ADOBE® FLEX® 4.6 を使用したデータへのアクセス

法律上の注意

法律上の注意については、http://help.adobe.com/ja_JP/legalnotices/index.html を参照してください。

目次

第 1 章：データサービスへのアクセスの概要

Flex のデータアクセスと他のテクノロジーとの比較

Flex におけるデータソースやデータの操作は、ユーザーインターフェイスに HTML を使用しているアプリケーションとは異なります。

クライアント側の処理とサーバー側の処理

JSP とサーブレット、ASP、PHP、CFML などを使用して作成される HTML テンプレートとは異なり、Flex ではクライアントコードをサーバーコードから分離します。アプリケーションのユーザーインターフェイスは、バイナリの SWF ファイルにコンパイルされて、クライアントに送信されます。

アプリケーションがデータサービスに対して要求を行うとき、SWF ファイルは再コンパイルされず、ページ更新も不要です。リモートサービスはデータのみを返します。Flex は、返されたデータをクライアントアプリケーションのユーザーインターフェイスコンポーネントにバインドします。

例えば、Flex では、ユーザーがアプリケーションの Button コントロールをクリックすると、クライアント側のコードが Web サービスを呼び出します。Web サービスからの結果データは、ページを更新しなくてもバイナリ SWF ファイルに返されます。そのため、結果データはアプリケーションの動的コンテンツとして使用できます。

```
<?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)"/>
```

この Flex の例を、次に示す例と比較してください。次の例では、JSP カスタムタグを使用して Web サービスを呼び出す JSP コードを示します。ユーザーが JSP を要求すると、クライアントではなくサーバー上で Web サービスの要求が行われます。その結果を使用して HTML ページのコンテンツが生成されます。アプリケーションサーバーでは、HTML ページ全体を再生成してからユーザーの Web ブラウザーに返送します。

```
<%@ 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") %>
```

データソースへのアクセス

Flex と他の Web アプリケーションテクノロジーとのもう 1 つの違いは、Flex ではデータソースと直接通信しないことです。リモートサービスに接続して、サーバー側のデータソースとやり取りするには、データアクセスコンポーネントを使用します。

次の例は、データソースに直接アクセスする ColdFusion ページを示しています。

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

Flex で同様の機能を実現するには、HTTPService、Web サービスまたは RemoteObject コンポーネントを使用して、データソースから結果を返すサーバー側のオブジェクトを呼び出します。

イベント、サービスの呼び出しおよびデータバインディング

Flex はイベント駆動型のテクノロジーです。ユーザーのアクションまたはプログラムのイベントによって、サービスへのアクセスをトリガーできます。例えば、ユーザーによるボタンのクリックは、サービスの呼び出しをトリガーするために使用できるユーザーアクションイベントです。プログラムのイベントの例としては、アプリケーションによる DataGrid などのユーザーインターフェイスコンポーネントの作成の完了などがあります。DataGrid の creationComplete イベントを使用すると、リモートサービスを呼び出して、DataGrid に値を設定できます。

Flex では、サービスの呼び出しは非同期に行われます。クライアントアプリケーションは、データが返されるのを待つ必要はありません。非同期でのサービスの呼び出しは、大きいデータセットを取得または更新するときに役立ちます。データが取得または更新されるまでクライアントアプリケーションがブロックされることはありません。

サービスの呼び出しから返されたデータは、サービスの呼び出しと関連付けられた CallResponder に格納されます。その後、ユーザーインターフェイスコンポーネントは、データバインディングを使用して、返されたデータを CallResponder から取得します。

Flex のデータバインディングを使用すると、ユーザーインターフェイスコンポーネントをデータソースで動的に更新できます。例えば、Flex コンポーネントは、text 属性を CallResponder の lastResult 属性と関連付けることができます。CallResponder のデータが変化すると、Flex コンポーネントは自動的に更新されます。

また、Flex では双方向のデータバインディングも実装されています。双方向のデータバインディングでは、Flex コンポーネントまたはデータソースでデータが変更されると、対応するデータソースまたは Flex コンポーネントが自動的に更新されます。双方向のデータバインディングは、フォームコンポーネントまたは Flex データコンポーネントに対するユーザー入力でリモートデータを更新するときに便利です。

関連項目

7 ページの「Flash Builder によるデータ中心型アプリケーションの構築」

Flash Builder でのデータサービスへのアクセス

Flex Builder 3 では、Flex データアクセスコンポーネントを使用して、データサービスへのリモートプロシージャコールを実装します。Flash Builder を使用すると、このプロセスが簡略化されます。

Flash Builder には、次のような機能を持つウィザードやその他のツールが用意されています。

- データサービスへのアクセス
- データサービスから返されるデータの設定
- サービスから返されるデータのページング
- サーバーデータに複数の更新を同期するデータ管理機能
- データサービスにアクセスするためのクライアントコードの生成
- サービスから返されるデータのユーザーインターフェイスコンポーネントへのバインド

サービスにアクセスするための Flash Builder のワークフロー

Flash Builder を使用して、データサービスにアクセスするアプリケーションを作成するときは、次のワークフローを使用します。

1 状況に応じて、先にデータサービスに接続するか、先にユーザーインターフェイスを作成するかを選択します。

リモートサービスに接続します。リモートサービスへの接続から始める場合は、後でユーザーインターフェイスを作成します。

ユーザーインターフェイスを作成します。ユーザーインターフェイスの作成から始める場合は、後でリモートサービスに接続します。

注意： どちらから始めるかは個人的な好みの問題です。例えば、ユーザーインターフェイスのデザインの計画が既にある場合は、ユーザーインターフェイスを最初に作成できます。逆に、最初にデータに接続してから、Flash Builder を利用してアプリケーションのコンポーネントを生成することもできます。

2 データ操作をアプリケーションコンポーネントにバインドします。

3 （オプション）データの取得と更新を管理します。

Flash Builder のツールを使用すると、返されるデータのページングを実装し、データセットの更新を調節できます。

大量のデータレコードを返すときは、通常、ページングを実装し、必要に応じてレコードのセットを取得します。

複数のレコードを更新するアプリケーションの場合は、データ管理機能を実装できます。データ管理機能には、次の機能が含まれます。

- コミット機能。変更された複数のレコードを同時に更新します。
- 取り消し機能。サーバーに書き込まれる前に、変更を取り消します。

- コード生成機能。レコードが追加、削除、または変更されたときにユーザーインターフェイスコンポーネントを自動的に更新するコードを生成します。

4 アプリケーションを実行し、データフローをモニタリングします。

アプリケーションが完成したら、アプリケーションを実行して稼働状況を確認します。Flash Builder のネットワークモニターを使用して、アプリケーションとサービスの間で受け渡されるデータを確認します。ネットワークモニターは、エラーを診断し、パフォーマンスを分析するのに役立ちます。

また、Flash Builder には、強力なデバッグとプロファイリングの環境も用意されています。ネットワークモニターおよび Flash プロファイラーは Flash Builder Premium で使用できます。

関連項目

7 ページの「Flash Builder によるデータ中心型アプリケーションの構築」

Flash Builder でサポートされるサービスの拡張

Flash Builder のウィザードとツールは、次の種類のサービス実装へのアクセスをサポートします。

- PHP サービス
- ColdFusion サービス
- BlazeDS
- LiveCycle Data Services
- HTTP（REST 形式）サービス
- Web サービス（SOAP）
- 静的な XML ファイル

Ruby on Rails などの他の種類のサービスをツールでサポートする必要がある場合は、Flash Builder の実装を拡張できます。Flash Builder Extensibility Reference を参照してください。

データアクセスコンポーネント

データアクセスコンポーネントを使用すると、クライアントアプリケーションからネットワーク上の操作およびサービスを呼び出すことができます。データアクセスコンポーネントでは、リモートプロシージャコールを使用してサーバー環境とやり取りをします。データアクセスコンポーネントには、RemoteObject コンポーネント、HTTPService コンポーネント、WebService コンポーネントの 3 つがあります。

データアクセスコンポーネントは、外部データへのアクセス手段として呼び出しと応答モデルが適切な選択肢となるクライアントアプリケーション向けに設計されています。これらのコンポーネントを使用して、クライアントはリモートサービスに非同期要求を発行できます。リモートサービスでは要求を処理した後、データをアプリケーションに返します。

データアクセスコンポーネントはリモートサービスを呼び出します。その後、サービスからの応答データを、ActionScript オブジェクトまたはサービスから返されたその他のフォーマットに格納します。クライアントアプリケーションではデータアクセスコンポーネントを使用して次の 3 種類のサービスを操作できます。

- リモートオブジェクトサービス（RemoteObject）
- SOAP ベースの Web サービス（WebService）
- REST ベースの Web サービスを含む HTTP サービス（HTTPService）

Adobe® Flash® Builder™ には、データアクセスコンポーネントの実装をサービスラッパーにラップするためのウィザードとツールが用意されています。サービスラッパーはデータアクセスコンポーネントの機能をカプセル化します。そのため、低レベルの実装の大部分はユーザーからは見えません。これにより、サービスの実装とサービスにアクセスするクライアントアプリケーションの構築に集中することができます。Flash Builder を使用してデータサービスにアクセスする方法について詳しくは、7 ページの「Flash Builder によるデータ中心型アプリケーションの構築」を参照してください。

サービスへのアクセスの提供

Adobe Flash Player のデフォルトでは、アプリケーションの読み込みに使用されたものと完全には一致しないホストへのアクセスがブロックされます。要求をプロキシする際に、LiveCycle Data Services または BlazeDS などのサーバー側アプリケーションを使用しない場合は、アプリケーションをホストするサーバー上で HTTP サービスまたは Web サービスを実行するか、HTTP サービスまたは Web サービスをホストするリモートサーバーで crossdomain.xml ファイルを定義する必要があります。crossdomain.xml ファイルは、サーバーのデータとドキュメントをどのドメインの SWF ファイルが利用できるかを示す XML ファイルで、特定のドメインのみを指定することも、すべてのドメインを指定することもできます。crossdomain.xml ファイルは、アプリケーションの接続先になるサーバーの Web ルートに存在している必要があります。

HTTPService コンポーネント

HTTPService コンポーネントを使用すると、HTTP の GET 要求または POST 要求を送信でき、HTTP 応答のデータをクライアントアプリケーションに含めることもできます。Flex を使用してデスクトップアプリケーションを構築している場合（Adobe AIR® で実行している場合）は、HTTP PUT と DELETE がサポートされます。

LiveCycle Data Services または BlazeDS を使用している場合は、HTTPProxyService を使用して、追加の HTTP メソッドを使用できます。HTTPProxyService を使用すると、GET、POST、HEAD、OPTIONS、PUT、TRACE、DELETE 要求を送信できます。

HTTP 要求を受け入れて応答を送信する任意の HTTP URI を HTTP サービスとして使用できます。一般に、この種のサービスは REST 形式の Web サービスとも呼ばれます。REST は Representational State Transfer の略で、分散されたハイパーメディアシステムのためのアーキテクチャによる形式です。

HTTPService コンポーネントは、同じ機能を SOAP Web サービスやリモートオブジェクトサービスとしては公開できない場合に役立ちます。例えば、HTTPService コンポーネントを使用すると、Web サービスまたは Remoting Service の宛先として使用できない JavaServer Pages（JSP）、サーブレットおよび ASP ページとやり取りすることができます。

HTTPService オブジェクトの send() メソッドを呼び出すと、指定した URI に対して HTTP 要求が実行され、HTTP 応答が返されます。オプションで、指定した URI に引数を渡すこともできます。

Flash Builder には、HTTP サービスにインタラクティブに接続できるようにするワークフローが用意されています。詳しくは、12 ページの「HTTP サービスへのアクセス」を参照してください。

関連項目

12 ページの「HTTP サービスへのアクセス」

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

WebService コンポーネント

WebService コンポーネントを使用すると、SOAP Web サービスにアクセスできます。SOAP Web サービスはメソッドを備えたソフトウェアモジュールです。Web サービスのメソッドは、一般に操作と呼ばれます。Web サービスのインターフェイスは、Web Services Description Language（WSDL）を使用して定義されます。Web サービスは、様々なプラットフォーム上で実行されているソフトウェアモジュールが相互に通信するための標準規格に準拠しています。Web サービスについて詳しくは、World Wide Web Consortium の Web サイト（www.w3.org/2002/ws/）の、Web サービスのセクションを参照してください。

クライアントアプリケーションは、URL として使用可能な Web Services Description Language（WSDL）ドキュメントでインターフェイスを定義する Web サービスとやり取りできます。WSDL は、Web サービスで認識できるメッセージ、それらのメッセージへの応答の形式、Web サービスがサポートするプロトコル、およびメッセージの送信先を記述するための標準形式です。

Flex では、www.w3.org/TR/wsdl に記載されている WSDL 1.1 がサポートされています。Flex では、RPC エンコードとドキュメントリテラルの両方の Web サービスがサポートされています。

Flex では、SOAP メッセージ形式で HTTP 経由でトランスポートされる Web サービスの要求と結果がサポートされています。SOAP で提供されている XML ベース形式の定義を使用すると、Flex で構築されたアプリケーションなどの Web サービスクライアントと Web サービスの間で構造化および型指定された情報を交換できます。

Web サービスが標準として確立されている環境では、WebService コンポーネントを使用して SOAP 準拠の Web サービスに接続できます。WebService コンポーネントは、エンタープライズ環境内にあるものの、Web アプリケーションのソースパスで使用できるとは限らないオブジェクトにも役立ちます。

Flash Builder には、Web サービスにインタラクティブに接続できるようにするワークフローが用意されています。詳しくは、15 ページの「Web サービスへのアクセス」を参照してください。

RemoteObject コンポーネント

リモートオブジェクトサービスを使用すると、REST 形式のサービスまたは Web サービスの場合とは異なり、XML としてフォーマットすることなく、ネイティブフォーマットでビジネスロジックに直接アクセスできます。これにより、既存のロジックを XML として公開するために必要な時間を節約できます。リモートオブジェクトサービスのもう 1 つの利点は、回線での通信の速度です。データの交換はやはり HTTP または https を介して行われますが、データ自体はバイナリ表現にシリアライズされます。RemoteObject コンポーネントを使用すると、回線を通過するデータが少なくなり、クライアント側でのメモリの使用量が減り、処理時間が短くなります。

サーバー上のデータにアクセスするときは、ColdFusion、PHP、BlazeDS、LiveCycle Data Services でサーバー側の型指定を使用できます。クライアントアプリケーションは、指定されたオブジェクトのメソッドのリモート呼び出しにより、Java オブジェクト、ColdFusion コンポーネント（内部的には Java オブジェクト）または PHP クラスに直接アクセスします。サーバー上のオブジェクトは、それぞれのネイティブデータ型を引数として使用し、その引数でデータベースをクエリし、ネイティブデータ型の値を返します。

サーバー側の型指定が使用できない場合、Flash Builder にはクライアント側の型指定を実装するためのツールがあります。サービスから返されるデータの型を設定および定義すには、Flash Builder を使用します。クライアント側の型指定を使用すると、クライアントアプリケーションからデータベースにクエリを送信して、適切な型のデータを取得できます。サービスによって返されるデータの型が定義されていないすべてのサービスに対し、クライアント側での型指定が必要です。

Flash Builder には、リモートオブジェクトサービスにインタラクティブに接続できるようにするワークフローが用意されています。詳しくは、8 ページの「データサービスへの接続」を参照してください。

第 2 章：Flash Builder によるデータ中心型アプリケーションの構築

Flash Builder ツールを使用すると、データサービスにアクセスするアプリケーションを簡単に作成できます。まず、アプリケーション用の Flex プロジェクトを作成します。次に、データサービスに接続し、サービスからのデータに対するアクセス設定を行い、アプリケーションのユーザーインターフェイスを構築します。場合によっては、ユーザーインターフェイスを作成してからデータサービスにアクセスすることもあります。

データサービスにアクセスするための Flex プロジェクトの作成

Flex は、リモートオブジェクト、HTTP（REST スタイル）サービス、または SOAP Web サービスとしてデータサービスにアクセスします。

リモートオブジェクトを使用すると、次の種類のデータサービスにアクセスできます。

- ColdFusion サービス
- AMF ベースの PHP サービス
- BlazeDS
- LiveCycle Data Services

LiveCycle Service Discovery ウィザードの使用方法については、Using LiveCycle Discovery を参照してください。

リモートオブジェクトとしてアクセスされるサービスを利用する場合は、該当するアプリケーションサーバーの種類に基づいて設定された Flex プロジェクトを作成します。新規 Flex プロジェクトウィザードの指示に従って、次に示すアプリケーションサーバー用のプロジェクトを設定します。

サーバーの種類	サポートされるリモートオブジェクトサービス
PHP	• AMF ベースの PHP サービス
ColdFusion	• ColdFusion Flash Remoting • BlazeDS • LiveCycle Data Services
J2EE	• BlazeDS • LiveCycle Data Services

HTTP（REST スタイル）サービスと SOAP Web サービスには、どの Flex プロジェクト設定からでも接続できます。これには、サーバーテクノロジーが指定されていないプロジェクトも含まれます。

リモートオブジェクトにアクセスするように設定されたプロジェクトは、指定されたリモートオブジェクトにのみアクセスできます。例えば、ColdFusion 用に設定されたプロジェクトから AMF ベースの PHP サービスにアクセスすることはできません。ただし、Web サービスまたは HTTP サービスとして PHP サービスに接続すれば、このプロジェクトから PHP サービスに接続することも可能です。

関連項目

1 ページの「データサービスへのアクセスの概要」

プロジェクトのサーバーの種類の変更

Flex プロジェクトが設定されていないサービスにアクセスしようとすると、Flash Builder で警告が表示されます。Flex プロジェクトで適切なサーバー設定が指定されていない場合、Flash Builder にはプロジェクトプロパティダイアログボックスへのリンクが表示されます。プロジェクトプロパティダイアログボックスで、データサービスにアクセスするようにプロジェクトを設定できます。サーバー設定が指定されていないプロジェクトから、AMF ベースの PHP サービスへのアクセスを試みると、Flash Builder で警告が表示されます。

別の種類のサービスにアクセスするように Flex プロジェクトが設定されている場合は、新しい Flex プロジェクトを設定するか、現在のプロジェクトの設定を変更します。プロジェクトのサーバー設定を変更すると、以前に設定されていたサービスにはアクセスできなくなります。例えば、プロジェクト設定を ColdFusion から PHP に変更した場合は、そのプロジェクトでアクセスしていた ColdFusion サービスが利用できなくなります。

同じプロジェクトから様々な種類のサービスにアクセスする場合は、HTTP サービスまたは Web サービスとしてサービスにアクセスできます。

クロスドメインポリシーファイル

アプリケーションの SWF ファイルから、異なるドメインに存在するサービスにアクセスする場合は、クロスドメインポリシーファイルが必要です。通常、AMF ベースのサービスでは、サービスがアプリケーションとして同じドメイン上に存在しているので、クロスドメインポリシーファイルは必要ありません。

データサービスへの接続

データサービスに接続するには、Flash Builder のサービスウィザードを使用します。

通常、リモートオブジェクトサービスの場合は、該当するアプリケーションサーバーの種類で Flex プロジェクトを作成します。Flash Builder では、サービスを分析して、サービスから返されるデータの型を設定できます。

リモートオブジェクトサービスには、ColdFusion、PHP、BlazeDS、および LiveCycle Data Services で実装されるデータサービスが含まれます。

LiveCycle Service Discovery ウィザードの使用方法については、Using LiveCycle Discovery を参照してください。

関連項目

7 ページの「データサービスにアクセスするための Flex プロジェクトの作成」

ColdFusion サービスへのアクセス

ColdFusion コンポーネント（CFC）として実装された ColdFusion データサービスにアクセスするには、Flash Builder のサービスウィザードを使用します。Flex は、リモートオブジェクトとしてこれらのサービスにアクセスします。

アプリケーションサーバーの種類として ColdFusion が指定されている Flex プロジェクトを使用します。Flex プロジェクトを作成するときに、「リモートオブジェクトアクセスサービスを使用」を指定し、「ColdFusion Flash リモーティング」を使用します。

ColdFusion データサービスへの接続

この手順では、既に ColdFusion サービスが実装され、ColdFusion サービスにアクセスする Flex プロジェクトが作成されているものと想定しています。

1 Flash Builder のデータメニューから「ColdFusion に接続」を選択して、サービスウィンドウを開きます。

2 ColdFusion サービスを設定ダイアログボックスで、サービスを実装する CFC の場所を参照します。

注意： ColdFusion サービスがまだ実装されていない場合は、単一のデータベーステーブルからサンプルサービスを生成できます。生成されたサンプルは、データサービスにアクセスする方法の例として使用できます。9 ページの「データベーステーブルからのサンプル ColdFusion サービスの生成」を参照してください。

3 （オプション）サービスの詳細を変更します。

サービス名	サービスの名前を指定します。 サービスのファイル名に基づいてサービス名が生成されます。 サービスに使用できる名前には制限があります。19 ページの「データサービス名の指定」を参照してください。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
データ型パッケージ	サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルを含むパッケージの名前を指定します。 デフォルトでは、valueObjects パッケージが作成されます。

4 （オプション）「次へ」をクリックして、サービス操作を確認します。

5 「終了」をクリックすると、サービスにアクセスする ActionScript ファイルが生成されます。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

次の手順：25 ページの「データサービス操作のデータ型の設定」。

データベーステーブルからのサンプル ColdFusion サービスの生成

Flash Builder では、独自のサービスのプロトタイプとして使用できる、サンプルの ColdFusion サービスを生成できます。サンプルサービスは単一のデータベーステーブルにアクセスし、作成、読み取り、更新、削除のメソッドを備えています。

Flash Builder で、生成されたサービスからの戻り値のデータ型が設定され、ページングやデータ管理などのデータアクセス機能が有効になります。

重要： 生成されたサービスは、信頼できる開発環境のみで使用してください。生成されたコードを使用すると、ネットワーク経由でサーバーにアクセスできるすべてのユーザーが、データベーステーブル内のデータにアクセスし、変更や削除を実行できます。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護されたサービスの作成については、Data Services のセキュリティ保護を参照してください。

次の手順では、既に ColdFusion サービスにアクセスする Flex プロジェクトが作成され、ColdFusion データソースが利用可能であるものと想定しています。

1 Flash Builder のデータメニューから「ColdFusion に接続」を選択して、サービスウィザードを開きます。

2 ColdFusion サービスを設定ダイアログボックスで、サンプルサービスを生成するためのリンクをクリックします。

3 「RDS データソースから生成」を選択し、ColdFusion データソースとテーブルを指定します。

テーブルで主キーが定義されていない場合は、テーブルの主キーを選択します。

注意： 使用可能な ColdFusion データソースがない場合は、「テンプレートから生成」を選択します。一般的なサービス操作を含むサンプルの ColdFusion コンポーネント（CFC）が生成されます。CFC に含まれる特定の関数を非コメント化し、操作を変更して、プロトタイプとして使用できるサンプルサービスを作成します。

4 デフォルトの場所を使用するか、新しい場所を指定します。「OK」をクリックします。

サンプルサービスが生成されます。デフォルト値を変更する場合は、サービス名とパッケージの場所を変更します。

5 （オプション）「次へ」をクリックして、サービスの操作を確認します。

6 「終了」をクリックします。

サンプルサービスにアクセスする ActionScript ファイルが生成されます。また、ColdFusion CFC ファイルの編集用に登録されているシステム上のエディターでサンプルサービスが開かれます。

PHP サービスへのアクセス

PHP で実装されるデータサービスに接続するには、Flash Builder のサービスウィザードを使用します。Flex では Action Message Format（AMF）を使用して、クライアントアプリケーションとデータサービスの間でデータがシリアル化されます。Flash Builder では、PHP で実装されるサービスにアクセスできるように Zend AMF フレームワークがインストールされます。19 ページの「Zend Framework のインストール」を参照してください。

PHP データサービスにアクセスするには、アプリケーションサーバーの種類として PHP が指定された Flex プロジェクトを使用します。データサービスは、プロジェクトを PHP 用に設定するときに指定した Web ルート下で使用可能なものである必要があります。次のように、サービスディレクトリ内にサービスを配置してください。

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

関連項目

7 ページの「データサービスにアクセスするための Flex プロジェクトの作成」

PHP データサービスへの接続

この手順では、既に PHP サービスが実装され、PHP サービスにアクセスする Flex プロジェクトが作成されているものと想定しています。

1 Flash Builder のデータメニューから「PHP に接続」を選択して、サービスウィザードを開きます。

2 PHP サービスを設定ダイアログボックスで、サービスを実装する PHP ファイルを参照します。

注意： PHP サービスがまだ実装されていない場合は、単一のデータベーステーブルからサンプルサービスを生成できます。生成されたサンプルは、データサービスにアクセスする方法の例として使用できます。11 ページの「データベーステーブルからのサンプル PHP サービスの生成」を参照してください。

