Accès aux données avec ADOBE® FLEX® 4.6

Informations juridiques

Pour consulter les informations juridiques, rendez-vous à l'adresse suivante : http://help.adobe.com/fr_FR/legalnotices/index.html.

Dernière mise à jour le 16/11/2012

Sommaire

Chapitre 1 : Présentation de l'accès aux services de données

Accès aux données dans Flex comparé à d'autres technologies

Flex n'utilise pas les sources de données et les données de la même manière que les applications dont l'interface utilisateur fait appel à HTML.

Traitement côté client et traitement côté serveur

Contrairement aux ensembles de modèles HTML créés à l'aide de servlets, d'environnements JSP, ASP, PHP ou CFML, Flex sépare le code client du code serveur. L'interface utilisateur de l'application est compilée dans un fichier binaire SWF envoyé au client.

Lorsque l'application effectue une demande à un service de données, le fichier SWF n'est pas recompilé et aucune réactualisation de page n'est requise. Le service distant ne renvoie que des données. Flex lie les données renvoyées à des composants de l'interface utilisateur dans l'application client.

Lorsqu'un utilisateur clique sur le contrôle Button d'une application par exemple, le code côté client appelle un service Web. Les données de résultat provenant du service Web sont renvoyées dans le fichier SWF binaire sans réactualisation de page. Les données de résultat peuvent alors être utilisées comme contenu dynamique dans l'application.

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

Comparons cet exemple Flex à l'exemple suivant, dans lequel du code JSP est utilisé pour appeler un service Web avec une balise personnalisée JSP. Lorsqu'un utilisateur demande le code JSP, la demande du service Web est effectuée sur le serveur et non pas sur le client. Le résultat est utilisé pour générer du contenu dans la page HTML. Le serveur d'application régénère la page HTML entière avant de la renvoyer au navigateur Web de l'utilisateur.

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

Accès à la source de données

Une autre différence entre Flex et les autres technologies d'application Web réside dans l'absence (dans Flex) de communication directe avec une source de données. Vous utilisez un composant d'accès aux données pour vous connecter à un service distant et interagir avec la source de données côté serveur.

L'exemple suivant présente une page ColdFusion accédant directement à une source de données :

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

Une fonctionnalité similaire est obtenue dans Flex en ayant recours à un service HTTPService, un service Web ou un composant RemoteObject afin d'appeler un objet côté serveur qui renvoie les résultats d'une source de données.

Evénements, appels de service et liaison de données

Flex est une technologie orientée événement. Une action de l'utilisateur ou un événement de programme peut déclencher l'accès à un service. Par exemple, l'utilisateur cliquant sur un bouton constitue un événement d'action qui peut être utilisé pour déclencher un appel de service. Un événement de programme peut être une application terminant la création d'un composant de l'interface utilisateur tel qu'un composant DataGrid. L'événement creationComplete pour le composant DataGrid peut être utilisé pour appeler un service distant qui renseignera ce composant.

Dans Flex, les appels de service sont asynchrones. L'application client n'a pas besoin d'attendre les données renvoyées. Les appels de service asynchrones présentent des avantages lors de l'extraction ou de la mise à jour de jeux de données volumineux. L'application client n'est pas bloquée par l'attente de l'extraction ou de la mise à jour des données.

Les données renvoyées par un appel de service sont stockées dans une propriété CallResponder associée à l'appel de service. Les composants de l'interface utilisateur font alors appel à la liaison de données pour extraire de la propriété CallResponder les données renvoyées.

La liaison de données dans Flex vous permet de mettre à jour dynamiquement un composant de l'interface utilisateur avec une source de données. Par exemple, un composant Flex peut associer son attribut text à l'attribut lastResult d'une propriété CallResponder. En cas de modification des données de CallResponder, le composant Flex est automatiquement mis à jour.

Flex implémente également la liaison de données bidirectionnelle qui garantit la mise à jour automatique d'un composant Flex ou d'une source de données dont les données sont modifiées. Un autre avantage de la liaison de données bidirectionnelle réside dans la mise à jour de données distantes à partir des saisies effectuées par un utilisateur dans un composant Form ou dans un composant de données Flex.

Voir aussi

« Création d'applications centrées sur les données avec Flash Builder » à la page 8

Utilisation de Flash Builder pour accéder aux services de données

Flex Builder 3 permet d'implémenter des appels de procédures distants à des services de données à l'aide des composants d'accès aux données Flex. Cependant, Flash Builder simplifie ce processus.

Flash Builder fournit des assistants et des outils permettant d'effectuer les opérations suivantes :

- Accès aux services de données
- Configuration des données renvoyées par le service de données
- Assistance pour la pagination des données renvoyées par le service
- Assistance pour la fonctionnalité de gestion des données qui synchronise plusieurs mises à jour des données du serveur
- Génération de code client pour l'accès aux services de données
- Liaison des données renvoyées par le service aux composants de l'interface utilisateur

Flux de travaux Flash Builder pour l'accès aux services

Appliquez le flux de travaux suivant lorsque vous utilisez Flash Builder pour créer une application qui accède à des services de données.

1 Selon les circonstances, vous commencerez soit par vous connecter à un service de données, soit par construire l'interface utilisateur.

Connexion au service distant : si vous commencez par vous connecter au service distant, vous devez ensuite construire l'interface utilisateur.

Construction de l'interface utilisateur : si vous commencez par construire l'interface utilisateur, vous devez ensuite vous connecter au service distant.

Remarque : *la première action effectuée relève d'un choix personnel. Par exemple, si la conception d'une interface utilisateur est déjà prévue, vous pouvez commencer par construire l'interface utilisateur. A l'inverse, vous pouvez commencer par vous connecter aux données et laisser Flash Builder vous aider à générer les composants d'application.*

2 Liez les opérations de données à des composants d'applications.

3 (Facultatif) Gérez l'extraction et la mise à jour des données.

Les outils Flash Builder vous permettent d'implémenter la pagination des données renvoyées et de coordonner la mise à jour des jeux de données.

Lors du retour de gros volumes d'enregistrements de données, vous implémentez généralement la pagination pour extraire un jeu d'enregistrements selon vos besoins.

Pour les applications mettant à jour plusieurs enregistrements, vous pouvez implémenter des fonctions de gestion de données. Les fonctions de gestion de données incluent :

- Fonction de validation pour la mise à jour simultanée des enregistrements modifiés
- Mécanisme d'annulation des modifications avant leur écriture sur le serveur
- Génération de code pour la mise à jour automatique des composants d'interface utilisateur au fur et à mesure que des enregistrements sont ajoutés, supprimés ou modifiés

4 Exécutez l'application et surveillez le flux de données.

Une fois l'application terminée, exécutez-la pour en observer le fonctionnement. Utilisez le Moniteur de réseau Flash Builder pour afficher les données transmises entre l'application et le service. Le Moniteur de réseau est utile pour diagnostiquer les erreurs et analyser les performances.

Flash Builder fournit également des environnements robustes de débogage et de définition de profil. Le Moniteur de réseau et le profileur Flash sont disponibles avec Flash Builder Premium.

Voir aussi

« Création d'applications centrées sur les données avec Flash Builder » à la page 8

Développement des services pris en charge par Flash Builder

Les assistants et les outils Flash Builder prennent en charge l'accès aux implémentations des types de services suivants :

- Services PHP
- Services ColdFusion
- BlazeDS
- LiveCycle Data Services
- Services HTTP (de style REST)
- Services Web (SOAP)
- Fichiers XML statiques

Si vous avez besoin d'une prise en charge d'outils pour d'autres types de services (Ruby on Rails, par exemple), vous pouvez développer l'implémentation Flash Builder. Voir Référence d'extensibilité Flash Builder.

Composants d'accès aux données

Les composants d'accès aux données permettent à une application client d'appeler des opérations et des services à travers un réseau. Les composants d'accès aux données utilisent des appels de procédure distante pour interagir avec les environnements de serveur. Les trois composants d'accès aux données sont les composants RemoteObject, HTTPService et WebService.

Les composants d'accès aux données sont conçus pour les applications client dans lesquelles un modèle d'appel et de réponse constitue un bon choix pour accéder aux données externes. Ces composants permettent au client d'effectuer des demandes asynchrones à des services distants qui traitent les demandes, puis de renvoyer les données à l'application.

Un composant d'accès aux données appelle un service distant. Il stocke ensuite les données de réponse du service dans un objet ActionScript ou tout autre format renvoyé par le service. Utilisez les composants d'accès aux données dans l'application client pour qu'elle fonctionne avec trois types de services :

- Services d'objets distants (RemoteObject)
- Services Web basés sur SOAP (WebServices)
- Services HTTP, incluant les services Web basés sur REST (HTTPService)

Adobe® Flash® Builder™ fournit des assistants et des outils permettant d'envelopper l'implémentation d'un composant d'accès aux données dans une enveloppe de service. L'enveloppe de service encapsule la fonctionnalité du composant d'accès aux données, rendant invisible une grande partie de l'implémentation de niveau inférieur et vous permettant ainsi de vous concentrer sur l'implémentation de services et la génération d'applications client pour accéder à ces services. Pour plus d'informations sur l'utilisation de Flash Builder pour accéder aux services de données, voir « Création d'applications centrées sur les données avec Flash Builder » à la page 8.

Accès aux services

Par défaut, Adobe Flash Player bloque l'accès à tout hôte qui ne correspond pas exactement à celui utilisé pour charger une application. Si vous n'utilisez pas LiveCycle Data Services ou BlazeDS pour traiter les demandes par proxy, un service HTTP ou Web doit résider sur le serveur hébergeant l'application ou le serveur distant hébergeant le service HTTP ou Web doit définir un fichier `crossdomain.xml`. Le fichier `crossdomain.xml` permet à un serveur d'indiquer que ses données et ses documents sont disponibles pour les fichiers SWF traités à partir de certains domaines ou de tous les domaines. Le fichier `crossdomain.xml` doit figurer à la racine Web du serveur que l'application contacte.

Composants HTTPService

Utilisez les composants HTTPService pour l'envoi de requêtes HTTP GET ou POST et afin d'inclure les données de réponses HTTP dans une application client. Si vous utilisez Flex pour créer des applications d'ordinateur (exécutées dans Adobe AIR®), les requêtes HTTP PUT et DELETE sont prises en charge.

Si vous utilisez LiveCycle Data Services ou BlazeDS, vous pouvez faire appel à un HTTPProxyService grâce auquel vous pourrez utiliser des méthodes HTTP supplémentaires. Le service HTTPProxyService vous permet d'envoyer des requêtes GET, POST, HEAD, OPTIONS, PUT, TRACE ou DELETE.

Un service HTTP peut consister en n'importe quel URI HTTP qui accepte des demandes HTTP et envoie des réponses. Un autre nom courant pour ce type de service est service Web de style REST. REST signifie REpresentational State Transfer, un style architectural pour les systèmes hypermédia distribués.

Les composants HTTPService s'avèrent utiles lorsque vous ne pouvez pas exposer la même fonctionnalité en tant que service Web SOAP ou service d'objet distant. Vous pouvez par exemple utiliser des composants HTTPService pour interagir avec des pages JavaServer (JSP), des servlets et des pages ASP qui ne sont pas disponibles comme services Web ou destinations de service distant.

Lorsque vous appelez la méthode `send()` de l'objet HTTPService, elle effectue une demande HTTP auprès de l'URI spécifié et une réponse HTTP est renvoyée. Si vous le souhaitez, vous pouvez transmettre des arguments à l'URI spécifié.

Flash Builder fournit des flux de travaux qui vous permettent de vous connecter de manière interactive à des services HTTP. Pour plus d'informations, voir « Accès aux services HTTP » à la page 13.

Voir aussi

« Accès aux services HTTP » à la page 13

Thèse : Representational State Transfer (REST) par Roy Thomas Fielding

Composants WebService

Les composants WebService vous permettent d'accéder aux services Web SOAP (modules logiciels comportant des méthodes). Les méthodes de service Web sont aussi appelées *opérations*. Les interfaces de service Web sont définies à l'aide du langage WSDL (Web Services Description Language). Les services Web fournissent aux modules logiciels exécutés sur différentes plateformes une méthode conforme aux standards leur permettant d'interagir entre eux. Pour plus d'informations sur les services Web, voir la section relative aux services Web du site World Wide Web Consortium, www.w3.org/2002/ws/.

Les applications client peuvent interagir avec des services Web qui définissent leurs interfaces dans un document WSDL, disponible en tant qu'URL. WSDL est un format standard permettant de décrire les messages qu'un service Web comprend, le format des réponses de ce service à ces messages, les protocoles que le service Web prend en charge et l'adresse à laquelle envoyer les messages.

Flex prend en charge WSDL 1.1, décrit à l'adresse www.w3.org/TR/wsdl, ainsi que les services Web codés RPC et littéral document.

Il prend également en charge les demandes et les résultats de service Web formatés en tant que messages SOAP et transportés via HTTP. SOAP fournit la définition du format XML que vous pouvez utiliser pour échanger des informations structurées et typées entre un client de service Web (une application créée avec Flex, par exemple) et un service Web.

Vous pouvez utiliser un composant WebService pour vous connecter à un service Web SOAP lorsque les services Web constituent un standard établi dans l'environnement. Les composants WebService sont également utiles pour les objets situés dans un environnement d'entreprise mais qui ne sont pas nécessairement disponibles sur le chemin source de l'application Web.

Flash Builder fournit des flux de travaux qui vous permettent de vous connecter de manière interactive à des services Web. Pour plus d'informations, voir « Accès aux services Web » à la page 16.

Composants RemoteObject

Les services d'objets distants vous permettent d'accéder à la logique commerciale directement dans son format natif au lieu de la formater en tant que XML, comme vous le faites avec les services Web ou de style REST. Vous économisez ainsi le temps nécessaire pour exposer la logique existante au format XML. Un autre avantage des services d'objets distants réside dans la vitesse de communication à travers le réseau. Les échanges de données s'effectuent encore via HTTP ou https, mais les données elles-mêmes sont sérialisées dans une représentation binaire. L'utilisation de composants RemoteObject réduit le volume de données qui transitent sur le réseau, diminue la mémoire utilisée côté client et accélère le traitement.

ColdFusion, PHP, BlazeDS et LiveCycle Data Services peuvent utiliser une définition de type côté serveur lors de l'accès à des données du serveur. L'application client accède à un objet Java, à un composant ColdFusion (qui constitue un objet Java en interne) ou à une classe PHP directement par l'invocation distante d'une méthode sur un objet désigné. L'objet sur le serveur utilise ses propres types de données natifs en tant qu'arguments, interroge une base de données avec ces arguments, puis renvoie les valeurs dans leurs types de données natifs.

Lorsque la définition de type côté serveur n'est pas disponible, Flash Builder dispose d'outils pour implémenter la définition de type côté client. Utilisez Flash Builder afin de configurer et de définir des types pour les données renvoyées du service. La définition de type côté client permet à l'application client d'interroger une base de données et d'extraire des données correctement typées. La définition de type côté client est requise pour un service qui ne définit pas le type de données renvoyées par le service.

Flash Builder fournit des flux de travaux qui vous permettent de vous connecter de manière interactive à des services d'objets distants. Pour plus d'informations, voir « Connexion à des services de données » à la page 9.

Chapitre 2 : Création d'applications centrées sur les données avec Flash Builder

Les outils Flash Builder peuvent vous aider à créer des applications accédant à des services de données. Commencez par créer un projet Flex pour vos applications. Vous pouvez ensuite établir une connexion à un service de données, configurer l'accès aux données du service, puis créer une interface utilisateur pour une application. Dans certains cas, vous devez commencer par créer l'interface utilisateur, puis accéder au service de données.

Création d'un projet Flex pour l'accès aux services de données

Flex accède aux services de données en tant qu'objet distant, service HTTP (de type REST) ou service Web SOAP.

Utilisez un objet distant pour accéder aux types de services de données suivants :

- Services ColdFusion
- Services PHP au format AMF
- BlazeDS
- LiveCycle Data Services

Pour plus d'informations sur l'utilisation de l'assistant LiveCycle Service Discovery, voir Utilisation de LiveCycle Discovery.

Pour tout service auquel l'accès s'effectue par le biais d'un objet distant, créez un projet Flex configuré pour le type de serveur d'application approprié. L'assistant de nouveau projet Flex vous guide au cours des étapes de configuration d'un projet pour les types de serveurs d'applications répertoriés ci-dessous.

Type de serveur	**Services d'objets distants pris en charge**
PHP	• Services PHP au format AMF
ColdFusion	• ColdFusion Flash Remoting • BlazeDS • LiveCycle Data Services
J2EE	• BlazeDS • LiveCycle Data Services

Vous pouvez vous connecter à des services HTTP et Web SOAP à partir de n'importe quel projet Flex, y compris les projets sans technologie de serveur spécifiée.

Un projet configuré pour accéder à un objet distant peut accéder uniquement à un service d'objets distants pour lequel il est configuré. Par exemple, vous ne pouvez pas accéder à un service PHP basé sur AMF à partir d'un projet configuré pour ColdFusion. Cependant, vous pouvez vous connecter à un service PHP à partir d'un tel projet si vous vous y connectez en tant que service Web ou HTTP.

Voir aussi

« Présentation de l'accès aux services de données » à la page 1

Changement du type de serveur d'un projet

Flash Builder vous avertit si vous tentez d'accéder à un service pour lequel aucun projet Flex n'est configuré. Si le projet Flex ne spécifie pas la configuration de serveur correcte, Flash Builder fournit un lien vers la boîte de dialogue Propriétés du projet. Cette boîte de dialogue vous permet de configurer le projet afin d'accéder au service de données. Flash Builder vous avertit par exemple lorsque vous tentez d'accéder à un service PHP basé sur AMF à partir d'un projet qui ne spécifie aucune configuration de serveur.

