. Clients und Server müssen die Zeit dementsprechend für Zeitzonen anpassen.
java.util.Collection (beispielsweise java.util.ArrayList)	mx.collections.ArrayCollection
java.lang.Object[]	Array
java.util.Map	Object (untyped). Beispielsweise wird ein java.util.Map[] in ein Array (von Objekten) konvertiert.
java.util.Dictionary	Object (untyped)
org.w3c.dom.Document	XML object
null	null
java.lang.Object (andere als zuvor aufgelistete Typen)	Typed Object Objekte werden mithilfe von JavaBean-Introspektionsregeln serialisiert und enthalten auch public-Felder. static-, transient- oder nonpublic-Felder sowie Bean-Eigenschaften, die „nonpublic“ oder „static“ sind, werden ausgeschlossen.

Konfigurieren der AMF-Serialisierung für einen Kanal

Sie können die in früheren Versionen von Flex verwendete legacy-AMF-Serialisierung unterstützen und andere Serialisierungseigenschaften in Kanaldefinitionen in der services-config.xml-Datei konfigurieren.

Die folgende Tabelle beschreibt die Eigenschaften, die Sie im `<serialization>`-Element einer Kanaldefinition festlegen können:

Eigenschaft	**Beschreibung**
`<ignore-property-errors>true</ignore-property-errors>`	Der Standardwert lautet `true`. Legt fest, ob der Endpunkt einen Fehler ausgeben soll, sobald ein eingehendes Clientobjekt unerwartete Eigenschaften aufweist, die für das Serverobjekt nicht festgelegt werden können.
`<log-property-errors>false</log-property-errors>`	Der Standardwert lautet `false`. Im Fall von `true` werden unerwartete Eigenschaftenfehler protokolliert.
`<legacy-collection>false</legacy-collection>`	Der Standardwert lautet `false`. Im Fall von `true` werden Instanzen von java.util.Collection als ActionScript-Arrays zurückgegeben. Im Fall von `false` werden Instanzen von java.util.Collection als mx.collections.ArrayCollection zurückgegeben.
`<legacy-map>false</legacy-map>`	Der Standardwert lautet `false`. Im Fall von `true` werden java.util.Map-Instanzen als ECMA-Array oder assoziatives Array statt als anonymes Objekt serialisiert.
`<legacy-xml>false</legacy-xml>`	Der Standardwert lautet `false`. Im Falle von `true` werden org.w3c.dom.Document-Instanzen als flash.xml.XMLDocument-Instanzen statt als eigentliche (E4X-fähige) XML-Instanzen serialisiert.
`<legacy-throwable>false</legacy-throwable>`	Der Standardwert lautet `false`. Im Fall von `true` werden java.lang.Throwable-Instanzen als AMF-Statusinfoobjekte serialisiert (statt normaler Bean-Serialisierung einschließlich schreibgeschützter Eigenschaften).

Eigenschaft	Beschreibung
`<type-marshaller>className</type-marshaller>`	Gibt eine Implementierung von flex.messaging.io.TypeMarshaller an, wodurch ein Objekt in eine Instanz einer gewünschten Klasse übersetzt wird. Wird beim Aufrufen einer Java-Methode oder beim Füllen einer Java-Instanz verwendet, wenn der Typ des Eingabeobjekts der Deserialisierung (ein anonymes ActionScript-Objekt wird beispielsweise immer als java.util.HashMap deserialisiert) nicht mit der Ziel-API übereinstimmt (beispielsweise java.util.SortedMap). Dadurch kann der Typ in den gewünschten Typ umgewandelt werden.
`<restore-references>false</restore-references>`	Der Standardwert lautet `false`. Ein erweiterter Switch, mit dem Deserialisierer Objektverweise überwachen, sobald eine Typübersetzung erfolgen soll. Wenn beispielsweise ein anonymes Objekt für eine Eigenschaft des Typs java.util.SortedMap gesendet wurde, wird das Objekt zuerst wie üblich in java.util.Map deserialisiert und anschließend in eine geeignete Implementierung von SortedMap übersetzt (beispielsweise java.util.TreeMap). Wenn andere Objekte auf dasselbe anonyme Objekt in einem Objektdiagramm verweisen, stellt diese Einstellung diese Verweise wieder her, anstatt überall SortedMap-Implementierungen zu erstellen. Beachten Sie, dass sich bei größeren Datenmengen die Leistung beträchtlich verlangsamen kann, wenn für diese Eigenschaft `true` festgelegt wird.
`<instantiate-types>true</instantiate-types>`	Der Standardwert lautet `true`. Erweiterter Switch, der, falls auf `false` gesetzt, verhindert, dass der Deserialisierer Instanzen stark typisierter Objekte erstellt und stattdessen die Typinformationen beibehält und die raw-Eigenschaften in einer Map-Implementierung (insbesondere flex.messaging.io.ASObject) deserialisiert. Beachten Sie, dass alle Klassen unter demflex*-Paket immer instanziiert werden.

Verwenden einer benutzerdefinierten Serialisierung

Wenn das Standardverfahren zum Serialisieren und Deserialisieren von Daten zwischen ActionScript auf dem Client und Java auf dem Server nicht Ihre Anforderungen erfüllt, können Sie Ihr eigenes Serialisierungsschema erstellen. Sie implementieren die ActionScript-basierte flash.utils.IExternalizable-Schnittstelle auf dem Client und die entsprechende Java-basierte java.io.Externalizable-Schnittstelle auf dem Server.

Ein häufiger Grund für das Verwenden einer benutzerdefinierten Serialisierung besteht darin, dass das Übergeben aller Eigenschaften der client- oder serverseitigen Darstellung eines Objekts auf Netzwerkebene vermieden werden kann. Wenn Sie eine benutzerdefinierte Serialisierung implementieren, können Sie Ihre Klassen so kodieren, dass spezifische Eigenschaften, die nur für Client oder Server gelten, nicht übermittelt werden. Wenn Sie das Standardschema zur Serialisierung verwenden, werden alle public-Eigenschaften zwischen dem Client und dem Server übermittelt.

Clientseitig wird die Identität einer Klasse, die die flash.utils.IExternalizable-Schnittstelle implementiert, in den Serialisierungsstream geschrieben. Die Klasse serialisiert und rekonstruiert den Status ihrer Instanzen. Die Klasse implementiert die `writeExternal()`- und `readExternal()`-Methoden der IExternalizable-Schnittstelle, um Inhalt und Format des Serialisierungsstreams zu steuern, nicht jedoch den Klassennamen oder -typ eines Objekts und seine übergeordneten Typen. Diese Methoden ersetzen das systemeigene AMF-Serialisierungsverhalten. Diese Methoden müssen zu ihren jeweiligen Remote-Entsprechungen symmetrisch sein, um den Status der Klasse speichern zu können.