3 （オプション）サービスの詳細を変更します。

サービス名	サービスの名前を指定します。 サービスのファイル名に基づいてサービス名が生成されます。 サービスに使用できる名前には制限があります。19 ページの「データサービス名の指定」を参照してください。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
データ型パッケージ	サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルを含むパッケージの名前を指定します。 デフォルトでは、valueObjects パッケージが作成されます。

4 「次へ」をクリックして、サービス操作を確認します。

PHP サービスにアクセスする際に、サポートされているバージョンの Zend Framework がインストールされていない場合、サポートされる最小バージョンの Zend Framework をインストールするよう求められます。19 ページの「Zend Framework のインストール」を参照してください。

5 「終了」をクリックします。

サービスにアクセスする ActionScript ファイルが生成されます。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

次の手順：25 ページの「データサービス操作のデータ型の設定」。

データベーステーブルからのサンプル PHP サービスの生成

Flash Builder では、独自のサービスのプロトタイプとして使用できる、サンプルの PHP サービスを生成できます。サンプルサービスは単一の MySQL データベーステーブルにアクセスし、作成、読み取り、更新、削除のメソッドを備えています。

Flash Builder で、生成されたサービスからの戻り値のデータ型が設定され、ページングやデータ管理などのデータアクセス機能が有効になります。

重要： 生成されたサービスは、信頼できる開発環境のみで使用してください。生成されたコードを使用すると、ネットワーク経由でサーバーにアクセスできるすべてのユーザーが、データベーステーブル内のデータにアクセスし、変更や削除を実行できます。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護されたサービスの作成については、Data Services のセキュリティ保護を参照してください。

次の手順では、既に PHP サービスにアクセスする Flex プロジェクトが作成され、MySQL データソースが利用可能であるものと想定しています。

1 Flash Builder のデータメニューから「PHP に接続」を選択して、サービスウィザードを開きます。

2 PHP サービスを設定ダイアログボックスで、サンプルサービスを生成するためのリンクをクリックします。

3 「データベースから生成」を選択して、データベースへの接続に必要な情報を指定します。「データベースに接続」をクリックします。

注意： 使用可能な PHP データソースがない場合は、「テンプレートから生成」を選択します。一般的なサービス操作を含むサンプルプロジェクトが生成されます。プロジェクトに含まれる特定の領域を非コメント化し、操作を変更して、プロトタイプとして使用できるサンプルサービスを作成します。

4 データベースでテーブルを選択し、主キーを指定します。

5 デフォルトの場所を使用するか、新しい場所を指定します。「OK」をクリックします。

PHP サービスにアクセスする際に、サポートされているバージョンの Zend Framework がインストールされていない場合、サポートされる最小バージョンの Zend Framework をインストールするよう求められます。19 ページの「Zend Framework のインストール」を参照してください。

サンプルサービスが生成されます。デフォルト値を変更する場合は、サービス名とパッケージの場所を変更します。

6 （オプション）「次へ」をクリックして、サービスの操作を確認します。

7 「終了」をクリックします。

サンプルサービスにアクセスする ActionScript ファイルが生成されます。また、PHP ファイルの編集用に登録されているシステム上のエディターでサンプルサービスが開かれます。

HTTP サービスへのアクセス

REST ベースの HTTP データサービスに接続するには、Flash Builder のサービスウィザードを使用します。任意の Flex プロジェクトから HTTP サービスに接続できます。プロジェクトのサーバーテクノロジーを指定する必要はありません。

クライアントアプリケーションの SWF ファイルから、異なるドメインに存在するサービスにアクセスする場合は、クロスドメインポリシーファイルが必要です。詳しくは、Using cross-domain policy files を参照してください。

HTTP サービスの設定

REST ベースの HTTP サービスへアクセスする際には、様々な方法でサービスへのアクセス方法を設定できます。HTTP サービスを設定ウィザードでは、次のものがサポートされています。

- ベース URL を接頭辞として使用

  1 つのサービスから提供される複数の操作にアクセスする場合、ベース URL を接頭辞として使用すると便利です。サービスのベース URL を指定する場合は、それぞれの HTTP 操作への相対パスのみを指定します。

  複数のサービスにアクセスする場合は、ベース URL を使用できません。

- クエリパラメーター付き URL

  操作の URL を指定する際には、サービス操作に対するクエリパラメーターを含めることができます。HTTP サービスを設定ウィザードのパラメーターテーブルには、操作 URL に含まれる各パラメーターが表示されます。

- 区切られたパラメーターを使用する RESTful サービス

  Flash Builder では、GET クエリパラメーターではなく区切られたパラメーターを使用する RESTful サービスへのアクセスがサポートされています。例えば、次の URL を使用して RESTful サービスにアクセスするとします。

  ```
  http://restfulService/items/itemID
  ```

  操作 URL でパラメーターを指定するには、波括弧（{}）を使用します。次に例を示します。

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

  この場合、HTTP サービスを設定ウィザードのパラメーターテーブルには、次の情報が表示されます。

名前	データ型	パラメータータイプ
items	String	URL
itemID	String	URL

  RESTful サービスパラメーターを指定する場合、データ型は常に文字列として設定され、パラメータータイプは常に URL として設定されます。

  注意： 操作の URL を指定する際には、RESTful サービスパラメーターと同時にクエリパラメーターも指定できます。

- ローカルファイルへのパス（操作 URL の場合）

  操作 URL の場合は、HTTP サービスを実装するローカルファイルへのパスを指定できます。例えば、操作 URL に対しては次のようなパスを指定できます。

  ```
  c:/MyHttpServices/MyHttpService.xml
  ```

- GET 操作と POST 操作の追加

  HTTP サービスを設定する際には、操作を追加できます。操作を追加するには、操作テーブルの「追加」をクリックします。

  操作メソッドとして GET または POST を指定します。

- 操作へのパラメーターの追加

操作テーブルで選択されている操作にパラメーターを追加できます。操作テーブルで操作を選択し、パラメーターテーブルで「追加」をクリックします。

追加するパラメーターの名前とデータ型を指定します。パラメータータイプ（GET または POST）は、操作メソッドに対応します。

- コンテンツタイプ（POST 操作の場合）

  POST 操作の場合は、コンテンツタイプを指定できます。選択できるコンテンツタイプには、application/x-www-form-urlencoded または application/xml があります。

  コンテンツタイプとして applicaton/xml を選択すると、編集できないクエリパラメーターが生成されます。strXML はデフォルトの名前です。実行時に実際のパラメーターを指定します。

名前	データ型	パラメータータイプ
strXML	String	POST

コンテンツタイプとして application/xml を選択した場合は、他のパラメーターを追加できません。

HTTP サービスへの接続

1 Flash Builder のデータメニューから「HTTP に接続」を選択して、サービスウィザードを開きます。

2 （オプション）すべての操作の接頭辞として使用するベース URL を指定します。

3 「操作」で、アクセスする操作ごとに次の項目を指定します。

- 操作メソッド（GET または POST）
- サービス操作への URL

  この URL には操作のパラメーターを含めます。REST スタイルのサービスパラメーターを指定するには、波括弧（{}）を使用します。

  Flash Builder では、次のプロトコルへのアクセスがサポートされています。

  http://

  https://

  C:/ や /Applications/ などの標準的な絶対パス
- 操作の名前

4 選択した URL に含まれる操作パラメーターごとに、パラメーターの名前とデータ型を指定します。

5 （オプション）選択した操作のパラメーターを追加または削除するには、「追加」または「削除」をクリックします。

6 （オプション）サービスの詳細を変更します。

サービス名	サービスの名前を指定します。 サービスのファイル名に基づいてサービス名が生成されます。 サービスに使用できる名前には制限があります。19 ページの「データサービス名の指定」を参照してください。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
データ型パッケージ	サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルを含むパッケージの名前を指定します。 デフォルトでは、valueObjects パッケージが作成されます。

7 （オプション）サービス用に生成されたパッケージ名を変更します。

8 「終了」をクリックします。

サービスにアクセスする ActionScript ファイルが生成されます。

HTTP サービスに接続した後、サービス操作の戻り値の型を設定します。戻り値の型を設定するときには、操作パラメーターの型も設定します。25 ページの「データサービス操作のデータ型の設定」を参照してください。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

次の手順：25 ページの「データサービス操作のデータ型の設定」。

HTTP サービスを実装する XML ファイルへのアクセス

HTTP サービスを実装する静的な XML ファイルにアクセスできます。静的な XML ファイルにアクセスするには、ローカルファイルへのパス、または URL を指定します。

サービスでは、XML 応答を返す GET メソッドが使用されます。この機能は、Flex での HTTP サービスの概要を理解し、クライアントアプリケーションで模擬データを試作する場合に便利です。

サービスにアクセスする際には、XML 応答を返すノードを指定します。Flash Builder では、このノードを使用して、戻り値のデータ型が自動的に設定されます。サービスに接続した後は、サービスに対する操作をユーザーインターフェイスコンポーネントにバインドできます。

XML サービスファイルへの接続

1 Flash Builder のデータメニューから「HTTP」を選択して、サービスウィザードを開きます。

2 「ローカルファイル」または「URL」を指定し、ファイルを参照します。

3 必要な応答を含むファイル内のノードを選択します。

応答が配列かどうかを指定します。

Flash Builder により、選択したノードの戻り値の型が設定されます。

4 サービスの詳細を変更します。

サービス名	サービスの名前を指定します。 サービスのファイル名に基づいてサービス名が生成されます。 サービスに使用できる名前には制限があります。19 ページの「データサービス名の指定」を参照してください。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
データ型パッケージ	サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルを含むパッケージの名前を指定します。 デフォルトでは、valueObjects パッケージが作成されます。

5 （オプション）サービス用に生成されたパッケージ名を変更します。

6 「終了」をクリックします。

サービスにアクセスする ActionScript ファイルが生成されます。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

Web サービスへのアクセス

Web サービス（SOAP）に接続するには、Flash Builder のサービスウィザードを使用します。Web サービスには、任意の Flex プロジェクトから接続できます。プロジェクトのサーバーテクノロジーを指定する必要はありません。

クライアントアプリケーションの SWF ファイルから、異なるドメインに存在するサービスにアクセスする場合は、クロスドメインポリシーファイルが必要です。

関連項目

クロスドメインポリシーファイルの使用

Web サービスへの接続

1 Flash Builder のデータメニューから「Web サービス」を選択して、サービスウィザードを開きます。

2 （BlazeDS ／ Data Services）LiveCycle Data Services または BlazeDS がインストールされている場合は、プロキシ経由で Web サービスにアクセスできます。

「BlazeDS または Data Services プロキシ送信先を経由」を選択します。

宛先を指定します。「次へ」をクリックして、ステップ 5 に進みます。

注意： プロキシを経由した Web サービスへのアクセスは、Flex プロジェクトでアプリケーションサーバーの種類として J2EE が指定されている場合にのみ有効です。

3 SOAP サービスの URI を入力します。

4 （オプション）サービスの詳細を変更します。

サービス名	サービスの名前を指定します。 WSDL URI に基づいてサービス名が生成されます。 サービスに使用できる名前には制限があります。19 ページの「データサービス名の指定」を参照してください。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
データ型パッケージ	サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルを含むパッケージの名前を指定します。 デフォルトでは、dataValues パッケージが作成されます。

5 （オプション）サービスのコード生成を設定します。

サービス **ポート**	使用可能なサービスからサービスを選択します。 WSDL URI に基づいてサービス名が生成されます。
操作リスト	クライアントアプリケーションでアクセスするサービスから操作を選択します。

6 「終了」をクリックします。

サービスにアクセスする ActionScript ファイルが生成されます。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

Web サービスに接続した後、サービス操作の戻り値の型を設定します。詳しくは、25 ページの「データサービス操作のデータ型の設定」を参照してください。

BlazeDS へのアクセス

Adobe BlazeDS® がインストールされていて、Remote Development Services（RDS）サーバーが設定されている場合は、BlazeDS サービスにのみアクセスできます。BlazeDS のインストールと設定については、LiveCycle Data Services ES のマニュアルを参照してください。

通常、BlazeDS データサービスにアクセスするには、アプリケーションサーバーの種類として J2EE が指定されている Flex プロジェクトを使用します。

関連項目

7 ページの「データサービスにアクセスするための Flex プロジェクトの作成」

BlazeDS サービスへの接続

この手順では、既に BlazeDS がインストールされ、Remote Development Server が設定され、BlazeDS サービスにアクセスする Flex プロジェクトが作成されているものと想定しています。

1 Flash Builder のデータメニューから「BlazeDS に接続」を選択して、サービスウィザードを開きます。

2 インポートする宛先を選択します。

3 （オプション）サービスの詳細を変更します。

サービス名	サービスの名前を指定します。 宛先に基づいてサービス名が生成されます。 サービスに使用できる名前には制限があります。19 ページの「データサービス名の指定」を参照してください。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
データ型パッケージ	サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルを含むパッケージの名前を指定します。 デフォルトでは、valueObjects パッケージが作成されます。

4 「終了」をクリックします。

サービスにアクセスする ActionScript ファイルが生成されます。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

LiveCycle Data Services へのアクセス

LiveCycle Data Services がインストールされていて、Remote Development Services（RDS）サーバーが設定されている場合は、LiveCycle Data Services から利用可能なサービスにのみアクセスできます。詳しくは、LiveCycle Data Services のマニュアルを参照してください。

LiveCycle Data Services にアクセスするには、アプリケーションサーバーの種類として J2EE または ColdFusion が指定されている Flex プロジェクトを使用します。

LiveCycle Data Services のサービスタイプ

LiveCycle Data Services に接続するときは、次のタイプのデータサービスを宛先として使用できます。

- Remoting Service

  Remoting Service は、AMF 型指定を使用して実装されます。このサービスでは、サーバー側のデータ管理は行えません。Flash Builder ツールを使用して、クライアント側のデータ管理を設定できます。31 ページの「データ管理の有効化」を参照してください。

- データサービス

  データサービスは、サーバー側のデータ管理を実装するサービスです。詳しくは、LiveCycle Data Services のマニュアルを参照してください。

- Web サービス

  Web サービスは、LiveCycle Data Services の宛先として設定された LiveCycle Data Services プロキシを経由して利用できます。通常、Web サービスに接続する場合、サーバー側の型指定は提供されません。

データ型の設定とデータ管理

Flash Builder には、クライアント側のデータ設定とクライアント側のデータ管理に必要なツールが用意されています。使用できる Flash Builder ツールは、LiveCycle Data Services の宛先の種類によって異なります。

- Remoting Service

  Remoting Service では、AMF 型指定がサービスで実装されます。Remoting Service の宛先では、戻り値のデータ型を設定しません。

  ただし、Flash Builder を使用して、クライアント側のデータ管理用のコードを生成できます。31 ページの「データ管理の有効化」を参照してください。

- データサービス

  データサービスでは、サーバー側のデータ型が実装されます。データサービスの宛先に対しては、戻り値のデータ型を設定しません。

  データサービスの宛先では、サーバー側のデータ管理も可能です。データサービスの宛先に対しては、クライアント側のデータ管理は使用しません。

- Web サービス

  通常、LiveCycle Data Service プロキシを経由して使用できる Web サービスの宛先では、サーバー側の型指定は実装されません。Flash Builder ツールを使用して、Web サービス操作の戻り値の型を設定できます。25 ページの「データサービス操作のデータ型の設定」を参照してください。

  Flash Builder を使用して、クライアント側のデータ管理用のコードを生成できます。31 ページの「データ管理の有効化」を参照してください。

LiveCycle Data Service の宛先（データサービスおよび Remoting Service の宛先）への接続

この手順では、既に LiveCycle Data Services がインストールされ、Remote Development Server が設定され、LCDS サービスにアクセスする Flex プロジェクトが作成されているものと想定しています。

1 Flash Builder のデータメニューから「データとサービスに接続」を選択して、サービスウィザードを開きます。

2 サービスタイプを選択ダイアログボックスで、「LCDS」を選択します。「次へ」をクリックします。

3 必要に応じて、ログインに必要な情報を入力します。

4 （オプション）サービスの詳細を変更します。

サービス名	サービス名は指定しません。サービス名は Flash Builder によって生成されます。宛先に基づいてサービス名が生成されます。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
宛先	LiveCycle Data Services サーバーから使用できる宛先を 1 つ以上指定します。
データ型パッケージ	データ型パッケージの名前を指定します。このパッケージには、サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルが含まれます。 デフォルトでは、valueObjects パッケージが作成されます。

5 「終了」をクリックします。

サービスにアクセスする ActionScript ファイルが生成されます。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

LiveCycle Data Service の宛先（Web サービスの宛先）への接続

この手順では、既に LiveCycle Data Services がインストールされ、Remote Development Server が設定され、DS サービスにアクセスする Flex プロジェクトが作成されているものと想定しています。

1 Flash Builder のデータメニューから「データとサービスに接続」を選択して、サービスウィザードを開きます。

2 サービスタイプを選択ダイアログボックスで、「Web サービス」を選択します。「次へ」をクリックします。

3 「LCDS または Blazeds のプロキシ送信先を経由」を選択します。

4 必要に応じて、ログインに必要な情報を入力します。

5 宛先を選択します。

6 （オプション）サービスの詳細を変更します。「次へ」をクリックします。

サービス名	サービスの名前を指定します。 宛先の名前に基づいてサービス名が生成されます。 サービスに使用できる名前には制限があります。19 ページの「データサービス名の指定」を参照してください。
サービスパッケージ	サービスにアクセスする生成済みの ActionScript ファイルを含むパッケージの名前を指定します。 サービス名に基づいてパッケージが生成され、services パッケージに追加されます。
データ型パッケージ	サービスから取得されるデータ型を定義する生成済みの ActionScript クラスファイルを含むパッケージの名前を指定します。 デフォルトでは、dataValues パッケージが作成されます。

7 （オプション）サービスのコード生成を設定します。

サービス **ポート**	使用可能なサービスとポートから、サービスとポートを選択します。
操作リスト	クライアントアプリケーションでアクセスするサービスから操作を選択します。

8 「終了」をクリックします。

サービスにアクセスする ActionScript ファイルが生成されます。

注意： サービスに接続した後は、サービスのプロパティを変更できます。データとサービスビューで、サービスを選択します。コンテキストメニューで、「プロパティ」を選択します。

関連項目

7 ページの「データサービスにアクセスするための Flex プロジェクトの作成」

データサービス名の指定

Flash Builder からアクセスできるデータサービスの名前には制限があります。アプリケーションをコンパイルするまで明らかにならない制限もあります。

サービス名を指定するときには、次のガイドラインに従ってください。

- サービス名の先頭には数字を使用できません。
- ActionScript のキーワードをサービス名にすることはできません。
- カスタムクラスを含め、ActionScript クラスの名前をサービス名に使用することはできません。
- （PHP のみ）サービス名にアンダースコアが含まれていると、Flash Builder にサービスをインポートできません。

注意： MXML ファイルの名前とは異なるサービス名を使用することをお勧めします。

Zend Framework のインストール

PHP サービスに初めてアクセスすると、Flash Builder はサポートされているバージョンの Zend Framework がインストールされているかどうかを確認します。サポートされているバージョンの Zend Framework が見つからない場合、Zend Framework をインストールしてよいかどうかの確認を求めるメッセージが Flash Builder に表示されます。承認した場合は、最小バージョンの Zend Framework がインストールされます。拒否した場合は、PHP サービスにアクセスするときに Zend Framework を手動でインストールします。

デフォルトの Flash Builder インストール

Flash Builder をインストールすると、Web サーバーのルートディレクトリ内の ZendFramework フォルダーに Zend Framework がインストールされます。

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

PHP サービスにアクセスする Flex プロジェクトの場合、Flash Builder ではプロジェクト出力フォルダー内に次の設定が作成されます。

- amf_config.ini
- gateway.php

実稼働サーバー

実稼働サーバーの場合は、ZendFramework フォルダーを Web ルートの外に移動することをお勧めします。amf_config.ini で定義されている zend_path 変数を更新します。

zend_path 変数がコメント化されている場合は、zend_path 変数をコメント解除します。Zend Framework インストールの場所を指定します。

Zend Framework の手動インストール

Zend Framework は手動でインストールすることもできます。

1 Zend Framework の最新リリースをダウンロードします。

最小パッケージまたは完全パッケージをインストールできます。Flash Builder では最小パッケージがインストールされます。

2 ダウンロードしたバージョンをシステム上の任意の場所に抽出します。

3 PHP サービスにアクセスするには、Flex プロジェクトフォルダーの amf_config.ini で定義されている zend_path 変数を更新します。

zend_path 変数がコメント化されている場合は、zend_path 変数をコメント解除します。Zend Framework のインストール場所への絶対パスを指定します。

Zend Framework インストールのトラブルシューティング

Zend Framework への接続中に発生したエラーを解決するためのヒントを次に示します。

Zend Framework の手動インストール

Zend Framework を手動でインストールした場合は、amf_config.ini ファイル内の zend_path 変数を確認します。

amf_config.ini はプロジェクト出力フォルダーにあります。

次の点を確認します。

- zend_path がコメント解除されていること。
- Zend Framework インストールへのパスが正しく指定されていること。
  - このパスは、ローカルファイルシステム上のインストール先への絶対パスです。マップされたネットワークリソースへのパスを指定することはできません。
  - このパスは、Zend Framework インストールのライブラリフォルダーへのパスです。通常、ライブラリフォルダーは次の場所にあります。

    （Windows）C:¥apache¥PHP¥Frameworks¥ZendFramework¥library

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

Zend Framework の Flash Builder インストール

Flash Builder とともに Zend Framework をインストールした場合は、次の点を確認します。

- Web ルートフォルダーの場所

  Flash Builder とともにインストールした場合、Zend Framework はプロジェクトの Web ルートフォルダーにインストールされます。Web ルートフォルダーの場所を確認します。プロジェクト／プロパティ／ Flex サーバーを選択します。

- PHP を使用するように Web サーバーが設定されていることを確認します。
- amf_config.ini ファイル内の zend_path 変数を確認します。

  amf_config.ini はプロジェクト出力フォルダーにあります。

  次の点を確認します。

  - zend_path がコメント解除されていること。
  - 指定したパスが、プロジェクトの Web ルートにある Zend Framework インストールを指していること。

- このパスは、ローカルファイルシステム上のインストール先への絶対パスです。マップされたネットワークリソースへのパスを指定することはできません。

単一サーバーインスタンスの使用

データサービスに接続すると、プロジェクト内の各アプリケーションがサービスにアクセスできるようになります。デフォルトでは、サーバーにアクセスすると、アプリケーションごとに独自のサービスインスタンスが作成されます。

この動作を変更して、1 つのプロジェクトにサービスインスタンスが 1 つだけ存在するように設定することもできます。この場合、プロジェクト内の各アプリケーションは同じサービスインスタンスにアクセスします。通常、複数のアプリケーションからのデータアクセスを連携させる必要がある場合は、単一のサーバーインスタンスを作成します。

単一サービスインスタンスへのアクセスはプロジェクトごとに指定できますが、すべてのプロジェクトに適用される環境設定として指定することもできます。

プロジェクトごとに単一サーバーインスタンスへのアクセスを指定する方法

1 プロジェクト／プロパティ／データとサービスを選択します。

2 単一サーバーインスタンスを使用するチェックボックスをオンにします。「OK」をクリックします。

環境設定として単一サーバーインスタンスへのアクセスを指定する方法

1 設定ダイアログボックスを開きます。

2 Flash Builder ／データとサービスを選択します。

3 単一サーバーインスタンスを使用するチェックボックスをオンにします。「OK」をクリックします。

クライアントアプリケーションの作成

MXML コードエディターを使用して、ユーザーインターフェイスを作成できます。

コードエディターを使用してアプリケーションのコンポーネントを定義したら、サービスから返されたデータをユーザーインターフェイスコンポーネントにバインドできます。ユーザーとアプリケーションのやり取りに使用するイベントハンドラーを必要に応じて生成します。

データとサービスビューに表示されるサービス操作からフォームを生成することもできます。

コントロールへのサービス操作のバインディング

データにバインドダイアログを使用して、サービス操作をユーザーインターフェイスコンポーネントにバインドします。

データにバインドダイアログボックスには、データとサービスビューのツールバーにあるデータメニューからアクセスできます。

サービス操作をコンポーネントにバインドする際、クライアントアプリケーションからサービス操作にアクセスするための MXML コードと ActionScript コードが Flash Builder によって生成されます。