Si le projet Flex a été précédemment configuré pour l'accès à un autre type de service, configurez un nouveau projet Flex ou modifiez la configuration du projet actuel. En cas de modification de la configuration de serveur d'un projet, vous ne pourrez plus accéder aux services précédemment configurés. Par exemple, si vous modifiez une configuration de projet de ColdFusion vers PHP, les services ColdFusion auxquels vous tenterez d'accéder dans le projet ne seront plus disponibles.

Vous pouvez accéder à différents types de services à partir du même projet en configurant des services en tant que services HTTP ou Web.

Fichier de régulation interdomaines

Un fichier de régulation interdomaines est requis pour l'accès aux services se trouvant sur un autre domaine à partir du fichier SWF pour l'application. Situés dans le même domaine que l'application, les services au format AMF ne nécessitent généralement pas de fichier de régulation interdomaines.

Connexion à des services de données

Utilisez l'assistant de service Flash Builder pour vous connecter à un service de données.

Pour les services d'objets distants, vous créez généralement un projet Flex ayant un type de serveurs d'applications correspondant. Flash Builder introspecte le service et peut configurer des types de retour pour les données renvoyées par le service.

Les services d'objets distants incluent les services de données implémentés dans ColdFusion, PHP, BlazeDS et LiveCycle Data Services.

Pour plus d'informations sur l'utilisation de l'assistant LiveCycle Service Discovery, voir Utilisation de LiveCycle Discovery.

Voir aussi

« Création d'un projet Flex pour l'accès aux services de données » à la page 8

Accès aux services ColdFusion

Utilisez l'assistant de service Flash Builder pour accéder au service de données ColdFusion implémenté en tant que composant ColdFusion (CFC). Flex accède à ces services en qualité d'objets distants.

Utilisez un projet Flex spécifiant ColdFusion comme type de serveurs d'applications. Lors de la création du projet Flex, activez l'option Utiliser le service d'accès aux objets distants et sélectionnez ColdFusion Flash Remoting.

Connexion à des services de données ColdFusion

Cette procédure part du principe que vous avez implémenté un service ColdFusion et créé un projet Flex afin d'accéder aux services ColdFusion.

1 Dans le menu Données de Flash Builder, sélectionnez Connexion à ColdFusion pour ouvrir l'assistant de connexion à un service.

2 Dans la boîte de dialogue Configuration du service ColdFusion, recherchez l'emplacement du fichier CFC qui implémente le service.

Remarque : *si vous n'avez pas implémenté de service ColdFusion, Flash Builder peut générer un exemple de service à partir d'un tableau de base de données simple. Utilisez l'exemple généré en guise d'illustration de l'accès aux services de données. Voir « Génération d'un exemple de service ColdFusion à partir d'un tableau de base de données » à la page 10.*

3 (Facultatif) Modifiez les détails du service.

Nom du service	Spécifiez un nom pour le service. Flash Builder génère un nom pour le service, basé le nom du fichier du service. Les noms possibles pour un service sont soumis à certaines restrictions. Voir « Affectation de noms aux services de données » à la page 20.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Package du type de données	Spécifiez un nom pour le package contenant les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `valueObjects`.

4 (Facultatif) Cliquez sur Suivant pour afficher les opérations de service.

5 Cliquez sur Terminer pour générer des fichiers ActionScript qui accèdent au service.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Prochaine étape : « Configuration des types de données pour les opérations de service de données » à la page 27.

Génération d'un exemple de service ColdFusion à partir d'un tableau de base de données

Flash Builder peut générer un exemple de service ColdFusion que vous pouvez utiliser en tant que prototype pour vos propres services. L'exemple de service accède à un tableau de base de données simple et dispose de méthodes de création, de lecture, de mise à jour et de suppression.

Flash Builder configure les types de données de retour pour les services générés et active des fonctionnalités d'accès aux données telles que la pagination ou la gestion de données.

Important : *utilisez le service généré uniquement dans un environnement de développement fiable. Le code généré permet à toute personne disposant d'un accès réseau à votre serveur d'accéder aux données du tableau de données et de les modifier ou supprimer. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour plus d'informations sur la rédaction de services sécurisés, voir Sécurisation des services de données.*

La procédure suivante part du principe que vous avez créé un projet Flex pour accéder aux services ColdFusion et disposez de sources de données ColdFusion.

1 Dans le menu Données de Flash Builder, sélectionnez Connexion à ColdFusion pour ouvrir l'assistant de connexion à un service.

2 Dans la boîte de dialogue Configuration du service ColdFusion, cliquez sur le lien afin de générer un exemple de service.

3 Sélectionnez l'option Générer à partir d'une source de données RDS et indiquez une source de données et une table ColdFusion.

   Si la table ne définit pas de clé primaire, sélectionnez-en une.

   Remarque : *en l'absence de source de données ColdFusion disponible, sélectionnez l'option Générer à partir d'un modèle. Flash Builder rédige un exemple de composant ColdFusion (CFC) avec des opérations de service courantes. Supprimez les marques de commentaires de certaines fonctions du CFC et modifiez les opérations pour créer un exemple de service que vous pourrez utiliser comme prototype.*

4 Utilisez l'emplacement par défaut ou spécifiez-en un nouveau. Cliquez sur OK.

   Flash Builder génère l'exemple de service. Modifiez le nom du service et les emplacements de package afin de remplacer les valeurs par défaut.

5 (Facultatif) Cliquez sur Suivant pour afficher les opérations du service.

6 Cliquez sur Terminer.

   Flash Builder génère des fichiers ActionScript qui accèdent à l'exemple de service. Il ouvre également l'exemple de service dans un éditeur du système enregistré pour l'édition de fichiers ColdFusion CFC.

Accès aux services PHP

Utilisez l'assistant de service Flash Builder pour vous connecter à un service de données implémenté sous PHP. Flex utilise le format AMF (Action Message Format) pour sérialiser les données entre l'application client et le service de données. Flash Builder installe la structure Zend AMF pour fournir un accès aux services implémentés sous PHP. Voir « Installation de Zend Framework » à la page 21.

Accédez aux services de données PHP à partir d'un projet Flex dont le type de serveurs d'applications spécifié est PHP. Le service de données doit être disponible sous la racine Web définie à la configuration du projet pour PHP. Placez le service dans un répertoire de services, comme illustré ci-dessous :

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

Voir aussi

« Création d'un projet Flex pour l'accès aux services de données » à la page 8

Connexion à des services de données PHP

Cette procédure part du principe que vous avez implémenté un service PHP et créé un projet Flex afin d'accéder aux services PHP.

1 Dans le menu Données de Flash Builder, sélectionnez Connexion à PHP pour ouvrir l'assistant de connexion à un service.

2 Dans la boîte de dialogue Configuration du service PHP, recherchez le fichier PHP implémentant le service.

Remarque : *si vous n'avez pas implémenté de service PHP, Flash Builder peut générer un exemple de service à partir d'un tableau de base de données simple. Utilisez l'exemple généré en guise d'illustration de l'accès aux services de données. Voir « Génération d'un exemple de service PHP à partir d'un tableau de base de données » à la page 12.*

3 (Facultatif) Modifiez les détails du service.

Nom du service	Spécifiez un nom pour le service. Flash Builder génère un nom pour le service, basé le nom du fichier du service. Les noms possibles pour un service sont soumis à certaines restrictions. Voir « Affectation de noms aux services de données » à la page 20.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Package du type de données	Spécifiez un nom pour le package contenant les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `valueObjects`.

4 Cliquez sur Suivant pour afficher les opérations de service.

Si vous ne disposez pas de la version prise en charge de Zend Framework pour l'accès aux services PHP, Flash Builder vous invite à en installer la version minimale. Voir « Installation de Zend Framework » à la page 21.

5 Cliquez sur Terminer.

Flash Builder génère des fichiers ActionScript qui accèdent au service.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Prochaine étape : « Configuration des types de données pour les opérations de service de données » à la page 27.

Génération d'un exemple de service PHP à partir d'un tableau de base de données

Flash Builder peut générer un exemple de service PHP que vous pouvez utiliser en tant que prototype pour vos propres services. L'exemple de service accède à un tableau de base de données MySQL simple et dispose de méthodes de création, de lecture, de mise à jour et de suppression.

Flash Builder configure les types de données de retour pour les services générés et active des fonctionnalités d'accès aux données telles que la pagination ou la gestion de données.

Important : *utilisez le service généré uniquement dans un environnement de développement fiable. Le code généré permet à toute personne disposant d'un accès réseau à votre serveur d'accéder aux données du tableau de données et de les modifier ou supprimer. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour plus d'informations sur la rédaction de services sécurisés, voir Sécurisation des services de données.*

La procédure suivante part du principe que vous avez créé un projet Flex pour accéder aux services PHP et disposez de sources de données MySQL.

1 Dans le menu Données de Flash Builder, sélectionnez Connexion à PHP pour ouvrir l'assistant de connexion à un service.

2 Dans la boîte de dialogue Configuration du service PHP, cliquez sur le lien afin de générer un exemple de service.

3 Sélectionnez Générer à partir d'une base de données et spécifiez les informations de connexion à la base de données. Cliquez sur Connexion à la base de données.

Remarque : *en l'absence de source de données PHP disponible, sélectionnez l'option Générer à partir d'un modèle. Flash Builder rédige un exemple de projet avec des opérations de service courantes. Supprimez les marques de commentaires de certaines zones du projet et modifiez les opérations pour créer un exemple de service que vous pourrez utiliser comme prototype.*

4 Sélectionnez un tableau dans la base de données et spécifiez la clé primaire.

5 Utilisez l'emplacement par défaut ou spécifiez-en un nouveau. Cliquez sur OK.

Si vous ne disposez pas de la version prise en charge de Zend Framework pour l'accès aux services PHP, Flash Builder vous invite à en installer la version minimale. Voir « Installation de Zend Framework » à la page 21.

Flash Builder génère l'exemple de service. Modifiez le nom du service et les emplacements de package afin de remplacer les valeurs par défaut.

6 (Facultatif) Cliquez sur Suivant pour afficher les opérations du service.

7 Cliquez sur Terminer.

Flash Builder génère des fichiers ActionScript qui accèdent à l'exemple de service. Il ouvre également l'exemple de service dans un éditeur du système enregistré pour l'édition de fichiers PHP.

Accès aux services HTTP

Utilisez l'assistant de service Flash Builder pour vous connecter aux services HTTP basés sur REST. Vous pouvez vous connecter à des services HTTP à partir de n'importe quel projet Flex. Il n'est pas nécessaire de spécifier une technologie de serveur pour le projet.

Un fichier de régulation interdomaines est requis pour l'accès aux services dont le domaine ne correspond pas à celui du fichier SWF de l'application client. Voir Utilisation de fichiers de régulation interdomaines.

Configuration des services HTTP

Il existe plusieurs façons de configurer l'accès aux services HTTP basés sur REST. L'assistant de configuration du service HTTP prend en charge les options suivantes.

- URL de base en tant que préfixe

  Cette option s'avère pratique pour accéder à plusieurs opérations à partir d'un service unique. Si vous spécifiez une URL de base vers le service, il vous suffit ensuite de spécifier pour chaque opération le chemin relatif vers les opérations HTTP.

  L'utilisation d'une URL de base ne permet pas d'accéder à plusieurs services.

- URL avec paramètres de requête

  Lors de la spécification d'une URL vers une opération, vous pouvez inclure les paramètres de requête pour les opérations de service. L'assistant de configuration du service HTTP renseigne le tableau Paramètres avec chaque paramètre inclus dans l'URL de l'opération.

- Services RESTful avec des paramètres délimités

  Flash Builder prend en charge l'accès aux services RESTful qui utilisent des paramètres délimités au lieu du paramètre de requête GET. Par exemple, supposons que vous utilisez l'URL suivante pour accéder à un service RESTful :

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

  Utilisez des accolades ({}) pour spécifier les paramètres de l'URL de l'opération, Par exemple :

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

L'assistant de configuration du service HTTP renseigne ensuite le tableau Paramètres.

Nom	Type de données	Type de paramètre
items	String	URL
itemID	String	URL

Lorsque vous spécifiez des paramètres pour un service RESTful, Type de données et Type de paramètre sont toujours configurés en tant que String et URL respectivement.

Remarque : *vous pouvez associer des paramètres de service RESTful avec des paramètres de requête lors de la spécification de l'URL vers une opération.*

- Chemin vers un fichier local pour une URL d'opération

  Pour une URL d'opération, vous pouvez spécifier un chemin vers un fichier local qui implémente les services HTTP. Par exemple, spécifiez ce qui suit pour une URL d'opération :

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

- Ajout d'opérations GET et POST

  Vous pouvez ajouter des opérations supplémentaires lors de la configuration d'un service HTTP. Cliquez sur le bouton Ajouter du tableau Opérations.

  Spécifiez la méthode de l'opération (GET ou POST).

- Ajout de paramètres à une opération

  Vous pouvez ajouter des paramètres à l'opération que vous aurez préalablement sélectionnée dans le tableau Opérations. Sélectionnez une opération, puis cliquez sur le bouton Ajouter du tableau Paramètres.

  Spécifiez un nom et un type de données pour le paramètre ajouté. Le type de paramètre (GET ou POST) correspond à la méthode de l'opération.

- Type de contenu pour les opérations POST

  Pour les opérations POST, vous pouvez spécifier le type de contenu. Il peut s'agir de `application/x-www-form-urlencoded` ou de `application/xml`.

  Si vous sélectionnez `application/xml` comme type de contenu, Flash Builder génère un paramètre de requête non modifiable, dont le nom par défaut est strXML. Vous pourrez spécifier le paramètre actuel à l'exécution.

Nom	Type de données	Type de paramètre
strXML	String	POST

Vous ne pouvez pas ajouter de paramètres supplémentaires pour le type de contenu `application/xml`.

Connexion à des services HTTP

1. Dans le menu Données de Flash Builder, sélectionnez Connexion à HTTP pour ouvrir l'assistant de connexion à un service.
2. (Facultatif) Spécifiez une URL de base à utiliser en tant que préfixe pour toutes les opérations.
3. Sous Opérations, spécifiez les éléments suivants pour chaque opération à laquelle vous souhaitez accéder :
   - Méthode de l'opération (GET ou POST)
   - URL de l'opération de service

Vous pouvez inclure dans l'URL n'importe quel paramètre d'opération. Utilisez des accolades ({}) pour spécifier des paramètres de service de style REST.

Flash Builder prend en charge l'accès aux protocoles suivants :

`http://`

`https://`

Chemins d'accès standard tels que `C:/` ou `/Applications/`

- Nom de l'opération

4 Spécifiez le nom et le type de données de chaque paramètre d'opération de l'URL sélectionnée.

5 (Facultatif) Cliquez sur Ajouter ou Supprimer pour ajouter ou supprimer des paramètres pour l'opération sélectionnée.

6 (Facultatif) Modifiez les détails du service.

Nom du service	Spécifiez un nom pour le service. Flash Builder génère un nom pour le service, basé le nom du fichier du service. Les noms possibles pour un service sont soumis à certaines restrictions. Voir « Affectation de noms aux services de données » à la page 20.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Package du type de données	Spécifiez un nom pour le package contenant les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `valueObjects`.

7 (Facultatif) Modifiez le nom du package généré pour le service.

8 Cliquez sur Terminer.

Flash Builder génère des fichiers ActionScript qui accèdent au service.

Une fois la connexion au service HTTP établie, configurez les types de retour des opérations de service. Lors de la configuration du type de retour, les types de paramètres de l'opération sont également configurés. Voir « Configuration des types de données pour les opérations de service de données » à la page 27.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Prochaine étape : « Configuration des types de données pour les opérations de service de données » à la page 27.

Accès à un fichier XML implémentant des services HTTP

Vous pouvez accéder à un fichier XML statique qui implémente un service HTTP. Ce fichier XML statique peut être un fichier local ou être disponible sous la forme d'une URL.

Le service utilise une méthode GET qui renvoie une réponse XML. Cette fonctionnalité est utile pour apprendre à connaître les services HTTP dans Flex et pour établir des prototypes de données fictives dans les applications client.

Lors de l'accès au service, spécifiez le nœud renvoyant la réponse XML. Flash Builder utilise ce nœud pour configurer automatiquement un type de retour pour les données. Une fois la connexion au service établie, vous pouvez lier des opérations du service à des composants d'interface utilisateur.

Connexion à un fichier de service XML

1 Dans le menu Données de Flash Builder, sélectionnez Connexion à HTTP pour ouvrir l'assistant de connexion à un service.

2 Spécifiez Fichier local ou URL, puis accédez au fichier.

3 Sélectionnez un nœud dans le fichier contenant la réponse souhaitée.

Indiquez si la réponse est un tableau Array.

Flash Builder configure un type de retour pour le nœud sélectionné.

4 Modifiez les détails du service.

Nom du service	Spécifiez un nom pour le service. Flash Builder génère un nom pour le service, basé le nom du fichier du service. Les noms possibles pour un service sont soumis à certaines restrictions. Voir « Affectation de noms aux services de données » à la page 20.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Package du type de données	Spécifiez un nom pour le package contenant les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `valueObjects`.

5 (Facultatif) Modifiez le nom du package généré pour le service.

6 Cliquez sur Terminer.

Flash Builder génère des fichiers ActionScript qui accèdent au service.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Accès aux services Web

Utilisez l'assistant de service Flash Builder pour vous connecter aux services Web (SOAP). Vous pouvez vous connecter à des services Web à partir de n'importe quel projet Flex. Il n'est pas nécessaire de spécifier une technologie de serveur pour le projet.

Un fichier de régulation interdomaines est requis pour l'accès aux services se trouvant sur un autre domaine à partir du fichier SWF pour l'application client.