Serverseitig führt eine Java-Klasse, die die java.io.Externalizable-Schnittstelle implementiert, Funktionen aus, die analog einer ActionScript-Klasse sind, die die flash.utils.IExternalizable-Schnittstelle implementiert.

Hinweis: *Wenn die präzise Serialisierung nach Referenzen erforderlich ist, sollten Sie keine Typen verwenden, die die IExternalizable-Schnittstelle mit HTTPChannel implementieren. Wenn Sie dies tun, gehen Verweise zwischen wiederkehrenden Objekten verloren und scheinen am Endpunkt geklont zu sein.*

Das folgende Beispiel zeigt den vollständigen Quellcode für die Client-Version (ActionScript) einer Product-Klasse, die einer Java-basierten Product-Klasse auf dem Server zugeordnet wird. Das Client-Product implementiert die IExternalizable-Schnittstelle, das Server-Product implementiert die Externalizable-Schnittstelle.

```
// Product.as
package samples.externalizable {

import flash.utils.IExternalizable;
import flash.utils.IDataInput;
import flash.utils.IDataOutput;

[RemoteClass(alias="samples.externalizable.Product")]
public class Product implements IExternalizable {
    public function Product(name:String=null) {
        this.name = name;
    }

    public var id:int;
    public var name:String;
    public var properties:Object;
    public var price:Number;

    public function readExternal(input:IDataInput):void {
        name = input.readObject() as String;
        properties = input.readObject();
        price = input.readFloat();
    }

    public function writeExternal(output:IDataOutput):void {
        output.writeObject(name);
        output.writeObject(properties);
        output.writeFloat(price);
    }
}
}
```

Das Client-Product verwendet zwei Arten von Serialisierung. Es verwendet die Standardserialisierung, die mit der java.io.Externalizable-Schnittstelle und der AMF 3-Serialisierung kompatibel ist. Das folgende Beispiel zeigt die `writeExternal()`-Methode des Client-Product, das beide Serialisierungstypen verwendet:

```
public function writeExternal(output:IDataOutput):void {
    output.writeObject(name);
    output.writeObject(properties);
    output.writeFloat(price);
}
```

Wie das folgende Beispiel zeigt, ist die `writeExternal()`-Methode des Server-Product beinahe mit der Clientversion dieser Methode identisch:

```
public void writeExternal(ObjectOutput out) throws IOException {
    out.writeObject(name);
    out.writeObject(properties);
    out.writeFloat(price);
}
```

In der `writeExternal()`-Methode im Client-Product ist die `flash.utils.IDataOutput.writeFloat()`-Methode ein Beispiel für Standardserialisierungsmethoden, die die Spezifikationen für die Java-Methoden `java.io.DataInput.readFloat()` für das Arbeiten mit primitiven Typen erfüllen. Diese Methode sendet die `price`-Eigenschaft, die eine Gleitkommazahl ist, an das Server-Product.

Das Beispiel für die AMF 3-Serialisierung der `writeExternal()`-Methode im Client-Product ist der Aufruf der `flash.utils.IDataOutput.writeObject()`-Methode, die dem Aufruf der `java.io.ObjectInput.readObject()`-Methode in der `readExternal()`-Methode des Server-Product zugeordnet wird. Die `flash.utils.IDataOutput.writeObject()`-Methode sendet die `properties`-Eigenschaft, die ein „Object“ ist, und die name-Eigenschaft, die ein „String“ ist, an das Server-Product. Dies ist möglich, weil der AMFChannel-Endpunkt eine Implementierung der java.io.ObjectInput-Schnittstelle hat, die erwartet, dass von der `writeObject()`-Methode gesendete Daten als AMF 3 formatiert werden.

Wenn dagegen die `readObject()`-Methode in der `readExternal()`-Methode des Server-Product aufgerufen wird, verwendet sie AMF 3-Deserialisierung; deshalb wird angenommen, dass die ActionScript-Version des `properties`-Werts vom Typ „Map“ und `name` vom Typ „String“ ist.

Das folgende Beispiel zeigt die vollständige Quelle der Server-Product-Klasse:

```
// Product.java
package samples.externalizable;

import java.io.Externalizable;
import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;
import java.util.Map;

/**
* This Externalizable class requires that clients sending and
* receiving instances of this type adhere to the data format
* required for serialization.
*/
public class Product implements Externalizable {
    private String inventoryId;
    public String name;
    public Map properties;
    public float price;

    public Product()
    {
    }

        /**
        * Local identity used to track third-party inventory. This property is
        * not sent to the client because it is server specific.
        * The identity must start with an 'X'.
        */
        public String getInventoryId() {
            return inventoryId;
        }
```

```
        public void setInventoryId(String inventoryId) {
            if (inventoryId != null && inventoryId.startsWith("X"))
            {
                this.inventoryId = inventoryId;
            }
            else
            {
                throw new IllegalArgumentException("3rd party product
                inventory identities must start with 'X'");
            }
        }

        /**
         * Deserializes the client state of an instance of ThirdPartyProxy
         * by reading in String for the name, a Map of properties
         * for the description, and
         * a floating point integer (single precision) for the price.
         */
        public void readExternal(ObjectInput in) throws IOException,
            ClassNotFoundException {
            // Read in the server properties from the client representation.
            name = (String)in.readObject();
            properties = (Map)in.readObject();
            price = in.readFloat();
            setInventoryId(lookupInventoryId(name, price));
        }

        /**
         * Serializes the server state of an instance of ThirdPartyProxy
         * by sending a String for the name, a Map of properties
         * String for the description, and a floating point
         * integer (single precision) for the price. Notice that the inventory
         * identifier is not sent to external clients.
         */
        public void writeExternal(ObjectOutput out) throws IOException {
            // Write out the client properties from the server representation.
            out.writeObject(name);
            out.writeObject(properties);
            out.writeFloat(price);
        }

        private static String lookupInventoryId(String name, float price) {
            String inventoryId = "X" + name + Math.rint(price);
            return inventoryId;
        }
}
```

Das folgende Beispiel zeigt die `readExternal()`-Methode des Server-Product:

```
public void readExternal(ObjectInput in) throws IOException,
    ClassNotFoundException {
    // Read in the server properties from the client representation.
    name = (String)in.readObject();
    properties = (Map)in.readObject();
    price = in.readFloat();
    setInventoryId(lookupInventoryId(name, price));
    }
```

Die Methode `writeExternal()` des Client-Product sendet die `id`-Eigenschaft während der Serialisierung nicht an den Server, weil sie für die Serverversion des Product-Objekts nicht nützlich ist. Analog dazu sendet die `writeExternal()`-Methode des Server-Product die `inventoryId`-Eigenschaft nicht an den Client, weil es sich um eine serverspezifische Eigenschaft handelt.