サービス操作の戻り値の型

サービス操作をコントロールにバインドすると、Flash Builder では、サービス操作の戻り値のデータ型が使用されます。通常は、コンポーネントにバインドする前に、サービス操作の戻り値の型を設定します。

サービス操作の戻り値の型がまだ設定されていない場合は、データにバインドダイアログボックスで、この設定を行うように求められます。

25 ページの「データサービス操作のデータ型の設定」を参照してください。

サービス操作への DataGrid コントロールのバインディング（データにバインドダイアログボックス）

この手順では、データサービスに接続しているものと想定しています。

1. アウトラインビューで DataGrid コントロールを選択します。または、MXML エディターでカーソルを <s:DataGrid> タグ内に置きます。
2. DataGrid が選択されている状態で、Flash Builder データメニューから「データにバインド」を選択して、データにバインドダイアログを開きます。
3. 「新規サービスの呼び出し」を選択してから、「サービス」および「操作」を選択します。

   サービス操作が既にコンポーネントにバインドされている場合は、その結果を使用できます。この場合は、「既存の呼び出し結果」を指定して、使用する操作を選択します。
4. （オプション）「戻り値の型を変更」を選択します。

   サービス操作の戻り値の型を再設定する場合は、「戻り値の型を変更」を選択します。

   操作の戻り値の型をまだ設定していない場合は、「戻り値の型を設定」を選択します。

   25 ページの「データサービス操作のデータ型の設定」を参照してください。
5. 「OK」をクリックします。

   DataGrid コンポーネントが変化して、データベースから取得されたフィールドが表示されます。

   DataGrid および AdvancedDataGrid コンポーネントの構成を参照してください。
6. アプリケーションを保存した後、実行してみます。

操作に対するサービス呼び出しの生成

Flash Builder では、サービスの操作を呼び出す ActionScript メソッドを生成できます。メソッドはユーザーインターフェイスコンポーネントにバインドされるのではなく、アプリケーションコードで使用できます。

Flash Builder でこのような ActionScript メソッドを生成すると、サービス呼び出しから返されるデータへのアクセスを可能にする CallResponder も作成されます。35 ページの「CallResponder」を参照してください。

操作に対するサービス呼び出しの生成

この手順では、データサービスに接続しているものと想定しています。

1. データとサービスビューで、操作を選択します。
2. 操作のコンテキストメニューから、「サービスの呼び出しを生成」を選択します。

   Flash Builder で、操作を呼び出すメソッドが生成され、生成されたメソッドが MXML エディターのソースモードで表示されます。Flash Builder で、サービスの呼び出し結果を保持するための CallResponder が作成されます。

   このオプションは、データとサービスのツールバーからも使用できます。

アプリケーション用のフォームの生成

フォームは、ユーザーから情報を収集するために Web アプリケーションで一般的に使用される方法の 1 つです。Flash Builder では、サービスの呼び出しから取得したデータ用のフォーム、またはリモートデータにアクセスするために使用するカスタムデータ型用のフォームを生成できます。

Flash Builder でフォームを生成すると、Form レイアウトコンテナが作成され、サービスから取得した特定のデータを表示または編集するためのコンポーネントが追加されます。

Flash Builder では、次の種類のフォームが生成されます。

フォーム	説明
データ型	同じデータ型のフィールドを表すコンポーネントが格納されます。
マスター／詳細フォーム	通常、「マスター」コンポーネントは、サービスから取得されたデータを一覧表示するデータコントロールです。 「詳細」フォームは、マスターコンポーネントで選択された個々のアイテムを表します。
サービスの呼び出し	2 つのフォームを作成します。一方のフォームでは、操作への入力を指定します。もう一方のフォームには、返されたデータが表示されます。

フォームを生成するときは、含めるフィールド、各フィールドを表すユーザーインターフェイスコントロールの種類、およびフォームを編集可能にするかどうかを指定します。

フォームの生成

この手順では、サービスの呼び出し用のフォームを生成する方法を示します。他の種類のフォームを生成する手順も同様です。

1 フォームを生成ウィザードを実行するために、データとサービスビューで操作を選択します。次のいずれかの操作を実行します。

- 操作のコンテキストメニューから、「フォームを生成」を選択します。
- Flash Builder のデータメニューで、「フォームを生成」を選択します。

2 フォームを生成ウィザードで、「フォームを生成する対象 : サービスの呼び出し」を選択します。

3 「新規サービスの呼び出し」または「既存の呼び出し結果」を選択します。

サービスの呼び出し用に既に生成されたコードを使用するには、「既存の呼び出し結果」を指定します。

それ以外の場合は、「新規サービスの呼び出し」を指定して、フォームのサービスと操作を選択します。

4 （オプション）操作の種類に応じて、生成されるフォームに関するオプションがいくつか用意されています。

パラメーターを受け入れる操作の場合は、パラメーターにフォームを含めることができます。

値を返す操作の場合は、戻り値にフォームを含めることができます。

フォームを編集可能にするか、読み取り専用にするかを選択できます。

5 （オプション）入力の型または戻り値の型を設定します。

選択した操作が入力パラメーターを持つ場合、または値を返す場合は、入力の型または戻り値の型を設定できます。

フォームを生成する前に、操作の入力の型と戻り値の型を設定します。これらの型が既に設定されている場合は、再度設定することもできます。

25 ページの「データサービス操作のデータ型の設定」を参照してください。

6 「次へ」をクリックします。プロパティコントロールマッピングダイアログボックスで、フォームに含めるフィールドと、データを表すコントロールの種類を選択します。

7 「終了」をクリックします。

Flash Builder で複数のフォームを生成すると、フォームの上に別のフォームが重なって配置されることがあります。生成されたフォームを再配置するには、フォームのコンポーネントではなくフォーム全体を確実に選択して移動します。

フォームの上に別のフォームが重なって配置されている場合、フォームの選択時に混乱が生じる場合があります。コードエディターで、いずれかのフォームに相当するタグを選択します。

マスター／詳細フォームの生成

マスター／詳細フォームを作成するには、アプリケーションにデータコントロールコンポーネントを追加し、操作の結果をコントロールにバインドします。

例えば、DataGrid コンポーネントを追加し、getItems_paged() などの操作の結果を DataGrid にバインドします。

1 アウトラインビューで、DataGrid などのデータコントロールを選択します。

2 データメニューから、「詳細フォームを生成」を選択します。

3 「フォームの生成」の説明に従って、フォームを生成します。

データ型用のフォームの生成

カスタムデータ型のフィールドを表すコンポーネントを含んだフォームを生成するには、先にデータ型の設定を行う必要があります。25 ページの「データサービス操作のデータ型の設定」を参照してください。

1 データとサービスビューで、カスタムデータ型を選択します。

2 コンテキストメニューで、「フォームを生成」を選択します。

3 「フォームを生成する対象 : データ型」が選択されていることを確認し、データ型を選択します。

4 （オプション）フォームを編集可能にするかどうかを選択できます。

5 「終了」をクリックします。

リモートデータを取得するイベントハンドラーの生成

データサービス操作をコンポーネントにバインドすると、サービスからデータを取得してコンポーネントに設定するイベントハンドラーが生成されます。

例えば、getAllItems() などの操作を DataGrid にバインドすると、イベントハンドラー creationComplete が Flash Builder によって生成されます。DataGrid は、生成されたイベントハンドラーを参照します。呼び出しの結果が、DataGrid のデータプロバイダーになります。

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

アプリケーションを実行すると、DataGrid が作成された後に、サービスから取得されたデータがイベントハンドラーによって DataGrid に挿入されます。

イベントハンドラーを生成する場合は、生成されたハンドラーをそのまま使用するか、必要に応じて別のハンドラーに置き換えることができます。例えば、DataGrid 上のイベントハンドラー creationComplete を Application の creationComplete ハンドラーで置き換え可能です。

また、Button や Text など、ユーザー入力を受け付けるコントロールのイベントハンドラーを生成、または作成できます。

ユーザーインターフェイスコンポーネントのイベントハンドラーの生成

1. DataGrid や Button などのユーザーインターフェイスコンポーネントを含むアプリケーションを作成します。
2. Flash Builder では、イベントハンドラーの生成に役立つコンテンツアシストを利用できます。Control+ スペースまたは Cmd+ スペース（Mac）を押して、「イベントハンドラーを生成」を選択します。
3. Flash Builder によって、イベントハンドラーの一意の名前が生成され、スクリプトブロックにイベントハンドラーが配置されます。

   コードエディターでイベントハンドラーの生成されたスタブが強調表示されます。イベントハンドラーの残りのコードを入力します。イベントハンドラーのコーディングに役立つコンテンツアシストを使用できます。

データサービス操作のデータ型の設定

データサービスに接続するときは、サービス操作によって返されるデータのデータ型が Flash Builder によって認識されている必要があります。サポートされるデータ型は、データサービスまたはリモートサービスとデータを交換できるように、AMF によって認識される型です。

多くのデータサービスでは、返されるデータのデータ型をサーバー上で定義しています（サーバー側の型指定）。ただし、サーバーで型が定義されていない場合は、クライアントアプリケーションで返されるデータの型を設定する必要があります（クライアント側の型指定）。

パラメーターを指定するサービス操作では、サービスでアクセスするデータに対応する型も指定する必要があります。クライアント側の型指定を使用する場合は、入力パラメーターの型を設定します。

クライアント側の型指定の型を設定した場合、Flash Builder では AMF データ型のみが認識されます。使用できる型は、複雑なデータを表すカスタムデータ型、または操作がデータを返さないことを示す void です。

複雑なデータを返すサービス操作の場合は、ユーザー定義型を設定できます。例えば、従業員データベースからレコードを取得する場合は、返される複雑なデータを Employee として定義します。この場合、Employee のカスタムデータ型には、データベースレコードの各フィールドに対応するエントリが含まれます。

クライアント側の型指定のデータ型

データ型	説明
ActionScript の型	Boolean Boolean[] ByteArray ByteArray[] Date Date[] int int[] Number Number[] Object Object[] String String[]
データを返さない	void
ユーザー定義型	**CustomType** **CustomType**[]

ユーザー定義型（Employee）

フィールド	データ型
emp_no	Number
first_name	String
last_name	String
hire_date	Date
birth_date	Date

サービスへのアクセスの認証

通常、データサービスでは、サービスへのアクセスを許可する前にユーザー認証が求められます。HTTP プロトコルを使用してアクセス可能な PHP、BlazeDS、および ColdFusion サービスでは、追加の認証を要求できます。これらのサービスでは HTTP 認証とリモート認証の両方を要求される場合があります。

Flash Builder には、次の作業を行うときのサービス認証に関するオプションが用意されています。

- 操作の戻り値の型の設定

  27 ページの「操作からの戻り値のデータ型の設定」を参照してください。

- 「操作をテスト」インターフェイスの使用

  29 ページの「サービス操作のテスト」を参照してください。

「認証が必要です」を指定すると、Flash Builder でサービス認証ダイアログボックスが開きます。アクセスするサービスの種類に応じて、基本認証またはリモート認証を指定できます。

基本認証

基本認証は、HTTP サービスおよび Web サービスへのアクセスを可能にします。これらのサービスにアクセスするには、ユーザー名とパスワードを入力します。

セッション全体を通して、入力した資格情報を使用するように設定する場合は、「ユーザー名とパスワードを記憶」を指定します。

リモート認証

リモート認証は、リモートオブジェクトサービスへのアクセスを可能にします。リモートオブジェクトサービスとは、ColdFusion、PHP、BlazeDS、または LiveCycle Data Services（）を使用するリモートオブジェクトとして実装されたサービスです。

Flash Builder では、リモートオブジェクトサービスを実装しないプロジェクトに対しては、リモート認証ログインインターフェイスは提供されません。

リモートオブジェクトサービスにアクセスするには、ユーザー名とパスワードを入力します。

セッション全体を通して、入力した資格情報を使用するように設定する場合は、「ユーザー名とパスワードを記憶」を指定します。

操作の入力パラメーターの設定

クライアント側の型指定では、データサービスから使用可能な操作の入力パラメーターを設定します。

次の手順では、既に Flash Builder でデータサービスに接続していて、設定可能な入力パラメーターを必要とする操作がそのデータサービスに含まれているものと想定しています。

1. データとサービスビューで、設定可能な入力パラメーターが含まれる操作を選択します。操作のコンテキストメニューから、「入力の型を設定」を選択します。
2. 入力の型を設定ダイアログボックスで、操作の引数ごとに、使用可能な型の一覧からデータ型を選択します。「OK」をクリックします。

   サービスの戻り値のカスタムデータ型が既に定義されている場合は、それらの型も選択できます。

サーバー側の型指定の場合、入力パラメーターのデータ型はサービスで指定されます。

操作からの戻り値のデータ型の設定

操作から返されるデータ型が定義されたサービスでは、サーバー側の型指定を使用できます。操作から返されるデータ型が定義されていないサービスの場合、Flash Builder では、クライアント側の型指定を使用して戻り値のデータ型が定義されます。

Flash Builder により、サービス操作から返されるデータが分析され、データ型が決定されます。操作の戻り値の型を設定する際には、次の 2 つのオプションがあります。

- サンプルデータからの戻り値の型を自動検出する

  サービスでサーバー側の型指定が実装されている場合は、サービスで定義されたデータ型が Flash Builder で検出されます。

  サービスでサーバー側の型指定が実装されていない場合は、クライアントアプリケーション用のカスタム型が Flash Builder で作成されます。クライアント側の型指定では、カスタム型の名前を手動で指定します。一般的には、返されるデータの内容を示す名前を使用します。例えば、データベーステーブルからブックの配列が返される場合は、型の名前として Book と指定します。
- 既存の型を使用する

既存の型とは、サービスで定義されている型、ActionScript 型、または以前に定義されたカスタム型です。

Flash Builder でデータを分析するために使用される手順は、データサービスの種類によって若干異なります。例えば、HTTP サービスの戻り値の型を分析して設定する手順は、PHP サービスや ColdFusion サービスでの手順とは異なります。

データ型のマージと変更

サーバーデータの分析時には、別のデータ型からフィールドをマージしたり、既存のデータ型に基づいてデータ型を作成できます。カスタムデータ型を変更する方法には次のようなものがあります。

- 既存のデータ型に対して新しい名前を使用する

  返されたデータをクライアントアプリケーションで複数の方法で使用する場合は、新しい名前を使用します。

  例えば、従業員データを取得して、クライアントアプリケーションの従業員サマリーテーブルと従業員詳細テーブルで使用する場合などです。

- フィールドをマージする

  返されたフィールドを既存のデータ型に追加できます。フィールドの追加は、複数のソースからのデータを関連付けるときに役立ちます。例えば、複数のデータベーステーブルから取得したデータを返す JOIN 操作の場合などです。

  また、複数のサービスから受け取ったデータを使用する場合にも役立ちます。例えば、HTTP サービスと ColdFusion サービスの両方から受け取った Book データをマージする場合などです。

カスタムデータ型の設定（PHP サービスまたは ColdFusion サービス）

この手順では、PHP または ColdFusion で実装されたデータサービスへ既に接続しているものと想定しています。

1. データとサービスビューで、操作のコンテキストメニューから「戻り値の型を設定」を選択します。
2. 操作に引数が含まれる場合は、引数の値を入力します。引数の正しいデータ型を指定します。
3. （カスタム型を新規作成または修正）この操作からの戻り値の型を自動検出するように指定します。

   サービスで認証が要求される場合は、「認証が必要です」を選択し、必要に応じて資格情報を入力します。26 ページの「サービスへのアクセスの認証」を参照してください。

   Flash Builder で操作が分析され、カスタムデータ型が構築されます。

   カスタムデータ型の名前を指定します。

   カスタムデータ型が既に定義されている場合は、返されるフィールドを既存のカスタムデータ型の定義に追加できます。
4. （既存の型を使用）このオプションは、ActionScript 型または設定済みの型を指定する場合に使用します。
5. 「終了」をクリックします。

カスタムデータ型の設定（HTTP サービス）

この手順では、HTTP サービスへ既に接続しているものと想定しています。

1. データとサービスビューで、操作のコンテキストメニューから「戻り値の型を設定」を選択します。
2. （カスタム型を新規作成）この操作からの戻り値の型を自動検出するように指定します。

   サービスで認証が要求される場合は、「認証が必要です」を選択し、必要に応じて資格情報を入力します。

   Flash Builder で操作が分析され、カスタムデータ型が構築されます。操作のパラメーター値を渡すメソッドを選択し、「次へ」をクリックします。

   - （パラメーター値を入力）各パラメーターの値を指定します。

     パラメーターのデータ型も指定できます。指定しない場合は、デフォルトのデータ型が自動的に選択されます。

- （サービスの URL を入力）パラメーターと値を含めて HTML サービスの URL を入力します。次に例を示します。

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

- （XML/JSON 応答を入力）XML/JSON 応答をテキストボックスにコピーします。

  このオプションは、オフラインの場合、または HTTP サービスが開発中で、サーバーからの応答が分かっている場合に使用します。

3 （カスタム型を新規作成、続き）カスタムデータ型の名前を指定するか、返されるデータからノードを選択します。

返されるデータのノードを選択した場合は、そのノードの戻り値のカスタムデータ型が作成されます。

データが配列として返されるかどうかを指定します。

サービスから XML ファイルが返される場合は、「ルートを選択」ドロップダウンリストが有効になります。データ型を指定するには、XML ファイルからノードを選択します。

4 （既存の型を使用）このオプションは、ActionScript 型または設定済みの型を指定する場合に使用します。

5 「終了」をクリックします。

サービス操作のテスト

Flash Builder では、サービス操作をテストして、操作から返されるデータを表示できます。この機能は、サービスの動作を検証する上で役立ちます。

重要： 一部の操作（更新や削除）では、サーバー上のデータが変更されます。

サービス操作のテスト

この手順では、データサービスに接続しているものと想定しています。

1 データとサービスビューから、サービスの操作を選択します。コンテキストメニューで、「操作をテスト」を選択します。

操作をテストビューが開き、選択した操作が表示されます。入力パラメーターが必要な操作の場合は、操作をテストビューにパラメーターが一覧表示されます。

2 必須の入力パラメーターで、「値を入力」フィールドをクリックして、パラメーターの値を指定します。

複雑な型を要求するパラメーターの場合は、フィールド内の「...」ボタンをクリックして、引数を入力エディターを開きます。エディターで値を指定します。

引数を入力エディターでは、複雑な型を ActionScript で表現するための JSON 表記法を使用できます。

3 サーバーから認証が要求される場合は、「認証が必要です」を選択します。「テスト」をクリックします。

必要に応じて、認証情報を入力します。26 ページの「サービスへのアクセスの認証」を参照してください。

サービスから返されたデータが表示されます。

4 （オプション）操作をテストビューで、テスト可能なその他のサービスや操作を選択します。

サーバーからのデータのアクセスの管理

ページング ページングとは、リモートサービスから大きなデータセットをインクリメンタルに取得することです。

例えば、10,000 件のレコードがあるデータベースにアクセスして、20 行の DataGrid にそのデータを表示するとします。この場合、ページング操作を実装して、20 セットずつインクリメンタルに行を取り出すことができます。ユーザーが（DataGrid でスクロールして）さらにデータを要求する場合、次のページのレコードが取り出されて表示されます。

データ管理 Flash Builder では、データ管理とは、クライアントアプリケーションからサーバー上のデータに更新を同期することです。データ管理を使用すると、サーバーを更新することなく、クライアントアプリケーション内の 1 つまたは複数の項目を変更できます。その後、1 回の操作でサーバーにすべての変更をコミットできます。データを更新しないで変更を元に戻すこともできます。

データ管理には、従業員レコードの更新などのクライアントアプリケーションからのイベントに対応するいくつかの操作（作成、取得、更新、削除）の連係が含まれます。

Flash Builder でデータ管理を有効にすると、ユーザーインターフェイスコンポーネントを自動的に更新するためのコードも生成されます。例えば、DataGrid とサーバー上のデータの同期を維持するためのコードが生成されます。

ページングの有効化

次のシグニチャを持つページング関数を実装するデータサービスに対しては、ページングを有効にできます。

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

関数名	関数には、任意の有効な名前を使用できます。
startIndex	取得するデータの開始行。 startIndex のデータ型は、クライアント操作で Number として定義します。
numItems	各ページで取得するデータの行数。 numItems のデータ型は、クライアント操作で Number として定義します。
myDataType	データサービスから返されるデータ型。

サービスからのページングを実装するときに、count() 操作も実装できます。count() 操作は、サービスから返されるアイテムの数を返します。Flash Builder で count() 操作を使用するには、次のシグニチャを実装する必要があります。

```
count(): Number
```

関数名	関数には、任意の有効な名前を使用できます。
Number	操作から取得したレコードの数。

Flex は、count 操作を使用して、大きいデータセットを取得するユーザーインターフェイスコンポーネントを正しく表示します。例えば、count() 操作は、DataGrid のスクロールバーのサムのサイズを決定するのに役立ちます。

リモートサービスの中には count() 操作を備えていないものがあります。その場合もページングは機能しますが、ページングされたデータを表示するコントロールで、データセットのサイズが正しく表されない可能性があります。

フィルターされるクエリに対するページング操作

データベースからの結果がフィルターされるクエリに対してページングを有効にできます。クエリで結果をフィルターする際には、ページング関数と count 関数に対して次のシグニチャを使用します。

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

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

filterParam1	オプションのフィルターパラメーター。このパラメーターは getItems_PagedFiltered() と countFiltered() で同じです。
filterParam2	オプションのフィルターパラメーター。このパラメーターは getItems_PagedFiltered() と countFiltered() で同じです。

PHP で実装されている getItems_pagedFiltered() 関数のコードスニペットを次に示します。このコードは MySQL データベースにアクセスします。このコードスニペットは、オプションフィルターパラメーターの使用方法を示しています。

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

操作に対するページングの有効化

この手順では、getItems_paged() 操作と count() 操作の両方が、リモートサービスで既にコーディングされているものと想定しています。また、25 ページの「データサービス操作のデータ型の設定」で説明されているように、操作の戻り値のデータ型が設定されているものと想定しています。

1. データとサービスビューで、getItems_paged() 操作のコンテキストメニューから「ページングを有効にする」を選択します。
2. 事前にデータ型の一意キーを指定していない場合は、このデータ型のインスタンスを一意に示す属性を指定します。「次へ」をクリックします。

   通常、この属性は主キーです。
3. （オプション）カスタムのページサイズを定義するには、取り出すアイテムの数を指定します。

   カスタムのページサイズを指定しない場合は、サービスレベルでデフォルトのページサイズが設定されます。デフォルトのページサイズは、1 ページにつき 20 件のレコードです。
4. （オプション）count() 操作を指定します。「終了」をクリックします。

   count() 操作により、スクロールバーの縮小サイズなど、ユーザーインターフェイス要素が正しく表示されます。

   操作に対して、ページングが有効になります。

   データとサービスビューで、ページングを実装する関数のシグニチャに startIndex パラメーターと numItems パラメーターが含まれなくなります。これらの値は動的に追加されます。これらの値は、ユーザーが指定したカスタムページサイズ、またはデフォルトのページサイズ（1 ページにつき 20 件のレコード）に基づいて決定されます。

データ管理の有効化

サービスのデータ管理を有効にするには、次に示す 1 つ以上の関数をサービスで実装します。データ管理機能では、これらの関数を使用して、リモートサーバー上のデータに更新を同期します。

- 追加（createItem）

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

  createItem() の戻り値の型は、データベースの主キーの型です。
- すべてのプロパティの取得（getItem）

  ```
  getItem(itemID:Number): myDatatype
  ```
- 更新（updateItem）

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

- 削除（deleteItem）

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

これらの関数には次のシグニチャが必要です。

関数名	関数には、任意の有効な名前を使用できます。
item **originalItem**	データサービスから返されるデータ型のアイテム
itemID	アイテムの一意の識別子。通常は、データベースの主キー
changes[]	指定したアイテム内のフィールドに対応する配列。この引数は、updateItem() の 1 つのバージョンでのみ使用されます。
myDataType	データサービスから使用可能なアイテムのデータ型。通常は、サービスからデータを取得するときにカスタムデータ型を定義します。

autoCommit フラグ

データ管理機能を使用すると、サーバー上のデータに更新を同期できます。クライアントアプリケーションでデータが変更された場合、***service***.commit() メソッドを呼び出すまでサーバー上のデータは更新されません。