Voir aussi

Utilisation des fichiers de régulation interdomaines

Connexion à des services Web

1 Dans le menu Données de Flash Builder, sélectionnez Connexion au service Web pour ouvrir l'assistant de connexion à un service.

2 (BlazeDS/Data Services) Si vous avez installé LiveCycle Data Services ou BlazeDS, vous pouvez accéder au service Web via un proxy.

Sélectionnez l'option Par une destination proxy BlazeDS/Data Services.

Spécifiez une destination. Cliquez sur Suivant et passez à l'étape 5.

Remarque : *l'option d'accès aux services Web via un proxy est activée uniquement si le type de serveurs d'applications spécifié pour le projet Flex est J2EE.*

3 Saisissez un URI pour le service SOAP.

4 (Facultatif) Modifiez les détails du service.

Nom du service	Spécifiez un nom pour le service. Flash Builder génère un nom pour le service basé sur l'URI WSDL. Les noms possibles pour un service sont soumis à certaines restrictions. Voir « Affectation de noms aux services de données » à la page 20.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Package du type de données	Spécifiez un nom pour le package contenant les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `dataValues`.

5 (Facultatif) Configurez la génération de code pour le service.

Service **Port**	Sélectionnez l'un des services disponibles. Flash Builder génère un nom pour le service basé sur l'URI WSDL.
Liste d'opérations	Sélectionnez les opérations du service auquel vous souhaitez accéder dans votre application client.

6 Cliquez sur Terminer.

Flash Builder génère des fichiers ActionScript qui accèdent au service.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Une fois la connexion au service Web établie, configurez les types de retour des opérations de service. Pour plus d'informations, voir « Configuration des types de données pour les opérations de service de données » à la page 27.

Accès à BlazeDS

Vous pouvez accéder aux services BlazeDS uniquement si vous avez installé Adobe® BlazeDS et configuré un serveur RDS (Remote Development Services). Pour plus d'informations sur l'installation et la configuration de BlazeDS, voir la documentation de LiveCycle Data Services ES.

En règle générale, vous accédez aux services de données BlazeDS à partir d'un projet Flex dont le type de serveurs d'applications spécifié est J2EE.

Voir aussi

« Création d'un projet Flex pour l'accès aux services de données » à la page 8

Connexion à des services BlazeDS

Cette procédure part du principe que vous avez installé BlazeDS, configuré un serveur de développement distant et créé un projet Flex afin d'accéder aux services BlazeDS.

1 Dans le menu Données de Flash Builder, sélectionnez Connexion à BlazeDS pour ouvrir l'assistant de connexion à un service.

2 Sélectionnez une destination à importer.

3 (Facultatif) Modifiez les détails du service.

Nom du service	Spécifiez un nom pour le service. Flash Builder génère un nom pour le service basé sur la destination. Les noms possibles pour un service sont soumis à certaines restrictions. Voir « Affectation de noms aux services de données » à la page 20.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Package du type de données	Spécifiez un nom pour le package contenant les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `valueObjects`.

4 Cliquez sur Terminer.

Flash Builder génère des fichiers ActionScript qui accèdent au service.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Accès à LiveCycle Data Services

Vous pouvez accéder aux services disponibles à partir de LiveCycle Data Services uniquement si vous avez installé LiveCycle Data Services et configuré un serveur RDS (Remote Development Services). Pour plus d'informations, voir la documentation de LiveCycle Data Services.

Vous pouvez accéder à LiveCycle Data Services à partir d'un projet Flex dont le type de serveur d'applications spécifié est soit J2EE soit ColdFusion.

Types de service pour LiveCycle Data Services

Lors de la connexion à LiveCycle Data Services, les types de services de données suivants sont disponibles en tant que destinations.

- Service distant

  Les services distants sont implémentés à l'aide de la définition de types AMF. Ils ne fournissent aucune gestion de données côté serveur. Vous pouvez utiliser les outils Flash Builder pour configurer la gestion de données côté client. Voir « Activation de la gestion des données » à la page 34.

- Service de données

  Les services de données implémentent la gestion de données côté serveur. Pour plus d'informations, voir la documentation LiveCycle Data Services.

- Service Web

  Les services Web sont disponibles par le biais d'un proxy LiveCycle Data Services configuré en tant que destination LiveCycle Data Services. La définition de type côté serveur n'est généralement pas fournie lors de la connexion à un service Web.

Configuration et gestion de types de données

Flash Builder fournit des outils pour la configuration et la gestion des données côté client. Les outils Flash Builder disponibles dépendent du type de destination LiveCycle Data Services.

- Service distant

  Les services distants implémentent la définition de types AMF sur le service. Il n'est pas nécessaire de configurer des types de données de retour pour les destinations de service distant.

  Cependant, vous pouvez utiliser Flash Builder pour générer du code pour la gestion de données côté client. Voir « Activation de la gestion des données » à la page 34.

- Service de données

  Les services de données implémentent les types de données côté serveur. Il n'est pas nécessaire de configurer des types de données de retour pour les destinations de service de données.

  Les destinations de service de données fournissent également la gestion des données côté serveur. Vous ne pouvez pas utiliser la gestion des données côté client avec les destinations de service de données.

- Service Web

  En règle générale, les destinations de service Web disponibles par le biais d'un proxy LiveCycle Data Service n'implémentent pas la définition de type côté serveur. Vous pouvez utiliser les outils Flash Builder pour configurer des types de retour pour les opérations de service Web. Voir « Configuration des types de données pour les opérations de service de données » à la page 27.

  Vous pouvez utiliser Flash Builder pour générer du code pour la gestion des données côté client. Voir « Activation de la gestion des données » à la page 34.

Connexion aux destinations LiveCycle Data Service (destinations de service de données et de service distant)

Cette procédure part du principe que vous avez installé LiveCycle Data Services, configuré un serveur de développement distant et créé un projet Flex afin d'accéder aux services LCDS.

1. Dans le menu Données de Flash Builder, sélectionnez Connexion aux données/services pour ouvrir l'assistant de connexion à un service.
2. Dans la boîte de dialogue Sélection d'un type de service, choisissez LCDS. Cliquez sur Suivant.
3. Saisissez votre identifiant de connexion, le cas échéant.
4. (Facultatif) Modifiez les détails du service.

Nom du service	N'indiquez aucun nom de service. Flash Builder génère un nom de service. Flash Builder génère un nom pour le service basé sur la destination.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Destinations	Spécifiez une ou plusieurs destinations disponibles à partir du serveur LiveCycle Data Services.
Package du type de données	Spécifiez un nom pour le package de type de données. Ce package contient les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `valueObjects`.

5. Cliquez sur Terminer.

   Flash Builder génère des fichiers ActionScript qui accèdent au service.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Connexion aux destinations LiveCycle Data Service (destinations de service Web)

Cette procédure part du principe que vous avez installé LiveCycle Data Services, configuré un serveur de développement distant (RDS) et créé un projet Flex pour l'accès aux services DS.

1. Dans le menu Données de Flash Builder, sélectionnez Connexion aux données/services pour ouvrir l'assistant de connexion à un service.
2. Dans la boîte de dialogue Sélection d'un type de service, choisissez Service Web. Cliquez sur Suivant.
3. Sélectionnez l'option Par une destination proxy LCDS/BlazeDS.
4. Saisissez votre identifiant de connexion, le cas échéant.
5. Sélectionnez la destination.
6. (Facultatif) Modifiez les détails du service. Cliquez sur Suivant.

Nom du service	Spécifiez un nom pour le service. Flash Builder génère un nom pour le service basé sur le nom de la destination. Les noms possibles pour un service sont soumis à certaines restrictions. Voir « Affectation de noms aux services de données » à la page 20.
Package de services	Spécifiez un nom pour le package contenant les fichiers ActionScript générés qui accèdent au service. Flash Builder génère un package basé sur le nom du service et le place dans un package `services`.
Package du type de données	Spécifiez un nom pour le package contenant les fichiers de classe ActionScript générés qui définissent les types de données récupérés à partir du service. Par défaut, Flash Builder crée le package `dataValues`.

7. (Facultatif) Configurez la génération de code pour le service.

Service **Port**	Sélectionnez un service et un port à partir des services et ports disponibles.
Liste d'opérations	Sélectionnez les opérations du service auquel vous souhaitez accéder dans votre application client.

8. Cliquez sur Terminer.

   Flash Builder génère des fichiers ActionScript qui accèdent au service.

Remarque : *une fois la connexion établie, vous pouvez modifier les propriétés du service. Sélectionnez le service dans la vue Données/Services. Dans le menu contextuel, sélectionnez Propriétés.*

Voir aussi

« Création d'un projet Flex pour l'accès aux services de données » à la page 8

Affectation de noms aux services de données

Il existe des restrictions sur les noms autorisés pour les services de données dont l'accès s'effectue à partir de Flash Builder. Certaines restrictions ne sont pas apparentes tant que vous ne compilez pas votre application.

Les règles d'attribution des noms pour les services sont :

- Le nom du service ne peut pas commencer par un nombre.

- Les noms de service ne peuvent pas être des mots-clés ActionScript.
- Vous ne pouvez pas utiliser un nom de classe ActionScript, y compris les classes personnalisées, en tant que nom de service.
- (PHP uniquement) Flash Builder ne peut pas importer un service dont le nom comporte des traits de soulignement.

Remarque : *il est recommandé d'utiliser des noms de service différant des noms de vos fichiers MXML.*

Installation de Zend Framework

Lors de l'accès initial aux services PHP, Flash Builder détermine si la version de Zend Framework prise en charge est installée. Si cette version est introuvable, Flash Builder vous invite à en confirmer l'installation. Si vous acceptez, Flash Builder installe une version minimale de Zend Framework. Si vous refusez, installez manuellement Zend Framework si vous souhaitez accéder aux services PHP.

Installation par défaut de Flash Builder

Flash Builder installe Zend Framework dans un dossier `ZendFramework` situé dans le répertoire racine de votre serveur Web :

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

Pour les projets Flex accédant aux services PHP, Flash Builder crée les fichiers de configuration suivants dans le dossier de sortie du projet :

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

Serveurs de production

Pour les serveurs de production, Adobe vous recommande de déplacer le dossier `ZendFramework` hors de la racine Web. Mettez à jour la variable `zend_path` définie dans le fichier `amf_config.ini`.

Si la variable `zend_path` est commentée, supprimez-en les marques de commentaire. Spécifiez l'emplacement de l'installation Zend Framework.

Installation manuelle de Zend Framework

Vous avez la possibilité d'installer manuellement Zend Framework.

1. Téléchargez la dernière version de Zend Framework.

   Vous pouvez installer le package minimal ou le package complet. Flash Builder installe le package minimal.

2. Extrayez la version téléchargée dans un emplacement du système.
3. Dans le dossier du projet Flex permettant d'accéder aux services PHP, mettez à jour la variable `zend_path` définie dans le fichier `amf_config.ini`.

   Si la variable `zend_path` est commentée, supprimez-en le commentaire. Spécifiez le chemin absolu vers l'emplacement de l'installation Zend Framework.

Dépannage d'une installation Zend Framework

Voici quelques conseils pour résoudre les erreurs susceptibles de survenir lors de la connexion à Zend Framework.

Installation manuelle de Zend Framework

Si vous avez installé manuellement Zend Framework, examinez la variable zend_path dans le fichier `amf_config.ini`.

Le fichier `amf_config.ini` se trouve dans le dossier de sortie du projet.

Vérifiez ce qui suit :

- `zend_amf` n'est pas commenté.
- Le chemin spécifié vers votre installation Zend Framework est correct :
  - Il s'agit d'un chemin absolu vers une destination située sur le système de fichiers local. Vous ne pouvez pas spécifier un chemin vers une ressource réseau mappée.
  - Le chemin mène au dossier de bibliothèque de l'installation Zend Framework. Habituellement, le dossier de bibliothèque est situé dans les emplacements suivants :

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

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

Installation de Zend Framework par Flash Builder

Si Zend Framework a été installé par Flash Builder, vérifiez ce qui suit :

- L'emplacement du dossier racine Web

  Flash Builder installe Zend Framework dans le dossier racine Web du projet. Vérifiez-en l'emplacement. Sélectionnez Projet > Propriétés > Serveur Flex.
- Assurez-vous que le serveur Web est configuré pour utiliser PHP.
- La variable zend_path du fichier `amf_config.ini`

  Le fichier `amf_config.ini` se trouve dans le dossier de sortie du projet.

  Vérifiez ce qui suit :

  - `zend_amf` n'est pas commenté.
  - Le chemin spécifié pointe vers l'installation Zend Framework à la racine Web du projet.
  - Il s'agit d'un chemin absolu vers une destination située sur le système de fichiers local. Vous ne pouvez pas spécifier un chemin vers une ressource réseau mappée.

Utilisation d'une instance de serveur unique

Une fois la connexion à un service de données établie, chaque application d'un projet peut accéder à ce service. Par défaut, chaque application crée sa propre instance de service lorsqu'elle accède au serveur.

Vous pouvez modifier ce comportement de sorte qu'un projet ne comporte qu'une seule instance de service. Chaque application dans le projet accède à la même instance de service. En règle générale, vous créez une instance de serveur unique lorsque vous souhaitez coordonner l'accès aux données à partir d'applications multiples.

Vous pouvez spécifier l'accès à une seule instance de service par projet ou en tant que préférence pour tous les projets.

Accès à une instance de serveur unique pour un projet

1 Cliquez sur Projet > Propriétés > Données/Services.

2 Sélectionnez la case à cocher pour l'utilisation d'une instance de serveur unique. Cliquez sur OK.

Spécification d'une instance de serveur unique en tant que préférence

1 Ouvrez la boîte de dialogue Préférences.

2 Sélectionnez Flash Builder > Données/Services.

3 Sélectionnez la case à cocher pour l'utilisation d'une instance de serveur unique. Cliquez sur OK.

Génération de l'application client

Vous utilisez l'éditeur de code MXML pour créer une interface utilisateur.

Après avoir défini les composants de l'application à l'aide de l'éditeur de code, vous pouvez lier les données renvoyées par le service aux composants de l'interface utilisateur. Générez des gestionnaires d'événement en fonction des interactions de l'utilisateur avec l'application.

Vous pouvez aussi générer un formulaire à partir des opérations de service disponibles dans la vue Données/Services.

Liaison d'opérations de service à des contrôles

Utilisez la boîte de dialogue Lier aux données pour lier une opération de service à un composant d'interface utilisateur.

La boîte de dialogue Lier aux données est disponible à partir du menu Données de la barre d'outils dans la vue Données/Services.

Lorsque vous liez une opération de service à un composant, Flash Builder génère du code MXML et ActionScript pour accéder à l'opération de service à partir de l'application client.

Types de retour pour les opérations de service

Lorsque vous liez une opération de service à un contrôle, Flash Builder utilise le type des données renvoyées par l'opération. En règle générale, vous configurez le type de retour pour une opération de service avant de la lier à un composant.

Si le type de retour d'une opération de service n'a pas été configuré, la boîte de dialogue Lier aux données vous invite à terminer cette étape.

Voir « Configuration des types de données pour les opérations de service de données » à la page 27.

Liaison d'un contrôle DataGrid à une opération de service (boîte de dialogue Lier aux données)

Cette procédure part du principe que vous êtes connecté à un service de données.

1 Dans la vue Structure, sélectionnez le contrôle DataGrid. Vous pouvez également placer votre curseur dans la balise <s:DataGrid> dans l'éditeur MXML.

2 Ouvrez ensuite la boîte de dialogue Lier aux données en sélectionnant l'option Lier aux données du menu Données de Flash Builder.

3 Sélectionnez Nouvel appel de service, puis un service et une opération.

Si vous avez lié une opération de service à un composant, vous pouvez utiliser ces résultats. Dans ce cas, spécifiez Résultat d'appel existant et sélectionnez l'opération à utiliser.

4 (Facultatif) Sélectionnez Modifier type de retour.

Sélectionnez Modifier type de retour si vous voulez reconfigurer le type de retour pour l'opération de service.

Si le type de retour de l'opération n'a pas été précédemment configuré, sélectionnez Configurer le type de retour.

Voir « Configuration des types de données pour les opérations de service de données » à la page 27.

5 Cliquez sur OK.

Le composant DataGrid change pour afficher les champs extraits de la base de données.

Voir Configuration de composants DataGrid et AdvancedDataGrid.

6 Enregistrez et exécutez l'application.

Génération d'un appel de service à une opération

Flash Builder peut générer une méthode ActionScript qui appelle une opération de service. Cette méthode n'est pas liée à un composant d'interface utilisateur, mais peut être utilisée dans le code de l'application.

Outre la méthode ActionScript, Flash Builder génère une propriété CallResponder donnant accès aux données renvoyées à partir de l'appel de service. Voir « CallResponder » à la page 38.

Génération d'un appel de service à une opération

Cette procédure part du principe que vous êtes connecté à un service de données.

1 Dans la vue Données/Services, sélectionnez une opération.

2 Dans le menu contextuel de l'opération, sélectionnez Générer l'appel de service.

Flash Builder génère une méthode pour appeler l'opération et affiche la méthode générée en mode Source dans l'éditeur MXML. Il crée une propriété CallResponder qui contient le résultat de l'appel de service.

Cette option est également disponible à partir de la barre d'outils Données/Services.

Création d'un formulaire pour une application

Les formulaires sont l'une des méthodes les plus courantes que les applications Web utilisent pour recueillir des informations des utilisateurs. Flash Builder peut générer des formulaires pour les données obtenues à partir des appels de service ou pour les types de données personnalisés utilisés pour accéder aux données distantes.

Lors de la génération d'un formulaire, Flash Builder crée un conteneur de présentation Form et ajoute des composants afin d'afficher ou de modifier les données spécifiques récupérées à partir du service.