Beachten Sie, dass die Namen der Product-Eigenschaften während der Serialisierung weder in die eine noch in die andere Richtung gesendet werden. Da der Status der Klasse festgelegt und zu handhaben ist, werden die Eigenschaften in einer definierten Reihenfolge ohne ihre Namen gesendet und von der `readExternal()`-Methode in der entsprechenden Reihenfolge gelesen.

Explizite Übergabe und Bindung von Parametern

Es gibt zwei unterschiedliche Möglichkeiten, HTTPService-, WebService- und RemoteObject-Komponenten aufzurufen: explizite Übergabe und Bindung von Parametern. Bei der expliziten Übergabe von Parametern stellen Sie Eingaben für einen Dienst in Form von Parametern für eine ActionScript-Funktion bereit. Diese Art des Aufrufens eines Dienstes ähnelt stark dem Aufrufen von Methoden in Java. Sie können Flex-Datenvalidatoren nicht automatisch zusammen mit der expliziten Übergabe von Parametern verwenden.

Mit der Bindung von Parametern können Sie Daten von Steuerelementen oder Modellen der Benutzeroberfläche kopieren, um Parameter anzufordern. Die Bindung von Parametern steht nur für Datenzugriffskomponenten zur Verfügung, die Sie in MXML deklarieren. Sie können Validatoren auf Parameterwerte anwenden, bevor Sie Anforderungen an Dienste übergeben. Weitere Informationen über Datenbindung und Datenmodelle finden Sie unter Data binding und Storing data. Weitere Informationen zur Datenüberprüfung finden Sie unter Validating Data.

Wenn Sie die Parameterbindung verwenden, deklarieren Sie Parameter-Tags für eine RemoteObject-Methode, die in einem `<mx:arguments>`-Tag unter einem `<mx:method>`-Tag, HTTPService-Parameter-Tags, die in einem `<mx:request>`-Tag oder Parameter-Tags für eine WebService-Methode, die in einem `<mx:request>`-Tag unter einem `<mx:operation>`-Tag verschachtelt sind. Mit der `send()`-Methode senden Sie die Anforderung.

Explizite Parameterübergabe mit RemoteObject- und WebService-Komponenten

Die Art und Weise, wie die explizite Parameterübergabe mit RemoteObject- und WebService-Komponenten erfolgt, ist sehr ähnlich. Das folgende Beispiel zeigt MXML-Code zum Deklarieren einer RemoteObject-Komponente und Aufrufen eines Dienstes unter Verwendung expliziter Parameterübergabe im click-Ereignis-Listener des Button-Steuerelements Ein ComboBox-Steuerelement stellt Daten für den Dienst zur Verfügung. Einfache Ereignis-Listener verarbeiten die result- und fault-Ereignisse auf Dienstebene.

```
<?xml version="1.0"?>
<!-- fds\rpc\RPCParamPassing.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
    verticalGap="10">
    <mx:Script>
        <![CDATA[
            import mx.controls.Alert;
            [Bindable]
            public var empList:Object;
        ]]>
    </mx:Script>

    <mx:RemoteObject
        id="employeeRO"
        destination="SalaryManager"
        result="empList=event.result"
        fault="Alert.show(event.fault.faultString, 'Error');"/>

    <mx:ComboBox id="dept" width="150">
        <mx:dataProvider>
            <mx:ArrayCollection>
                <mx:source>
                    <mx:Object label="Engineering" data="ENG"/>
                    <mx:Object label="Product Management" data="PM"/>
                    <mx:Object label="Marketing" data="MKT"/>
                </mx:source>
            </mx:ArrayCollection>
        </mx:dataProvider>
    </mx:ComboBox>

   <mx:Button label="Get Employee List" click="employeeRO.getList(dept.selectedItem.data);"/>
</mx:Application>
```

Explizite Parameterübergabe mit HTTPService-Komponenten

Explizite Parameterübergabe mit HTTPService-Komponenten unterscheidet sich von der mit RemoteObject- und WebService-Komponenten. Sie rufen einen Dienst immer mit der `send()`-Methode einer HTTPService-Komponente auf. Diese unterscheidet sich von RemoteObject- und WebService-Komponenten, bei denen Sie Methoden aufrufen, die clientseitige Versionen der Methoden oder Operationen des RPC-Dienstes sind.

Bei expliziter Parameterübergabe können Sie ein Objekt angeben, das Name/Wert-Paare als Parameter für die `send()`-Methode enthält Ein `send()`-Methodenparameter muss ein einfacher Basistyp sein; Sie können keine komplexen verschachtelten Objekte verwenden, weil es keine generische Möglichkeit gibt, diese in Name/Wert-Paare zu konvertieren.

Wenn Sie keinen Parameter für die `send()`-Methode angeben, verwendet die HTTPService-Komponente beliebige in einem `<mx:request>`-Tag angegebene Abfrageparameter.

Die folgenden Beispiele zeigen zwei Möglichkeiten, einen HTTP-Dienst unter Verwendung der `send()`-Methode mit einem Parameter aufzurufen. Das zweite Beispiel zeigt auch, wie die `cancel()`-Methode zum Abbrechen eines HTTP-Dienstaufrufs aufgerufen werden kann.

```
<?xml version="1.0"?>
<!-- fds\rpc\RPCSend.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">

    <mx:Script>
        <![CDATA[
            public function callService():void {
                // Cancel all previous pending calls.
                myService.cancel();

                var params:Object = new Object();
                params.param1 = 'val1';
                myService.send(params);
            }
        ]]>
    </mx:Script>

    <mx:HTTPService
        id="myService"
        destination="Dest"
        useProxy="true"/>
<!-- HTTP service call with a send() method that takes a variable as its parameter. The value
of the variable is an Object. -->
    <mx:Button click="myService.send({param1: 'val1'});"/>

<!-- HTTP service call with an object as a send() method parameter that provides query
parameters. -->
    <mx:Button click="callService();"/>
</mx:Application>
```

Parameterbindung mit RemoteObject-Komponenten

Bei der Parameterbindung mit RemoteObject-Komponenten deklarieren Sie immer Methoden in einem `<mx:method>`-Tag der RemoteObject-Komponente.