この機能を無効にする場合は、autoCommit フラグを true に設定します。autoCommit が true になっていると、サーバーデータの更新がキャッシュに保留されず、即座に実行されます。38 ページの「サービスのデータ管理の有効化」を参照してください。

deleteItemOnRemoveFromFill フラグ

データ管理が有効になっているアイテムを削除するときには、deleteItemOnRemoveFromFill フラグを使用します。このフラグはデフォルトで true に設定されています。アイテムを削除すると、そのアイテムはデータベースから即座に削除されます。

deleteItemOnRemoveFromFill を false に設定すると、commit() メソッドが呼び出されるまで削除操作が保留されます。次の例は、DataGrid の creationComplete イベントハンドラーのコードを示しています。DataGrid で選択されている Employee アイテムをユーザーが削除した場合、そのアイテムは commit() メソッドが呼び出されるまでデータベースから削除されません。

```
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();
}
```

操作に対するデータ管理の有効化

この手順では、必要な操作をリモートサービスで既に実装しているものと想定しています。また、カスタムデータ型を使用する操作の戻り値のデータ型が設定されているものと想定しています。25 ページの「データサービス操作のデータ型の設定」を参照してください。

1 データとサービスビューで、データ型ノードを展開します。

2 データ型のコンテキストメニューから、「データ管理を有効にする」を選択します。

3 事前にデータ型の一意キーを指定していない場合は、このデータ型のインスタンスを一意に示す属性を指定します。「次へ」をクリックします。

通常、この属性は主キーです。

4 既に実装されている、追加、すべてのプロパティの取得、更新、および削除操作を指定します。「終了」をクリックします。

注意：これらの関数をすべて実装する必要はありません。アプリケーションに必要な関数のみを実装してください。

操作に対して、データ管理が有効になります。

Flash Builder のコード生成（クライアントアプリケーション用）

Flash Builder では、リモートサービス操作にアクセスできるようにするためのクライアントコードが生成されます。コードが生成されるのは、次の各状況です。

- データサービスへの接続
- データとサービスビューのデータサービスの更新
- 操作の戻り値の型の設定
- ユーザーインターフェイスコントロールへのサービス操作のバインディング
- サービス操作のページングの有効化
- サービスのデータ管理の有効化
- イベントハンドラーまたはサービス呼び出しの生成

サービスクラス

データサービスに接続するには、サービスウィザードを使用します。サービスに接続すると、Flash Builder では、サービス操作にアクセスできるようにするための ActionScript クラスファイルが生成されます。

RemoteObject にアクセスするサービスの場合は、生成されたクラスによって RemoteObjectServiceWrapper クラスが拡張されます。通常、PHP、ColdFusion、BlazeDS、および LiveCycle Data Services を使用して実装されるサービスは、RemoteObject にアクセスします。

HTTP サービスの場合は、生成されたクラスによって HTTPServiceWrapper クラスが拡張されます。

Web サービスの場合は、生成されたクラスによって WebServiceWrapper クラスが拡張されます。

Flash Builder では、サービスウィザードに入力されたサービスの名前に基づいて、生成されたクラスファイルに名前が付けられます。このクラスは、デフォルトでは、プロジェクトのメインソースフォルダーに配置されます。通常、このフォルダーは src です。パッケージの名前は、サービス名に基づいて付けられます。例えば、Flash Builder では、EmployeeService クラスに対して、次の ActionScript クラスが生成されます。

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

スーパークラスには、EmployeeService クラスの実装が格納されています。

スーパークラスは、生成されたクラスなので、編集しないでください。スーパークラスを変更しても上書きされます。実装を変更すると、予期しない動作が発生する可能性があります。

この例では、EmployeeService.as を使用して、生成されたスーパークラスを拡張し、実装を追加しています。

関連項目

8 ページの「データサービスへの接続」

カスタムデータ型のクラス

多くのリモートデータサービスでは、サーバー側の型指定が用意されています。これらのサービスでは、複雑なデータがカスタムデータ型として返されます。

型指定されたデータを返さないリモートデータサービスに対しては、Flash Builder でクライアント側の型指定が用意されます。クライアント側の型指定では、Flash Builder の接続ウィザードを使用して、サービスから返される複雑なデータのデータ型の定義と設定を行います。例えば、従業員データベースレコードを返すサービスに対して、Employee というデータ型を定義して設定するとします。

Flash Builder では、サービスから返される各データ型の実装に対して ActionScript クラスが生成されます。Flash Builder では、このクラスを使用して、値オブジェクトが作成されます。次に、この値オブジェクトが、リモートサービスからのデータのアクセスに使用されます。

例えば、Employee データ型を含む EmployeeService クラス用に次の ActionScript クラスが生成されます。

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

スーパークラスには、EmployeeService と Employee データ型の実装がそれぞれ含まれています。

生成されたスーパークラスは編集しないでください。スーパークラスを変更しても上書きされます。実装を変更すると、予期しない動作が発生する可能性があります。

この例では、EmployeeService.as と Employee.as を使用して、生成されたスーパークラスを拡張し、実装を追加しています。

ユーザーインターフェイスコントロールへのサービス操作のバインディング

21 ページの「コントロールへのサービス操作のバインディング」に、サービス操作から返されるデータをユーザーインターフェイスコントロールにバインドする方法を示します。サービス操作をコントロールにバインドするときに、Flash Builder では次のコードが生成されます。

- CallResponder とサービスタグを含む Declarations タグ
- サービスコールを呼び出すイベントハンドラー
- 操作から返されたデータとコントロールの間のデータバインディング

Declarations タグ

Declarations タグとは、現在のクラスのデフォルトでない非表示のプロパティを宣言する MXML エレメントです。サービス操作をユーザーインターフェイスにバインドするときに、Flash Builder では、CallResponder とサービスタグを含む Declarations タグが生成されます。CallResponder と生成されたサービスクラスは、コンテナエレメントのプロパティで、通常このコンテナエレメントは Application タグです。

次の例は、リモートの EmployeeService へのアクセスを可能にする Declarations タグを示しています。

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

CallResponder

CallResponder は、サービスへのコールの実行結果を管理します。CallResponder には、サービスコールから返された Async トークンが設定されたトークンプロパティが格納されています。CallResponder には、最後に成功したサービスコールの結果が設定された lastResult プロパティも格納されています。CallResponder にイベントハンドラーを追加すれば、lastResult プロパティ経由で返されたデータにアクセスできるようになります。

Flash Builder で CallResponder が生成される際には、バインドされているサービス操作の名前に基づいて id プロパティが生成されます。次のコードサンプルは、EmployeeService の 2 つの操作の CallResponder を示しています。getAllItems() 操作は、DataGrid の creationComplete イベントハンドラーにバインドされています。削除操作は、DataGrid の選択アイテムにバインドされています。DataGrid には、作成直後の getAllItems() サービスコールから取得されたアイテムが表示されます。「Delete Item」Button コントロールは、DataGrid で選択したレコードをデータベースから削除します。

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

イベントハンドラー

サービスの操作をユーザーインターフェイスコンポーネントにバインドする際、Flash Builder では CallResponder のイベントハンドラーが生成されます。このイベントハンドラーは、操作の結果を管理します。ActionScript コードブロックにイベントハンドラーを作成して、ユーザーインターフェイスコンポーネントのプロパティからそのイベントハンドラーを参照することもできます。

一般に、List や DataGrid などのコントロールには、サービスから返された値を設定します。デフォルトでは、Flash Builder により、コントロールの creationComplete イベントハンドラーが生成されて、コントロールの作成直後に実行されます。その他のコントロールに対しては、Flash Builder により、コントロールのデフォルトイベントのハンドラーが生成されます。例えば、Button の場合は、Flash Builder により、Button のクリックイベント用のイベントが生成されます。

コントロールのイベントプロパティは、生成されたイベントハンドラーに設定されます。次の例は、DataGrid 用に生成された creationComplete イベントハンドラーを示しています。

```
<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>
```

Button など、ユーザーイベントに応答するコントロールのイベントハンドラーを生成できます。次の例は、DataGrid に値を設定する Button 用に生成されたイベントハンドラーを示しています。

```
<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)"/>
```

データバインディング

ユーザーインターフェイスを構築する際には、サービス操作をコントロールにバインドします。21 ページの「コントロールへのサービス操作のバインディング」を参照してください。

Flash Builder では、データを表示するユーザーインターフェイスコントロールに、サービス操作から返されたデータをバインドするコードが生成されます。

DataGrid コントロールに値を設定するために Flash Builder で生成されたコード例を次に示します。getAllItems() 操作では、Employee というカスタムデータ型に対して一連の従業員レコードが返されます。

DataGrid の dataProvider プロパティは、getAllItemsResult という CallResponder に保存されている結果にバインドされます。Flash Builder では、Employee レコードに対して返された各フィールドに対応する DataGridColumn で DataGrid が自動的に更新されます。各列の headerText プロパティと dataField プロパティは、Employee レコードで返されたデータに従って設定されます。

```
<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>
```

サービス操作のページングの有効化

ページングを有効にすると、生成されたサービスの実装が Flash Builder によって変更されます。ページングされたデータでデータコントロール（DataGrid や List など）に値を設定すると、データコントロールで表示可能なレコード数とデータベース内のレコード総数が Flash Builder によって判断されます。Flash Builder は、ページングを実装するために使用したサービス操作に引数としてこれらの値を渡します。

ページングを有効にした後に、クライアントアプリケーションコードを変更する必要はありません。

詳しくは、30 ページの「ページングの有効化」を参照してください。

サービスのデータ管理の有効化

Flash Builder では、データ管理とは、サーバー上のデータに対して一連の更新を同期することです。サービスから返されるカスタムデータ型に対するデータ管理を有効にできます。データ管理を有効にすると、サーバーを更新することなく、クライアントアプリケーション内の 1 つまたは複数の項目を変更できます。その後、1 回の操作でサーバーにすべての変更をコミットできます。サーバーのデータを更新しないで変更を元に戻すこともできます。31 ページの「データ管理の有効化」に、この機能の実装方法を示します。

データ管理を有効にすると、生成されたサービスクラスの実装とカスタムデータ型用に生成されたクラスが Flash Builder によって変更されます。Flash Builder で DataManager が生成されてこの機能が実装されます。

サーバーデータへの更新の同期

管理対象のデータ型に対してサービス操作を呼び出すと、クライアントアプリケーションに変更が反映されます。ただし、DataManager の commit() メソッドを呼び出すまでサーバーのデータを更新しないように指定することもできます。

データ管理が有効になっているサービスには autoCommit フラグがあります。autoCommit はデフォルトでは false に設定されています。

autoCommit フラグは、変更を即座にコミットするか、***service***.commit() が呼び出されるまで待つかを決定します。

autoCommit が false に設定されている場合、クライアントアプリケーションでサービスに加えられた更新はすべて、***service***.commit() が呼び出されるまでキャッシュに保留されます。変更を破棄するには、サービスの revertChanges() メソッドを呼び出します。

autoCommit が true に設定されている場合、更新は即座にサーバーに送信されます。revertChanges() を呼び出して変更を破棄することはできません。

deleteItemOnRemoveFromFill フラグは、削除されたアイテムをデータベースから即座に削除するかどうかを決定します。true に設定した場合、アイテムは ***service***.commit() が呼び出されるまで削除されません。

次のコードを使用すると、サーバーデータの更新に対するデータ管理同期が無効になります。管理対象の型のデータが変更されると、サーバー上のデータも即座に更新されます。

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

次のコードでは、データ管理機能を使用して、サーバーデータに更新を同期しています。管理対象の型のデータに加えられた変更は、サービスの commit() が呼び出されるまで更新されません。また、削除されたアイテムも、commit() が呼び出されるまでデータベースから削除されません。

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

変更の破棄

DataManager には、revertChanges() メソッドが用意されています。revertChanges() メソッドは、クライアントアプリケーション内で表示されているデータを、最後の commit 呼び出しの前にサーバーから取得された値に復元します。

クライアントアプリケーションで管理対象のデータ型に加えられた変更を破棄するには、commit() を呼び出す前に revertChanges() を呼び出します。

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

管理対象のデータ型に加えられた変更をコミットするには、commit() メソッドを呼び出します。

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

bookService インスタンスから commit() メソッドを直接呼び出すこともできます。サービスインスタンスから直接 commit メソッドを呼び出すと、管理対象のすべてのデータ型に対するすべての変更がコミットされます。

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

注意： サービスインスタンスから直接 revertChanges() を呼び出して、管理対象のすべてのデータ型に対する変更を破棄することはできません。このメソッドは、管理対象の特定のデータ型に対してのみ呼び出すことができます。

デフォルトのデータ管理の動作を無効にして、変更を破棄できないようにするには、autoCommit を true に設定します。例えば、bookService のインスタンスがあり、Book データ型に対するデータ管理が有効になっている場合は、autoCommit を true に設定します。

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

これにより、管理対象の型のデータが変更されると、サーバー上のデータも即座に更新されるようになります。

データサービスにアクセスするアプリケーションのデプロイ

開発環境からデプロイ環境にアプリケーションを移行する際には、様々な事項を考慮する必要があります。アプリケーションのデプロイプロセスはアプリケーション、アプリケーションの要件、およびデプロイ環境によって異なります。

例えば、企業従業員だけがアクセスできる社内 Web サイトでアプリケーションをデプロイするプロセスは、一般の Web サイトで同じアプリケーションをデプロイする場合とは異なります。

考慮すべき事項の概要については、Deploying applications を参照してください。ここには、Deployment checklist も掲載されています。このチェックリストでは、アプリケーションを実稼働環境にデプロイするときに一般的に発生するシステム設定上の問題について説明しています。また、このマニュアルには、デプロイ時によく発生する問題の解決に役立つトラブルシューティングのヒントもあります。

サービスへのアクセスをコーディングする際のベストプラクティス

Flash Builder ツールを使用すると、データベース内のデータにアクセスするためのクライアントコードを生成できます。この機能は PHP と ColdFusion の両方に対して用意されています。ただし、このコードはあくまでもプロトタイプです。セキュリティ保護されたアプリケーションを作成するためのテンプレートとしては使用しないでください。

デフォルトでは、このコードを使用すると、サーバーへのアクセス権を持つすべてのユーザーがデータベーステーブルに対して挿入、選択、更新、削除の操作を実行できます。生成されたコードを変更する場合、またはサービスにアクセスするコードを記述する場合に検討すべきベストプラクティスをいくつか紹介します。詳しくは、Data Services のセキュリティ保護を参照してください。

使用しない関数の削除

アプリケーションで使用する予定がない関数を削除またはコメントアウトします。

認証の追加

ユーザー認証を追加して、信頼できるユーザーのみがデータベース情報にアクセスできるようにします。

承認チェックの追加

認証が必要な場合は、承認チェックを追加します。アプリケーションへのアクセスをユーザーに認証した場合でも、そのユーザーが特定のクエリーを実行することを承認するかどうかを確認する必要があります。

例えば、選択操作をすべてのユーザーに許可し、削除操作を一部のユーザーのみに制限することができます。

また、ユーザー A が select クエリーを使用して自分自身の情報を取得することを承認する一方で、ユーザー B の情報へのアクセスを承認しないことも可能です。

データ検証

データ検証は必ず追加してください。例えば、insert 文に入力されたデータを検証して、不良データや悪意のあるデータがデータベースに保存されないようにします。

クライアント側の検証では、ユーザーから Web サーバーへの要求の送信を阻止できません。データ検証はハッカーの不正行為を防止し、保存される情報の品質を確保します。

取得するデータ量の制限

select メソッドでは、テーブル内のあらゆる情報を選択できます。これにより、膨大な量の情報がネットワーク経由で転送される場合があります。必要なデータのみを取得してください。

例えば、ユーザーテーブルに対して SELECT * を実行すると、ユーザー名とパスワードがネットワーク経由で返されます。

機密データに対する SSL 使用の検討

セキュリティ保護されたプロトコルを使用すると、送信する情報のプライバシーを確保できます。

アプリケーションのリリースバージョンを含むソースファイルの書き出し

アプリケーションのリリースビルドを書き出す場合、Flash Builder には「「ソースの表示」を有効にする」オプションが用意されています。このオプションを使用すると、アプリケーションを実装するソースファイルの内容をユーザーが表示できるようになります。サーバープロジェクトの場合、ソースファイルには services フォルダーが含まれ、このフォルダーにはサービス実装へのアクセスに必要なファイルが含まれています。

重要： サービスファイルに「ソースの表示」オプションを含める際には注意が必要です。サービスファイルではデータベースへのアクセスに関する詳細が公開されますが、これにはユーザー名やパスワードなどの機密情報が含まれる可能性があります。サービスに「ソースの表示」オプションが含まれている場合、起動されるアプリケーションにアクセスできるすべてのユーザーが機密データにもアクセスできることになります。

関連項目

Flex セキュリティ

セキュリティ保護されたサービスの作成

Adobe のマニュアルに記載されている例は、Flash Builder のコード生成機能を使用して作成されたチュートリアルやアプリケーションなども含め、基本的には見本を示すことを目的としています。これらの例は、クライアントアプリケーションからデータサービスへのアクセス方法を示しています。ただし、分かりやすさに重点を置いているため、データへのアクセスを保護するためのベストプラクティスは示されていません。

Flash Builder のマニュアルには、生成されたコードから作成したアプリケーションなど、様々な例が記載されています。これらの例は、信頼できる開発環境でデプロイする必要があります。信頼できる開発環境の例には、ローカルコンピューターや、ファイアウォールの内側の場所などがあります。セキュリティ対策を強化しない限り、ネットワークにアクセスできるすべての人がデータベースにアクセスできることになります。

サービスを作成する際のベストプラクティスには、次のようなものがあります。

- サービスのメソッドを呼び出す前にユーザー認証を行います。
- サービス認証を使用して、特定の処理を一部のユーザーのみに許可します。

  例えば、RemoteObject の呼び出しによって従業員データを変更できるアプリケーションがあると想定します。この場合は、RemoteObject 認証を使用して、管理職だけが従業員データを変更できるように設定できます。
- プログラムセキュリティを使用して、サービスへのアクセスを制限します。
- 宣言型セキュリティ制約をサービス全体に適用します。
- Web サービス（<mx:WebService>）または HTTP サービス（<mx:HTTPService>）にアクセスする場合は、次のいずれかの条件に該当して必要があります。
  - サービスの実装が呼び出し元のアプリケーションと同じドメインに存在していること。
  - サービスのホストシステムに、アプリケーションのドメインからのアクセスを明示的に許可する crossdomain.xml ファイルが存在していること。

関連項目

Flex セキュリティ

Data Services のセキュリティ保護

セキュリティ保護されたアプリケーションの作成

Adobe® Flash® Player は、Flash で構築されたアプリケーションを実行します。コンテンツは、SWF ファイル形式で正確に記述されたバイナリ形式の一連の命令として、Web プロトコルを介して Flash Player に配信されます。SWF ファイル自体は、通常はサーバーでホストされ、要求に応じてクライアントコンピューターにダウンロードされたり、表示されたりします。コンテンツのほとんどは、バイナリ ActionScript 命令で構成されています。ActionScript は Flash で使用される ECMA 標準ベースのスクリプト言語です。ActionScript には、クライアント側のユーザーインターフェイス要素の作成および操作と、データの処理を可能にする API が用意されています。

Flex のセキュリティモデルでは、クライアントとサーバーの両方が保護されます。セキュリティについては、次の 2 つの一般的な側面を考慮する必要があります。

- サーバーのリソースにアクセスするユーザーの承認および認証
- クライアント上のサンドボックス内で Flash Player を実行する

Flex は、あらゆる J2EE アプリケーションサーバーの Web アプリケーションセキュリティに対応しています。さらに、プリコンパイルされた Flex のアプリケーションを、基礎となるサーバーテクノロジーの認証および承認スキームと統合すると、ユーザーによるアプリケーションへのアクセスを防止できます。Flex フレームワークには、Web サービス、HTTP サービス、および EJB などのサーバーベースリソースへのアクセスを制御できるいくつかのセキュリティメカニズムが組み込まれています。

Flash Player はセキュリティサンドボックス内で動作するので、悪意のあるアプリケーションコードによるクライアントの乗っ取りを防止できます。

注意： Adobe AIR で実行される SWF コンテンツは、ブラウザーで実行されるコンテンツとは異なるセキュリティ規則に従います。詳しくは、AIR マニュアルの「AIR のセキュリティ」を参照してください。

セキュリティに関する様々なトピックへのリンクについては、Adobe Developer Connection で Security Topic Center を参照してください。

関連項目

Flex セキュリティ

第 3 章：データ中心型アプリケーションでのサービスの実装

Action Message Format（AMF）

Flex では、リモートオブジェクトサービスと AMF を使用して、ColdFusion、PHP、BlazeDS および LiveCycle Data Services で実装されたサービスにアクセスします。AMF には、データベースとクライアントアプリケーションの間でデータを交換するためのメッセージング機能が用意されています。

ColdFusion、BlazeDS および LiveCycle Data Services には、リモートオブジェクトサービス用の AMF フレームワークが用意されています。PHP で実装されているサービスの場合、Flash Builder では Zend AMF フレームワークが使用されます。

ColdFusion サービスと PHP サービスでは、サーバー側の型指定を使用できます。サーバー側の型指定の場合、返されるデータの型はサービスで定義されます。ただし、サーバーの実装で戻り値のデータ型が定義されていない場合は、Flash Builder でクライアント側の型指定が用意されます。Flash Builder ではサービスからのデータがサンプリングされるので、戻り値の型をクライアントアプリケーションで設定できます。

クライアント側の型指定とサーバー側の型指定

Flex の場合、クライアントアプリケーションは、データにアクセスして表示するメソッドで、サービス呼び出しから返されたデータのデータ型を使用します。

ただし、次のサービスの例では型指定のないデータが返されます。

- 43 ページの「ColdFusion サービスの実装」
- 50 ページの「PHP サービスの実装」
- 63 ページの「複数のソースからサービスを実装する例」

例えば、getAllEmployees() 操作では、データベースのレコードを表す、型指定のないオブジェクトの配列がサービスから返されます。Flash Builder には、クライアント側の型指定を可能にするツールが用意されています。Flash Builder のツールを使用し、返されるデータを分析して、データのカスタムデータ型を定義します。

従業員レコードから返されたオブジェクトに対しては、Employee というカスタムデータ型を定義します。レコードの各列が、Employee データ型のプロパティになります。

Employee というカスタムデータ型を使用すれば、返されるデータにクライアントアプリケーションでアクセスして正しく表示できます。

Flash Builder では、サーバー側の型指定を実装するサービスにもアクセスできます。サーバー側の型指定の例については、Flash Builder server-side type examples を参照してください。

ColdFusion サービスの実装

ColdFusion でサービスを実装する際には、ColdFusion コンポーネント（CFC）ファイルとしてサービスを実装します。各 CFC には、サービス操作を提供する関数が含まれます。

ColdFusion サービスは、Adobe ColdFusion® Builder™ などの任意の IDE で作成できます。Flash Builder には、ColdFusion ファイルの編集用に最適化されたエディターは用意されていません。ただし、Flash Builder で ColdFusion ファイルを開くと、システムで ColdFusion ファイルに関連付けられているアプリケーションが Flash Builder によって起動されます。

ColdFusion サービスの例

サービス操作の関数を含む ColdFusion コンポーネント（CFC）を作成すれば、基本的な ColdFusion サービスを実装できます。次の例（employeeService.cfc）は、2 つの関数を実装する EmployeeService を示しています。getAllIEmployees() 関数を使用すると、データベース内のすべての従業員レコードが返されます。getEmployees() 関数を使用すると、emp_no 引数に基づいて単一の従業員レコードが返されます。

この例は、クライアント側の型指定を示しています。サービスは、型指定のないデータを返します。Flash Builder はクライアント側の型指定を使用して、返されたデータを分析し、データ型を定義します。

その後の例は、ページングとデータ管理のサービスを実装する方法を示しています。

Flash Builder では、サーバー側の型指定を実装するサービスにもアクセスできます。43 ページの「クライアント側の型指定とサーバー側の型指定」を参照してください。

サーバー側の型指定の例は、このマニュアルの執筆時点ではまだ作成されていませんでした。サーバー側の型指定の例については、Flash Builder server-side type examples を参照してください。

ColdFusion で基本的なサービスを実装する例

この例は、ColdFusion で基本的なサービスを実装する方法を示しています。この例は、データベーステーブルにアクセスするときに Flash Builder で生成されるコードに基づいて作成されています。9 ページの「データベーステーブルからのサンプル ColdFusion サービスの生成」を参照してください。

この例では、クライアント側の型指定を実装します。43 ページの「クライアント側の型指定とサーバー側の型指定」を参照してください。

サーバー側の型指定の例については、Flash Builder server-side type examples を参照してください。

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護された ColdFusion サービスの作成については、ColdFusion のマニュアルのユーザーセキュリティについてを参照してください。

```
<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>
```

EmployeeService の重要な点は次のとおりです。

- employees データベースに接続し、データベース内の employees テーブルにアクセスします。
- オブジェクトの配列を返します。

  Flex フレームワークを使用してプログラミングするときには、サービスからデータのみが返されます。データのフォーマッティングとプレゼンテーションは、クライアントアプリケーションで処理します。このモデルは、返されるデータが HTML テンプレートでフォーマットされている従来の ColdFusion CFM アプリケーションとは異なります。