Flash Builder génère les types de formulaires suivants.

Form	Description
Type de données	Ce formulaire contient les composants représentant les champs d'un type de données.
Détail des données principales	Le composant « principal » correspond en règle générale à un contrôle de données répertoriant les données obtenues d'un service. Le formulaire « détaillé » représente des éléments individuels sélectionnés dans le composant principal.
Appel de service	Créez deux formulaires. Le premier spécifie les entrées d'une opération. Le second affiche les données renvoyées.

Lors de la génération d'un formulaire, définissez les champs à inclure, spécifiez le type du contrôle d'interface utilisateur utilisé pour représenter chaque champ et indiquez si le formulaire est modifiable ou non.

Création d'un formulaire

Cette procédure indique comment générer un formulaire pour un appel de service. Les procédures pour générer d'autres types de formulaires sont similaires.

1 Pour exécuter l'assistant Générer le formulaire, sélectionnez une opération depuis la vue Données/Services. Effectuez ensuite l'une des opérations suivantes :

- Dans le menu contextuel de l'opération, sélectionnez Générer le formulaire.
- Dans le menu Données de Flash Builder, sélectionnez Générer le formulaire.

2 Dans l'assistant Générer le formulaire, sélectionnez Appel de service dans le champ Générer le formulaire pour.

3 Sélectionnez Nouvel appel de service ou Résultat d'appel existant.

Spécifiez Résultat d'appel existant pour utiliser le code généré précédemment pour un appel de service.

Sinon, activez Nouvel appel de service et sélectionnez un service et une opération pour le formulaire.

4 (Facultatif) Les options disponibles sur le formulaire généré varient en fonction de l'opération.

Si l'opération accepte les paramètres, vous pouvez choisir d'inclure un formulaire pour les paramètres.

Si l'opération renvoie une valeur, vous pouvez choisir d'inclure un formulaire pour la valeur renvoyée.

Vous pouvez choisir de rendre le formulaire modifiable ou de le rendre uniquement accessible en lecture.

5 (Facultatif) Configurez les types d'entrée et les types de retour.

Si l'opération sélectionnée dispose de paramètres d'entrée ou renvoie une valeur, vous pouvez configurer le type d'entrée ou le type de retour.

Configurez les types d'entrée et de retour de l'opération pour pouvoir générer le formulaire. Si vous les avez déjà configurés précédemment, vous avez ici la possibilité de les configurer à nouveau.

Voir « Configuration des types de données pour les opérations de service de données » à la page 27.

6 Cliquez sur Suivant. Dans la boîte de dialogue Mappage de contrôle de propriété, sélectionnez les champs à inclure dans le formulaire et le type de contrôle pour représenter les données.

7 Cliquez sur Terminer.

Les formulaires générés par Flash Builder peuvent être superposés. Pour réarranger les formulaires générés, assurez-vous d'avoir bien sélectionné et déplacé un formulaire et non un composant de ce dernier.

La sélection de formulaires superposés peut s'avérer difficile. Dans l'éditeur de code, sélectionnez la balise correspondant à un formulaire.

Génération d'un formulaire de détails

Pour générer un formulaire de détails, ajoutez un composant de contrôle de données à l'application et liez les résultats d'une opération au contrôle.

Ajoutez par exemple un composant DataGrid, puis liez à ce composant les résultats d'une opération telle que `getItems_paged()`.

1. Dans la vue Structure, sélectionnez un contrôle de données tel que DataGrid.
2. Dans le menu Données, sélectionnez Générer le formulaire de détails.
3. Générez le formulaire en suivant les instructions fournies dans le paragraphe Création d'un formulaire.

Création d'un formulaire pour un type de données

Pour générer un formulaire avec des composants représentant les champs d'un type de données personnalisé, commencez par configurer le type de données. Voir « Configuration des types de données pour les opérations de service de données » à la page 27.

1. Dans la vue Données/Services, sélectionnez un type de données personnalisé.
2. Dans le menu contextuel, sélectionnez Générer le formulaire.
3. Assurez-vous que l'option Générer le formulaire pour Type de données est sélectionnée, puis choisissez un type de données.
4. (Facultatif) Rendez le formulaire modifiable si vous le souhaitez.
5. Cliquez sur Terminer.

Génération de gestionnaires d'événement pour l'extraction de données distantes

Lorsque vous reliez une opération de service de données à un composant, Flash Builder crée un gestionnaire d'événement qui renseigne le composant avec les données extraites du service.

Par exemple, si vous reliez une opération `getAllItems()` à un DataGrid, Flash Builder génère un gestionnaire d'événement `creationComplete`. Le composant DataGrid référence le gestionnaire d'événements généré. Les résultats de l'appel deviennent le fournisseur de données du composant DataGrid.

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

Lorsque vous exécutez l'application, le gestionnaire d'événement renseigne le composant DataGrid créé avec les données extraites du service.

Lorsque vous créez des gestionnaires d'événement, vous pouvez accepter les gestionnaires créés ou les remplacer par des gestionnaires adaptés à vos besoins. Vous pouvez par exemple remplacer le gestionnaire d'événement `creationComplete` sur DataGrid par un gestionnaire `creationComplete` sur Application.

Vous pouvez également générer ou créer des gestionnaires d'événement pour des contrôles acceptant les interactions avec l'utilisateur, par exemple les contrôles Button et Text.

Création d'un gestionnaire d'événement pour un composant d'interface utilisateur

1. Créez une application contenant un composant d'interface utilisateur tel que DataGrid ou Button.
2. Flash Builder comporte un assistant de contenu facilitant la génération du gestionnaire d'événement. Appuyez sur Ctrl+Espace ou Cmd+Espace (Mac) et sélectionnez l'option Générer un gestionnaire d'événement.
3. Flash Builder génère un nom unique pour le gestionnaire d'événement et place ce dernier dans le bloc Script.

   Flash Builder met en surbrillance le stub généré pour le gestionnaire d'événement dans l'éditeur de code. Complétez le code restant pour le gestionnaire d'événement. Utilisez l'assistant de contenu afin de coder le gestionnaire d'événement.

Configuration des types de données pour les opérations de service de données

Lors de la connexion à un service de données, Flash Builder doit connaître le type des données renvoyées par une opération de service. Les types de données pris en charge sont ceux reconnus par AMF pour échanger des données avec un service de données ou un service distant.

De nombreux services de données définissent le type des données renvoyées sur le serveur (définition de type côté serveur). Toutefois, si le serveur ne définit pas le type, l'application client doit configurer le type des données renvoyées (définition de type côté client).

Les opérations de service qui spécifient les paramètres doivent également spécifier un type correspondant aux données auxquelles le service permet d'accéder. La définition de type côté client permet de configurer le type des paramètres d'entrée.

Lors de la configuration des types pour la définition de type côté client, Flash Builder reconnaît uniquement les types de données AMF. Le type peut également être un type de données personnalisé représentant des données complexes ou un type vide pour indiquer que l'opération ne renvoie aucune donnée.

Vous pouvez configurer des types définis par l'utilisateur pour les opérations de service qui renvoient des données complexes. Si vous récupérez par exemple des enregistrements d'une base de données d'employés, vous allez définir le retour de données complexes comme Employé. Dans ce cas, le type de données personnalisé pour Employé contient des entrées pour chaque champ de l'enregistrement de base de données.

Types de données pour la définition de type côté client

Type de données	Description
Types ActionScript	Boolean Boolean[] ByteArray ByteArray[] Date Date[] int int[] Number Number[] Object Object[] String String[]
Aucune donnée renvoyée	void
Type défini par l'utilisateur	*CustomType* *CustomType*[]

Type défini par l'utilisateur (Employé)

Champ	Type de données
emp_no	Number
first_name	String
last_name	String
hire_date	Date
birth_date	Date

Authentification de l'accès aux services

Les services de données nécessitent généralement une authentification de l'utilisateur avant d'autoriser l'accès aux services. Les services PHP, BlazeDS et ColdFusion donnant accès à l'aide du protocole HTTP peuvent requérir une authentification supplémentaire. Dans certains cas, ces types de services nécessitent une authentification HTTP et une authentification distante.

Flash Builder offre une option d'authentification du service lorsque vous effectuez les opérations suivantes :

- Configuration du type de retour pour une opération

  Voir « Configuration du type de retour pour les données d'une opération » à la page 29.

- Utilisation de l'interface Opération de test

  Voir « Test des opérations de service » à la page 31.

Lorsque vous sélectionnez l'option Authentification requise, Flash Builder ouvre la boîte de dialogue Authentification des services. Suivant le type de service auquel vous accédez, vous pouvez choisir l'authentification de base ou l'authentification distante.

Authentification de base

L'authentification de base permet d'accéder aux services HTTP et Web. Fournissez le nom d'utilisateur et le mot de passe pour accéder à ces services.

Sélectionnez les options de mémorisation du nom d'utilisateur et du mot de passe si vous souhaitez que Flash Builder utilise les autorisations spécifiées tout au long de la session.

Authentification à distance

L'authentification distante permet d'accéder aux services d'objets distants. Les services d'objets distants sont des services implémentés en tant qu'objets distants à l'aide de ColdFusion, PHP, BlazeDS ou LiveCycle Data Services.

Flash Builder ne fournit pas l'interface d'authentification distante pour les projets qui n'implémentent pas de services d'objets distants.

Fournissez le nom d'utilisateur et le mot de passe pour accéder aux services d'objets distants.

Sélectionnez les options de mémorisation du nom d'utilisateur et du mot de passe si vous souhaitez que Flash Builder utilise les autorisations spécifiées tout au long de la session.

Configuration des paramètres d'entrée d'une opération

Pour la définition de type côté client, configurez les paramètres d'entrée des opérations disponibles à partir du service de données.

La procédure suivante part du principe qu'une connexion à un service de données a été établie dans Flash Builder et que le service de données dispose d'opérations nécessitant des paramètres d'entrée configurables.

1. Dans la vue Données/Services, sélectionnez une opération contenant des paramètres d'entrée configurables. Dans le menu contextuel de l'opération, sélectionnez Configurer les types d'entrée.
2. Dans la boîte de dialogue Configurer les types d'entrée, sélectionnez un type de données dans la liste pour chaque argument de l'opération. Cliquez sur OK.

   Vous pourrez également sélectionner les types de données de retour personnalisés préalablement définis.

Pour la définition de type côté serveur, le service spécifie le type de données pour les paramètres d'entrée.

Configuration du type de retour pour les données d'une opération

Un service qui définit le type des données renvoyées par une opération fournit la définition de type côté serveur. Si un service ne définit pas le type des données renvoyées par une opération, Flash Builder utilise la définition de type côté client pour définir le type des données renvoyées.

Flash Builder introspecte les données renvoyées par une opération de service afin d'en déterminer le type. Lorsque vous configurez le type de retour d'une opération, vous avez deux options :

- Détecter automatiquement le type de retour à partir de l'exemple de données

  Si le service implémente la définition de type côté serveur, Flash Builder détecte le type de données défini par le service.

  Si le service n'implémente pas la définition de type côté serveur, Flash Builder crée un type personnalisé pour l'application client. Pour la définition de type côté client, spécifiez un nom pour le type personnalisé. En règle générale, le nom décrit les données renvoyées. Par exemple, si l'opération renvoie un tableau de livres à partir d'un tableau de base de données, nommez le type Livre.

- Utiliser un type de données existant

  Un type existant peut être défini par le service, un type ActionScript ou un type personnalisé précédemment défini.

La procédure utilisée par Flash Builder pour introspecter les données varie légèrement en fonction du type de service de données. Par exemple, la procédure utilisée pour introspecter et configurer le type de retour pour un service HTTP est différente de celle utilisée pour les services PHP ou ColdFusion.

Fusion et modification de types de données

Lors de l'introspection des données de serveur, vous pouvez fusionner les champs d'un autre type de données ou créer un type de données basé sur un type de données existant. Voici quelques méthodes de modification d'un type de données personnalisé.

- Utilisation d'un nouveau nom pour un type de données existant

  Utilisez un nouveau nom si vous prévoyez d'utiliser les données renvoyées de différentes manières dans l'application client.

  Par exemple, vous pouvez extraire des données employé qui peuvent être utilisées dans le résumé de l'employé et dans les tableaux d'informations des employés dans l'application client.

- Fusion de champs

  Vous pouvez ajouter des champs renvoyés à un type de données existant. L'ajout de champs supplémentaires peut être utile lors de l'association de données provenant de plusieurs sources, par exemple pour une opération JOIN renvoyant des données extraites de plusieurs tableaux de base de données.

  Un autre exemple est celui de données provenant de différents services, par exemple la fusion de données de type Livre reçues d'un service HTTP et d'un service ColdFusion.

Configuration d'un type de données personnalisé (services PHP ou ColdFusion)

Cette procédure part du principe que vous êtes connecté à un service de données implémenté avec PHP ou ColdFusion.

1. Dans la vue Données/Services, sélectionnez Configurer le type de retour dans le menu contextuel d'une opération.
2. Si l'opération comporte des arguments, entrez les valeurs d'argument. Spécifiez le type de données correct pour l'argument.
3. (Type personnalisé nouveau ou modifié) Sélectionnez l'option de détection automatique du type des données renvoyées par cette opération.

   Si ce service nécessite une authentification, sélectionnez Authentification requise et fournissez les autorisations requises. Voir « Authentification de l'accès aux services » à la page 28.

   Flash Builder introspecte l'opération et crée un type de données personnalisé.

Spécifiez un nom pour le type de données personnalisé.

Si vous avez préalablement défini un type de données personnalisé, vous pouvez ajouter les champs renvoyés à la définition du type de données personnalisé existant.

4 (Utilisation du type existant) Utilisez cette option pour spécifier un type ActionScript ou un type que vous avez précédemment configuré.

5 Cliquez sur Terminer.

Configuration d'un type de données personnalisé (services HTTP)

Cette procédure part du principe que vous êtes connecté à un service HTTP.

1 Dans la vue Données/Services, sélectionnez Configurer le type de retour dans le menu contextuel d'une opération.

2 (Nouveau type personnalisé) Sélectionnez l'option de détection automatique du type des données renvoyées par cette opération.

Si ce service nécessite une authentification, sélectionnez Authentification requise et fournissez les autorisations requises.

Flash Builder introspecte l'opération et crée un type de données personnalisé. Choisissez une méthode permettant à Flash Builder de passer les valeurs des paramètres de l'opération et cliquez sur Suivant.

- (Saisie des valeurs des paramètres) Pour chaque paramètre, spécifiez une valeur.

  Vous pouvez également spécifier le type de données des paramètres. Flash Builder sélectionne automatiquement le type de données par défaut.

- (Saisie de l'URL du service) Saisissez l'URL du service HTML, en incluant les paramètres et les valeurs dans l'URL. Par exemple :

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

- (Saisie de la réponse XML/JSON) Copiez la réponse XML/JSON dans la zone de texte.

  Utilisez cette option si vous êtes hors ligne ou si le service HTTP est encore en cours de développement, mais vous connaissez la réponse provenant du serveur.

3 (Nouveau type personnalisé, suite) Spécifiez un nom de type de données personnalisé ou sélectionnez un nœud provenant des données renvoyées.

Si vous sélectionnez un nœud pour les données renvoyées, Flash Builder crée un type de données personnalisé pour les données renvoyées pour ce nœud.

Indiquez si les données sont renvoyées sous forme de tableau.

Si le service renvoie un fichier XML, la liste déroulante Sélection de la racine est activée. Sélectionnez un nœud dans le fichier XML pour spécifier un type de données.

4 (Utilisation du type existant) Utilisez cette option pour spécifier un type ActionScript ou un type que vous avez précédemment configuré.

5 Cliquez sur Terminer.

Test des opérations de service

Vous pouvez utiliser Flash Builder pour tester des opérations de service et afficher les données renvoyées par une opération. Cette fonction est utile pour vérifier le comportement des services.

Important : *certaines opérations, telles que la mise à jour ou la suppression, entraînent une modification des données sur le serveur.*

Test d'une opération de service

Cette procédure part du principe que vous êtes connecté à un service de données.

1 Dans la vue Données/Services, sélectionnez une opération de service. Dans le menu contextuel, sélectionnez Opération de test.

La vue Opération de test s'ouvre et affiche l'opération sélectionnée. Si l'opération nécessite des paramètres d'entrée, la vue Opération de test comporte une liste des paramètres.

2 Pour chaque paramètre d'entrée requis, cliquez sur le champ Entrer une valeur et spécifiez une valeur.

Si le paramètre nécessite un type complexe, cliquez sur les points de suspension pour ouvrir un éditeur de saisie d'argument. Saisissez une valeur dans l'éditeur.

L'éditeur de saisie d'argument accepte la notation JSON pour la représentation de types complexes dans ActionScript.

3 Si le serveur requiert une authentification, sélectionnez l'option Authentification requise. Cliquez sur Tester.

Fournissez les autorisations le cas échéant. Voir « Authentification de l'accès aux services » à la page 28.

Flash Builder affiche les données renvoyées par le service.

4 (Facultatif) Dans la vue Opération de test, sélectionnez et testez d'autres services et opérations disponibles.

Gestion de l'accès aux données à partir du serveur

Pagination : la pagination est l'extraction incrémentielle de jeux de données volumineux d'un service distant.

Supposons par exemple que vous souhaitiez accéder à une base de données contenant 10 000 enregistrements pour en afficher les données dans une grille composée de 20 lignes. Vous pouvez dans ce cas implémenter une opération de pagination chargée d'extraire les lignes par incréments de 20 jeux. Une demande de données supplémentaires (matérialisée par exemple par le défilement de la grille de données par l'utilisateur) conduit à l'extraction et à l'affichage de la page d'enregistrements suivante.