Ein `<mx:method>`-Tag kann ein `<mx:arguments>`-Tag enthalten, das untergeordnete Tags für die Methodenparameter enthält. Die name-Eigenschaft eines `<mx:method>`-Tags muss mit einem der Methodennamen des Dienstes übereinstimmen. Die Reihenfolge der argument-Tags muss mit der Reihenfolge der Methodenparameter des Dienstes übereinstimmen. Sie können argument-Tags entsprechend benennen, damit sie mit den tatsächlichen Namen der entsprechenden Methodenparameter so exakt wie möglich übereinstimmen, aber dies ist nicht erforderlich.

Hinweis: *Wenn argument-Tags innerhalb eines `<mx:arguments>`-Tags denselben Namen aufweisen, schlagen Dienstaufrufe fehl, wenn die Remotemethode kein Array als einzige Eingabequelle erwartet. Es erfolgt dazu keine Warnung, wenn die Anwendung kompiliert wird.*

Sie können Daten an Methodenparameter einer RemoteObject-Komponente binden. Sie referenzieren die Tag-Namen der Parameter für Datenbindung und -überprüfung.

Das folgende Beispiel zeigt eine Methode mit zwei Parametern, die an Texteigenschaften von TextInput-Steuerelementen gebunden sind. Ein PhoneNumberValidator-Validator wird `arg1` zugewiesen. Dies ist der Name des ersten argument-Tags.

```
<?xml version="1.0"?>
<!-- fds\rpc\ROParamBind1.mxml. -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:RemoteObject
        id="ro"
        destination="roDest">

        <mx:method name="setData">
            <mx:arguments>
                <arg1>{text1.text}</arg1>
                <arg2>{text2.text}</arg2>
            </mx:arguments>
        </mx:method>
    </mx:RemoteObject>
    <mx:TextInput id="text1"/>
    <mx:TextInput id="text2"/>

    <mx:PhoneNumberValidator source="{ro.setData.arguments}" property="arg1"/>
</mx:Application>
```

Flex sendet die Werte für das argument-Tag in der Reihenfolge an die Methode, die die MXML-Tags angeben.

Das folgende Beispiel verwendet Parameterbindung in einem `<mx:method>`-Tag der RemoteObject-Komponente, um die Daten eines ausgewählten ComboBox-Elements an die `employeeRO.getList`-Methode zu binden, wenn Benutzer auf ein Button-Steuerelement klicken. Bei der Parameterbindung rufen Sie einen Dienst unter Verwendung der `send()`-Methode ohne Parameter auf.

```
<?xml version="1.0"?>
<!-- fds\rpc\ROParamBind2.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" verticalGap="10">
    <mx:Script>
        <![CDATA[
            import mx.controls.Alert;
                import mx.utils.ArrayUtil;
        ]]>
    </mx:Script>
    <mx:RemoteObject
        id="employeeRO"
        destination="roDest"
        showBusyCursor="true"
        fault="Alert.show(event.fault.faultString, 'Error');">
        <mx:method name="getList">
            <mx:arguments>
                <deptId>{dept.selectedItem.data}</deptId>
            </mx:arguments>
        </mx:method>
    </mx:RemoteObject>
    <mx:ArrayCollection id="employeeAC"
        source="{ArrayUtil.toArray(employeeRO.getList.lastResult)}"/>

    <mx:HBox>
        <mx:Label text="Select a department:"/>
        <mx:ComboBox id="dept" width="150">

            <mx:dataProvider>
```

```
                <mx:ArrayCollection>
                    <mx:source>
                        <mx:Object label="Engineering" data="ENG"/>
                        <mx:Object label="Product Management" data="PM"/>
                        <mx:Object label="Marketing" data="MKT"/>
                    </mx:source>
                </mx:ArrayCollection>
            </mx:dataProvider>
        </mx:ComboBox>
            <mx:Button label="Get Employee List"
                click="employeeRO.getList.send()"/>
    </mx:HBox>
    <mx:DataGrid dataProvider="{employeeAC}" width="100%">
        <mx:columns>
            <mx:DataGridColumn dataField="name" headerText="Name"/>
            <mx:DataGridColumn dataField="phone" headerText="Phone"/>
            <mx:DataGridColumn dataField="email" headerText="Email"/>
        </mx:columns>
    </mx:DataGrid>
</mx:Application>
```

Wenn Sie nicht sicher sind, ob das Ergebnis eines Dienstaufrufs ein Array oder ein einzelnes Objekt enthält, können Sie die `toArray()`-Methode der mx.utils.ArrayUtil-Klasse verwenden, um sie in ein Array zu konvertieren (siehe dieses Beispiel). Wenn Sie die `toArray()`-Methode an ein einzelnes Objekt übergeben, wird ein Array mit diesem Objekt als einziges Arrayelement zurückgegeben. Wenn Sie ein Array an die Methode übergeben, wird dasselbe Array zurückgegeben. Informationen zum Arbeiten mit ArrayCollection-Objekten finden Sie unter Data providers and collections.

Parameterbindung mit HTTPService-Komponenten

Wenn ein HTTP-Dienst Abfrageparameter übernimmt, können Sie sie als untergeordnete Tags eines `<mx:request>`-Tags deklarieren. Die Namen der Tags müssen mit den Namen der Abfrageparameter übereinstimmen, die der Dienst erwartet.

Das folgende Beispiel verwendet Parameterbindung in einem `<mx:request>`-Tag der HTTPService-Komponente, um die Daten eines ausgewählten ComboBox-Elements an die `employeeSrv`-Anforderung zu binden, wenn Benutzer auf ein Button-Steuerelement klicken. Bei der Parameterbindung rufen Sie einen Dienst unter Verwendung der `send()`-Methode ohne Parameter auf. Dieses Beispiel zeigt eine `url`-Eigenschaft für die HTTPService-Komponente, jedoch ist die Art und Weise des Dienstaufrufs identisch, egal ob Sie direkt eine Verbindung zum Dienst herstellen oder über ein Ziel gehen.

```
<?xml version="1.0"?>
<!-- fds\rpc\HttpServiceParamBind.mxml. Compiles -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" verticalGap="20">
    <mx:Script>
        <![CDATA[
            import mx.utils.ArrayUtil;
        ]]>
    </mx:Script>

    <mx:HTTPService
        id="employeeSrv"
        url="employees.jsp">
        <mx:request>
            <deptId>{dept.selectedItem.data}</deptId>
        </mx:request>
    </mx:HTTPService>
    <mx:ArrayCollection
        id="employeeAC"
        source=
            "{ArrayUtil.toArray(employeeSrv.lastResult.employees.employee)}"/>
    <mx:HBox>
        <mx:Label text="Select a department:"/>
        <mx:ComboBox id="dept" width="150">
            <mx:dataProvider>
                <mx:ArrayCollection>
                    <mx:source>
                        <mx:Object label="Engineering" data="ENG"/>
                        <mx:Object label="Product Management" data="PM"/>
                        <mx:Object label="Marketing" data="MKT"/>
                    </mx:source>
                </mx:ArrayCollection>
            </mx:dataProvider>
        </mx:ComboBox>
        <mx:Button label="Get Employee List" click="employeeSrv.send();"/>
    </mx:HBox>
    <mx:DataGrid dataProvider="{employeeAC}"
        width="100%">
        <mx:columns>
            <mx:DataGridColumn dataField="name" headerText="Name"/>
            <mx:DataGridColumn dataField="phone" headerText="Phone"/>
            <mx:DataGridColumn dataField="email" headerText="Email"/>
        </mx:columns>
    </mx:DataGrid>
</mx:Application>
```