  Flex は、返されたレコードセットをオブジェクトの配列として処理します。各行は、データベースから取得されたレコードを表しています。データベースレコードの各列は、返されたオブジェクトのプロパティになります。これにより、クライアントアプリケーションは、一連のプロパティが設定されたオブジェクトとして、返されたデータにアクセスできるようになります。

  返されるオブジェクトのデータ型を設定する必要があります。43 ページの「クライアント側の型指定とサーバー側の型指定」を参照してください。
- ColdFusion サーバーには、エラー処理機能が用意されています。

  ColdFusion のエラー処理機能は、サービスをデバッグするときに役立ちます。ColdFusion Administrator で、詳細なデバッグ情報を提供するようにデバッグとロギングの設定を変更します。

  Flash Builder の「操作をテスト」インターフェイスに、ColdFusion サーバーから返された情報が表示されます。

  サービスのテストについて詳しくは、60 ページの「リモートサービスのデバッグ」を参照してください。
- cfqueryparam を使用してデータベースクエリーを作成します。

  cfqueryparam を使用すると、サーバーへの呼び出しに含まれる SQL インジェクション文を防御できます。詳しくは、ColdFusion のマニュアルで cfqueryparam によるセキュリティの強化を参照してください。
- サービス内の関数へのアクセスを許可する前に、ユーザーを認証します。

  サンプルコードでは、ユーザー認証の方法は示されていません。ユーザーセキュリティについては、ColdFusion のマニュアルのユーザーセキュリティについてを参照してください。

関連項目

25 ページの「データサービス操作のデータ型の設定」

8 ページの「ColdFusion サービスへのアクセス」

9 ページの「データベーステーブルからのサンプル ColdFusion サービスの生成」

ColdFusion でページングを実装する例

Flash Builder ツールでは、リモートサービスから取得されたデータのページングを実装できます。ページングとは、大きなデータセットをインクリメンタルに取得することです。

Flash Builder では、ページングを実装するために特定の関数シグニチャが必要です。次のコード例は、ページングされるデータ用の ColdFusion サービスを実装する 1 つの方法を示しています。

この EmployeeServicePaged の例は、データベーステーブルにアクセスするときに Flash Builder で生成されるコードに基づいて作成されています。9 ページの「データベーステーブルからのサンプル ColdFusion サービスの生成」を参照してください。

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。サンプルを使用すると、ネットワーク経由でサーバーにアクセスできるすべてのユーザーが、データベーステーブル内のデータにアクセスし、変更や削除を実行できます。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護された ColdFusion サービスの作成については、ColdFusion のマニュアルのユーザーセキュリティについてを参照してください。

```
<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>
```

EmployeeServicePaged サービスは、型指定のないデータを返します。Flash Builder ツールを使用して、getEmployees_Paged() の戻り値の型を設定します。戻り値の型を設定した後、getEmployees_Paged() 操作に対するページングを有効にします。

関連項目

44 ページの「ColdFusion サービスの例」

25 ページの「データサービス操作のデータ型の設定」

29 ページの「サーバーからのデータのアクセスの管理」

ColdFusion でデータ管理操作を実装する例

Flash Builder ツールでは、リモートサービスに対するデータ管理機能を実装できます。データ管理とは、クライアントアプリケーションからサーバー上のデータに更新を同期することです。

Flash Builder では、データ管理を実装するために特定の関数シグニチャの組み合わせが必要です。次のコード例は、データ管理用の ColdFusion を実装する 1 つの方法を示しています。

次の EmployeeServiceDM の例は、データベーステーブルにアクセスするときに Flash Builder で生成されるコードに基づいて作成されています。9 ページの「データベーステーブルからのサンプル ColdFusion サービスの生成」を参照してください。

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護された ColdFusion サービスの作成については、ColdFusion のマニュアルのユーザーセキュリティについてを参照してください。

```
<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>
```

EmployeeServiceDM サービスは、型指定のないデータを返します。Flash Builder ツールを使用して、getAllEmployeess() と getEmployees() の戻り値の型を設定します。これらの操作で返されるカスタムデータ型には Employee を使用します。

戻り値の型を設定した後、Employee データ型に対するデータ管理を有効にします。

関連項目

44 ページの「ColdFusion サービスの例」

25 ページの「データサービス操作のデータ型の設定」

29 ページの「サーバーからのデータのアクセスの管理」

Adobe ColdFusion Builder を使用した CFC の生成

Adobe® ColdFusion® Builder™ には Adobe CFC Generator が付属しています。CFC Generator を使用すると、一連のデータベーステーブルから ORM CFC または従来の CFC を生成できます。ColdFusion Builder で生成した CFC は、Flash Builder でデータサービスとして使用できます。Adobe CFC Generator では、サーバー側の型指定を実装するサービスが作成されます。

詳しくは、Adobe CFC Generator の使用を参照してください。

注意： ColdFusion のオブジェクトリレーショナルマッピング（ORM）では、オブジェクトモデルを使用して、リレーショナルデータベースのデータを格納および取得するためのマッピング方針を定義します。ColdFusion ORM を参照してください。

PHP サービスの実装

PHP でサービスを実装する際には、PHP のクラスとして実装するのが一般的です。PHP のクラスは必ずしもオブジェクト指向クラスである必要はありません。各クラスは、サービス操作を実現する関数ライブラリでもかまいません。

PHP サービスは、DreamWeaver や Zend Studio などの任意の編集環境で作成できます。Flash Builder には、PHP ファイルの編集用に最適化されたエディターは用意されていません。ただし、Flash Builder で PHP ファイルを開くと、システムで PHP ファイルに関連付けられているアプリケーションが Flash Builder によって起動されます。また、Flash Builder には、PHP ファイルの編集に使用できるプレーンテキストエディターも用意されています。

PHP で実装されたサービスへの AMF を使用したアクセス

PHP データサービスは、Action Message Format（AMF）を使用して利用できます。AMF は Flash クライアントと Web サーバーの間のメッセージング機能を提供します。Flash Builder では、Zend AMF フレームワークを使用して PHP データサービス用の AMF メッセージ機能を実装します。

Zend AMF については、Zend Framework プログラマ向けリファレンスを参照してください。

Zend Framework のインストールについては、19 ページの「Zend Framework のインストール」を参照してください。

Flash Builder for PHP で Zend を使用する方法については、Zend 社の Web サイトを参照してください。

注意： Flash Builder では Zend AMF フレームワークが使用されますが、PHP サービスを作成するときは、Zend コンポーネントを使用する必要はありません。Zend コンポーネントは Flash Builder で問題なく動作しますが、任意の PHP 開発環境を使用してサービスを作成することもできます。

PHP サービスの例

サービス操作の関数を含む PHP クラスファイルを作成すれば、基本的な PHP サービスを実装できます。次の例は、2 つの関数を実装する EmployeeService を示しています。

- getAllIEmployees()

  データベース内のすべての従業員レコードを取得します。

- getEmployeeByID($itemID)

  1 件の従業員レコードを返します。

この例は、クライアント側の型指定を示しています。サービスは、型指定のないデータを返します。Flash Builder はクライアント側の型指定を使用して、返されたデータを分析し、データ型を定義します。

その後の例は、ページングとデータ管理のサービスを実装する方法を示しています。

Flash Builder では、サーバー側の型指定を実装するサービスにもアクセスできます。43 ページの「クライアント側の型指定とサーバー側の型指定」を参照してください。

サーバー側の型指定の例は、このマニュアルの執筆時点ではまだ作成されていませんでした。サーバー側の型指定の例については、Flash Builder server-side type examples を参照してください。

PHP の基本的なサービスの例

この例は、PHP で基本的なサービスを実装する方法を示しています。この例は、データベーステーブルにアクセスするときに Flash Builder で生成されるコードに基づいて作成されています。11 ページの「データベーステーブルからのサンプル PHP サービスの生成」を参照してください。

この例は、クライアント側の型指定を示しています。43 ページの「クライアント側の型指定とサーバー側の型指定」を参照してください。

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護されたサービスを作成する方法については、39 ページの「データサービスにアクセスするアプリケーションのデプロイ」を参照してください。

```
<?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);
        }
    }
}


?>
```

EmployeeService の重要な点は次のとおりです。

- employees データベースに接続して、localhost のポート 3306 でアクセスします。データベースの employees テーブルにアクセスします。
- サービスに接続し、データベースのテーブルにアクセスするためのクラス変数を提供します。

  これらの変数は、クラスの関数で使用できます。

  これらの変数の値は、それぞれのシステムの値に置き換える必要があります。
- クライアントアプリケーションにオブジェクトの配列を返します。

  Flex フレームワークを使用してプログラミングするときには、サービスからデータのみが返されます。データのフォーマッティングとプレゼンテーションは、クライアントアプリケーションで処理します。

  このモデルは、返されるデータが HTML テンプレートでフォーマットされている従来の PHP サービスとは異なります。
- getEmployeesByID($itemID) 関数を使用すると、入力パラメーターがデータ型にバインドされます。

  変数の数と文字列型の長さは、文のパラメーターと一致している必要があります。prepare 文の '?' は、パラメーターのプレースホルダーです。

  mysqli では次の型が認識されます。

  - integer (i)
  - double (d)
  - string (s)
  - blob (b)
- 結果をバインドして、オブジェクトの配列（$row[]）を作成します。

  Flex は、レコードセットをオブジェクトの配列として処理します。各オブジェクトは、データベースから取得されたレコードを表しています。データベースレコードの各列は、返されたオブジェクトのプロパティになります。これにより、クライアントアプリケーションは、一連のプロパティが設定されたオブジェクトとして、返されたデータにアクセスできるようになります。

  サーバーでは戻り値のデータ型が定義されないため、返されるオブジェクトのデータ型を構成する必要があります。43 ページの「クライアント側の型指定とサーバー側の型指定」を参照してください。
- データベースへの接続を初期化するコンストラクター関数を提供します。

- mysqli prepare 文を使用してデータベースクエリーを作成します。

  prepare 文を使用すると、サーバーへの呼び出しに含まれる SQL インジェクション文を防御できます。この文が用意されるまで、サーバー上で文は実行されません。

- サービス内の関数へのアクセスを許可する前に、ユーザーを認証します。

  サンプルコードでは、ユーザー認証の方法は示されていません。ユーザーセキュリティについては、ColdFusion のマニュアルのユーザーセキュリティについてを参照してください。この ColdFusion マニュアルに記載されているユーザー認証と承認のセキュリティ原則は、PHP サービスにもあてはまります。

- エラーが発生すると例外をスローします。

  例外で提供される情報は、サービスの実装をデバッグするときに役立ちます。Flash Builder の「操作をテスト」インターフェイスに、例外から返された情報が表示されます。

  サービスのテストについて詳しくは、60 ページの「リモートサービスのデバッグ」を参照してください。

- EmployeeService.php というファイル名が、サービスの PHP クラス名と一致します。

  ファイル名とクラス名が一致しないと、サービスへのアクセス時にエラーが発生します。

関連項目

25 ページの「データサービス操作のデータ型の設定」

10 ページの「PHP サービスへのアクセス」

11 ページの「データベーステーブルからのサンプル PHP サービスの生成」

PHP でページングを実装する例

Flash Builder ツールでは、リモートサービスから取得されたデータのページングを実装できます。ページングとは、大きなデータセットをインクリメンタルに取得することです。

Flash Builder では、ページングを実装するために特定の関数シグニチャが必要です。次のコード例は、ページングされるデータ用の PHP サービスを実装する 1 つの方法を示しています。

この例は、データベーステーブルにアクセスするときに Flash Builder で生成されるコードに基づいて作成されています。11 ページの「データベーステーブルからのサンプル PHP サービスの生成」を参照してください。

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護されたサービスを作成する方法については、39 ページの「データサービスにアクセスするアプリケーションのデプロイ」を参照してください。

```
<?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);
        }
    }
}
?>
```

EmployeeServicePaged サービスは、型指定のないデータを返します。Flash Builder ツールを使用して、getEmployees_Paged() の戻り値の型を設定します。戻り値の型を設定した後、getEmployees_Paged() 操作に対するページングを有効にします。

関連項目

50 ページの「PHP サービスの例」

25 ページの「データサービス操作のデータ型の設定」

29 ページの「サーバーからのデータのアクセスの管理」

PHP でデータ管理を実装する例

Flash Builder ツールでは、リモートサービスに対するデータ管理機能を実装できます。データ管理とは、クライアントアプリケーションからサーバー上のデータに更新を同期することです。

Flash Builder では、データ管理を実装するために特定の関数シグニチャの組み合わせが必要です。次のコード例は、データ管理用の PHP サービスを実装する 1 つの方法を示しています。

この例は、データベーステーブルにアクセスするときに Flash Builder で生成されるコードに基づいて作成されています。11 ページの「データベーステーブルからのサンプル PHP サービスの生成」を参照してください。

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護されたサービスを作成する方法については、39 ページの「データサービスにアクセスするアプリケーションのデプロイ」を参照してください。

```
<?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, '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);
}

/**
```

```
     * 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);
        }
    }
}
?>
```

EmployeeServiceDM サービスは、型指定のないデータを返します。Flash Builder ツールを使用して、getAllEmployeess() と getEmployeesByID() の戻り値の型を設定します。これらの操作で返されるカスタムデータ型には Employee を使用します。

戻り値の型を設定した後、Employee データ型に対するデータ管理を有効にします。

関連項目

50 ページの「PHP サービスの例」

25 ページの「データサービス操作のデータ型の設定」

29 ページの「サーバーからのデータのアクセスの管理」

リモートサービスのデバッグ

リモートサービスにアクセスするアプリケーションをデバッグするには、いくつかの方法があります。

- Flash Builder の操作をテストビュー

  Flash Builder の操作をテストビューでは、サービス操作を呼び出し、返されたデータを表示できます。操作をテストビューには、サービスで表示されるすべてのエラーメッセージが表示されます。

- サーバー側のスクリプト

  より詳細にサービスをデバッグするには、サーバーコードをテストして出力ストリーム情報をログファイルに書き込むスクリプトを作成します。

- Flash Builder のネットワークモニター

  サービスにアクセスするアプリケーションを Flash Builder で作成した後は、ネットワークモニターを使用します。ネットワークモニターを使用すると、サーバーとクライアントの間で送信されるデータを確認できます。

Flash Builder の操作をテストビュー

Flash Builder の操作をテストビューを使用すると、サービスから操作を呼び出して、操作の結果を表示できます。結果には、サービスから返されたエラーメッセージも含まれます。

操作をテストビューを使用すると、独自に作成したサービスや、HTTP サービスまたは Web サービスで使用可能なサービスの操作から返されたデータを表示できます。

サービス操作のテスト

この手順では、テストするサービスが既に作成されているか、HTTP サービスまたは Web サービスにアクセスできるものと想定しています。

1. Flash Builder のデータとサービスビューで、テストするサービス操作にナビゲートします。
2. サービス操作のコンテキストメニューから、「操作をテスト」を選択します。
3. （オプション）操作をテストビューで、「認証が必要です」を選択して、サービスへのログイン情報を入力します。
4. パラメーターが必要な操作の場合は、「値を入力」フィールドをクリックして、パラメーターの値を入力します。

   パラメーターで複雑な型を使用する必要がある場合は、「値を入力」フィールド内の「...」ボタンをクリックして、JSON 表記法を使用できるエディターを開きます。JSON 表記法を使用してパラメーターの値を入力します。
5. 「テスト」をクリックして、操作の結果を表示します。

サーバーコードのテスト用スクリプト

Flash Builder でサーバーに接続を試みる前に、テストスクリプトを使用してサーバーコードの表示とデバッグを行います。テストスクリプトには、次のような利点があります。

- Web ブラウザーでテスト結果を確認できます。

  コードを変更したら、ブラウザーのページを更新するだけで結果を確認できます。
- 出力ストリームに結果をエコーまたはプリントできます。AMF からは直接実行できません。
- エラー表示が適切にフォーマットされます。通常は、AMF を使用してキャプチャーされたエラーよりも完全です。

ColdFusion スクリプト

次に示す tester.cfm スクリプトを使用して、関数の呼び出しをダンプします。

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

tester2.cfm では、呼び出すメソッドと引数を URL で指定します。

```
<!--- 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>
```

例えば、EmployeeService の getItemID() メソッドを、次の URL で呼び出します。

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

tester3.cfm では、操作の呼び出しを記録するログを書き込み、cfdump を使用して入力引数をダンプします。

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

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

PHP スクリプト

次に示す tester.php スクリプトを使用して、関数の呼び出しをダンプします。

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

次のコードを PHP サービスに追加して、コード実行中にメッセージをログに記録します。

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

次のコードを PHP サービスに追加して、ログファイルへのダンプを有効にします。

```
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);
```

ネットワークモニター

ネットワークモニターは、Flash Builder の Flex デバッグパースペクティブから使用できます。データのモニターに使用する前に、モニターを有効化します。ネットワークモニターの有効化と使用について詳しくは、データサービスにアクセスするアプリケーションの監視を参照してください。

複数のソースからサービスを実装する例

通常、アプリケーションは複数のソースからのデータにアクセスし、データを関連付けた結果をアプリケーションで表示します。この例は、employees データベースに含まれる次の 3 つのテーブルからのデータを関連付ける方法を示しています。

- Departments

  各レコードには、部署番号と部署名のフィールドが含まれています。

- Dept_emp

  各レコードには、emp_no、dept_no、from_date、to_date フィールドが含まれています。

- Employees

  各レコードには、emp_no、birth_date、first_name、last_name、gender、hire_date フィールドが含まれています。

サンプルアプリケーションには 2 つの DataGrid があり、1 つは Departments 用で、もう 1 つは Employees 用です。

Departments には、すべての部署が表示されます。部署を 1 つ選択すると、その部署のすべての従業員が Employees DataGrid に表示されます。

Employees DataGrid で従業員を 1 名選択すると、フォームの値が設定され、その従業員レコードを更新できるようになります。

サービスの作成

この例では、1 つのサービスのみを作成します。このサービスには、次の操作が含まれます。

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

EmployeeService（PHP）

EmployeeService.php は、単一の関数を含むサービスを実装します。GetEmployeesByID() は、部署 ID を引数として取り、指定された部署のすべての従業員を返します。また、この関数は、従業員がその部署に配属された日付と、その部署から異動した日付も返します。GetEmployeesByDept() は、次の SQL クエリーを実行します。

```
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
```

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護されたサービスを作成する方法については、39 ページの「データサービスにアクセスするアプリケーションのデプロイ」を参照してください。

```
<?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, '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);
    }


    /**
     * 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 は、単一の関数を含むサービスを実装します。GetEmployeesByID() は、部署 ID を引数として取り、指定された部署のすべての従業員を返します。また、この関数は、従業員がその部署に配属された日付と、その部署から異動した日付も返します。GetEmployeesByDept() は、次の SQL クエリーを実行します。

```
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
```

重要： サンプルサービスは、あくまでもプロトタイプです。サンプルサービスは、信頼できる開発環境のみで使用してください。このサービスをデプロイする前に、セキュリティを強化し、アクセスを適切に制限する必要があります。セキュリティ保護された ColdFusion サービスの作成については、ColdFusion のマニュアルのユーザーセキュリティについてを参照してください。

```
<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>?>
```

サーバープロジェクトへのサービスのインポート

1 Flash Builder で、Associations という名前の Flex プロジェクトを作成します。

（PHP）プロジェクトを作成するときに、アプリケーションサーバーの種類として PHP を指定します。

（PHP）プロジェクトを作成すると、PHP 設定の Web ルートとして設定されたフォルダー内に出力フォルダーが作成されます。PHP_Associations プロジェクトのデフォルトの名前は、PHP_Associations-debug です。

（ColdFusion）プロジェクトを作成するときに、アプリケーションサーバーの種類として ColdFusion を指定します。次に ColdFusion Flash リモーティングを指定します。

2 （PHP）PHP_Associations-debug の中に、services というフォルダーを作成します。EmployeeService.php を services フォルダーにコピーします。

3 （ColdFusion）ColdFusion 設定の Web ルートに Associations というフォルダーを作成します。EmployeeService.chc を Associations フォルダーにコピーします。

4 EmployeeService をプロジェクトにインポートします。

PHP_Associations が Flash Builder でアクティブなプロジェクトであることを確認します。

データ／ PHP に接続を選択します。PHP クラスを指定するために、services フォルダーを参照して、EmployeeService.php を選択します。「終了」をクリックします。

詳しくは、10 ページの「PHP データサービスへの接続」を参照してください。

5 EmployeeService の操作の戻り値の型を設定します。

- DepartmentService

  getAllDepartments() のコンテキストメニューから、「戻り値の型を設定」を選択します。

  「次へ」をクリックして、戻り値の型を自動検出します。

  カスタムの戻り値の型として **Department** を指定します。「終了」をクリックします。

- EmployeeService

  getEmployeesByDept() で、「戻り値の型を設定」を選択します。

  「次へ」をクリックして、戻り値の型を自動検出します。

  パラメーターの値として **d007** を指定します。「次へ」をクリックします。

  カスタムの戻り値の型として **Employee** を指定します。「終了」をクリックします。

詳しくは、25 ページの「データサービス操作のデータ型の設定」を参照してください。

第 4 章：サーバー側のデータへのアクセス

Adobe® Flex® のデータアクセスコンポーネントは、リモートプロシージャコールを使用して、PHP、Adobe ColdFusion、Microsoft ASP.NET などのサーバー環境と通信します。これらのデータアクセスコンポーネントは、Adobe Flex フレームワークで作成されたクライアントアプリケーションにデータを提供し、バックエンドのデータソースにデータを送信します。データアクセスコンポーネントの概要については、4 ページの「データアクセスコンポーネント」を参照してください。

HTTPService コンポーネントの使用

HTTPService コンポーネントは、PHP ページ、ColdFusion ページ、Javaserver Pages（JSP）、Java サーブレット、Ruby on Rails、Microsoft ASP ページなど、任意のサーバー側テクノロジーとともに組み合わせて使用できます。さらに、REST ベースの Web サービスへのアクセスにも HTTPService を使用します。

HTTPService コンポーネントについての API リファレンス情報は、mx.rpc.http.mxml.HTTPService を参照してください。

PHP および SQL データの操作

HTTPService コンポーネントを PHP や SQL データベース管理システムとともに使用すると、データベースクエリの結果をアプリケーションで表示できます。また、これらのコンポーネントを使用して、データの挿入、更新、削除などの操作をデータベースに対して実行することもできます。GET または POST で PHP ページを呼び出して、データベースクエリを実行できます。その後、クエリ結果データを XML 構造にフォーマットし、HTTP 応答でアプリケーションに XML 構造を返すことができます。アプリケーションに返された結果は、1 つまたは複数のユーザーインターフェイスコントロールで表示できます。

MXML コード

次の例のアプリケーションは、POST メソッドで PHP ページを呼び出しています。PHP ページは、users という名前の MySQL データベーステーブルをクエリします。その後、クエリ結果を XML としてフォーマットし、XML をアプリケーションに返します。アプリケーションでは、XML は DataGrid コントロールの dataProvider プロパティにバインドされていて、DataGrid コントロールに表示されます。また、アプリケーションは新しいユーザーのユーザー名と電子メールアドレスを PHP ページに送信し、PHP ページはユーザーデータベーステーブルへの挿入を実行します。

```
<?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>
```

HTTPService の send() メソッドが、PHP ページの呼び出しを行います。この呼び出しは、MXML ファイルのスクリプトブロックの send_data() メソッドで行われます。

HTTPService コンポーネントの resultFormat プロパティは object に設定されているので、データは ActionScript オブジェクトのグラフとしてアプリケーションに返送されます。これは、resultFormat プロパティのデフォルト値です。または、e4x の resultFormat を使用してデータを XMLList オブジェクトとして返すこともできます。このオブジェクトに対しては、ECMAScript for XML（E4X）操作を実行できます。resultFormat プロパティを e4x に切り替えるには、MXML コードに対して次のような小さい変更を行う必要があります。

注意：結果のフォーマットが e4x の場合は、DataGrid にバインドするときのドット表記に XML 構造のルートノードを含めないようにします。

この例で返される XML には、名前空間の情報は含まれません。名前空間を含む XML の操作については、115 ページの「結果を e4x 結果形式の XML として処理する方法」を参照してください。

```
...
<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}">
...
```

e4x の結果フォーマットを使用するときは、オプションとして、lastResult プロパティを XMLListCollection オブジェクトにバインドし、さらにそのオブジェクトを DataGrid.dataProvider プロパティにバインドできます。次のコードスニペットはこれを示したものです。