Gestion des données : dans Flash Builder, la gestion de données est la synchronisation des mises à jour des données sur le serveur à partir de l'application client. Elle permet de modifier un ou plusieurs éléments dans une application client sans mettre à jour le serveur, puis de valider l'ensemble des modifications apportées au serveur en une seule opération. Vous pouvez également annuler les modifications sans mettre à jour les données.

La gestion de données implique la coordination de plusieurs opérations (création, récupération, mise à jour et suppression) répondant aux événements de l'application client, tels que la mise à jour de l'enregistrement de l'employé.

Lors de l'activation de la gestion des données avec Flash Builder, ce dernier génère un code mettant automatiquement à jour les composants de l'interface utilisateur. Par exemple, Flash Builder génère un code assurant la synchronisation des composants DataGrid avec les données sur le serveur.

Activation de la pagination

Vous pouvez activer la pagination pour un service de données qui implémente une fonction de pagination avec la signature suivante :

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

Nom de la fonction	Vous pouvez utiliser n'importe quel nom valide pour la fonction.
startIndex	La première ligne de données à extraire. Définissez le type de données pour startIndex en tant que nombre dans l'opération client.
numItems	Le nombre de lignes de données à extraire dans chaque page. Définissez le type de données pour numItems en tant que nombre dans l'opération client.
myDataType	Le type de données renvoyé par le service de données.

Lors de l'implémentation de la pagination à partir d'un service, vous pouvez aussi implémenter une opération `count()`. L'opération `count()` renvoie le nombre d'éléments renvoyés par le service. Flash Builder requiert que l'opération `count()` implémente la signature suivante :

```
count(): Number
```

Nom de la fonction	Vous pouvez utiliser n'importe quel nom valide pour la fonction.
Number	Nombre d'enregistrements extraits de l'opération.

Flex utilise l'opération count pour afficher correctement les composants de l'interface utilisateur qui extraient des jeux de données volumineux. L'opération `count()` contribue par exemple à déterminer la taille du curseur de la barre de défilement d'un objet DataGrid.

Certains services distants ne fournissent pas d'opération `count()`. La pagination continue à fonctionner, mais le contrôle qui affiche les données paginées risque de ne pas représenter correctement la taille du jeu de données.

Opérations de pagination des interrogations filtrées

Activez la pagination des interrogations pour filtrer les résultats envoyés par la base de données. Lorsque vous filtrez les résultats de l'interrogation, utilisez ces signatures pour les fonctions de pagination et count.

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

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

filterParam1	Paramètre de filtre facultatif. Ce paramètre est le même dans les fonctions getItems_PagedFiltered() et countFiltered().
filterParam2	Paramètre de filtre facultatif. Ce paramètre est le même dans les fonctions getItems_PagedFiltered() et countFiltered().

Voici un fragment de code d'une fonction `getItems_pagedFiltered()` implémentée dans PHP pour accéder à la base de données MySQL. Il indique comment utiliser le paramètre de filtre facultatif.

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

Activation de la pagination pour une opération

Cette procédure part du principe que vous avez codé les opérations `getItems_paged()` et `count()` du service distant et que vous avez configuré le type de données de retour pour l'opération, ainsi que décrit dans « Configuration des types de données pour les opérations de service de données » à la page 27.

1 Dans la vue Données/Services, ouvrez le menu contextuel de l'opération `getItems_paged()` et sélectionnez Activer la pagination.

2 En l'absence de clé unique pour le type de données, spécifiez les attributs qui caractériseront une instance de ce type de données de manière univoque. Cliquez sur Suivant.

Cet attribut est généralement la clé primaire.

3 (Facultatif) Spécifiez le nombre d'éléments à récupérer pour définir un format de page personnalisé.

Si vous ne spécifiez pas de format de page personnalisé, un format par défaut est défini au niveau du service. Par défaut, une page peut contenir 20 enregistrements.

4 (Facultatif) Spécifiez l'opération `count()`. Cliquez sur Terminer.

L'opération `count()` permet à Flash Builder d'afficher correctement les éléments de l'interface utilisateur, tels que la taille du curseur de la barre de défilement.

La pagination est activée pour l'opération.

Dans la vue Données/Services, la signature de la fonction qui implémente la pagination n'inclut plus les paramètres startIndex et numItems. Flash Builder ajoute maintenant ces valeurs de façon dynamique et les détermine en fonction de la taille des pages définie par l'utilisateur ou de la taille des pages par défaut (20 enregistrements par page).

Activation de la gestion des données

Pour activer la gestion de données pour un service, le service implémente une ou plusieurs des fonctions suivantes. La gestion de données utilise ces fonctions pour synchroniser les mises à jour des données sur le serveur distant :

- Ajout (createItem)

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

Le type de retour de la méthode createItem() est le type de la clé principale de la base de données.

- Obtention de toutes les propriétés (getItem)

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

- Mise à jour (updateItem)

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

- Suppression (deleteItem)

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

Dans Flash Builder, ces fonctions doivent avoir les signatures suivantes.

Nom de la fonction	Vous pouvez utiliser n'importe quel nom valide pour la fonction.
item **originalItem**	Elément du type de données renvoyé par le service de données.
itemID	Identifiant univoque de l'élément, généralement la clé primaire dans la base de données.
changes[]	Tableau correspondant aux champs d'un élément Item spécifié. Cet argument est utilisé uniquement dans une version de la méthode `updateItem()`.
myDataType	Type de données de l'élément du service de données. En général, vous définissez un type de données personnalisé lorsque vous extrayez des données d'un service.

Indicateur autoCommit

La gestion de données permet de synchroniser les mises à jour des données sur le serveur. Les modifications apportées aux données dans l'application client ne sont pas mises à jour sur le serveur tant que l'appel de la méthode `service.commit()` n'a pas été effectué.

Cependant, si vous souhaitez désactiver cette fonctionnalité, définissez l'indicateur autoCommit sur true. Si autoCommit est défini sur true, les mises à jours des données du serveur ne sont pas mises en cache, mais sont effectuées immédiatement. Voir « Activation de la gestion de données pour un service » à la page 41.

Indicateur deleteItemOnRemoveFromFill

Lorsque vous supprimez des éléments alors que la gestion des données est activée, utilisez l'indicateur deleteItemOnRemoveFromFill. Par défaut, cet indicateur est défini sur true. Lorsque vous supprimez un élément, cet élément est immédiatement retiré de la base de données.

Définissez deleteItemOnRemoveFromFill sur false pour différer la suppression jusqu'à l'appel de la méthode commit(). L'exemple suivant illustre le code pour la création d'un gestionnaire d'événement complet pour un contrôle DataGrid. Si l'utilisateur supprime un élément Employee sélectionné dans le DataGrid, l'élément sélectionné n'est retiré de la base de données qu'à l'appel de la méthode commit().

```
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();
}
```

Activation de la gestion de données pour une opération

Cette procédure part du principe que vous avez implémenté les opérations requises dans le service distant et que vous avez configuré le type de données de retour pour les opérations utilisant un type de données personnalisé. Voir « Configuration des types de données pour les opérations de service de données » à la page 27.

1. Dans la vue Données/Services, développez le nœud Types de données.
2. Accédez au menu contextuel d'un type de données et sélectionnez Activer la gestion de données.
3. En l'absence de clé unique pour le type de données, spécifiez les attributs qui caractériseront une instance de ce type de données de manière univoque. Cliquez sur Suivant.

   Cet attribut est généralement la clé primaire.
4. Spécifiez les opérations d'ajout, d'obtention de toutes les propriétés, de mise à jour et de suppression que vous avez implémentées. Cliquez sur Terminer.

Remarque : *il n'est pas nécessaire d'implémenter toutes ces fonctions. Implémentez uniquement celles qui sont requises pour votre application.*

La gestion des données est activée pour l'opération.

Génération du code Flash Builder pour les applications client

Flash Builder génère du code client donnant accès aux opérations du service distant. La génération de code est déclenchée par les opérations suivantes :

- Connexion à un service de données
- Actualisation du service de données dans la vue Données/Services
- Configuration d'un type de retour pour une opération
- Liaison d'une opération de service à un contrôle de l'interface utilisateur
- Activation de la pagination pour une opération de service
- Activation de la gestion de données pour un service
- Création d'un gestionnaire d'événement ou d'un appel de service

Classes du service

Utilisez l'assistant de service pour vous connecter à un service de données. Lorsque vous vous connectez à un service, Flash Builder génère un fichier de classe ActionScript donnant accès aux opérations du service.

Pour les services qui accèdent à un composant RemoteObject, la classe générée étend la classe RemoteObjectServiceWrapper. En règle générale, les services implémentés avec PHP, ColdFusion, BlazeDS et LiveCycle Data Services accèdent à un RemoteObject.

Pour les services HTTP, la classe générée étend la classe HTTPServiceWrapper.

Pour les services Web, la classe générée étend la classe WebServiceWrapper.

Flash Builder base le nom du fichier de la classe générée sur le nom attribué au service dans l'assistant de service. Par défaut, Flash Builder place cette classe dans le dossier source principal d'un projet. En règle générale, ce dossier est nommé `src`. Le nom du package est basé sur celui du service. Par exemple, Flash Builder génère les classes ActionScript suivantes pour une classe EmployeeService.

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

La super-classe contient l'implémentation de la classe EmployeeService.

La super-classe est une classe générée. Ne l'éditez jamais. Les modifications apportées à la super-classe risquent d'être remplacées. Les modifications apportées à l'implémentation peuvent provoquer un comportement non défini.

Dans cet exemple, utilisez `EmployeeService.as` pour étendre la super-classe générée et ajouter l'implémentation.

Voir aussi

« Connexion à des services de données » à la page 9

Classes de types de données personnalisés

De nombreux services de données distants fournissent une définition de type côté serveur. Ces services renvoient des données complexes comme type de données personnalisé.

Pour les services de données distants ne renvoyant pas de données typées, Flash Builder fournit une définition de type côté client. Avec la définition de type côté client, vous utilisez l'assistant de connexion Flash Builder pour définir et configurer le type des données complexes renvoyées par le service. Vous pouvez par exemple définir et configurer un type de données Employé pour un service renvoyant des enregistrements d'une base de données d'employés.

Flash Builder génère une classe ActionScript pour l'implémentation de chaque type de données personnalisé envoyé par le service. Il utilise cette classe pour créer des objets de valeur, qu'il utilise ensuite pour accéder aux données à partir du service distant.

Par exemple, Flash Builder génère les classes ActionScript suivantes pour une classe EmployeeService qui contient un type de données Employee :

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

Les super-classes contiennent respectivement l'implémentation de la classe EmployeeService et le type de données Employee.

N'éditez jamais une super-classe générée. Les modifications apportées à la super-classe risquent d'être remplacées. Les modifications apportées à l'implémentation peuvent provoquer un comportement non défini.

Dans cet exemple, utilisez `EmployeeService.as` et `Employee.as` pour étendre la super-classe générée et ajouter l'implémentation.

Liaison d'une opération de service à un contrôle de l'interface utilisateur

Pour obtenir des informations sur la liaison des données renvoyées par des opérations de service à un contrôle de l'interface utilisateur, voir « Liaison d'opérations de service à des contrôles » à la page 23. Lorsque vous liez une opération de service à un contrôle, Flash Builder génère le code suivant :

- Une balise Declarations contenant une propriété CallResponder et une balise de service
- Un gestionnaire d'événement pour appeler l'appel de service
- Une liaison de données entre le contrôle et les données renvoyées par l'opération

Balise Declarations

Une balise Declarations est un élément MXML qui déclare des propriétés de la classe actuelle qui ne sont pas des propriétés par défaut ni visuelles. Lors de la liaison d'une opération de service à une interface utilisateur, Flash Builder génère une balise Declarations contenant une propriété CallResponder et une balise de service. La propriété CallResponder et la classe de service générée sont les propriétés de l'élément de conteneur, qui est généralement la balise Application.

L'exemple suivant présente une balise Declarations donnant accès à un service EmployeeService distant :

```
<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

La propriété CallResponder gère les résultats des appels effectués à un service. Elle contient une propriété de jeton définie sur le jeton Async renvoyé par un appel de service. La propriété CallResponder contient également une propriété lastResult, définie sur le dernier résultat réussi de l'appel de service. Vous ajoutez des gestionnaires d'événement à la propriété CallResponder pour donner accès aux données renvoyées par l'intermédiaire de la propriété lastResult.

Lorsque Flash Builder génère une propriété CallResponder, il génère une propriété ID en fonction du nom de l'opération de service à laquelle elle est liée. L'exemple de code suivant illustre les propriétés CallResponder de deux opérations d'un service EmployeeService. L'opération `getAllItems` est liée au gestionnaire d'événement creationComplete d'un composant DataGrid. L'opération delete est liée à l'élément sélectionné dans le composant DataGrid. Immédiatement après sa création, le composant DataGrid affiche les éléments extraits de l'appel de service `getAllItems`. Le contrôle Delete Item Button supprime de la base de données l'enregistrement sélectionné dans le composant DataGrid.

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

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

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

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

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

Gestionnaires d'événement

Lorsque vous reliez une opération de service à un composant de l'interface utilisateur, Flash Builder crée un gestionnaire d'événement pour la propriété CallResponder. Le gestionnaire d'événement gère les résultats de l'opération. Vous pouvez également créer un gestionnaire d'événement dans un bloc de code ActionScript et référencer ce gestionnaire d'événement à partir d'une propriété d'un composant de l'interface utilisateur.

Vous renseignez généralement les contrôles tels que List et DataGrid avec les données renvoyées par un service. Par défaut, Flash Builder génère pour le contrôle un gestionnaire d'événement creationComplete qui est déclenché immédiatement après la création du contrôle. Pour d'autres contrôles, il génère un gestionnaire pour l'événement par défaut. Pour un bouton par exemple, il génère un événement pour l'événement click.

La propriété event du contrôle est définie sur le gestionnaire d'événement généré. L'exemple suivant illustre le gestionnaire d'événement creationComplete généré pour un contrôle DataGrid :

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

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

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

Vous pouvez générer des gestionnaires d'événement pour des contrôles répondant aux événements user (les contrôles Button, par exemple). L'exemple suivant présente un gestionnaire d'événement généré pour un bouton qui renseigne un contrôle DataGrid :

```
<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)"/>
```

Liaison de données

Lorsque vous créez une interface utilisateur, vous liez les opérations de service aux contrôles. Voir « Liaison d'opérations de service à des contrôles » à la page 23.

Flash Builder génère du code qui lie les données renvoyées par une opération de service au contrôle de l'interface utilisateur affichant les données.

L'exemple suivant illustre le code que Flash Builder génère pour renseigner un contrôle DataGrid. L'opération `getAllItems` renvoie un ensemble d'enregistrements d'employés pour le type de données personnalisé Employé.

La propriété dataProvider du contrôle DataGrid est liée aux résultats stockés dans la propriété CallResponder, getAllItemsResult. Flash Builder met automatiquement à jour le contrôle DataGrid en fonction des colonnes DataGridColumn correspondant à chaque champ renvoyé pour un enregistrement Employé. Les propriétés headerText et dataField de chaque colonne sont définies en fonction des données renvoyées dans un enregistrement Employé.