Wenn Sie nicht sicher sind, ob das Ergebnis eines Dienstaufrufs ein Array oder ein einzelnes Objekt enthält, können Sie die `toArray()`-Methode der mx.utils.ArrayUtil-Klasse verwenden, um sie in ein Array zu konvertieren (siehe vorheriges Beispiel). Wenn Sie die `toArray()`-Methode an ein einzelnes Objekt übergeben, wird ein Array mit diesem Objekt als einziges Arrayelement zurückgegeben. Wenn Sie ein Array an die Methode übergeben, wird dasselbe Array zurückgegeben. Informationen zum Arbeiten mit ArrayCollection-Objekten finden Sie unter Data providers and collections.

Parameterbindung mit WebService-Komponenten

Bei der Parameterbindung mit einer WebService-Komponente deklarieren Sie Methoden immer in den `<mx:operation>`-Tags der WebService-Komponente. Ein `<mx:operation>`-Tag kann ein `<mx:request>`-Tag enthalten, das die XML-Knoten enthält, die die Methode erwartet. Die name-Eigenschaft eines `<mx:operation>`-Tags muss mit einem der Webservicemethodennamen übereinstimmen.

Sie können Daten an Parameter der Webervicemethoden binden. Sie referenzieren die Tag-Namen der Parameter für Datenbindung und -überprüfung.

Das folgende Beispiel verwendet Parameterbindung in einem `<mx:operation>`-Tag der WebService-Komponente, um die Daten eines ausgewählten ComboBox-Elements an die `employeeWS.getList`-Methode zu binden, wenn Benutzer auf ein Button-Steuerelement klicken. Das `<deptId>`-Tag entspricht direkt dem `deptId`-Parameter der getList-Methode. Bei der Parameterbindung rufen Sie einen Dienst unter Verwendung der `send()`-Methode ohne Parameter auf. Dieses Beispiel zeigt eine destination-Eigenschaft für die WebService-Komponente, jedoch ist die Art und Weise des Dienstaufrufs identisch, egal ob Sie direkt eine Verbindung zum Dienst herstellen oder über ein Ziel gehen.

```
<?xml version="1.0"?>
<!-- fds\rpc\WebServiceParamBind.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" verticalGap="10">
    <mx:Script>
        <![CDATA[
            import mx.utils.ArrayUtil;
            import mx.controls.Alert;
        ]]>
    </mx:Script>

    <mx:WebService
        id="employeeWS"
        destination="wsDest"
        showBusyCursor="true"
        fault="Alert.show(event.fault.faultString)">
        <mx:operation name="getList">
            <mx:request>
                <deptId>{dept.selectedItem.data}</deptId>
            </mx:request>
        </mx:operation>
    </mx:WebService>
    <mx:ArrayCollection
        id="employeeAC"
        source="{ArrayUtil.toArray(employeeWS.getList.lastResult)}"/>
    <mx:HBox>
        <mx:Label text="Select a department:"/>
        <mx:ComboBox id="dept" width="150">
            <mx:dataProvider>
```

```
                <mx:ArrayCollection>
                    <mx:source>
                        <mx:Object label="Engineering" data="ENG"/>
                        <mx:Object label="Product Management" data="PM"/>
                        <mx:Object label="Marketing" data="MKT"/>
                    </mx:source>
                </mx:ArrayCollection>
            </mx:dataProvider>
        </mx:ComboBox>
            <mx:Button label="Get Employee List"
                click="employeeWS.getList.send()"/>
    </mx:HBox>
    <mx:DataGrid dataProvider="{employeeAC}" width="100%">
        <mx:columns>
                <mx:DataGridColumn dataField="name" headerText="Name"/>
                <mx:DataGridColumn dataField="phone" headerText="Phone"/>
                <mx:DataGridColumn dataField=" to email" headerText="Email"/>
        </mx:columns>
    </mx:DataGrid>
</mx:Application>
```

Sie können auch manuell einen ganzen SOAP-Anforderungstext in XML mit allen korrekten in einem `<mx:request>`-Tag definierten Namespace-Informationen angeben. Dazu legen Sie den Wert des `format`-Attributs des `<mx:request>`-Tags auf `xml` fest (siehe folgendes Beispiel):

```
<?xml version="1.0"?>
<!-- fds\rpc\WebServiceSOAPRequest.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" verticalGap="10">
    <mx:WebService id="ws" wsdl="http://api.google.com/GoogleSearch.wsdl"
        useProxy="true">
        <mx:operation name="doGoogleSearch" resultFormat="xml">
            <mx:request format="xml">
                <ns1:doGoogleSearch xmlns:ns1="urn:GoogleSearch"
                    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
                    <key xsi:type="xsd:string">XYZ123</key>
                    <q xsi:type="xsd:string">Balloons</q>
                    <start xsi:type="xsd:int">0</start>
                    <maxResults xsi:type="xsd:int">10</maxResults>
                    <filter xsi:type="xsd:boolean">true</filter>
                    <restrict xsi:type="xsd:string"/>
                    <safeSearch xsi:type="xsd:boolean">false</safeSearch>
                    <lr xsi:type="xsd:string" />
                    <ie xsi:type="xsd:string">latin1</ie>
                    <oe xsi:type="xsd:string">latin1</oe>
                </ns1:doGoogleSearch>
            </mx:request>
        </mx:operation>
    </mx:WebService>
</mx:Application>
```

Verarbeiten von Dienstergebnissen

Nachdem eine RPC-Komponente einen Dienst aufruft, werden die vom Dienst zurückgegebenen Daten in einem lastResult-Objekt platziert. Der `resultFormat`-Eigenschaftswert der HTTPService-Komponenten und der WebService-Komponentenmethoden lautet standardmäßig `object` und die zurückgegebenen Daten werden als einfache Baumstruktur der ActionScript-Objekte dargestellt. Flex interpretiert die XML-Daten, die ein Webservice oder HTTP-Dienst zurückgibt, um Basistypen wie „String“, „Number“, „Boolean“ und „Date“ entsprechend darzustellen. Um mit stark typisierten Objekten zu arbeiten, füllen Sie diese Objekte mithilfe der von Flex erstellten Objektbaumstruktur.