```
<fx:Declarations>
...
    <mx:XMLListCollection id="xc"
         source="{userRequest.lastResult.user}"/>
...
</fx:Declarations>
...
    <mx:DataGrid id="dgUserRequest" x="22" y="128" dataProvider="{xc}">
...
```

MySQL データベーススクリプト

このアプリケーションの PHP コードは、sample という名前の MySQL データベースの users という名前のデータベーステーブルを使用します。次の MySQL スクリプトでテーブルを作成します。

```
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 コード

このアプリケーションは次の PHP ページを呼び出します。この PHP コードは、SQL データベースの挿入とクエリを実行し、XML 構造でアプリケーションにクエリ結果を返します。

```
<?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)
?>
```

ColdFusion および SQL データの操作

HTTPService コンポーネントを ColdFusion ページや SQL データベース管理システムとともに使用すると、データベースクエリの結果をアプリケーションで表示できます。また、これらのコンポーネントを使用して、データの挿入、更新、削除などの操作をデータベースに対して実行することもできます。まず、GET または POST で ColdFusion ページを呼び出して、データベースクエリを実行します。その後、クエリ結果データを XML 構造にフォーマットし、HTTP 応答でアプリケーションに XML 構造を返します。アプリケーションに返された結果は、ユーザーインターフェイスコントロールで表示できます。

MXML コード

次の例のアプリケーションは、POST メソッドで ColdFusion ページを呼び出しています。ColdFusion ページは、users という名前の MySQL データベーステーブルをクエリします。その後、クエリ結果を XML としてフォーマットし、XML をアプリケーションに返します。アプリケーションでは、XML は DataGrid コントロールの dataProvider プロパティにバインドされていて、DataGrid コントロールに表示されます。また、アプリケーションは新しいユーザーのユーザー名と電子メールアドレスを ColdFusion ページに送信し、ColdFusion ページはユーザーデータベーステーブルへの挿入を実行します。

```
<?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>
```

HTTPService の send() メソッドが、ColdFusion ページの呼び出しを行います。この呼び出しは、MXML ファイルのスクリプトブロックの send_data() メソッドで行われます。

HTTPService コンポーネントの resultFormat プロパティは object に設定されているので、データは ActionScript オブジェクトのグラフとしてアプリケーションに返送されます。これは、resultFormat プロパティのデフォルト値です。または、e4x の結果フォーマットを使用してデータを XMLList オブジェクトとして返すこともできます。このオブジェクトに対しては、ECMAScript for XML（E4X）操作を実行できます。resultFormat プロパティを e4x に切り替えるには、MXML コードに対して次のような小さい変更を行う必要があります。

注意： 結果のフォーマットが e4x の場合は、DataGrid にバインドするときのドット表記に XML 構造のルートノードを含めないようにします。

この例で返される XML には、名前空間の情報は含まれません。名前空間を含む XML の操作については、115 ページの「結果を e4x 結果形式の XML として処理する方法」を参照してください。

```
...
<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}">
...
```

e4x の結果フォーマットを使用するときは、オプションとして、lastResult プロパティを XMLListCollection オブジェクトにバインドし、さらにそのオブジェクトを DataGrid の dataProvider プロパティにバインドできます。次のコードスニペットはこれを示したものです。

```
<fx:Declarations>
...
    <mx:XMLListCollection id="xc"
         source="{userRequest.lastResult.user}"/>
...
</fx:Declarations>
...
    <mx:DataGrid id="dgUserRequest" x="22" y="128" dataProvider="{xc}">
...
```

SQL スクリプト

このアプリケーションの ColdFusion コードは、sample という名前の MySQL データベースの users という名前のデータベーステーブルを使用します。次の MySQL スクリプトでテーブルを作成します。

```
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 コード

「ColdFusion および SQL データの操作」の節に掲載されているアプリケーションは、次の ColdFusion アプリケーション（returncfxml.cfm）を呼び出します。この ColdFusion コードは、SQL データベースの挿入とクエリを実行し、アプリケーションにクエリ結果を返します。ColdFusion ページでは、cfquery タグを使用して、データベースにデータを挿入し、データベースをクエリします。また、cfxml タグを使用して、クエリ結果を XML 構造にフォーマットします。

```
<!--- 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>
```

Javaserver Pages の操作

Flex HTTPService コンポーネントを JSP ページや SQL データベース管理システムとともに使用すると、データベースクエリの結果をアプリケーションで表示できます。また、これらのコンポーネントを使用して、データの挿入、更新、削除などの操作をデータベースに対して実行することもできます。まず、GET または POST で JSP ページを呼び出して、データベースクエリを実行します。その後、クエリ結果データを XML 構造にフォーマットし、HTTP 応答でアプリケーションに XML 構造を返します。アプリケーションに返された結果は、ユーザーインターフェイスコントロールで表示できます。

MXML コード

次の例のアプリケーションは、SQL データベースからデータを取得する JSP ページを呼び出します。その後、データベースクエリ結果を XML としてフォーマットし、XML をアプリケーションに返します。アプリケーションでは、XML は DataGrid コントロールの dataProvider プロパティにバインドされていて、DataGrid コントロールに表示されます。

```
<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>
```

HTTPService の send() メソッドが、JSP ページの呼び出しを行います。この呼び出しは、MXML ファイルの Button の click イベントで行われます。

HTTPService コンポーネントの resultFormat プロパティは object に設定されているので、データは ActionScript オブジェクトのグラフとしてアプリケーションに返送されます。これは、resultFormat プロパティのデフォルト値です。または、e4x の結果フォーマットを使用してデータを XMLList オブジェクトとして返すこともできます。このオブジェクトに対しては、ECMAScript for XML（E4X）操作を実行できます。resultFormat プロパティを e4x に切り替えるには、MXML コードに対して次のような小さい変更を行う必要があります。

注意： 結果のフォーマットが e4x の場合は、DataGrid にバインドするときのドット表記に XML 構造のルートノードを含めないようにします。

この例で返される XML には、名前空間の情報は含まれません。名前空間を含む XML の操作については、115 ページの「結果を e4x 結果形式の XML として処理する方法」を参照してください。

```
...
    <s:HTTPService id="srv" url="catalog.jsp" resultFormat="e4x"/>
...
    <mx:DataGrid dataProvider="{srv.lastResult.product}" width="100%" height="100%"/>
```

e4x の結果フォーマットを使用するときは、オプションとして、lastResult プロパティを XMLListCollection オブジェクトにバインドし、さらにそのオブジェクトを DataGrid.dataProvider プロパティにバインドできます。

```
<fx:Declarations>
...
    <mx:XMLListCollection id="xc"
         source="{userRequest.lastResult.user}"/>
...
</fx:Declarations>
...
    <mx:DataGrid id="dgUserRequest" x="22" y="128" dataProvider="{xc}">
...
```

JSP コード

次に示す例は、このアプリケーションで使用される JSP ページです。この JSP ページは、データベースを直接呼び出しません。データは ProductService という名前の Java クラスから取得します。このクラスは、個別の製品を表す Product という名前の Java クラスを使用します。

```
<%@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>
```

ActionScript での HTTP サービスの呼び出し

次の例は、ActionScript スクリプトブロック内の HTTP サービス呼び出しを示しています。useHTTPService() メソッド呼び出しでは、サービスを宣言し、宛先を設定し、result および fault イベントリスナーを設定して、サービスの send() メソッドを呼び出しています。

```
<?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>
```

WebService コンポーネントの使用

Flex フレームワークで構築されたアプリケーションは、URL として使用可能な Web Services Description Language 1.1（WSDL 1.1）ドキュメントでインターフェイスを定義する SOAP ベースの Web サービスとやり取りできます。WSDL は、Web サービスで認識できるメッセージ、それらのメッセージへの応答の形式、Web サービスがサポートするプロトコル、およびメッセージの送信先を記述するための標準形式です。Flex Web サービス API は、通常、Simple Object Access Protocol（SOAP）1.1、XML Schema 1.0（バージョン 1999、2000 および 2001）、WSDL 1.1 RPC エンコード、RPC リテラル、ドキュメント / リテラル（bare および wrapped スタイルパラメーター）をサポートします。最も一般的な 2 種類の Web サービスでは、リモートプロシージャコール（RPC）エンコードまたはドキュメント / リテラルの SOAP バインディングを使用します。「エンコード」および「リテラル」という用語は、サービスで使用される WSDL から SOAP へのマッピングの種類を示します。

Flex では、SOAP メッセージ形式の Web サービス要求および結果がサポートされています。SOAP で提供されている XML ベース形式の定義を使用すると、Flex で構築されたアプリケーションなどの Web サービスクライアントと Web サービスの間で構造化および型指定された情報を交換できます。

Adobe® Flash® Player は、セキュリティサンドボックス内で動作します。このセキュリティサンドボックスでは、Flex で作成されたアプリケーションや、Flash で作成されたその他のアプリケーションが HTTP でアクセスできるリソースが制限されます。Flash で作成されたアプリケーションから HTTP でアクセスできるのは、同じプロトコルで提供されている同じドメイン内のリソースのみです。Web サービスは通常リモートの場所からアクセスされるため、これによって問題が発生する場合があります。LiveCycle Data Services と BlazeDS で使用可能なプロキシサービスは、リモートの Web サービスに対する要求を受け取り、その要求をリダイレクトして、応答をクライアントに返します。

LiveCycle Data Services または BlazeDS を使用していない場合、アプリケーションと同じドメインにある Web サービスにはアクセス可能ですが、他のドメインにあるサービスにアクセスするには、RPC サービスをホストしている Web サーバーに、アプリケーションのドメインからのアクセスを許可する crossdomain.xml（クロスドメインポリシー）ファイルをインストールする必要があります。

WebService コンポーネントについての API リファレンス情報は、mx.rpc.soap.mxml.WebService を参照してください。

WebService アプリケーションのサンプル

次のサンプルコードのアプリケーションでは、WebService コンポーネントを使用して Web サービス操作を呼び出しています。

MXML コード

次の例のアプリケーションは、users という名前の SQL データベーステーブルをクエリしてデータをアプリケーションに返す Web サービスを呼び出しています。アプリケーションでは、データは DataGrid コントロールの dataProvider プロパティにバインドされ、DataGrid コントロールに表示されます。また、アプリケーションは新しいユーザーのユーザー名と電子メールアドレスを Web サービスに送信し、Web サービスはユーザーデータベーステーブルへの挿入を実行します。Web サービスのバックエンドの実装は ColdFusion コンポーネントです。93 ページの「RemoteObject コンポーネントの使用」では、同じ ColdFusion コンポーネントがリモートオブジェクトとしてアクセスされています。

```
<?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 ドキュメント

次の例では、Web サービスの API を定義する WSDL ドキュメントを示します。

```
<?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>
```

ActionScript での Web サービスの呼び出し

次の例は、ActionScript スクリプトブロック内の Web サービス呼び出しを示しています。useWebService() メソッドを呼び出すと、サービスが宣言され、宛先が設定され、WSDL ドキュメントが取得されて、サービスの echoArgs() メソッドが呼び出されます。

注意：ActionScript で WebService コンポーネントを宣言するときは、WebService.loadWSDL() メソッドを呼び出します。

```
<?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>
```

予約されている操作名

WebService の操作には、サービス変数名を付けるだけでアクセスできます。ただし、操作名がサービスで定義されているメソッドと一致する場合は、名前付けの競合が発生する可能性があります。WebService コンポーネントの ActionScript で次のメソッドを使用することで、指定した名前の操作を返すことができます。

```
public function getOperation(name:String):Operation
```

WSDL ドキュメントの読み取り

WSDL ドキュメントは、Web ブラウザー、単純なテキストエディター、XML エディター、または WSDL ドキュメントを読みやすい形式で表示するユーティリティが組み込まれている Adobe Dreamweaver などの開発環境で表示できます。

WSDL ドキュメントには、次の表で説明するタグが含まれています。

タグ	説明
<binding>	Flex で作成されたアプリケーションなどのクライアントが Web サービスとの通信に使用するプロトコルを指定します。SOAP、HTTP GET、HTTP POST、MIME に対するバインディングが存在します。Flex は SOAP のバインディングのみをサポートします。
<fault>	メッセージ処理に問題が発生した結果として返されるエラーの値を指定します。
<input>	Flex で作成されたアプリケーションなどのクライアントが Web サービスに送信するメッセージを指定します。
<message>	WebService 操作で転送するデータを定義します。
<operation>	<input> タグ、<output> タグおよび <fault> タグの組み合わせを定義します。
<output>	Web サービスが Flex で作成されたアプリケーションなどの Web サービスクライアントに送信するメッセージを指定します。
<port>	バインディングとネットワークアドレスとの関連付けを指定する Web サービスエンドポイントを指定します。
<portType>	Web サービスで提供される 1 つまたは複数の操作を定義します。
<service>	<port> タグのコレクションを定義します。各サービスは 1 つの <portType> タグにマップされ、その <portType> タグで、操作にアクセスするための様々な方法を指定します。
<types>	Web サービスのメッセージで使用するデータ型を定義します。

RPC 指向およびドキュメント指向の操作

WSDL ファイルでは、RPC 指向またはドキュメント指向（ドキュメント / リテラル）の操作を指定できます。Flex は、どちらの操作スタイルもサポートします。

RPC 指向の操作を呼び出す場合、アプリケーションは、操作とそのパラメーターを指定する SOAP メッセージを送信します。ドキュメント指向の操作を呼び出す場合は、アプリケーションは XML ドキュメントを含む SOAP メッセージを送信します。

WSDL ドキュメントの各 <port> タグには、次の例に示すように、特定の <soap:binding> タグの名前を指定する binding プロパティが設定されています。

```
<binding name="InstantMessageAlertSoap" type="s0:InstantMessageAlertSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
        style="document"/>
```

関連付けられた <soap:binding> タグの style プロパティによって、操作のスタイルが決まります。この例では、スタイルは document です。

次の例に示すように、サービス内のどの操作でも、同じスタイルを指定したり、サービスに関連付けられているポート用に指定されているスタイルを上書きしたりできます。

```
<operation name="SendMSN">
    <soap:operation soapAction="http://www.bindingpoint.com/ws/imalert/
        SendMSN" style="document"/>
```

ステートフル Web サービス

Flex では、Java サーバーセッションを使用して、cookie にセッション情報を保存する Web サービスエンドポイントの状態が管理されます。この機能は、アプリケーションと Web サービスの間に介在するかたちで機能します。エンドポイントからアプリケーションに渡されるすべてのものに、エンドポイントの ID が追加されます。エンドポイントからセッション情報を送信すると、アプリケーションがそれを受信します。この機能には設定の必要はありません。プロキシサービスを使用する際に RTMP チャンネルを使用する宛先では、この機能はサポートされません。

SOAP ヘッダーの操作

SOAP ヘッダーは、SOAP エンベロープ内のオプションのタグであり、通常は認証情報などアプリケーションに固有の情報が含まれます。

Web サービス要求への SOAP ヘッダーの追加

Web サービスによっては、操作を呼び出すときに SOAP ヘッダーを同時に渡す必要があります。

SOAP ヘッダーは、WebService オブジェクトまたは Operation オブジェクトの addHeader() メソッドまたは addSimpleHeader() メソッドをイベントリスナー関数で呼び出すことにより、すべての Web サービス操作または個々の操作に追加できます。

addHeader() メソッドを使用する場合は、まず SOAPHeader オブジェクトと QName オブジェクトを別々に作成する必要があります。addHeader() メソッドには次のシグニチャがあります。

```
addHeader(header:mx.rpc.soap.SOAPHeader):void
```

SOAPHeader オブジェクトを作成するには、次のコンストラクターを使用します。

```
SOAPHeader(qname:QName, content:Object)
```

SOAPHeader() メソッドの最初のパラメーターで QName オブジェクトを作成するには、次のコンストラクターを使用します。

```
QName(uri:String, localName:String)
```

SOAPHeader() コンストラクターの content パラメーターは、次の形式に基づく、一連の名前と値のペアです。

```
{name1:value1, name2:value2}
```

addSimpleHeader() メソッドは、単一の名前と値の SOAP ヘッダーのショートカットです。addSimpleHeader() メソッドを使用する場合には、メソッドのパラメーターで SOAPHeader オブジェクトおよび QName オブジェクトを作成します。addSimpleHeader() メソッドには次のシグニチャがあります。

```
addSimpleHeader(qnameLocal:String, qnameNamespace:String, headerName:String,
    headerValue:Object):void
```

addSimpleHeader() メソッドは次のパラメーターを受け取ります。

- ***qnameLocal*** は、ヘッダーの QName のローカル名です。
- ***qnameNamespace*** は、ヘッダーの QName の名前空間です。
- ***headerName*** は、ヘッダーの名前です。
- ***headerValue*** は、ヘッダーの値です。単純な値の場合は String を、基本的な XML エンコードを使用する場合は Object を、ヘッダー XML を独自に指定する場合は XML を、それぞれ指定します。

次のコード例は、addHeader() メソッドと addSimpleHeader() メソッドを使用して SOAP ヘッダーを追加する方法を示しています。メソッドは headers というイベントリスナー関数で呼び出され、<mx:WebService> タグの load プロパティでイベントリスナーが割り当てられます。

```
<?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>
```

SOAP ヘッダーのクリア

オブジェクトに追加した SOAP ヘッダーを削除するには、次の WebService オブジェクトの例のように、WebService オブジェクトまたは操作オブジェクトの clearHeaders() メソッドを使用します。clearHeaders() は、ヘッダーを追加したレベル（WebService または操作）で呼び出す必要があります。

```
<?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>
```

別の URL への Web サービスのリダイレクト

Web サービスによっては、WSDL を処理して Web サービスを最初に呼び出した後で別のエンドポイント URL に変更する必要があります。例えば、セキュリティ資格情報を渡す必要のある Web サービスを使用するとします。Web サービスを呼び出してログイン資格情報を送信すると、資格情報が受け入れられ、サービスのビジネス操作を使用するために必要な実際のエンドポイント URL が返されます。ビジネス操作を呼び出す前に、WebService コンポーネントの endpointURI プロパティを変更する必要があります。

次の result イベントリスナーの例では、Web サービスから返されたエンドポイント URL を変数に格納し、その変数を関数に渡して後続の要求で使用するエンドポイント URL を変更します。

```
...
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;

}
...
```

セキュリティ資格情報を渡す必要のある Web サービスでは、後続の要求の SOAP ヘッダーに追加する必要のある ID も返される場合があります。詳しくは、83 ページの「SOAP ヘッダーの操作」を参照してください。

Web サービスデータのシリアライズ

ActionScript データのエンコード

次の表は、ActionScript 3.0 型から XML スキーマの複合型へのエンコードマッピングを示しています。

XML スキーマ定義	サポートされている ActionScript 3.0 型	メモ
最上位のエレメント		
xsd:element nillable == true	Object	入力値が null の場合、エンコードされた出力は xsi:nil 属性で設定されます。
xsd:element fixed != null	Object	入力値が無視され、代わりに固定値が使用されます。
xsd:element default != null	Object	入力値が null の場合、このデフォルト値が代わりに使用されます。
ローカルのエレメント		
xsd:element maxOccurs == 0	Object	入力値が無視され、エンコード出力から省略されます。
xsd:element maxOccurs == 1	Object	入力値が単一のエンティティとして処理されます。関連付けられている型が SOAP でエンコードされた配列である場合、配列および mx.collection.IList の実装はそのまま通過し、その型の SOAP エンコーダーで特別に処理されます。
xsd:element maxOccurs > 1	Object	反復不可能な値は処理前にラップされますが、入力値は反復可能である必要があります（配列や mx.collections.IList 実装など）。個々の項目は、定義に従って別々のエンティティとしてエンコードされます。
xsd:element minOccurs == 0	Object	入力値が未定義または null である場合、エンコードされた出力は省略されます。

次の表は、ActionScript 3.0 型から XML スキーマの組み込み型へのエンコードマッピングを示しています。

XML スキーマ型	サポートされている ActionScript 3.0 型	メモ
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 を使用します（行のラップなし）。
xsd:boolean	Boolean Number Object	常に true または false としてエンコードされます。 Number == 1 の場合は true、その他の場合は false です。 Object.toString() ==「true」または「1」の場合は true、その他の場合は false です。
xsd:byte xsd:unsignedByte	Number String	String はまず Number に変換されます。
xsd:date	Date Number String	Date UTC アクセッサーメソッドを使用します。 Number を使用して Date.time を設定します。 String は事前に書式設定されていると見なされ、そのままエンコードされます。
xsd:dateTime	Date Number String	Date UTC アクセッサーメソッドを使用します。 Number を使用して Date.time を設定します。 String は事前に書式設定されていると見なされ、そのままエンコードされます。
xsd:decimal	Number String	Number.toString() を使用します。Infinity、-Infinity および NaN は、この型には無効です。 String はまず Number に変換されます。
xsd:double	Number String	Number の範囲に制限されます。 String はまず Number に変換されます。
xsd:duration	Object	Object.toString() を呼び出します。
xsd:float	Number String	Number の範囲に制限されます。 String はまず Number に変換されます。
xsd:gDay	Date Number String	Date.getUTCDate() を使用します。 Number は日に直接使用されます。 String は日の Number として解析されます。
xsd:gMonth	Date Number String	Date.getUTCMonth() を使用します。 Number は月に直接使用されます。 String は月の Number として解析されます。
xsd:gMonthDay	Date String	Date.getUTCMonth() および Date.getUTCDate() を使用します。 String は月と日の部分用に解析されます。

XML スキーマ型	サポートされている ActionScript 3.0 型	メモ
xsd:gYear	Date Number String	Date.getUTCFullYear() を使用します。 Number は年に直接使用されます。 String は年の Number として解析されます。
xsd:gYearMonth	Date String	Date.getUTCFullYear() および Date.getUTCMonth() を使用します。 String は年と月の部分用に解析されます。
xsd:hexBinary	flash.utils.ByteArray	mx.utils.HexEncoder を使用します。
xsd:integer および次の派生型 xsd:negativeInteger xsd:nonNegativeInteger xsd:positiveInteger xsd:nonPositiveInteger	Number String	Number の範囲に制限されます。 String はまず Number に変換されます。
xsd:int xsd:unsignedInt	Number String	String はまず Number に変換されます。
xsd:long xsd:unsignedLong	Number String	String はまず Number に変換されます。
xsd:short xsd:unsignedShort	Number String	String はまず Number に変換されます。
xsd:string および次の派生型 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() を呼び出します。
xsd:time	Date Number String	Date UTC アクセッサーメソッドを使用します。 Number を使用して Date.time を設定します。 String は事前に書式設定されていると見なされ、そのままエンコードされます。
xsi:nil	null	対応する XML スキーマエレメント定義で minOccurs > 0 となっている場合、null 値は xsi:nil を使用してエンコードされます。そうしないと、このエレメント全体が省略されます。

次の表は、ActionScript 3.0 型から SOAP でエンコードされた型へのマッピングを示しています。

SOAPENC 型	サポートされている ActionScript 3.0 型	メモ
soapenc:Array	Array mx.collections.IList	SOAP でエンコードされた配列は特殊扱いされ、RPC でエンコードされたスタイルの Web サービスでのみサポートされます。
soapenc:base64	flash.utils.ByteArray	xsd:base64Binary と同様にエンコードされます。
soapenc:*	Object	SOAP でエンコードされているその他の型は、その型の QName の localName に基づいて、XSD 名前空間にあるかのように処理されます。

XML スキーマおよび SOAP から ActionScript 3.0 へのデコード

次の表は、XML スキーマの組み込み型から ActionScript 3.0 型へのデコードマッピングを示しています。

XML スキーマ型	デコードされる ActionScript 3.0 型	メモ
xsd:anyType xsd:anySimpleType	String Boolean Number	コンテンツが空の場合 -> xsd:string。 コンテンツが Number にキャストし値が NaN の場合、または コンテンツが「0」または「-0」で始まる場合、または コンテンツが「E」で終わる場合に、 コンテンツが「true」または「false」の場合 -> xsd:boolean それ以外の場合 -> xsd:string。 それ以外で、コンテンツが有効な Number の場合、-> xsd:double。
xsd:base64Binary	flash.utils.ByteArray	mx.utils.Base64Decoder を使用します。
xsd:boolean	Boolean	コンテンツが「true」または「1」の場合は true、その他の場合は false です。
xsd:date	Date	タイムゾーン情報が存在しない場合は、ローカルの時間と見なされます。
xsd:dateTime	Date	タイムゾーン情報が存在しない場合は、ローカルの時間と見なされます。
xsd:decimal	Number	コンテンツは Number(content) を使用して作成されるため、Number の範囲に制限されます。
xsd:double	Number	コンテンツは Number(content) を使用して作成されるため、Number の範囲に制限されます。
xsd:duration	String	コンテンツは空白スペースを省略して返されます。
xsd:float	Number	コンテンツは Number(content) を使用して変換されるため、Number の範囲に制限されます。
xsd:gDay	uint	コンテンツは uint(content) を使用して変換されます。
xsd:gMonth	uint	コンテンツは uint(content) を使用して変換されます。
xsd:gMonthDay	String	コンテンツは空白スペースを省略して返されます。
xsd:gYear	uint	コンテンツは uint(content) を使用して変換されます。
xsd:gYearMonth	String	コンテンツは空白スペースを省略して返されます。
xsd:hexBinary	flash.utils.ByteArray	mx.utils.HexDecoder を使用します。