```
<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>
```

Activation de la pagination pour une opération de service

Lorsque vous activez la pagination, Flash Builder modifie l'implémentation du service généré. Lorsque vous renseignez un contrôle de données (un contrôle DataGrid ou List, par exemple) avec des données paginées, Flash Builder détermine le nombre d'enregistrements visibles dans le contrôle de données et le nombre total d'enregistrements dans la base de données. Il fournit ces valeurs en tant qu'arguments à l'opération de service que vous avez utilisée pour implémenter la pagination.

Vous n'avez pas à modifier de code d'application client une fois la pagination activée.

Pour plus d'informations, voir « Activation de la pagination » à la page 32.

Activation de la gestion de données pour un service

Dans Flash Builder, la gestion de données est la synchronisation d'un ensemble de mises à jour des données sur le serveur. Vous pouvez activer la gestion de données pour les types de données personnalisés renvoyés par un service. Avec la gestion de données activée, vous pouvez modifier un ou plusieurs éléments dans une application client sans mettre à jour le serveur. Vous pouvez ensuite valider l'ensemble des modifications apportées au serveur en une opération. Vous pouvez aussi annuler les modifications sans mettre à jour les données sur le serveur. Pour plus d'informations sur l'implémentation de cette fonction, voir « Activation de la gestion des données » à la page 34.

Lorsque vous activez la gestion des données, Flash Builder modifie l'implémentation de la classe de service générée et de la classe générée pour les types de données personnalisés. Il crée une classe DataManager pour implémenter cette fonctionnalité.

Synchronisation des mises à jour avec les données de serveur

Lorsque vous appelez des opérations de service pour un type de données géré, les changements sont reflétés dans l'application client. Toutefois, vous pouvez spécifier que les données sur le serveur ne soient mises à jour que lorsque vous appelez la méthode `commit()` de la classe DataManager.

Lorsque la gestion des données est activée pour un service, le service comporte un indicateur autoCommit. Par défaut, autoCommit est défini sur false.

L'indicateur autoCommit détermine si les changements sont validés immédiatement ou s'il faut attendre l'appel de *service*`.commit()`.

Si autoCommit est défini sur false, toutes les mises à jour du service dans l'application client sont mises en cache jusqu'à l'appel de *service*.commit(). Vous pouvez appeler la méthode revertChanges() du service pour ignorer les changements.

Si autoCommit est défini sur true, les mises à jours sont envoyées immédiatement au serveur. Vous ne pouvez pas appeler la méthode revertChanges() pour ignorer les changements.

L'indicateur deleteItemOnRemoveFromFill détermine si un élément supprimé est immédiatement retiré de la base de données. Si l'indicateur est défini sur true, l'élément n'est supprimé qu'à l'appel de *service*.commit().

Le code suivant désactive la synchronisation par gestion des données des mises à jour avec les données du serveur. Les modifications apportées aux données du type géré sont mises à jour immédiatement sur le serveur.

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

Le code suivant active la synchronisation de la gestion de données des mises à jour des données du serveur. Les modifications apportées aux données pour le type géré ne sont mises à jour qu'à l'appel de commit() pour le service. En outre, les éléments supprimés ne sont retirés de la base de données qu'à l'appel de commit().

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

Annulation des modifications

La classe DataManager fournit la méthode revertChanges qui permet de rétablir dans l'application client les valeurs extraites du serveur avant le dernier appel de la méthode commit.

Appelez revertChanges() avant d'appeler commit() pour annuler les modifications apportées à un type de données géré dans l'application client :

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

Pour valider les modifications apportées au type de données géré, appelez la méthode commit().

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

Vous pouvez également appeler la méthode commit directement à partir de l'instance bookService. L'appel de la méthode commit directement à partir de l'instance de service valide toutes les modifications pour l'ensemble des types de données gérés.

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

Remarque : *vous ne pouvez pas appeler revertChanges() directement à partir de l'instance de service pour annuler les modifications apportées à tous les types de données gérés. Vous ne pouvez l'appeler que pour un type de données géré spécifique.*

Si vous souhaitez remplacer le comportement par défaut de la gestion des données et désactiver la possibilité de rétablir les modifications, définissez l'indicateur autoCommit sur true. Par exemple, si vous disposé d'une instance de bookService et avez activé la gestion des données pour le type de données Book, définissez autoCommit sur true :

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

Les modifications apportées aux données pour le type géré sont désormais mises à jour immédiatement sur le serveur.

Déploiement des applications accédant aux services de données

Il existe de nombreux facteurs à considérer lorsque vous déplacez votre application d'un environnement de développement vers un environnement de déploiement. Le processus de déploiement d'une application dépend de votre application, de ses exigences et de votre environnement de déploiement.

Par exemple, le processus de déploiement d'une application d'un site Web interne accessible uniquement aux employés de l'entreprise est différent du processus de déploiement de la même application sur un site Web public.

La section Deploying applications présente les différents éléments à considérer et inclut une liste de contrôle de déploiement (Deployment checklist). La liste de contrôle aborde les problèmes courants de configuration de système que les clients ont pu rencontrer lors du déploiement des applications pour la mise en production. La documentation contient également des conseils de dépannage permettant de détecter les problèmes de déploiement les plus courants.

Meilleures pratiques pour le codage de l'accès aux services de données

Les outils Flash Builder vous permettent de générer du code client pour accéder aux données d'une base de données. Cette fonction est disponible pour PHP et ColdFusion. Cependant, ce code est destiné uniquement au prototypage. N'utilisez pas ce code comme modèle pour écrire des applications sécurisées.

Par défaut, le code généré permet à toute personne disposant d'un accès réseau à votre serveur d'insérer, de sélectionner, de mettre à jour ou de supprimer des éléments du tableau de base de données. Voici quelques recommandations à prendre en compte lors de la modification du code généré ou de la rédaction de code accédant aux services. Pour plus d'informations, voir Sécurisation des services de données.

Suppression de fonctions inutilisées

Supprimez ou ajoutez un commentaire à toute fonction que vous ne souhaitez pas utiliser dans votre application.

Ajout d'authentification

Ajoutez l'authentification utilisateur afin de vous assurer que seuls les utilisateurs de confiance peuvent accéder à vos informations de base de données.

Ajout des vérifications d'autorisation

Si l'authentification est nécessaire, ajoutez les vérifications d'autorisation. Même si vous avez donné accès à des utilisateurs authentifiés à votre application, vous devez vous assurer qu'ils sont autorisés à faire des requêtes spécifiques.

Par exemple, vous pouvez autoriser tous les utilisateurs à sélectionner, mais restreindre le nombre d'utilisateurs autorisés à supprimer.

Un autre exemple consiste à autoriser un utilisateur A à extraire ses propres informations à l'aide d'une requête de sélection tout en l'empêchant d'utiliser cette même requête pour accéder aux informations de l'utilisateur B.

Validation de données

Assurez-vous d'ajouter la validation de données. Par exemple, effectuez une validation des données fournies à toute instruction d'insertion afin de vous assurer qu'aucune donnée erronée ou malveillante ne soit acceptée dans la base de données.

La validation côté client ne permet pas de vous protéger des envois de requêtes manuelles à votre serveur Web. La validation des données protège contre les pirates et garantit la qualité des informations stockées.

Restriction de la quantité de données extraites

Les méthodes de sélection peuvent sélectionner la totalité d'un tableau. Dans certains cas, cela peut entraîner la présence d'une quantité trop importante d'informations sur le réseau. Extrayez uniquement les données dont vous avez besoin.

Par exemple, `SELECT *` à partir d'un tableau d'utilisateurs peut renvoyer le nom de l'utilisateur et son mot de passe sur le réseau.

Utilisation de SSL pour les données sensibles

L'utilisation d'un protocole sécurisé permet d'assurer la sécurité des informations que vous envoyez.

Exportation des fichiers source avec version validée d'une application

Lorsque vous exportez la version validée d'une application, Flash Builder fournit l'option Activer l'affichage de la source. Cette option permet aux utilisateurs d'afficher les fichiers source qui implémentent l'application. Dans le cadre des projets pour serveurs, les fichiers source incluent le dossier `services` qui contient les fichiers fournissant l'accès à l'implémentation du service.

Important : *faites preuve de prudence lorsque vous incluez des fichiers de service avec l'option d'affichage de la source. Lors de l'accès à la base de données, les fichiers de service exposent des détails et peuvent comporter des informations sensibles, telles que des noms d'utilisateurs et des mots de passe. Si des services sont inclus dans l'option d'affichage de la source, toute personne ayant accès à l'application lancée peut également accéder à des données sensibles.*

Voir aussi

Sécurité Flex

Ecriture de services sécurisés

Les exemples de la documentation Adobe, y compris les didacticiels et les applications créées à l'aide de la génération de code Flash Builder, sont instructifs de par leur nature. Ils illustrent comment accéder aux services de données à partir d'une application client. Cependant, dans la mesure où ces exemples sont conçus dans une optique de clarté, ils n'illustrent pas les meilleures pratiques en matière d'accès sécurisé aux données.

La documentation Flash Builder contient des exemples, dont des applications créées à partir de code généré. Ces exemples doivent être déployés dans un environnement de développement fiable. Un environnement de développement fiable peut être un ordinateur local ou un emplacement à l'intérieur d'un pare-feu. Sans mesures de sécurité supplémentaires, toute personne disposant d'un accès au réseau peut accéder à votre base de données.

Les meilleures pratiques en matière de rédaction de services incluent :

- Authentifiez l'utilisateur avant tout appel de méthode sur un service.
- Utilisez l'authentification de service pour autoriser uniquement certains utilisateurs à effectuer certaines actions.

  Par exemple, admettons que vous disposez d'une application permettant de modifier les données d'employés avec un appel RemoteObject. Dans ce cas, utilisez l'authentification RemoteObject afin de vous assurer que seuls les gestionnaires puissent modifier les données des employés.
- Utilisez la sécurité par programmation pour limiter l'accès aux services.
- Appliquez des contraintes de sécurité déclarative à des services entiers.
- Lorsque vous accédez à un service Web (`<mx:WebService>`) ou HTTP (`<mx:HTTPService>`), l'un des éléments suivants doit être vrai :
  - L'implémentation de service se trouve dans le même domaine que l'application qui l'appelle.

- Le système hôte du service possède un fichier `crossdomain.xml` qui autorise explicitement l'accès à partir du domaine de l'application.

Voir aussi

Sécurité Flex

Sécurisation des services de données

Rédaction d'applications sécurisées

Adobe® Flash® Player exécute des applications générées avec Flash. Le contenu est livré sous forme de séries d'instructions au format binaire à Flash Player à l'aide de protocoles Web dans un format de fichier SWF décrit précisément. Les fichiers SWF sont en règle générale hébergés sur un serveur, puis téléchargés et affichés sur l'ordinateur client lorsqu'ils sont demandés. La plupart du contenu correspond à des instructions ActionScript binaires. ActionScript est le langage de script basé sur les normes ECMA utilisé par Flash. ActionScript dispose d'API conçues pour permettre la création et la manipulation d'éléments de l'interface utilisateur côté client et pour travailler avec les données.

Le modèle de sécurité pour Flex protège le client et le serveur. Tenez compte des deux aspects généraux de sécurité suivants :

- Autorisation et authentification des utilisateurs accédant aux ressources d'un serveur
- Flash Player opérant dans un sandbox sur le client

Flex prend en charge l'utilisation de la sécurité d'applications Web de tout serveur d'application J2EE. En outre, les applications précompilées dans Flex peuvent s'intégrer au modèle d'authentification et d'autorisation de toute technologie de serveur sous-jacente afin d'empêcher les utilisateurs d'accéder à vos applications. La structure Flex inclut également plusieurs mécanismes de sécurité intégrés vous permettant de contrôler l'accès aux services Web, aux services HTTP et aux ressources basées sur serveur comme les EJB.

Flash Player s'exécute au sein d'un sandbox de sécurité qui empêche le piratage du client par du code d'application malveillant.

Remarque : *le contenu SWF exécuté dans Adobe AIR suit des règles de sécurité différentes de celles appliquées au contenu exécuté dans le navigateur. Pour plus de détails, voir la rubrique sur la sécurité Air de la documentation AIR.*

Pour obtenir des liens vers différentes rubriques de sécurité, voir Security Topic Center sur le portail Adobe Developer Connection.

Voir aussi

Sécurité Flex

Chapitre 3 : Implémentation de services pour des applications centrées sur les données

AMF (Action Message Format)

Flex utilise des services d'objets distants et AMF pour accéder aux services implémentés dans ColdFusion, PHP, BlazeDS et LiveCycle Data Services. AMF fournit la messagerie nécessaire pour échanger des données entre une base de données et l'application client.

ColdFusion, BlazeDS et LiveCycle Data Services fournissent chacun une structure AMF pour les services d'objets distants. Pour les services implémentés dans PHP, Flash Builder utilise la structure Zend AMF.

Les services ColdFusion et PHP peuvent fournir une définition de type côté serveur. Dans la définition de type côté serveur, le service définit le type des données renvoyées. Toutefois, si l'implémentation de service ne définit pas le type de données de retour, Flash Builder fournit la définition de type côté client. Il échantillonne des données du service, ce qui vous permet de configurer le type de retour dans l'application client.

Définition de type côté client et côté serveur

Dans Flex, une application client utilise le type des données renvoyées par un appel de service dans les méthodes qui accèdent et affichent les données.

Cependant, les exemples de services énumérés ci-dessous renvoient des données non typées.

- « Implémentation de services ColdFusion » à la page 47
- « Implémentation de services PHP » à la page 53
- « Exemple d'implémentation de services à partir de plusieurs sources » à la page 67

Par exemple, pour l'opération `getAllEmployees`, le service renvoie un tableau d'objets non typés représentant des enregistrements issus de la base de données. Flash Builder fournit des outils qui activent la définition de type côté client. Les outils Flash Builder vous permettent d'examiner en détail les données renvoyées et de définir un type personnalisé pour les données.

Vous pouvez définir le type de données personnalisé Employé pour l'objet renvoyé des enregistrements d'employés. Chaque colonne de l'enregistrement devient une propriété du type de données Employé.

A l'aide du type de données personnalisé Employé, l'application client peut accéder aux données renvoyées et les afficher correctement.

Flash Builder peut également accéder à des services implémentant une définition de type côté serveur. Pour obtenir des exemples de la définition de type côté serveur, voir Exemples de définition de type côté serveur dans Flash Builder.

Implémentation de services ColdFusion

Lors de l'implémentation de services dans ColdFusion, implémentez les services en tant que fichiers de composants ColdFusion (CFC). Chaque CFC contient des fonctions fournissant des opérations de service.

Vous pouvez créer des services ColdFusion dans n'impote quel IDE, tel qu'Adobe ColdFusion® Builder™. Flash Builder ne fournit pas d'éditeur spécifique aux fichiers ColdFusion. L'ouverture d'un fichier ColdFusion dans Flash Builder conduit toutefois au lancement de l'application du système associée aux fichiers ColdFusion.

Exemples de services ColdFusion

Vous pouvez implémenter un service ColdFusion de base en créant un composant ColdFusion (CFC) qui contient les fonctions pour les opérations de service. L'exemple suivant, `employeeService.cfc`, illustre un EmployeeService qui implémente deux fonctions. La fonction `getAllIEmployees()` extrait tous les enregistrements d'employés de la base de données. La fonction `getEmployees()` renvoie un enregistrement d'employé unique basé sur l'argument `emp_no` de la fonction.

Cet exemple illustre la définition de type côté client. Le service renvoie des données non typées. Flash Builder utilise la définition de type côté client pour introspecter les données renvoyées et définir le type de données.

Les exemples suivants illustrent comment implémenter les services pour la pagination et la gestion de données.

Vous pouvez également utiliser Flash Builder pour accéder à des services implémentant la définition de type côté serveur. Voir « Définition de type côté client et côté serveur » à la page 46.

Aucun exemple de définition de type côté serveur n'était disponible lorsque la rédaction de ce document a été achevée. Pour obtenir des exemples de définition de type côté serveur, voir Exemples de définition de type côté serveur dans Flash Builder.

Exemple d'implémentation d'un service de base dans ColdFusion

Cet exemple illustre comment implémenter un service de base dans ColdFusion. Cet exemple est basé sur le code généré par Flash Builder lors de l'accès au tableau de la base de données. Voir « Génération d'un exemple de service ColdFusion à partir d'un tableau de base de données » à la page 10.

Cet exemple illustre la définition de type côté client. Voir « Définition de type côté client et côté serveur » à la page 46.

Pour obtenir des exemples de définition de type côté serveur, voir Flash Builder server-side type examples.

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour obtenir des informations sur la rédaction de services ColdFusion sécurisés, voir la documentation ColdFusion About User Security.*

```
<cfcomponent output="false">

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

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

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

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

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

    </cffunction>

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

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

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

        <cfreturn qItem>

    </cffunction>

</cfcomponent>
```

Points forts d'EmployeeService :

- Connexion à la base de données des employés et accès au tableau des employés dans la base de données
- Renvoi d'un tableau d'objets

  Lors de la programmation à l'aide de la structure Flex, les services ne renvoient que des données. L'application client gère le formatage et la présentation des données. Ce modèle diffère des applications ColdFusion CFM traditionnelles, qui retournent des données formatées dans un modèle HTML.

  Flex gère les jeux d'enregistrements renvoyés en tant que tableau d'objets. Chaque ligne représente un enregistrement extrait de la base de données. Chaque colonne de l'enregistrement de base de données devient une propriété de l'objet renvoyé. L'application client peut ainsi accéder aux données renvoyées comme objets avec un ensemble de propriétés.

Configurez le type de données pour l'objet renvoyé. Voir « Définition de type côté client et côté serveur » à la page 46.

- Fonction de gestion des erreurs du serveur ColdFusion

  La gestion des erreurs fournie par ColdFusion est utile lors du débogage d'un service. Dans ColdFusion Administrator, modifiez les paramètres de débogage et de journalisation afin de fournir des informations de débogage plus performantes.

  L'interface Opération de test de Flash Builder affiche les informations renvoyées par le serveur ColdFusion.

  Pour plus d'informations sur les services de test, voir « Débogage de services distants » à la page 65.

- Utilisation de `cfqueryparam` pour créer des requêtes de base de données

  `cfqueryparam` est une défense contre les attaques par instructions d'injection SQL dans les appels au serveur. Pour plus d'informations, voir Enhancing security with cfqueryparam dans la documentation ColdFusion.

- Il authentifie les utilisateurs avant de donner l'accès aux fonctions dans ce service.

  L'exemple de code n'illustre pas comment authentifier les utilisateurs. Voir la documentation ColdFusion About User Security.

Voir aussi

« Configuration des types de données pour les opérations de service de données » à la page 27

« Accès aux services ColdFusion » à la page 9

« Génération d'un exemple de service ColdFusion à partir d'un tableau de base de données » à la page 10

Exemple d'implémentation de la pagination dans ColdFusion

Les outils Flash Builder permettent d'implémenter la pagination des données extraites d'un service distant. La pagination est l'extraction incrémentielle de jeux de données volumineux.

Pour implémenter la pagination, Flash Builder nécessite des signatures de fonction spécifiques. L'exemple de code suivant illustre une méthode d'implémentation d'un service ColdFusion pour des données paginées.

L'exemple EmployeeServicePaged est basé sur le code généré par Flash Builder lors de l'accès au tableau de la base de données. Voir « Génération d'un exemple de service ColdFusion à partir d'un tableau de base de données » à la page 10.

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. L'exemple permet à toute personne disposant d'un accès réseau à votre serveur d'accéder aux données du tableau de la base de données et de les modifier ou supprimer. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour obtenir des informations sur la rédaction de services ColdFusion sécurisés, voir la documentation ColdFusion About User Security.*