WebService- und HTTPService-Komponenten geben beide anonyme „Objects“ und Arrays zurück, die komplexe Typen sind. Wenn `makeObjectsBindable` den Wert `true` hat (die Standardeinstellung), werden „Objects“ in mx.utils.ObjectProxy-Instanzen und Arrays in mx.collections.ArrayCollection-Instanzen eingeschlossen.

Hinweis: *Bei ColdFusion wird die Groß- und Kleinschreibung nicht berücksichtigt, und intern werden alle Daten mit Großbuchstaben geschrieben. Berücksichtigen Sie dies beim Verarbeiten von ColdFusion-Webservices.*

Verarbeiten von result- und fault-Ereignissen

Wenn ein Dienstaufruf abgeschlossen ist, sendet die RemoteObject- oder WebService-Methode oder die HTTPService-Komponente ein result- oder ein fault-Ereignis. Ein *result*-Ereignis gibt an, dass das Ergebnis verfügbar ist. Ein *fault*-Ereignis gibt an, dass ein Fehler aufgetreten ist. Das result-Ereignis fungiert als Auslöser für das Aktualisieren von Eigenschaften, die an lastResult gebunden sind. Sie können fault- und result-Ereignisse explizit verarbeiten, indem Sie Ereignis-Listener zu RemoteObject- oder WebService-Methoden hinzufügen. Bei HTTPService-Komponenten legen Sie result- und fault-Ereignis-Listener für die Komponente selbst fest, da HTTPService-Komponenten nicht über mehrere Methoden verfügen.

Wenn Sie bei RemoteObject- oder WebService-Methoden keine Ereignis- Listener für result- bzw. fault-Ereignisse festlegen, werden die Ereignisse an die Komponentenebene weitergegeben. Sie können result- und fault-Ereignis-Listener auf Komponentenebene festlegen.

Im folgenden MXML-Beispiel legen die `result`- und `fault`-Ereignisse einer WebService-Methode Ereignis-Listener fest. Das `fault`-Ereignis der WebService-Komponente legt ebenfalls einen Ereignis-Listener fest:

```
<?xml version="1.0"?>
<!-- fds\rpc\RPCResultFaultMXML.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:Script>
        <![CDATA[
            import mx.rpc.soap.SOAPFault;
            import mx.rpc.events.ResultEvent;
            import mx.rpc.events.FaultEvent;
            import mx.controls.Alert;
            public function showErrorDialog(event:FaultEvent):void {
                // Handle operation fault.
                Alert.show(event.fault.faultString, "Error");
            }
            public function defaultFault(event:FaultEvent):void {
                // Handle service fault.
                if (event.fault is SOAPFault) {
                     var fault:SOAPFault=event.fault as SOAPFault;
                     var faultElement:XML=fault.element;
                     // You could use E4X to traverse the raw fault element returned in the
SOAP envelope.
                     // ...
```

```
                }
                Alert.show(event.fault.faultString, "Error");
            }
            public function log(event:ResultEvent):void {
                // Handle result.
            }
        ]]>
    </mx:Script>
    <mx:WebService id="WeatherService" wsdl="http://myserver:8500/flexapp/app1.cfc?wsdl"
        fault="defaultFault(event)">
        <mx:operation name="GetWeather"
            fault="showErrorDialog(event)"
            result="log(event)">
            <mx:request>
                <ZipCode>{myZip.text}</ZipCode>
            </mx:request>
        </mx:operation>
    </mx:WebService>
    <mx:TextInput id="myZip"/>
</mx:Application>
```

Im folgenden ActionScript-Beispiel wird einer WebService-Methode ein result-Ereignis-Listener hinzugefügt. Der WebService-Komponente wird ein fault-Ereignis-Listener hinzugefügt.

```
<?xml version="1.0"?>
<!-- fds\rpc\RPCResultFaultAS.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:Script>
        <![CDATA[
            import mx.rpc.soap.WebService;
            import mx.rpc.soap.SOAPFault;
            import mx.rpc.events.ResultEvent;
            import mx.rpc.events.FaultEvent;

            private var ws:WebService;

            public function useWebService(intArg:int, strArg:String):void {
                ws = new WebService();
                ws.destination = "wsDest";
                ws.echoArgs.addEventListener("result", echoResultHandler);
                ws.addEventListener("fault", faultHandler);
                ws.loadWSDL();
                ws.echoArgs(intArg, strArg);
            }
```

```
            public function echoResultHandler(event:ResultEvent):void {
                var retStr:String = event.result.echoStr;
                var retInt:int = event.result.echoInt;
                //do something
        }

            public function faultHandler(event:FaultEvent):void {
            //deal with event.fault.faultString, etc.
                if (event.fault is SOAPFault) {
                     var fault:SOAPFault=event.fault as SOAPFault;
                     var faultElement:XML=fault.element;
                     // You could use E4X to traverse the raw fault element returned in the
SOAP envelope.
                     // ...
                }
            }
        ]]>
    </mx:Script>
</mx:Application>
```

Sie können auch mithilfe des mx.rpc.events.InvokeEvent-Ereignisses anzeigen, wann eine Datenzugriffskomponente aufgerufen wurde. Das ist nützlich, wenn Methoden in der Warteschlange platziert und zu einem späteren Zeitpunkt aufgerufen werden.

Verarbeiten von Ergebnissen als XML mit dem E4X-Ergebnisformat

Sie können den `resultFormat`-Eigenschaftswert von HTTPService-Komponenten und WebService-Methoden auf e4x festlegen, um eine lastResult-Eigenschaft vom Typ XML zu erstellen. Auf die `lastResult`-Eigenschaft können Sie mithilfe von ECMAScript for XML (E4X)-Ausdrücken zugreifen. Der Stammknoten der XML-Struktur wird beim Verwenden eines `E4X`-XML-Objekts in einem Bindungsausdruck nicht in die Punktnotation aufgenommen. Dies weicht von der Syntax für eine auf „object“ gesetzte `lastResult`-Eigenschaft ab. Beispiel: Wenn die lastResult-Eigenschaft auf `e4x` gesetzt ist, würden Sie `{srv.lastResult.product}` verwenden. Wenn die lastResult-Eigenschaft auf „object“ gesetzt ist, würden Sie `{srv.lastResult.products.product}` verwenden.