XML スキーマ型	デコードされる ActionScript 3.0 型	メモ
xsd:integer および次の派生型 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	コンテンツは parseInt() を使用してデコードされます。
xsd:string および次の派生型 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	未加工のコンテンツが単純に文字列として返されます。
xsd:time	Date	タイムゾーン情報が存在しない場合は、ローカルの時間と見なされます。
xsi:nil	null	

次の表は、SOAP エンコード型から ActionScript 3.0 型へのデコードマッピングを示しています。

SOAPENC 型	デコードされる ActionScript 型	メモ
soapenc:Array	Array mx.collections.ArrayCollection	SOAP でエンコードされた配列は特殊扱いされます。makeObjectsBindable が true の場合、結果は ArrayCollection でラップされます。その他の場合は単純な配列が返されます。
soapenc:base64	flash.utils.ByteArray	xsd:base64Binary と同様にデコードされます。
soapenc:*	Object	SOAP でエンコードされているその他の型は、その型の QName の localName に基づいて、XSD 名前空間にあるかのように処理されます。

次の表は、カスタムデータ型から ActionScript 3.0 データ型へのデコードマッピングを示しています。

カスタム型	デコードされる ActionScript 3.0 型	メモ
Apache Map http://xml.apache.org/xml-soap:Map	Object	java.util.Map. Keys の SOAP 表現は、文字列として表現可能である必要があります。
Apache Rowset http://xml.apache.org/xml-soap:Rowset	オブジェクトの配列	
ColdFusion QueryBean http://rpc.xml.coldfusion:QueryBean	オブジェクトの配列 オブジェクトの mx.collections.ArrayCollection	makeObjectsBindable が true の場合、結果の配列は ArrayCollection でラップされます。

XML スキーマエレメントのサポート

次の XML スキーマ構造または構造属性は、Flex 4 では一部のみ実装されます。

```
<choice>
<all>
<union
```

次の XML スキーマ構造または構造属性は無視され、Flex 4 ではサポートされていません。

```
<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>
```

Web サービス型マッピングのカスタマイズ

Web サービスの呼び出しからデータを受信するとき、Flex では通常、SOAP メッセージ本文の XML 構造を模した型指定されていない匿名の ActionScript オブジェクトを作成します。Flex で特定のクラスのインスタンスを作成する場合は、mx.rpc.xml.SchemaTypeRegistry オブジェクトを使用して、QName オブジェクトを対応する ActionScript クラスに登録できます。

例えば、User.as というファイルに次のクラス定義があるとします。

```
package
{
     public class User
     {
       public function User() {}

       public var firstName:String;
       public var lastName:String;
     }
}
```

次に、Web サービスで、次の XML を返す getUser 操作を呼び出します。

```
<tns:getUserResponse xmlns:tns="http://example.uri">
     <tns:firstName>Ivan</tns:firstName>
     <tns:lastName>Petrov</tns:lastName>
</tns:getUserResponse>
```

getUser 操作を呼び出したときに汎用 Object ではなく自分の User クラスのインスタンスを確実に取得するには、自分のアプリケーションのメソッドで次の ActionScript コードを指定する必要があります。

```
SchemaTypeRegistry.getInstance().registerClass(new QName("http://example.uri", "getUserResponse"), User);
```

SchemaTypeRegistry.getInstance() は、registry 型のデフォルトのインスタンスを返す静的メソッドです。ほとんどの場合、必要なのはこれだけです。ただし、この操作によって、指定された QName は、アプリケーションのすべての Web サービス操作で同じ ActionScript クラスに登録されます。操作ごとに別々のクラスを登録する場合は、アプリケーションのメソッドで次のコードを指定する必要があります。

```
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;
```

カスタム Web サービスシリアライゼーションの使用

ActionScript オブジェクトを XML にシリアライズする方法および XML 応答メッセージをデシリアライズする方法を完全に制御するには、2 つの方法があります。推奨される方法は、E4X を直接操作することです。

XML のインスタンスを唯一のパラメーターとして Web サービス操作に渡すと、このインスタンスはシリアライズされた要求の <SOAP:Body> ノードの子として、そのまま渡されます。SOAP メッセージを完全に制御する必要がある場合は、この方法を使用します。同様に、Web サービスの応答をデシリアライズするときは、操作の resultFormat プロパティを e4x に設定できます。これにより、<SOAP:Body> ノードの子が応答メッセージに含まれた XMLList オブジェクトが返されます。ここから、適切な ActionScript オブジェクトを作成するために必要なカスタムロジックを実装できます。

2 つ目の、より手間のかかる方法は、mx.rpc.soap.ISOAPDecoder および mx.rpc.soap.ISOAPEncoder を独自に実装することです。例えば、ISOAPDecoder を実装する MyDecoder というクラスを作成済みの場合は、アプリケーションのメソッドで次のように指定できます。

```
myWS.someOperation.decoder = new MyDecoder();
```

someOperation を呼び出すと、Flex では MyDecoder クラスの decodeResponse() メソッドが呼び出されます。この時点以降、カスタム実装が SOAP メッセージ全体を処理し、想定される ActionScript オブジェクトを生成します。

RemoteObject コンポーネントの使用

Flex の RemoteObject コンポーネントを使用して、ColdFusion コンポーネントまたは Java クラスでメソッドを呼び出すことができます。

PHP オブジェクトおよび .NET オブジェクトを含む RemoteObject コンポーネントを、オープンソースプロジェクトの AMFPHP や SabreAMF、および Midnight Coders WebORB などのサードパーティソフトウェアとともに使用することもできます。詳しくは、次の Web サイトを参照してください。

- 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 コンポーネントではデータの送受信に AMF プロトコルが使用されます。一方、WebService コンポーネントと HTTPService コンポーネントでは HTTP プロトコルが使用されます。AMF は HTTP よりもはるかに高速ですが、一般にサーバー側でのコーディングと設定が複雑になります。

Flash Builder for PHP は、Zend Technologies 社とのパートナーシップに基づいて作成された開発ツールで、Zend Studio のコピーが統合されています。詳しくは、アドビの Web サイトを参照してください。

HTTPService コンポーネントや WebService コンポーネントと同様に、RemoteObject コンポーネントを使用すると、データベースクエリの結果をアプリケーションで表示できます。また、これらのコンポーネントを使用して、データの挿入、更新、削除などの操作をデータベースに対して実行することもできます。アプリケーションに返されたクエリの結果は、ユーザーインターフェイスコントロールで表示できます。

RemoteObject コンポーネントについての API リファレンス情報は、mx.rpc.remoting.mxml.RemoteObject を参照してください。

RemoteObject アプリケーションのサンプル

MXML コード

次の例のアプリケーションは、RemoteObject コンポーネントを使用して ColdFusion コンポーネントを呼び出します。ColdFusion コンポーネントは、users という名前の MySQL データベーステーブルをクエリします。その後、クエリ結果をアプリケーションに返します。Flex アプリケーションでは、クエリ結果は DataGrid コントロールの dataProvider プロパティにバインドされていて、DataGrid コントロールに表示されます。また、アプリケーションは新しいユーザーのユーザー名と電子メールアドレスを ColdFusion コンポーネントに送信し、ColdFusion コンポーネントはユーザーデータベーステーブルへの挿入を実行します。

```
<?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>
```

このアプリケーションでは、RemoteObject コンポーネントの destination プロパティは Coldfusion に設定され、source プロパティは ColdFusion コンポーネントの完全修飾名に設定されています。

これに対し、LiveCycle Data Services または BlazeDS を使用するときは、設定ファイルでリモートサービスの宛先の source プロパティに完全修飾クラス名を指定します。デフォルトでは、設定ファイルは remoting-config.xml です。宛先の名前は、RemoteObject コンポーネントの destination プロパティで指定します。また、宛先クラスには引数なしのコンストラクターも設定する必要があります。ColdFusion を使用するときは、必要に応じて、RemoteObject コンポーネントの source プロパティを使用する代わりに、この方法で宛先を設定できます。

ColdFusion コンポーネント

アプリケーションは次の ColdFusion コンポーネントを呼び出します。この ColdFusion コードは、SQL データベースの挿入とクエリを実行し、アプリケーションにクエリ結果を返します。ColdFusion ページでは、cfquery タグを使用して、データベースにデータを挿入し、データベースをクエリします。また、cfreturn タグを使用して、クエリ結果を ColdFusion クエリオブジェクトとしてフォーマットします。

```
<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>
```

ActionScript での RemoteObject コンポーネントの呼び出し

次の ActionScript の例では、useRemoteObject() メソッドを呼び出すと、サービスが宣言され、宛先が設定され、result および fault のイベントリスナーが設定されて、サービスの getList() メソッドが呼び出されます。

```
<?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>
```

ソースパスにある Java オブジェクトへのアクセス

RemoteObject コンポーネントを使用すると、LiveCycle Data Services 、BlazeDS または ColdFusion の Web アプリケーションのソースパスにあるステートレスおよびステートフルの Java オブジェクトにアクセスできます。Web アプリケーションの WEB–INF/classes ディレクトリにスタンドアロンのクラスファイルを配置すると、それらのクラスファイルがソースパスに追加されます。Web アプリケーションの WEB–INF/lib ディレクトリに、Java アーカイブ（JAR）ファイルに含まれているクラスを配置すると、それらのクラスがソースパスに追加されます。LiveCycle Data Services 、BlazeDS または ColdFusion の services-config.xml ファイル、または services-config.xml ファイルが参照によってインクルードしている remoting-config.xml などのファイルにある、リモートサービスの宛先の source プロパティに、完全修飾クラス名を指定します。また、クラスには引数なしのコンストラクターも設定する必要があります。ColdFusion の場合は、必要に応じて、RemoteObject コンポーネントの destination プロパティを Coldfusion に設定し、source プロパティを ColdFusion コンポーネントまたは Java クラスの完全修飾名に設定できます。

ステートレスオブジェクト（要求のスコープ）にアクセスするようにリモートサービスの宛先を設定した場合、Flex では、同じオブジェクトで複数のメソッドが呼び出されるのではなく、メソッドの呼び出しごとに別のオブジェクトが作成されます。オブジェクトのスコープは、要求のスコープ（デフォルト値）、アプリケーションのスコープ、またはセッションのスコープに設定できます。アプリケーションのスコープ内のオブジェクトは、そのオブジェクトのある Web アプリケーションで利用できます。セッションのスコープ内のオブジェクトは、そのクライアントセッション全体で利用できます。

ステートフルオブジェクトにアクセスするようにリモートオブジェクトの宛先を設定した場合、Flex では、オブジェクトがサーバー上に 1 回だけ作成され、メソッド呼び出し間で状態が維持されます。オブジェクトをアプリケーションのスコープまたはセッションのスコープに格納するとメモリの問題が発生する場合は、要求のスコープを使用します。

JNDI 上の EJB およびその他のオブジェクトへのアクセス

Java Naming and Directory Interface（JNDI）内のオブジェクトを検索してそのメソッドを呼び出すサービスファサードクラスとなっている、宛先のメソッドを呼び出すことにより、JNDI 内に格納されている Enterprise JavaBean（EJB）およびその他のオブジェクトにアクセスすることができます。

ステートレスオブジェクトまたはステートフルオブジェクトを使用し、JNDI を使用する EJB その他のオブジェクトのメソッドを呼び出すことができます。EJB の場合、JNDI から EJB オブジェクトを返して EJB のメソッドを呼び出す、サービスファサードクラスを呼び出すことができます。

Java クラスでは標準の Java コーディングパターンを使用し、初期コンテキストを作成して JNDI ルックアップを実行します。EJB でも標準のコーディングパターンを使用し、EJB ホームオブジェクトの create() メソッドと結果の EJB のビジネスメソッドを呼び出すメソッドをクラスに含めます。

次の例では、ファサードクラスの宛先で getHelloData() という名前のメソッドを使用しています。

```
<mx:RemoteObject id="Hello" destination="roDest">
    <mx:method name="getHelloData"/>
</mx:RemoteObject>
```

Java 側では、getHelloData() メソッドを使用して、EJB のビジネスメソッドを呼び出すために必要なものすべてをカプセル化できます。次の例に示す Java メソッドでは、次のアクションを実行します。

- EJB を呼び出すための新しい初期コンテキストを作成します。
- JNDI ルックアップを実行して EJB ホームオブジェクトを取得します。
- EJB ホームオブジェクトの create() メソッドを呼び出します。
- EJB の sayHello() メソッドを呼び出します。

```
...
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);
    }
...
```

予約されているメソッド名

次のメソッド名は Flex の Remoting ライブラリで使用されています。メソッドを作成する際にこれらのメソッド名を使用しないでください。

```
addHeader()
addProperty()
deleteHeader()
hasOwnProperty()
isPropertyEnumerable()
isPrototypeOf()
registerClass()
toLocaleString()
toString()
unwatch()
valueOf()
watch()
```

また、メソッドの名前の先頭にはアンダースコア文字（_）を使用しないでください。

RemoteObject のメソッド（操作）には、サービス変数名を付けるだけでアクセスできます。ただし、操作名がサービスで定義されているメソッドと一致する場合は、名前付けの競合が発生する可能性があります。RemoteObject コンポーネントの ActionScript で次のメソッドを使用することで、指定した名前の操作を返すことができます。

```
public function getOperation(name:String):Operation
```

ActionScript と Java の間のシリアライズ

LiveCycle Data Services および BlazeDS では、ActionScript（AMF 3）データ型と Java および ColdFusion データ型の間でデータのシリアライズが双方向に行われます。ColdFusion データ型について詳しくは、ColdFusion のマニュアルを参照してください。

ActionScript から Java へのデータの変換

メソッドのパラメーターによってアプリケーションから Java オブジェクトへデータを送信すると、データは ActionScript データ型から Java データ型に自動的に変換されます。LiveCycle Data Services または BlazeDS では、Java オブジェクトで適切なメソッドを検索するときに、それほど厳密でない変換を使用して一致を見つけます。

Boolean や String 値などのクライアント側の単純なデータ型は、通常、リモート API に完全に一致しますが、ただし、Flex では、Java オブジェクトで適切なメソッドを検索するときに、簡単な変換をいくつか試行します。

ActionScript 配列では、2 つの方法でエントリにインデックスを付けることができます。「厳密な配列」は、すべてのインデックスが数値である配列です。「結合配列」は、1 つ以上のインデックスが文字列に基づいている配列です。サーバーに送信する配列によって、Java オブジェクトでメソッドを呼び出す際に使用するパラメーターのデータ型が変わるため、どちらの配列をサーバーに送信するかを知っておくことは重要です。「密配列」は、すべての数値インデックスが 0（ゼロ）から始まって途切れなく連続している配列です。「疎配列」は、数値インデックスに途切れがある配列です。配列はオブジェクトのように処理され、数値インデックスはプロパティとなり、デシリアライズされて java.util.Map オブジェクトになります。これにより、多数の null エントリが送信されることがなくなります。

次の表は、単純なデータ型に関してサポートされている、ActionScript（AMF 3）から Java への変換を示しています。

ActionScript 型（AMF 3）	Java へのデシリアライゼーション	サポートされている Java 型のバインディング
Array（密）	java.util.List	java.util.Collection、**Object**[ ]（ネイティブ配列） 型がインターフェイスである場合は、次のインターフェイス実装にマップされます。 • List は ArrayList になります • SortedSet は TreeSet になります • Set は HashSet になります • Collection は ArrayList になります カスタムの Collection 実装の新しいインスタンスは、その型にバインドされます。
Array（疎）	java.util.Map	java.util.Map
Boolean "true" または "false" の文字列	java.lang.Boolean	Boolean、boolean、String
flash.utils.ByteArray	byte []	
flash.utils.IExternalizable	java.io.Externalizable	

ActionScript 型（AMF 3）	Java へのデシリアライゼーション	サポートされている Java 型のバインディング
Date	java.util.Date （協定世界時（UTC）用に書式設定）	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、double、long、float、int、short、byte の各プリミティブ型
null	null	プリミティブ
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（ゼロ） null が送信された場合、double、long、float、int、short、byte の各プリミティブ型
Object（汎用）	java.util.Map	Map インターフェイスが指定されている場合、java.util.Map では java.util.HashMap が、java.util.SortedMap では新しい java.util.TreeMap が作成されます。
String	java.lang.String	java.lang.String、java.lang.Boolean、java.lang.Number、java.math.BigInteger、java.math.BigDecimal、char[]、すべてのプリミティブ数値型
型指定された Object	型指定された Object リモートクラス名を指定する [RemoteClass] メタデータタグを使用した場合。Bean 型には、引数なしのパブリックコンストラクターが必要です。	型指定された Object
undefined	null	オブジェクトの場合は null、プリミティブの場合はデフォルト値
XML	org.w3c.dom.Document	org.w3c.dom.Document
XMLDocument （従来の XML 型）	org.w3c.dom.Document	org.w3c.dom.Document XMLDocument 型に対する従来の XML のサポートは、services-config.xml ファイルで定義されている任意のチャンネルで有効にできます。この設定は、サーバーからクライアントにデータを送り返す場合にのみ重要です。org.w3c.dom.Document インスタンスを ActionScript に送信する方法を制御します。詳しくは、Configuring AMF serialization on a channel を参照してください。

プリミティブ値は、Java では null に設定できません。Boolean 型や Number 型の値をクライアントから Java オブジェクトに渡すとき、Flex では、null 値をプリミティブ型のデフォルト値として解釈します。例えば、double、float、long、int、short、byte の場合は 0、char の場合は \u0000、boolean の場合は false です。デフォルト値を取得するのは Java のプリミティブ型のみです。

LiveCycle Data Services および BlazeDS は、java.lang.Throwable オブジェクトを、その他の型指定されたオブジェクトと同じように処理します。パブリックフィールドや Bean プロパティを検索するルールに従って処理され、型指定されたオブジェクトがクライアントに返されます。このルールは通常の Bean ルールと同様ですが、読み取り専用プロパティの getter を探す点が異なります。これにより、Java 例外からより多くの情報を得ることができます。Throwable オブジェクトの従来の動作が必要な場合は、チャンネルで legacy-throwable プロパティを true に設定できます。詳しくは、Configuring AMF serialization on a channel を参照してください。

厳密な配列は、java.util.Collection またはネイティブの Java Array API を実装する予定のメソッドに対して、パラメーターとして渡すことができます。

Java Collection には、Object 型を何個でも含めることができますが、Java 配列ではエントリの型をすべて同じにする必要があります（java.lang.Object[ ]、int[ ] など）。

LiveCycle Data Services および BlazeDS では、ActionScript の厳密な配列も、共通の Collection API インターフェイスの適切な実装に変換されます。例えば、ActionScript の厳密な配列を Java オブジェクトメソッド public void addProducts(java.util.Set products) に送信した場合、これは LiveCycle Data Services および BlazeDS によって java.util.HashSet インスタンスに変換されてから、パラメーターとして渡されます。これは、HashSet が、java.util.Set インターフェイスの適切な実装であるためです。同様に、LiveCycle Data Services および BlazeDS では、java.util.TreeSet のインスタンスが、java.util.SortedSet インターフェイスで型指定されたパラメーターに渡されます。

LiveCycle Data Services および BlazeDS では、java.util.ArrayList のインスタンスが、java.util.List インターフェイスおよび java.util.Collection を拡張するその他のインターフェイスで型指定されたパラメーターに渡されます。その後、これらの型は、mx.collections.ArrayCollection インスタンスとしてクライアントに送り返されます。通常の ActionScript 配列をクライアントに送り返す必要がある場合は、チャンネル定義のプロパティの serialization セクションで、legacy-collection エレメントを true に設定する必要があります。詳しくは、Configuring AMF serialization on a channel を参照してください。

ActionScript オブジェクトと Java オブジェクトの明示的なマッピング

LiveCycle Data Services および BlazeDS で暗黙的に処理されない Java オブジェクトの場合は、get/set メソッドを使用してパブリック Bean プロパティで見つかった値とパブリック変数が、オブジェクトのプロパティとしてクライアントに送信されます。プライベートプロパティ、定数、静的プロパティ、読み取り専用プロパティなどは、シリアライズされません。ActionScript オブジェクトの場合は、get/set アクセッサーで定義されたパブリックプロパティとパブリック変数がサーバーに送信されます。

LiveCycle Data Services および BlazeDS では、標準の Java クラスである java.beans.Introspector を使用して、JavaBean クラスのプロパティ記述子を取得します。また、リフレクションを使用して、任意のクラスのパブリックフィールドを収集します。フィールドより Bean プロパティのほうが優先的に使用されます。Java と ActionScript のプロパティ名は一致する必要があります。クライアントでの ActionScript クラスの内観方法は、ネイティブの Flash Player コードによって決まります。

ActionScript クラスでは、[RemoteClass(alias="")] メタデータタグを使用して、Java オブジェクトに直接マップする ActionScript オブジェクトを作成します。MXML ファイルを SWF ファイルにリンクさせ、実行時に使用できるようにするには、データの変換先の ActionScript クラスを MXML ファイル内で使用または参照する必要があります。これを行うには、次の例のように、結果オブジェクトをキャストします。

```
var result:MyClass = MyClass(event.result);
```

クラス自体では、依存関係もリンクされるようにするために、厳密に型指定された参照を使用してください。

次の例は、[RemoteClass(alias="")] メタデータタグを使用する ActionScript クラスのソースコードを示しています。

```
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;
    }
}
```

サーバー上の Java オブジェクトにマップせずに、サーバーからオブジェクト型を送り返す場合は、[RemoteClass] メタデータタグを、エイリアスを指定せずに使用できます。ActionScript オブジェクトは、サーバーに送信されたときは特殊な Map オブジェクトにシリアライズされますが、サーバーからクライアントに返されたオブジェクトは元の ActionScript 型になります。

特定のプロパティが任意の ActionScript クラスからサーバーに送信されるのを制限するには、ActionScript クラスの該当するプロパティの宣言の上で [Transient] メタデータタグを使用します。

Java から ActionScript へのデータの変換

Java メソッドから返されたオブジェクトは、Java から ActionScript に変換されます。また、LiveCycle Data Services および BlazeDS はオブジェクト内にあるオブジェクトも処理します。LiveCycle Data Services は、次の表の Java データ型を暗黙的に処理します。

Java 型	**ActionScript 型（AMF 3）**
java.lang.String	String
java.lang.Boolean、boolean	Boolean
java.lang.Integer、int	int 値 < 0xF0000000 || 値 > 0x0FFFFFFF の場合、AMF エンコードの要件により、値は Number 型に昇格されます。
java.lang.Short、short	int i < 0xF0000000 || i > 0x0FFFFFFF の場合、値は Number 型に昇格されます。
java.lang.Byte、byte[]	int i < 0xF0000000 || i > 0x0FFFFFFF の場合、値は Number 型に昇格されます。
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 型	ActionScript 型（AMF 3）
java. math.BigInteger	String
java.math.BigDecimal	String
java.util.Calendar	Date 日付は協定世界時（UTC）タイムゾーンで送信されます。クライアントとサーバーで、タイムゾーンに合わせて時刻を調整する必要があります。
java.util.Date	Date 日付は UTC タイムゾーンで送信されます。クライアントとサーバーで、タイムゾーンに合わせて時刻を調整する必要があります。
java.util.Collection（java.util.ArrayList など）	mx.collections.ArrayCollection
java.lang.Object[]	Array
java.util.Map	Object（型指定なし）。例えば、java.util.Map[] は、オブジェクトの配列に変換されます。
java.util.Dictionary	Object（型指定なし）
org.w3c.dom.Document	XML オブジェクト
null	null
java.lang.Object（上記以外の型）	型指定された Object オブジェクトは JavaBean 分析ルールを使用してシリアライズされ、パブリックフィールドも含みます。静的フィールド、一時的なフィールド、非パブリックフィールドや、非パブリックまたは静的な Bean プロパティは除外されます。

チャンネルでの AMF シリアライゼーションの設定

以前のバージョンの Flex で使用されていた従来の AMF 型シリアライゼーションをサポートし、services-config.xml ファイルのチャンネル定義でその他のシリアライゼーションプロパティを設定できます。