```
<cfcomponent output="false">

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

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

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

        <cfreturn qread.itemCount>

    </cffunction>

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

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

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

    </cffunction>
</cfcomponent>
```

Le service EmployeeServicePaged renvoie des données non typées. Utilisez les outils Flash Builder pour configurer le type de retour pour `getEmployees_Paged()`. Une fois le type de retour configuré, activez la pagination sur l'opération `getEmployees_Paged()`.

Voir aussi

« Exemples de services ColdFusion » à la page 47

« Configuration des types de données pour les opérations de service de données » à la page 27

« Gestion de l'accès aux données à partir du serveur » à la page 32

Exemple d'implémentation de la gestion de données dans ColdFusion

Les outils Flash Builder permettent d'implémenter la fonctionnalité de gestion des données pour les services distants. La gestion de données est la synchronisation des mises à jour des données sur le serveur à partir de l'application client.

Pour implémenter la gestion de données, Flash Builder nécessite une combinaison de signatures de fonction spécifiques. L'exemple de code suivant illustre une méthode d'implémentation d'un service ColdFusion pour la gestion de données.

L'exemple EmployeeServiceDM est basé sur le code généré par Flash Builder lors de l'accès au tableau de la base de données. Voir « Génération d'un exemple de service ColdFusion à partir d'un tableau de base de données » à la page 10.

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour obtenir des informations sur la rédaction de services ColdFusion sécurisés, voir la documentation ColdFusion About User Security.*

```
<cfcomponent output="false">

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

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

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

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

    </cffunction>

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

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

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

        <cfreturn qItem>

    </cffunction>

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

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

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

        </cfquery>

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

    </cffunction>

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

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

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

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

    </cffunction>

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

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

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

    </cffunction>
</cfcomponent>
```

Le service EmployeeServiceDM renvoie des données non typées. Utilisez les outils Flash Builder pour configurer le type de retour des opérations `getAllEmployeess()` et `getEmployees()`. Utilisez Employé comme type de données personnalisé renvoyé par ces opérations.

Une fois le type de retour configuré, activez la gestion de données sur le type de données Employé.

Voir aussi

« Exemples de services ColdFusion » à la page 47

« Configuration des types de données pour les opérations de service de données » à la page 27

« Gestion de l'accès aux données à partir du serveur » à la page 32

Génération de CFC à l'aide d'Adobe ColdFusion Builder

Adobe® ColdFusion® Builder™ fournit la fonctionnalité Adobe CFC Generator. Utilisez CFC Generator pour générer un CFC ORM ou un CFC traditionnel à partir d'un ensemble de tables de bases de données. Le CFC généré par ColdFusion Builder peut ensuite être utilisé en tant que service de données dans Flash Builder. Adobe CFC Generator crée des services qui implémentent la définition de type côté serveur.

Pour plus d'informations, voir Using Adobe CFC Generator.

Remarque : *la fonctionnalité de mappage objet/relationnel ColdFusion (ORM (object-relational mapping) utilise un modèle d'objets afin de définir une stratégie de mappage pour le stockage et la récupération de données à partir d'une base de données relationnelle). Voir ColdFusion ORM.*

Implémentation de services PHP

Lors de l'implémentation de services dans PHP, vous implémentez généralement les services en tant que classes PHP. Il n'est pas nécessaire que les classes dans PHP soient des classes orientées objet. Chaque classe peut plutôt constituer une bibliothèque de fonctions fournissant les opérations de service.

Vous pouvez créer des services PHP dans n'importe quel environnement d'édition (DreamWeaver ou Zend Studio, par exemple). Flash Builder ne fournit pas d'éditeur spécifique aux fichiers PHP. L'ouverture d'un fichier PHP dans Flash Builder conduit toutefois au lancement de l'application du système associée aux fichiers PHP. Pour plus de commodité, Flash Builder fournit également un éditeur de texte brut que vous pouvez utiliser pour éditer les fichiers PHP.

Utilisation d'AMF pour l'accès aux services implémentés dans PHP

Les services de données PHP sont accessibles en utilisant le format Action Message Format (AMF). AMF assure l'échange de messages entre un client Flash et le serveur Web. Flash Builder utilise la structure Zend AMF pour implémenter l'échange de messages AMF pour les services de données PHP.

Pour plus d'informations sur Zend AMF, voir Guide de référence du programmeur Zend Framework.

Pour plus d'informations sur l'installation de Zend Framework, voir « Installation de Zend Framework » à la page 21.

Pour plus d'informations sur l'utilisation de Zend avec Flash Builder pour PHP, voir le site Web de Zend.

Remarque : *l'utilisation de la structure Zend AMF par Flash Builder ne signifie pas que vous devez obligatoirement utiliser les composants Zend pour la création de services PHP. Malgré le bon fonctionnement des composants Zend avec Flash Builder, vous pouvez utiliser tout autre environnement de développement pour la création de services.*

Exemples de services PHP

Vous pouvez implémenter un service PHP de base en créant un fichier de classe PHP qui contient les fonctions des opérations de service. L'exemple suivant illustre un EmployeeService qui implémente deux fonctions :

- `getAllIEmployees()`

  Extrait tous les enregistrements d'employés de la base de données.

- `getEmployeeByID($itemID)`

  Renvoie un seul enregistrement d'employé.

Cet exemple illustre la définition de type côté client. Le service renvoie des données non typées. Flash Builder utilise la définition de type côté client pour introspecter les données renvoyées et définir le type de données.

Les exemples suivants illustrent comment implémenter les services pour la pagination et la gestion de données.

Vous pouvez également utiliser Flash Builder pour accéder à des services implémentant la définition de type côté serveur. Voir « Définition de type côté client et côté serveur » à la page 46.

Aucun exemple de définition de type côté serveur n'était disponible lorsque la rédaction de ce document a été achevée. Pour obtenir des exemples de définition de type côté serveur, voir Exemples de définition de type côté serveur dans Flash Builder.

Exemple de service de base PHP

Cet exemple indique comment implémenter un service de base dans PHP. Cet exemple est basé sur le code généré par Flash Builder lors de l'accès au tableau de la base de données. Voir « Génération d'un exemple de service PHP à partir d'un tableau de base de données » à la page 12.

Cet exemple illustre la définition de type côté client. Voir « Définition de type côté client et côté serveur » à la page 46.

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour plus d'informations sur la rédaction de services sécurisés, voir « Déploiement des applications accédant aux services de données » à la page 43.*

```
<?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);
        }
    }
}


?>
```

Points forts d'EmployeeService :

- Il se connecte à la base de données des employés, à laquelle il accède via le port 3306 de l'hôte local. Il accède au tableau des employés dans la base de données.
- Il fournit des variables de classe pour la connexion au service et l'accès aux tableaux dans la base de données.

  Vous pouvez utiliser ces variables dans des fonctions de la classe.

  Remplacez les valeurs de ces variables avec des valeurs de votre système.
- Il renvoie le tableau d'objets à l'application client.

  Lors de la programmation à l'aide de la structure Flex, les services ne renvoient que des données. L'application client gère le formatage et la présentation des données.

  Ce modèle diffère des services PHP traditionnels, qui retournent des données formatées dans un modèle HTML.
- Il fournit la fonction `getEmployeesByID($itemID)` qui lie le paramètre d'entrée aux types de données.

  Le nombre de variables et la longueur des types de chaînes doivent correspondre aux paramètres dans l'instruction. Le menu de la palette '`?` » de l'instruction de préparation est un espace réservé pour le paramètre.

  mysqli reconnaît les types suivants :

  - integer (`i`)
  - double (`d`)
  - string (`s`)
  - blob (`b`)
- Il lie les résultats, créant ainsi un tableau d'objets (`$row[]`).

  Flex gère les jeux d'enregistrements en tant que tableau d'objets. Chaque objet représente un enregistrement extrait de la base de données. Chaque colonne de l'enregistrement de base de données devient une propriété de l'objet renvoyé. L'application client peut ainsi accéder aux données renvoyées comme objets avec un ensemble de propriétés.

  Le serveur ne définissant pas le type des données renvoyées, vous devez configurer le type de données de l'objet renvoyé. Voir « Définition de type côté client et côté serveur » à la page 46.
- Il fournit une fonction constructeur pour l'initialisation de la connexion à la base de données.
- Il utilise les instructions de préparation mysqli pour la création de requêtes de base de données.

  L'utilisation des instructions de préparation est une défense contre les attaques par instructions d'injection SQL dans les appels au serveur. L'instruction est exécutée sur le serveur uniquement après sa préparation.
- Il authentifie les utilisateurs avant de donner l'accès aux fonctions dans ce service.

  L'exemple de code n'illustre pas comment authentifier les utilisateurs. Voir la documentation ColdFusion About User Security. Les principes de sécurité sur l'authentification et l'autorisation des utilisateurs dans cette documentation ColdFusion s'appliquent aux services PHP.
- Il renvoie une exception sur l'erreur.

  Les informations que vous fournissez dans les exceptions sont utiles lors du débogage de l'implémentation du service. L'interface Opération de test de Flash Builder affiche les informations renvoyées par les exceptions.

  Pour plus d'informations sur les services de test, voir « Débogage de services distants » à la page 65.
- Le nom du fichier `EmployeeService.php` correspond au nom de la classe PHP du service.

Si les noms du fichier et de la classe ne correspondent pas, des erreurs se produisent lorsque vous accédez au service.

Voir aussi

« Configuration des types de données pour les opérations de service de données » à la page 27

« Accès aux services PHP » à la page 11

« Génération d'un exemple de service PHP à partir d'un tableau de base de données » à la page 12

Exemple d'implémentation de la pagination dans PHP

Les outils Flash Builder permettent d'implémenter la pagination des données extraites d'un service distant. La pagination est l'extraction incrémentielle de jeux de données volumineux.

Pour implémenter la pagination, Flash Builder nécessite des signatures de fonction spécifiques. L'exemple de code suivant illustre une méthode d'implémentation d'un service PHP pour des données paginées.

Cet exemple est basé sur le code généré par Flash Builder lors de l'accès au tableau de la base de données. Voir « Génération d'un exemple de service PHP à partir d'un tableau de base de données » à la page 12.

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour plus d'informations sur la rédaction de services sécurisés, voir « Déploiement des applications accédant aux services de données » à la page 43.*

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

Le service EmployeeServicePaged renvoie des données non typées. Utilisez les outils Flash Builder pour configurer le type de retour pour `getEmployees_Paged()`. Une fois le type de retour configuré, activez la pagination sur l'opération `getEmployees_Paged()`.

Voir aussi

« Exemples de services PHP » à la page 54

« Configuration des types de données pour les opérations de service de données » à la page 27

« Gestion de l'accès aux données à partir du serveur » à la page 32

Exemple d'implémentation de la gestion de données dans PHP

Les outils Flash Builder permettent d'implémenter la fonctionnalité de gestion des données pour les services distants. La gestion de données est la synchronisation des mises à jour des données sur le serveur à partir de l'application client.

Pour implémenter la gestion de données, Flash Builder nécessite une combinaison de signatures de fonction spécifiques. L'exemple de code suivant illustre une méthode d'implémentation d'un service PHP pour la gestion de données.

Cet exemple est basé sur le code généré par Flash Builder lors de l'accès au tableau de la base de données. Voir « Génération d'un exemple de service PHP à partir d'un tableau de base de données » à la page 12.

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour plus d'informations sur la rédaction de services sécurisés, voir «* *Déploiement des applications accédant aux services de données » à la page 43.*

```
<?php

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

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

    var $connection;

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

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

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

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

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

```
    $rows = array();

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

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

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

    return $rows;
}

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

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

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

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

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

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

/**
 * Returns the item corresponding to the value specified for the primary key.
 *
 * Add authroization or any logical checks for secure access to your data
 *
 *
```

```
     * @return stdClass
     */
    public function createEmployees($item) {

        $stmt = mysqli_prepare($this->connection, "INSERT INTO $this->tablename
                               (emp_no, birth_date, first_name, last_name,
                               gender, hire_date) VALUES (?, ?, ?, ?, ?, ?)");
        $this->throwExceptionOnError();

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

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

        $autoid = mysqli_stmt_insert_id($stmt);

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

        return $autoid;
    }

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

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

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

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

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

    /**
     * Deletes the item corresponding to the passed primary key value from
     * the table.
     *
```

```
     * Add authroization or any logical checks for secure access to your data
     *
     *
     * @return void
     */
    public function deleteEmployees($itemID) {

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

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

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

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

Le service EmployeeServiceDM renvoie des données non typées. Utilisez les outils Flash Builder pour configurer le type de retour des opérations `getAllEmployeess()` et `getEmployeesByID()`. Utilisez Employé comme type de données personnalisé renvoyé par ces opérations.

Une fois le type de retour configuré, activez la gestion de données sur le type de données Employé.

Voir aussi

« Exemples de services PHP » à la page 54

« Configuration des types de données pour les opérations de service de données » à la page 27

« Gestion de l'accès aux données à partir du serveur » à la page 32

Débogage de services distants

Le débogage des applications accédant aux services distants peut se faire de différentes manières.

- Vue Opération de test de Flash Builder

  La vue Opération de test de Flash Builder vous permet d'appeler des opérations de service et d'afficher les données renvoyées. Elle affiche tous les messages d'erreur interceptés par le service.

- Scripts côté serveur

  Pour le débogage supplémentaire des services, vous pouvez écrire des scripts qui testent le code du serveur et des informations de flux de sortie dans les fichiers journaux.

- Moniteur de réseau Flash Builder

  Après avoir créé une application accédant à un service dans Flash Builder, utilisez le Moniteur de réseau pour visionner les données échangées entre le serveur et le client.

Vue Opération de test de Flash Builder

Utilisez la vue Opération de test de Flash Builder afin d'appeler les opérations d'un service et afficher les résultats de l'opération. Les résultats incluent tout message d'erreur renvoyé par un service.

Vous pouvez utiliser la vue Opération de test afin d'afficher les données renvoyées par les opérations sur les services que vous rédigez, les services disponibles à partir de HTTP ou les services Web.

Test d'une opération de service

Cette procédure part du principe que vous avez rédigé un service que vous testez ou que vous pouvez accéder à un service HTTP ou à un service Web.

1. Dans la vue Données/Services de Flash Builder, accédez à l'opération de service que vous souhaitez tester.
2. Dans le menu contextuel de l'opération, sélectionnez l'option Opération de test.
3. (Facultatif) Dans la vue Opération de test, sélectionnez Authentification requise afin de saisir les informations de connexion au service.
4. Si l'opération accepte les paramètres, cliquez sur le champ Entrer une valeur afin de fournir une valeur pour le paramètre.

   Si le paramètre nécessite un type complexe, cliquez sur les points de suspension dans le champ Entrer une valeur afin d'ouvrir un éditeur acceptant la notation JSON. Spécifiez la valeur pour le paramètre à l'aide de la notation JSON.
5. Cliquez sur Tester afin d'afficher le résultat de l'opération.

Scripts de test du code serveur

Utilisez des scripts de test pour afficher et déboguer le code serveur avant d'essayer de vous connecter au serveur dans Flash Builder. Les scripts de test présentent les avantages suivants :

- Vous pouvez visionner les résultats des tests dans un navigateur Web.

  Rafraîchissez l'affichage du navigateur pour visionner le résultat des modifications que vous apportez au code.

- Vous pouvez envoyer un écho ou une impression des résultats au flux de sortie, ce qu'il vous est impossible de faire directement à partir d'AMF.
- La mise en forme des erreurs affichées est conviviale. Les erreurs sont souvent plus complètes que celles saisies avec AMF.

Scripts ColdFusion

Utilisez le script suivant (`tester.cfm`) pour vider l'appel d'une fonction :

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

Spécifiez dans `tester2.cfm` la méthode et les arguments à appeler dans l'URL.

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

Vous pouvez par exemple appeler la méthode `getItemID` dans EmployeeService en utilisant l'URL suivante :

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

Le script `tester3.cfm` enregistre dans un journal les appels d'opérations et vide les arguments entrants en utilisant `cfdump`.

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

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

Scripts PHP

Utilisez le script suivant (`tester.php`) pour vider l'appel d'une fonction :

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

Ajoutez le code suivant au service PHP afin de consigner les messages au cours de l'exécution du code :

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

Ajoutez le code suivant au service PHP afin d'activer le vidage dans un fichier journal :

```
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);
```

Moniteur de réseau

Le Moniteur de réseau est accessible dans la perspective Débogage Flex de Flash Builder. Activez le moniteur avant de pouvoir l'utiliser pour analyser les données. Pour plus d'informations sur l'activation et l'utilisation du Moniteur de réseau, voir Surveillance des applications accédant aux services de données.

Exemple d'implémentation de services à partir de plusieurs sources

En règle générale, les applications accèdent aux données de sources différentes, affichant le résultat de l'association de données dans une application. Cet exemple illustre comment associer les données des trois tableaux suivants dans une base de données d'employés :

- Départements

  Chaque enregistrement contient les champs suivants : numéro et nom du département.

- Dept_emp

  Chaque enregistrement contient les champs suivants : emp_no, dept_no, from_date, to_date.

- Employés

  Chaque enregistrement contient les champs suivants : emp_no, birth_date, first_name, last_name, gender, hire_date.

L'exemple d'application dispose de deux composants DataGrid, un pour les départements et un pour les employés.

La liste Départements répertorie tous les départements. Lorsque vous sélectionnez un département, le composant DataGrid Employés répertorie tous les employés de ce département.

La sélection d'un employé dans le composant DataGrid Employés renseigne un formulaire vous permettant de mettre à jour l'enregistrement de l'employé sélectionné.

Création des services

Pour cet exemple, créez un service unique. Le service contient les opérations suivantes :

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

EmployeeService (PHP)

`EmployeeService.php` implémente un service contenant une seule fonction. `GetEmployeesByID()` accepte l'ID de département en tant qu'argument et renvoie tous les employés du département donné. La fonction renvoie également les dates auxquelles l'employé a rejoint et quitté le département. `GetEmployeesByDept()` exécute la requête SQL suivante :

```
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
```

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour plus d'informations sur la rédaction de services sécurisés, voir «* *Déploiement des applications accédant aux services de données » à la page 43.*

```
<?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` implémente un service contenant une seule fonction. `GetEmployeesByID()` accepte l'ID de département en tant qu'argument et renvoie tous les employés du département donné. La fonction renvoie également les dates auxquelles l'employé a rejoint et quitté le département. `GetEmployeesByDept()` exécute la requête SQL suivante :

```
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
```

Important : *les exemples de services sont destinés uniquement à l'établissement d'un prototype. Utilisez l'exemple de service uniquement dans un environnement de développement fiable. Avant de déployer ce service, assurez-vous d'augmenter la protection et de restreindre l'accès de façon adéquate. Pour obtenir des informations sur la rédaction de services ColdFusion sécurisés, voir la documentation ColdFusion About User Security.*

```
<cfcomponent output="false">

<!---
    This sample service contains functions that illustrate typical service operations.
    Use these functions as a starting point for creating your own service implementation.

    This code is for prototyping only.

    Authenticate the user before allowing them to call these methods. You can find more
    information at http://www.adobe.com/go/cf9_usersecurity
--->
    <cffunction name="getEmployeesByDept" output="false" access="remote"  returntype="any" >
        <cfargument name="dept_no" type="string" required="true" />

        <cfset var qItem="">
        <cfquery name="qItem" datasource="employees">
        SELECT employees.emp_no,
            employees.birth_date,
            employees.first_name,
            employees.last_name,
            employees.gender,
            employees.hire_date,
            dept_emp.from_date,
            dept_emp.to_date
        FROM employees, dept_emp
        WHERE dept_emp.emp_no = employees.emp_no and
        dept_emp.dept_no = <CFQUERYPARAM CFSQLTYPE="CF_SQL_VARCHAR"
VALUE="#ARGUMENTS.dept_no#">
        </cfquery>

        <cfreturn qItem>

    </cffunction>

</cfcomponent>?>
```

Importation de services dans un projet de serveur

1 Dans Flash Builder, créez un projet Flex nommé Associations.

(PHP) Lors de la création du projet, spécifiez PHP comme type de serveurs d'applications.