Wenn Sie direkt mit XML arbeiten, verwenden Sie vorzugsweise das Ergebnisformat von `e4x`. Sie können aber auch die `resultFormat`-Eigenschaft auf `xml` setzen, um ein `lastResult` -Objekt vom Typ flash.xml.XMLNode zu erstellen, das für das Arbeiten mit XML ein veraltetes Objekt ist. Sie können auch die `resultFormat`-Eigenschaft der HTTPService-Komponenten auf `flashvars` oder `text` festlegen, um Ergebnisse jeweils als ActionScript-Objekte, die Name/Wert-Paare enthalten, oder als raw-Text zu erstellen.

Hinweis: *Wenn Sie für die Dienstergebnisse die E4X-Syntax verwenden möchten, legen Sie die `resultFormat`-Eigenschaft Ihrer HTTPService- oder WebService-Komponente auf `e4x` fest. Der Standardwert ist `object`.*

Wenn Sie die `resultFormat`-Eigenschaft einer HTTPService-Komponente oder einer WebService-Methode auf `e4x` setzen, müssen Sie möglicherweise Namespace-Informationen verarbeiten, die im zurückgegebenen XML enthalten sind. Bei WebService-Komponenten sind Namespace-Informationen im Body-Element des vom Webservice zurückgegebenen SOAP-Envelope enthalten. Das folgende Beispiel zeigt Teile eines SOAP-Body, der Namespace-Informationen enthält. Diese Daten wurden von einem Webservice zurückgegeben, der Börsenkurse abruft. Die Namespace-Informationen sind fett formatiert.

```
...
<soap:Body>
<GetQuoteResponse
xmlns="http://ws.invesbot.com/">
<GetQuoteResult><StockQuote xmlns="">
<Symbol>ADBE</Symbol>
<Company>ADOBE SYSTEMS INC</Company>
<Price>&lt;big&gt;&lt;b&gt;35.90&lt;/b&gt;&lt;/big&gt;</Price>
...
</soap:Body>
...
```

Da in soap:Body Namespace-Informationen enthalten sind, erstellen Sie, wenn Sie die `resultFormat`-Eigenschaft der WebService-Methode auf `e4x` festlegen, ein Namespace-Objekt für den Namespace http://ws.invesbot.com/. Das folgende Beispiel zeigt eine Anwendung, die dies kann:

```
<?xml version="1.0"?>
<!-- fds\rpc\WebServiceE4XResult1.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
    pageTitle="Test" >
    <mx:Script>
        <![CDATA[
            import mx.controls.Alert;
            private namespace invesbot = "http://ws.invesbot.com/";
            use namespace invesbot;
        ]]>
    </mx:Script>
    <mx:WebService
        id="WS"
        destination="stockservice" useProxy="true"
        fault="Alert.show(event.fault.faultString), 'Error'">
        <mx:operation name="GetQuote" resultFormat="e4x">
            <mx:request>
                <symbol>ADBE</symbol>
            </mx:request>
        </mx:operation>
    </mx:WebService>
    <mx:HBox>
        <mx:Button label="Get Quote" click="WS.GetQuote.send()"/>
        <mx:Text
            text="{WS.GetQuote.lastResult.GetQuoteResult.StockQuote.Price}"
        />
    </mx:HBox>
</mx:Application>
```

Optional können Sie eine Variable für einen Namespace erstellen und darauf in einer Bindung an das Dienstergebnis zugreifen (siehe dazu folgendes Beispiel):

```
<?xml version="1.0"?>
<!-- fds\rpc\WebServiceE4XResult2.mxml -->
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" xmlns="*"
    pageTitle="Test" >
    <mx:Script>
        <![CDATA[
        import mx.controls.Alert;
        public var invesbot:Namespace =
            new Namespace("http://ws.invesbot.com/");
        ]]>
    </mx:Script>
    <mx:WebService
        id="WS"
        destination="stockservice" useProxy="true"
        fault="Alert.show(event.fault.faultString), 'Error'">
        <mx:operation name="GetQuote" resultFormat="e4x">
            <mx:request>
                <symbol>ADBE</symbol>
            </mx:request>
        </mx:operation>
    </mx:WebService>
    <mx:HBox>
        <mx:Button label="Get Quote" click="WS.GetQuote.send()"/>
        <mx:Text
          text="{WS.GetQuote.lastResult.invesbot::GetQuoteResult.StockQuote.Price}"
        />
    </mx:HBox>
</mx:Application>
```

Mit der E4X-Syntax greifen Sie auf XML-Elemente und -Attribute zu, die in einem lastResult-Objekt zurückgegeben werden. Die Syntax ist unterschiedlich, je nachdem, ob in XML ein oder mehrere Namespaces deklariert sind.

Kein Namespace

Das folgende Beispiel zeigt, wie der Wert eines Elements oder Attributs abgerufen werden kann, wenn kein Namespace für das Element oder Attribut angegeben wurde:

```
var attributes:XMLList = XML(event.result).Description.value;
```

Der vorherige Code gibt xxx für das folgende XML-Dokument zurück:

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <Description>
        <value>xxx</value>
    </Description>
</RDF>
```

Beliebiger Namespace

Das folgende Beispiel zeigt, wie der Wert eines Elements oder Attributs abgerufen werden kann, wenn ein beliebiger Namespace für das Element oder Attribut angegeben wurde:

```
var attributes:XMLList = XML(event.result).*::Description.*::value;
```

Der vorherige Code gibt xxx für eines der folgenden XML-Dokumente zurück:

XML-Dokument eins:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description>
        <rdf:value>xxx</rdf:value>
    </rdf:Description>
</rdf:RDF>
```

XML-Dokument zwei:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:cm="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <cm:Description>
        <rdf:value>xxx</rdf:value>
    </cm:Description>
</rdf:RDF>
```

Spezifischer Namespace

Das folgende Beispiel zeigt, wie der Wert eines Elements oder Attributs abgerufen werden kann, wenn der deklarierte rdf-Namespace für das Element oder Attribut angegeben wurde:

```
var rdf:Namespace = new Namespace("http://www.w3.org/1999/02/22-rdf-syntax-ns#");
var attributes:XMLList = XML(event.result).rdf::Description.rdf::value;
```

Der vorherige Code gibt `xxx` für das folgende XML-Dokument zurück:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description>
        <rdf:value>xxx</rdf:value>
    </rdf:Description>
</rdf:RDF>
```

Das folgende Beispiel zeigt eine alternative Möglichkeit, um den Wert eines Elements oder Attributs abzurufen, wenn der deklarierte rdf-Namespace für ein Element oder Attribut angegeben wurde:

```
namespace rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#";
use namespace rdf;
var attributes:XMLList = XML(event.result).rdf::Description.rdf::value;
```