チャンネル定義の <serialization> 要素で設定できるプロパティを、次の表で説明します。

プロパティ	説明
`<ignore-property-errors>true</ignore-property-errors>`	デフォルト値は true です。受信するクライアントオブジェクトに、サーバーオブジェクトで設定できない予期しないプロパティが含まれている場合にエンドポイントでエラーを発生させるかどうかを決定します。
`<log-property-errors>false</log-property-errors>`	デフォルト値は false です。true に設定すると、予期しないプロパティエラーがログに記録されます。
`<legacy-collection>false</legacy-collection>`	デフォルト値は false です。true に設定すると、java.util.Collection のインスタンスが ActionScript 配列として返されます。false に設定すると、java.util.Collection のインスタンスが mx.collections.ArrayCollection として返されます。
`<legacy-map>false</legacy-map>`	デフォルト値は false です。true に設定すると、java.util.Map インスタンスが、匿名オブジェクトではなく ECMA 配列または結合配列としてシリアライズされます。
`<legacy-xml>false</legacy-xml>`	デフォルト値は false です。true に設定すると、org.w3c.dom.Document インスタンスが組み込みの XML（E4X 対応）インスタンスではなく flash.xml.XMLDocument インスタンスとしてシリアライズされます。

プロパティ	説明
`<legacy-throwable>false</legacy-throwable>`	デフォルト値は false です。true に設定すると、java.lang.Throwable インスタンスが、読み取り専用プロパティを含む通常の Bean シリアライゼーションではなく、AMF ステータス情報オブジェクトとしてシリアライズされます。
`<type-marshaller>className</type-marshaller>`	任意のオブジェクトを必要なクラスのインスタンスに変換する flex.messaging.io.TypeMarshaller の実装を指定します。Java メソッドを呼び出すときや、Java インスタンスにデータを渡すときに、デシリアライゼーションからの入力オブジェクトの型（例えば、ActionScript の匿名オブジェクトは常に java.util.HashMap としてシリアライズされます）が、宛先 API（java.util.SortedMap など）と一致しない場合に使用します。つまり、必要な型にマーシャリングすることができます。
`<restore-references>false</restore-references>`	デフォルト値は false です。型の変換が必要なときに、デシリアライザでオブジェクト参照を追跡するための高度なスイッチです。例えば、java.util.SortedMap 型のプロパティに関して匿名オブジェクトが送信された場合、オブジェクトはまず通常どおりに java.util.Map にデシリアライズされてから、SortedMap の適切な実装（java.util.TreeMap など）に変換されます。他のオブジェクトがオブジェクトグラフ内で同じ匿名オブジェクトを指していた場合、これを設定すると、SortedMap 実装をすべての場所で作成する代わりに、これらの参照が復元されます。大量のデータがある場合、このプロパティを true に設定すると、パフォーマンスが大幅に低下することがあります。
`<instantiate-types>true</instantiate-types>`	デフォルト値は true です。高度なスイッチであり、false に設定すると、デシリアライザーは厳密に型指定されたオブジェクトのインスタンスの作成を停止し、代わりに型情報を保持して、Map 実装（flex.messaging.io.ASObject）の未加工のプロパティをデシリアライズします。flex.* パッケージの下のクラスは常にインスタンス化されます。

カスタムシリアライゼーションの使用

クライアント上の ActionScript とサーバー上の Java との間でデータをシリアライズおよびデシリアライズする際、標準のメカニズムでニーズが満たされない場合は、独自のシリアライゼーションスキームを作成できます。ActionScript ベースの flash.utils.IExternalizable インターフェイスをクライアントに実装し、対応する Java ベースの java.io.Externalizable インターフェイスをサーバーに実装します。

通常、カスタムシリアライゼーションを使用するのは、ネットワーク層でクライアント側またはサーバー側のオブジェクト表現のプロパティがすべて渡されるのを避けるためです。カスタムシリアライゼーションを実装するときは、クライアントのみまたはサーバーのみの特定のプロパティが渡されないように、クラスをコード化することができます。標準のシリアライゼーションスキームを使用すると、クライアントとサーバーの間ですべてのパブリックプロパティがやり取りされます。

クライアント側では、flash.utils.IExternalizable インターフェイスを実装するクラスの ID がシリアライゼーションストリームに書き込まれます。このクラスでは、インスタンスの状態をシリアライズし、再構築します。クラスでは IExternalizable インターフェイスの writeExternal() メソッドと readExternal() メソッドを実装して、シリアライゼーションストリームの内容と形式を制御しますが、オブジェクトとそのスーパータイプのクラス名や型は制御しません。これらのメソッドは、ネイティブの AMF シリアライゼーション動作より優先されます。クラスの状態を保存するため、これらのメソッドは、リモートの対応するメソッドと対称的である必要があります。

サーバー側では、java.io.Externalizable インターフェイスを実装する Java クラスによって、flash.utils.IExternalizable インターフェイスを実装する ActionScript クラスと同様の機能が実行されます。

注意：参照ごとの厳密なシリアライゼーションが必要な場合は、HTTPChannel を使用して IExternalizable インターフェイスを実装する型を使用しないでください。使用すると、循環オブジェクトの参照が失われ、エンドポイントでクローンが作成されたような状態になります。

次の例は、サーバー上の Java ベースの Product クラスにマップする、クライアント（ActionScript）側の Product クラスの完全なソースコードを示しています。クライアントの Product が IExternalizable インターフェイスを実装し、サーバーの Product が Externalizable インターフェイスを実装します。

```
// 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);
    }
}
}
```

クライアントの Product では 2 種類のシリアライゼーションを使用します。つまり、java.io.Externalizable インターフェイスと互換性のある標準のシリアライゼーションと、AMF 3 シリアライゼーションを使用します。次の例は、クライアントの Product の writeExternal() メソッドを示しています。このメソッドでは両方の種類のシリアライゼーションを使用しています。

```
public function writeExternal(output:IDataOutput):void {
    output.writeObject(name);
    output.writeObject(properties);
    output.writeFloat(price);
}
```

次の例に示すように、サーバーの Product の writeExternal() メソッドは、クライアント側のこのメソッドとほぼ同じです。

```
public void writeExternal(ObjectOutput out) throws IOException {
    out.writeObject(name);
    out.writeObject(properties);
    out.writeFloat(price);
}
```

クライアントの Product の writeExternal() メソッドで、flash.utils.IDataOutput.writeFloat() メソッドはプリミティブ型を操作する Java の java.io.DataInput.readFloat() メソッドの仕様に適合する標準のシリアライズメソッドの例です。このメソッドは、Float 型の price プロパティをサーバーの Product に送信します。

クライアントの Product の writeExternal() メソッド内の AMF 3 シリアライゼーションの例は、flash.utils.IDataOutput.writeObject() メソッドに対する呼び出しです。このメソッドは、サーバーの Product の readExternal() メソッド内の java.io.ObjectInput.readObject() メソッド呼び出しに対応します。flash.utils.IDataOutput.writeObject() メソッド

は、Object 型の properties プロパティと String 型の name プロパティをサーバーの Product に送信します。これが可能なのは、writeObject() メソッドから送信されたデータを AMF 3 形式として受信する java.io.ObjectInput インターフェイスが AMFChannel エンドポイントに実装されているためです。

また、サーバーの Product の readExternal() メソッドで readObject() メソッドが呼び出されると、AMF 3 デシリアライゼーションが使用されます。このため、ActionScript 側の properties 値は Map 型、name は String 型と見なされます。

次の例は、サーバーの Product クラスの完全なソースを示しています。

```
// 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;
        }
}
```

次の例は、サーバーの Product の readExternal() メソッドを示しています。

```
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));
    }
```

クライアントの Product の writeExternal() メソッドは、シリアライズの際に id プロパティをサーバーに送信しません。これは、サーバー側の Product オブジェクトではこのプロパティが役に立たないためです。同様に、サーバーの Product の writeExternal() メソッドは、サーバー固有のプロパティである inventoryId プロパティをクライアントに送信しません。

Product のプロパティの名前は、どちらの方向のシリアライゼーションでも送信されません。クラスの状態が固定であり管理可能であるため、プロパティは名前が付けられずに、適切に定義された順序で送信され、readExternal() ではそれらが適切な順序で読み取られます。

明示的なパラメーターの受け渡しとパラメーターのバインディング

HTTPService、WebService および RemoteObject のコンポーネントを呼び出すには、明示的なパラメーターの受け渡しとパラメーターのバインディングという 2 つの方法があります。明示的にパラメーターを渡す方法を使用する場合は、ActionScript 関数へのパラメーターの形式でサービスに入力します。この方法によるサービスの呼び出しは、Java でメソッドを呼び出す方法とよく似ています。明示的にパラメーターを渡す場合は、Flex のデータバリデーターを自動的に使用することはできません。

パラメーターのバインディングを使用すると、ユーザーインターフェイスのコントロールまたはモデルから、データを要求パラメーターにコピーできます。パラメーターのバインディングは、MXML で宣言されているデータアクセスコンポーネントに対してのみ使用できます。要求をサービスに送信する前に、パラメーター値にバリデーターを適用できます。データバインディングおよびデータモデルについて詳しくは、Data binding および Storing Data を参照してください。データの検証について詳しくは、Validating Data を参照してください。

パラメーターのバインディングを使用するときは、<mx:method> タグの下の <mx:arguments> タグにネストされた RemoteObject メソッドパラメータータグ、<mx:request> タグにネストされた HTTPService パラメータータグまたは <mx:operation> タグの下の <mx:request> タグにネストされた WebService 操作パラメータータグを宣言します。要求を送信するには send() メソッドを使用します。

RemoteObject コンポーネントおよび WebService コンポーネントで明示的にパラメーターを渡す方法

RemoteObject コンポーネントと WebService コンポーネントでは、明示的にパラメーターを渡す方法がよく似ています。次の例は、RemoteObject コンポーネントを宣言し、Button コントロールの click イベントリスナーで明示的にパラメーターを渡す方法でサービスを呼び出す MXML コードを示しています。ComboBox コントロールによってデータがサービスに提供されます。簡単なイベントリスナーによって、サービスレベルの result イベントと fault イベントが処理されます。

```
<?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>
```

HTTPService コンポーネントで明示的にパラメーターを渡す方法

HTTPService コンポーネントで明示的にパラメーターを渡す方法は、RemoteObject コンポーネントおよび WebService コンポーネントの場合とは異なります。必ず、HTTPService コンポーネントの send() メソッドを使用してサービスを呼び出します。これは、クライアント側の RPC サービスのメソッドまたは操作であるメソッドを呼び出す RemoteObject コンポーネントおよび WebService コンポーネントとは異なります。

明示的にパラメーターを渡す方法を使用する場合、名前と値のペアを格納したオブジェクトを send() メソッドのパラメーターとして指定できます。send() メソッドのパラメーターは、単純な基本型である必要があります。複雑にネストされたオブジェクトは、名前と値のペアに変換する一般的な方法がないので使用できません。

send() メソッドへのパラメーターを指定しない場合、HTTPService コンポーネントでは、<mx:request> タグで指定した任意のクエリパラメーターが使用されます。

次の例は、send() メソッドにパラメーターを指定して HTTP サービスを呼び出す 2 つの方法を示しています。2 つ目の例では、cancel() メソッドを呼び出して HTTP サービス呼び出しをキャンセルする方法も示しています。

```
<?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>
```

RemoteObject コンポーネントでのパラメーターのバインディング

RemoteObject コンポーネントでパラメーターのバインディングを使用する場合は、必ず RemoteObject コンポーネントの <mx:method> タグでメソッドを宣言します。

<mx:method> タグには、メソッドのパラメーターに使用する子タグを含む <mx:arguments> タグを含めることができます。<mx:method> タグの name プロパティは、サービスのいずれかのメソッド名に合わせる必要があります。引数タグの順序はサービスのメソッドパラメーターの順序に一致させる必要があります。引数タグの名前は、対応するメソッドパラメーターの名前と一致させることもできますが、別の名前にすることもできます。

注意：<mx:arguments> タグの内側に同じ名前の引数タグがある場合、リモートメソッドで配列を唯一の入力ソースとして想定していなければ、サービスの呼び出しは失敗します。アプリケーションのコンパイル時に、これに関する警告は表示されません。

データは、RemoteObject コンポーネントのメソッドパラメーターにバインドできます。データのバインドと検証を行うには、パラメーターのタグ名を参照します。

次の例は、TextInput コントロールの text プロパティに 2 つのパラメーターをバインドしたメソッドを示しています。PhoneNumberValidator バリデーターを、最初の引数タグの名前である arg1 に割り当てています。

```
<?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 により、引数タグの値が MXML タグで指定した順序でメソッドに渡されます。

次の例では、ユーザーが Button コントロールをクリックしたときに、RemoteObject コンポーネントの <mx:method> タグでパラメーターのバインディングを使用して、選択した ComboBox アイテムのデータを employeeRO.getList 操作にバインドします。パラメーターのバインディングを行う場合は、パラメーターなしの send() メソッドを使用してサービスを呼び出します。

```
<?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>
```

サービス呼び出しの結果に配列が含まれるのか単独のオブジェクトが含まれるのかわからない場合は、この例のように mx.utils.ArrayUtil クラスの toArray() メソッドを使用して、結果を配列に変換することができます。toArray() メソッドに単独のオブジェクトを渡すと、そのオブジェクトを唯一の配列エレメントとする配列が返されます。このメソッドに任意の配列を渡すと、同じ配列が返されます。ArrayCollection オブジェクトの使用について詳しくは、Data providers and collections を参照してください。

HTTPService コンポーネントでのパラメーターのバインディング

HTTP サービスがクエリパラメーターを取得した場合、それらを <mx:request> タグの子タグとして宣言できます。タグの名前は、サービスで想定されているクエリパラメーターの名前に一致させる必要があります。

次の例では、ユーザーが Button コントロールをクリックしたときに、HTTPService コンポーネントの <mx:request> タグでパラメーターのバインディングを使用して、選択した ComboBox アイテムのデータを employeeSrv 要求にバインドします。パラメーターのバインディングを行う場合は、パラメーターなしの send() メソッドを使用してサービスを呼び出します。この例では HTTPService コンポーネントの url プロパティを示していますが、サービスを呼び出す方法は、サービスに直接接続している場合でも、宛先を経由する場合でも同じです。

```
<?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>
```

サービス呼び出しの結果に配列と単独のオブジェクトのどちらが含まれるか不明な場合は、前の例で示したように、mx.utils.ArrayUtil クラスの toArray() メソッドを使用して配列に変換できます。toArray() メソッドに単独のオブジェクトを渡すと、そのオブジェクトを唯一の配列エレメントとする配列が返されます。このメソッドに任意の配列を渡すと、同じ配列が返されます。ArrayCollection オブジェクトの使用について詳しくは、Data providers and collections を参照してください。

WebService コンポーネントでのパラメーターのバインディング

WebService コンポーネントでパラメーターのバインディングを使用する場合は、必ず WebService コンポーネントの <mx:operation> タグで操作を宣言します。<mx:operation> タグには、操作で想定される XML ノードを含む <mx:request> タグを含めることができます。<mx:operation> タグの name プロパティは、Web サービスのいずれかの操作名に合わせる必要があります。

Web サービス操作のパラメーターにデータをバインドできます。データのバインドと検証を行うには、パラメーターのタグ名を参照します。

次の例では、ユーザーが Button コントロールをクリックしたときに、WebService コンポーネントの <mx:operation> タグでパラメーターのバインディングを使用して、選択した ComboBox アイテムのデータを employeeWS.getList 操作にバインドします。<deptId> タグは、getList 操作の deptId パラメーターに直接対応しています。パラメーターのバインディングを行う場合は、パラメーターなしの send() メソッドを使用してサービスを呼び出します。この例では WebService コンポーネントの destination プロパティを示していますが、サービスを呼び出す方法は、サービスに直接接続する場合でも宛先を経由する場合でも同じです。

```
<?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>
```

SOAP 要求のボディ全体を XML 内で手動で指定し、すべての正しい名前空間情報を <mx:request> タグで定義することもできます。これには、次の例のように、<mx:request> タグの format 属性の値を xml に設定します。

```
<?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>
```

サービス結果の処理

RPC コンポーネントでサービスを呼び出した後、サービスから返されたデータは lastResult オブジェクトに配置されます。デフォルトで、HTTPService コンポーネントおよび WebService コンポーネントの操作の resultFormat プロパティ値は object であり、返されるデータは ActionScript オブジェクトの単純なツリーとして表現されます。Flex は、Web サービスまたは HTTP サービスから返された XML データを、String、Number、Boolean および Date などの基本型を適切に表すように解釈します。厳密に型指定されたオブジェクトを処理するには、Flex で作成されたオブジェクトツリーを使用してオブジェクトを設定します。

WebService コンポーネントと HTTPService コンポーネントはいずれも、複合型である匿名のオブジェクトおよび配列を返します。makeObjectsBindable が true（デフォルト値）の場合、オブジェクトは mx.utils.ObjectProxy インスタンスにラップされ、配列は mx.collections.ArrayCollection インスタンスにラップされます。

注意：ColdFusion では大文字と小文字が区別されないため、ColdFusion のデータはすべて大文字に変換されます。ColdFusion の Web サービスを受信する場合は、このことを覚えておいてください。

result イベントと fault イベントの処理

サービスの呼び出しが完了すると、RemoteObject のメソッド、WebService の操作または HTTPService のコンポーネントから result イベントまたは fault イベントが送出されます。**result** イベントは、結果が使用可能であることを示します。**fault** イベントは、エラーが発生したことを示します。result イベントは、lastResult にバインドされているプロパティを更新するトリガーの役割を果たします。fault イベントおよび result イベントは、イベントリスナーを RemoteObject のメソッドまたは WebService の操作に追加することにより、明示的に処理できます。HTTPService コンポーネントの場合は、result および fault のイベントリスナーをコンポーネント自体で指定します。これは、HTTPService コンポーネントには複数の操作やメソッドがないためです。

RemoteObject のメソッドまたは WebService の操作で result イベントまたは fault イベントのイベントリスナーを指定しない場合、イベントはコンポーネントレベルに渡されます。コンポーネントレベルの result および fault のイベントリスナーを指定できます。

次の MXML の例では、WebService 操作の result および fault イベントでイベントリスナーを指定しています。また、WebService コンポーネントの fault イベントでもイベントリスナーを指定しています。

```
<?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>
```

次の ActionScript の例では、result イベントリスナーを WebService 操作に追加し、fault イベントリスナーを WebService コンポーネントに追加します。

```
<?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>
```

mx.rpc.events.InvokeEvent イベントを使用して、データアクセスコンポーネント要求がいつ呼び出されたのかを示すこともできます。これは、操作をキューに入れて後から呼び出す場合に便利です。

結果を e4x 結果形式の XML として処理する方法

HTTPService コンポーネントおよび WebService 操作の resultFormat プロパティ値を e4x に設定すると、XML 型の lastResult プロパティを作成できます。lastResult プロパティには、ECMAScript for XML（E4X）式を使用してアクセスできます。バインド式で XML オブジェクト E4X を使用する場合は、ドット表記に XML 構造のルートノードを含めないようにします。それに対し、lastResult プロパティを object に設定する場合は、ドット表記に XML 構造のルートノードを含めます。つまり、lastResult プロパティが e4x に設定されている場合は、{srv.lastResult.product} を使用し、object に設定されている場合は、{srv.lastResult.products.product} を使用します。

XML を直接処理する場合は、結果のフォーマットとして e4x を使用することを推奨しますが、resultFormat プロパティを xml に設定して flash.xml.XMLNode 型の lastResult オブジェクト（XML 処理用の従来のオブジェクト）を作成することもできます。また、HTTPService コンポーネントの resultFormat プロパティを flashvars または text に設定することで、前者は名前と値のペアを含む ActionScript オブジェクトとして、後者は純粋なテキストだけを含む ActionScript オブジェクトとして、それぞれ結果を作成できます。

注意：サービス結果で E4X の構文を使用する場合は、HTTPService または WebService コンポーネントの resultFormat プロパティを e4x に設定する必要があります。デフォルト値は object です。

HTTPService コンポーネントまたは WebService 操作の resultFormat プロパティを e4x に設定すると、返される XML に含まれる名前空間情報を処理しなければならない場合があります。WebService コンポーネントの場合は、Web サービスから返される SOAP エンベロープの本体に名前空間情報が含まれます。次の例は、名前空間情報が含まれている SOAP 本体の一部を示しています。このデータは、株式情報を取得する Web サービスから返されたものです。名前空間情報は太字テキストで示しています。

```
...
<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>
...
```

この soap:Body には名前空間情報が含まれているため、WebService 操作の resultFormat プロパティを e4x に設定する場合は、http://ws.invesbot.com/ 名前空間用の名前空間オブジェクトを作成します。次の例は、これを行うアプリケーションを示しています。

```
<?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>
```

オプションで、次の例のように、名前空間用の var を作成し、サービス結果へのバインディングでこの var にアクセスすることもできます。

```
<?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>
```

lastResult オブジェクトで返される XML のエレメントおよび属性にアクセスするには、E4X 構文を使用します。使用する構文は、XML で名前空間が宣言されているかどうかで異なります。

名前空間なし

次の例は、エレメントまたは属性に関して名前空間が指定されていない場合に、そのエレメントまたは属性の値を取得する方法を示しています。

```
var attributes:XMLList = XML(event.result).Description.value;
```

上記のコードは、次の XML ドキュメントの xxx を返します。

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <Description>
        <value>xxx</value>
    </Description>
</RDF>
```

任意の名前空間

次の例は、エレメントまたは属性に関して任意の名前空間が指定されている場合に、そのエレメントまたは属性の値を取得する方法を示しています。

```
var attributes:XMLList = XML(event.result).*::Description.*::value;
```

上記のコードは、次のいずれかの XML ドキュメントの xxx を返します。

XML ドキュメント 1：

```
<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 ドキュメント 2：

```
<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>
```

特定の名前空間

次の例は、エレメントまたは属性に関して宣言された rdf 名前空間が指定されている場合に、そのエレメントまたは属性の値を取得する方法を示しています。

```
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;
```

上記のコードは、次の XML ドキュメントの xxx を返します。

```
<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>
```

次の例は、宣言された rdf 名前空間がエレメントや属性で指定されている場合にエレメント値や属性値を取得する別の方法を示しています。

```
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;
```

上記のコードも、次の XML ドキュメントの xxx を返します。

```
<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>
```

.NET DataSet または DataTable を含む Web サービス結果の処理

Microsoft .NET Framework を使用して記述された Web サービスでは、特殊な .NET DataSet または DataTable オブジェクトをクライアントに返すことができます。.NET Web サービスでは、操作するデータ型に関する情報なしで、基本的な WSDL ドキュメントが提供されます。Web サービスから DataSet または DataTable が返されると、データ型の情報が SOAP メッセージの XML スキーマエレメントに埋め込まれ、これによってメッセージの残りの部分の処理方法が指定されます。この種の Web サービスからの結果を最適に処理するには、Flex WebService 操作の resultFormat プロパティを object に設定します。オプションで、WebService 操作の resultFormat プロパティを e4x に設定できますが、データを例えば DataGrid コントロールにバインドする場合など、応答と実装の対処方法の構造が通常とは異なるため、XML 形式と e4x 形式は不便です。

Flex WebService 操作の resultFormat プロパティを object に設定すると、.NET Web サービスから返された DataTable または DataSet は、1 つ以上の DataTable オブジェクトのマップが含まれる Tables プロパティを持つオブジェクトに自動的に変換されます。Tables マップの DataTable オブジェクトにはそれぞれ、Columns と Rows の 2 つのプロパティが含まれます。Rows プロパティにはデータが含まれます。event.result オブジェクトは、.NET の DataSet プロパティおよび DataTable プロパティに対応する次のプロパティを取得します。DataSet や DataTable の配列は、ここで説明する構造と同じですが、結果オブジェクトの最上位の配列にネストされています。

プロパティ	説明
`result.Tables`	テーブルデータを含むオブジェクトへのテーブル名のマップ。
`result.Tables["someTable"].Columns`	列名の配列（テーブルの DataSet スキーマまたは DataTable スキーマで指定されている順）。
`result.Tables["someTable"].Rows`	テーブルの各行のデータを表すオブジェクトの配列。例えば、{columnName1:value, columnName2:value, columnName3:value} など。

次の MXML アプリケーションでは、DataGrid コントロールに、.NET Web サービスから返された DataTable データが指定されます。

```
<?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>
```

次の例は、アプリケーションによって呼び出されるバックエンド Web サービスの実装である .NET C# クラスを示しています。このクラスでは、Microsoft SQL Server の Northwind サンプルデータベースを使用します。

```
<%@ 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;
    }
}
```