Der vorherige Code gibt auch `xxx` für das folgende XML-Dokument zurück:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description>
        <rdf:value>xxx</rdf:value>
    </rdf:Description>
</rdf:RDF>
```

Verarbeiten von Webserviceergebnissen, die „.NET DataSets“ oder „DataTables“ enthalten

Mit dem Microsoft .NET Framework erstellte Webservices können spezielle .NET DataSet- oder DataTable-Objekte an den Client zurückgeben. Ein .NET-Webservice stellt ein einfaches WSDL-Dokument ohne Informationen über den bearbeiteten Datentyp bereit. Wenn der Webservice „DataSet“ oder „DataTable“ zurückgibt, werden Datentypinformationen in ein XML-Schemaelement der SOAP-Meldung eingebettet, wodurch angegeben wird, wie der Rest der Meldung verarbeitet werden sollte. Um die Ergebnisse dieses Webservicetyps optimal zu verarbeiten, legen Sie die `resultFormat`-Eigenschaft einer Flex-WebService-Methode auf `object` fest. Sie können optional die `resultFormat`-Eigenschaft der WebService-Methode auf `e4x` festlegen, aber die XML- und e4x-Formate sind unpraktisch, weil Sie durch die ungewöhnliche Struktur der Antwort navigieren und Problemumgehungen implementieren müssen, wenn Sie Daten beispielsweise an ein DataGrid-Steuerelement binden möchten.

Wenn Sie die `resultFormat`-Eigenschaft einer Flex-WebService-Methode auf `object` festlegen, wird ein von einem .NET-Webservice zurückgegebener DataTable oder DataSet automatisch in ein Objekt mit einer Tables-Eigenschaft konvertiert, das eine Zuordnung eines oder mehrerer DataTable-Objekte enthält. Jedes DataTable-Objekt aus der Tables-Zuordnung enthält zwei Eigenschaften: Columns und Rows. Die Rows-Eigenschaft enthält die Daten. Das event.result-Objekt ruft die folgenden Eigenschaften entsprechend der DataSet- und DataTable-Eigenschaften in .NET ab. Arrays von „DataSets“ oder „DataTables“ haben dieselben hier beschriebenen Strukturen, sind aber in einem Array der obersten Ebene des Ergebnisobjekts verschachtelt.

Eigenschaft	**Beschreibung**
`result.Tables`	Zuordnung von Tabellennamen zu Objekten, die Tabellendaten enthalten.
`result.Tables["someTable"].Columns`	Array von Spaltennamen, angegeben in der im DataSet- oder DataTable-Schema für die Tabelle angegebenen Reihenfolge.
`result.Tables["someTable"].Rows`	Array von Objekten, die die Daten für jede Tabellenzeile darstellen. Beispiel: {columnName1:value, columnName2:value, columnName3:value}.

Die folgende MXML-Anwendung fügt von einem .NET-Webservice zurückgegebene DataTable-Daten in ein DataGrid-Steuerelement ein.

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns="*" xmlns:mx="http://www.adobe.com/2006/mxml" layout="vertical">
    <mx:WebService
        id="nwCL"
        wsdl="http://localhost/data/CustomerList.asmx?wsdl"
        result="onResult(event)"
        fault="onFault(event)" />
    <mx:Button label="Get Single DataTable" click="nwCL.getSingleDataTable()"/>
    <mx:Button label="Get MultiTable DataSet" click="nwCL.getMultiTableDataSet()"/>
    <mx:Panel id="dataPanel" width="100%" height="100%" title="Data Tables"/>

    <mx:Script>
        <![CDATA[
            import mx.controls.Alert;
            import mx.controls.DataGrid;
            import mx.rpc.events.FaultEvent;
            import mx.rpc.events.ResultEvent;

            private function onResult(event:ResultEvent):void {
                // A DataTable or DataSet returned from a .NET webservice is
                // automatically converted to an object with a "Tables" property,
                // which contains a map of one or more dataTables.
                if (event.result.Tables != null)
                {
                    // clean up panel from previous calls.
                    dataPanel.removeAllChildren();

                    for each (var table:Object in event.result.Tables)
                    {
                        displayTable(table);
                    }

                    // Alternatively, if a table's name is known beforehand,
                    // it can be accessed using this syntax:
```

```
                var namedTable:Object = event.result.Tables.Customers;
                //displayTable(namedTable);
            }
        }

        private function displayTable(tbl:Object):void {
            var dg:DataGrid = new DataGrid();
            dataPanel.addChild(dg);
            // Each table object from the "Tables" map contains two properties:
            // "Columns" and "Rows". "Rows" is where the data is, so we can set
            // that as the dataProvider for a DataGrid.
            dg.dataProvider = tbl.Rows;
        }

        private function onFault(event:FaultEvent):void {
            Alert.show(event.fault.toString());
        }
    ]]>
</mx:Script>

</mx:Application>
```

Das folgende Beispiel zeigt die .NET C#-Klasse, die von der Anwendung aufgerufene Back-End-Webserviceimplementierung. Diese Klasse verwendet die Microsoft SQL Server Northwind-Beispieldatenbank

:

```
<%@ WebService Language="C#" Class="CustomerList" %>
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Web.Services.Description;
using System.Data;
using System.Data.SqlClient;
using System;

public class CustomerList : WebService {
    [WebMethod]
    public DataTable getSingleDataTable() {
        string cnStr = "[Your_Database_Connection_String]";
        string query = "SELECT TOP 10 * FROM Customers";
        SqlConnection cn = new SqlConnection(cnStr);
        cn.Open();
        SqlDataAdapter adpt = new SqlDataAdapter(new SqlCommand(query, cn));
        DataTable dt = new DataTable("Customers");

        adpt.Fill(dt);
        return dt;
    }
```

```
    [WebMethod]
    public DataSet getMultiTableDataSet() {
        string cnStr = "[Your_Database_Connection_String]";
        string query1 = "SELECT TOP 10 CustomerID, CompanyName FROM Customers";
        string query2 = "SELECT TOP 10 OrderID, CustomerID, ShipCity,
        ShipCountry FROM Orders";
        SqlConnection cn = new SqlConnection(cnStr);
        cn.Open();

        SqlDataAdapter adpt = new SqlDataAdapter(new SqlCommand(query1, cn));
        DataSet ds = new DataSet("TwoTableDataSet");
        adpt.Fill(ds, "Customers");

        adpt.SelectCommand = new SqlCommand(query2, cn);
        adpt.Fill(ds, "Orders");

        return ds;
    }
}
```