(PHP) Après avoir créé le projet, Flash Builder crée un dossier de sortie dans le dossier de racine Web de votre configuration PHP. Le nom par défaut du projet PHP_Associations est `PHP_Associations-debug`.

(ColdFusion) Lors de la création du projet, spécifiez ColdFusion comme type de serveurs d'applications. Sélectionnez ensuite ColdFusion Flash Remoting.

2 (PHP) Dans `PHP_Associations-debug`, créez un dossier nommé `services`. Copiez `EmployeeService.php` dans le dossier `services`.

3 (ColdFusion) Créez un dossier nommé `Associations` dans la racine Web de votre configuration ColdFusion. Copiez `EmployeeService.chc` dans le dossier `Associations`.

4 Importez EmployeeService dans le projet.

Assurez-vous que PHP_Associations est le projet actif dans Flash Builder.

Sélectionnez Données > Connexion à PHP. Pour spécifier la classe PHP, accédez au dossier `services` et sélectionnez `EmployeeService.php`. Cliquez sur Terminer.

Pour plus d'informations, voir « Connexion à des services de données PHP » à la page 11.

5 Configurez le type de retour des opérations dans EmployeeService.

- DepartmentService

  Dans le menu contextuel de l'opération `getAllDepartments`, sélectionnez Configurer le type de retour.

  Cliquez sur Suivant pour lancer la détection automatique du type de retour.

  Spécifiez **Département** pour le type de retour personnalisé. Cliquez sur Terminer.

- EmployeeService

  Dans le menu contextuel de l'opération `getEmployeesByDept()`, sélectionnez Configurer le type de retour.

  Cliquez sur Suivant pour lancer la détection automatique du type de retour.

  Saisissez la valeur **d007** pour le paramètre. Cliquez sur Suivant.

  Spécifiez **Employé** comme type de retour personnalisé. Cliquez sur Terminer.

Pour plus d'informations, voir « Configuration des types de données pour les opérations de service de données » à la page 27.

Chapitre 4 : Accès aux données côté serveur

Les composants d'accès aux données Adobe® Flex® utilisent des appels de procédures distants pour interagir avec les environnements de serveur tels que PHP, Adobe ColdFusion et Microsoft ASP.NET. Ces composants fournissent des données aux applications client créées avec la structure Adobe Flex et envoient des données aux sources de données de back-end. Pour une présentation des composants d'accès aux données, voir « Composants d'accès aux données » à la page 4.

Utilisation de composants HTTPService

Vous pouvez utiliser un composant HTTPService avec tout type de technologie côté serveur, y compris les pages PHP, les pages ColdFusion, les pages JavaServer (JSP), les servlets Java, Ruby on Rails et les pages Microsoft ASP. Il vous est également possible d'utiliser HTTPService pour accéder aux services Web basés sur REST.

Pour obtenir des informations de référence API sur le composant HTTPService, voir mx.rpc.http.mxml.HTTPService.

Utilisation de données PHP et SQL

Les composants HTTPService peuvent être utilisés avec PHP et un système de gestion de base de données SQL pour afficher les résultats d'une interrogation de base de données dans une application. Vous pouvez également utiliser les composants pour insérer, mettre à jour et supprimer les données d'une base de données. Vous pouvez appeler une page PHP avec la méthode GET ou POST pour effectuer une interrogation de base de données, puis formater les données du résultat de l'interrogation dans une structure XML et renvoyer la structure XML à l'application dans le cadre de la réponse HTTP. Une fois le résultat renvoyé à l'application, vous pouvez l'afficher dans un ou plusieurs contrôles de l'interface utilisateur.

Code MXML

Dans l'exemple suivant, l'application appelle une page PHP avec la méthode POST. La page PHP interroge une table de base de données MySQL appelée users. Elle formate les résultats de l'interrogation sous XML et renvoie les données XML à l'application, dans laquelle elles sont liées à la propriété `dataProvider` d'un contrôle DataGrid et affichées dans ce contrôle DataGrid. L'application envoie également le nom d'utilisateur et l'adresse électronique des nouveaux utilisateurs à la page PHP, qui procède à une insertion dans la table de base de données des utilisateurs.

```
<?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>
```

La méthode `send()` du composant HTTPService effectue l'appel à la page PHP. Cet appel est effectué dans la méthode `send_data()` dans le bloc Script du fichier MXML.

La propriété `resultFormat` du composant HTTPService étant définie sur `object`, les données sont renvoyées à l'application en tant que graphique d'objets ActionScript. Il s'agit de la valeur par défaut de la propriété `resultFormat`. Une autre possibilité consiste à utiliser une propriété `resultFormat` définie sur `e4x` pour renvoyer les données en tant qu'objet XMLList sur lequel vous pouvez exécuter ECMAScript pour les opérations XML (E4X). Si vous sélectionnez la valeur `e4x` pour la propriété `resultFormat`, vous devrez apporter les quelques modifications suivantes au code MXML.

Remarque : *si le format de résultat est `e4x`, n'incluez pas le nœud racine de la structure XML dans la notation par point lors de la liaison au contrôle DataGrid.*

Les données XML renvoyées dans cet exemple ne contiennent aucune information d'espace de noms. Pour obtenir des informations sur l'utilisation de données XML ne contenant pas d'espaces de noms, voir « Traitement de résultats en tant que données XML avec le format de résultat E4X » à la page 127.

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

Le format de résultat `e4x` permet en outre de lier la propriété `lastResult` à un objet XMLListCollection, puis de lier cet objet à la propriété `DataGrid.dataProvider`, comme l'illustre le fragment de code suivant :

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

Script de base de données MySQL

Le code PHP de cette application utilise une table de base de données appelée users dans une base de données MySQL appelée sample. Le script MySQL suivant crée la table :

```
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 ;
```

Code PHP

Cette application appelle la page PHP suivante. Ce code PHP effectue des insertions dans la base de données SQL et l'interroge, puis retourne les résultats de l'interrogation à l'application dans une structure XML.

```
<?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)
?>
```

Utilisation de données ColdFusion et SQL

Les composants HTTPService peuvent être utilisés avec une page ColdFusion et un système de gestion de base de données SQL pour afficher les résultats d'une interrogation de base de données dans une application. Vous pouvez également utiliser les composants pour insérer, mettre à jour et supprimer les données d'une base de données. Vous pouvez appeler une page ColdFusion avec la méthode GET ou POST pour effectuer une interrogation de base de données, puis formater les données du résultat de l'interrogation dans une structure XML et renvoyer la structure XML à l'application dans le cadre de la réponse HTTP. Une fois le résultat renvoyé à l'application, vous pouvez l'afficher dans un ou plusieurs contrôles de l'interface utilisateur.

Code MXML

Dans l'exemple suivant, l'application appelle une page ColdFusion avec la méthode POST. La page ColdFusion interroge une table de base de données MySQL appelée users. Elle formate les résultats de l'interrogation sous XML et renvoie les données XML à l'application, dans laquelle elles sont liées à la propriété `dataProvider` d'un contrôle DataGrid et affichées dans ce contrôle DataGrid. L'application envoie aussi le nom d'utilisateur et l'adresse électronique des nouveaux utilisateurs à la page ColdFusion, qui procède à une insertion dans la table de base de données des utilisateurs.

```
<?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>
```

La méthode `send()` du composant HTTPService effectue l'appel à la page ColdFusion. Cet appel est effectué dans la méthode `send_data()` dans le bloc Script du fichier MXML.

La propriété `resultFormat` du composant HTTPService étant définie sur `object`, les données sont renvoyées à l'application en tant que graphique d'objets ActionScript. Il s'agit de la valeur par défaut de la propriété `resultFormat`. Une autre possibilité consiste à utiliser un format de résultat `e4x` pour renvoyer les données en tant qu'objet XMLList sur lequel vous pouvez exécuter ECMAScript pour les opérations XML (E4X). Si vous sélectionnez la valeur `e4x` pour la propriété `resultFormat`, vous devrez apporter les quelques modifications suivantes au code MXML.

Remarque : *si le format de résultat est `e4x`, n'incluez pas le nœud racine de la structure XML dans la notation par point lors de la liaison au contrôle DataGrid.*

Les données XML renvoyées dans cet exemple ne contiennent aucune information d'espace de noms. Pour obtenir des informations sur l'utilisation de données XML ne contenant pas d'espaces de noms, voir « Traitement de résultats en tant que données XML avec le format de résultat E4X » à la page 127.

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

Le format de résultat `e4x` permet de lier la propriété `lastResult` à un objet XMLListCollection, puis de lier cet objet à la propriété `dataProvider` de DataGrid, comme l'illustre le fragment de code suivant :

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

Script SQL

Le code ColdFusion de cette application utilise une table de base de données appelée users dans une base de données MySQL appelée sample. Le script MySQL suivant crée la table :

```
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 ;
```

Code ColdFusion

L'application répertoriée dans la section Utilisation de données ColdFusion et SQL appelle l'application ColdFusion suivante, `returncfxml.cfm`. Ce code ColdFusion effectue des insertions dans la base de données SQL et l'interroge, puis renvoie les résultats de l'interrogation à l'application. La page ColdFusion utilise la balise `cfquery` pour insérer des données dans la base de données et l'interroger. Elle fait appel à la balise `cfxml` pour formater les résultats de l'interrogation dans une structure XML.

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

Utilisation de pages JavaServer

Les composants Flex HTTPService peuvent être utilisés avec une page JSP et un système de gestion de base de données SQL pour afficher les résultats d'une interrogation de base de données dans une application. Vous pouvez également utiliser les composants pour insérer, mettre à jour et supprimer les données d'une base de données. Vous pouvez appeler une page JSP avec la méthode GET ou POST pour effectuer une interrogation de base de données, puis formater les données du résultat de l'interrogation dans une structure XML et renvoyer la structure XML à l'application dans le cadre de la réponse HTTP. Une fois le résultat renvoyé à l'application, vous pouvez l'afficher dans un ou plusieurs contrôles de l'interface utilisateur.

Code MXML

Dans l'exemple suivant, l'application appelle une page JSP et extrait des données d'une base de données SQL. Elle formate les résultats de l'interrogation de la base de données sous XML et retourne les données XML à l'application, dans laquelle elles sont liées à la propriété `dataProvider` d'un contrôle DataGrid et affichées dans ce contrôle DataGrid.

```
<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>
```

La méthode `send()` du composant HTTPService effectue l'appel à la page JSP. Cet appel est effectué dans l'événement `click` de l'objet Button dans le fichier MXML.

La propriété `resultFormat` du composant HTTPService étant définie sur `object`, les données sont renvoyées à l'application en tant que graphique d'objets ActionScript. Il s'agit de la valeur par défaut de la propriété `resultFormat`. Une autre possibilité consiste à utiliser un format de résultat `e4x` pour renvoyer les données en tant qu'objet XMLList sur lequel vous pouvez exécuter ECMAScript pour les opérations XML (E4X). Si vous sélectionnez la valeur `e4x` pour la propriété `resultFormat`, vous devrez apporter les quelques modifications suivantes au code MXML.

Remarque : *si le format de résultat est `e4x`, n'incluez pas le nœud racine de la structure XML dans la notation par point lors de la liaison au contrôle DataGrid.*

Les données XML renvoyées dans cet exemple ne contiennent aucune information d'espace de noms. Pour obtenir des informations sur l'utilisation de données XML ne contenant pas d'espaces de noms, voir « Traitement de résultats en tant que données XML avec le format de résultat E4X » à la page 127.

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

Lorsque vous utilisez le format de résultat e4x, vous pouvez si vous le souhaitez lier la propriété `lastResult` à un objet XMLListCollection, puis lier cet objet à la propriété DataGrid.dataProvider :

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

Code JSP

L'exemple suivant présente la page JSP utilisée dans cette application. Cette page JSP n'appelle pas de base de données directement. Elle obtient ses données d'une classe Java appelée ProductService, qui à son tour utilise une classe Java appelée Product pour représenter des produits particuliers.

```
<%@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>
```

Appel de services HTTP dans ActionScript

L'exemple suivant présente un appel de service HTTP dans un bloc de script ActionScript. L'appel de la méthode `useHTTPService()` déclare le service, définit la destination, configure des écouteurs d'événement result et fault et conduit à l'appel de la méthode `send()` du service.

```
<?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>
```

Utilisation de composants WebService

Les applications créées avec la structure Flex peuvent interagir avec des services Web basés sur SOAP qui définissent leurs interfaces dans un document WSDL 1.1 (Web Services Description Language 1.1), disponible sous forme d'URL. WSDL est un format standard permettant de décrire les messages qu'un service Web comprend, le format des réponses de ce service à ces messages, les protocoles que le service Web prend en charge et l'adresse à laquelle envoyer les messages. L'API du service Web Flex prend généralement en charge le protocole SOAP (Simple Object Access Protocol) 1.1, XML Schema 1.0 (versions 1999, 2000 et 2001), WSDL 1.1 codé RPC, littéral RPC et littéral document (paramètres de style brut et enveloppé). Les deux types de services Web les plus courants utilisent des liaisons SOAP codées (RPC) ou littéral document ; les termes *codées* et *littéral* indiquent le type de mappage WSDL sur SOAP qu'un service utilise.

Flex prend en charge les demandes et les résultats de service Web formatés en tant que messages SOAP. SOAP fournit la définition du format XML que vous pouvez utiliser pour échanger des informations structurées et typées entre un client de service Web (une application créée avec Flex, par exemple) et un service Web.

Adobe® Flash® Player fonctionne au sein d'un sandbox de sécurité qui limite les données auxquelles les applications créées avec Flex et les autres applications créées avec Flash peuvent accéder via HTTP. Les applications créées avec Flash peuvent accéder par HTTP uniquement aux ressources figurant dans le même domaine et par le même protocole les ayant traitées. Cela constitue un problème pour les services Web qui sont généralement atteints à partir d'emplacements distants. Le service Proxy, disponible dans LiveCycle Data Services et BlazeDS, intercepte les demandes aux services Web distants et les redirige, puis renvoie les réponses au client.

Si vous n'utilisez pas LiveCycle Data Services ou BlazeDS, vous pouvez accéder aux services Web dans le même domaine que l'application ou utiliser un fichier crossdomain.xml (de régulation interdomaines) permettant l'accès à partir du domaine de l'application et devant être installé sur le serveur Web hébergeant le service RPC.

Pour obtenir des informations de référence API sur le composant WebService, voir mx.rpc.soap.mxml.WebService.

Exemple d'application WebService

L'exemple de code suivant concerne une application qui utilise un composant WebService pour appeler des opérations de service Web.

Code MXML

Dans l'exemple suivant, l'application appelle un service Web. Ce service interroge une table de base de données SQL appelée users et renvoie des données à l'application, dans laquelle ces données sont liées à la propriété `dataProvider` d'un contrôle DataGrid où elles sont affichées. L'application envoie également le nom d'utilisateur et l'adresse électronique des nouveaux utilisateurs au service Web, qui procède à une insertion dans la table de base de données des utilisateurs. L'implémentation principale du service Web est un composant ColdFusion ; le même composant ColdFusion est accédé en tant qu'objet distant dans « Utilisation de composants RemoteObject » à la page 101.

```
<?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>
```

Document WSDL

L'exemple suivant présente le document WSDL qui définit l'API du service Web :

```
<?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>
```

Appel de services Web dans ActionScript

L'exemple suivant présente un appel de service Web dans un bloc de script ActionScript. L'appel de la méthode `useWebService()` déclare le service, définit la destination, récupère le document WSDL et conduit à l'appel de la méthode `echoArgs()` du service.

Remarque : *lorsque vous déclarez un composant WebService dans ActionScript, appelez la méthode* `WebService.loadWSDL()`.

```
<?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>
```

Noms d'opérations réservés

Pour accéder aux opérations WebService, il suffit de les nommer d'après une variable de service. Des conflits de noms risquent toutefois de se produire si le nom d'une opération correspond à une méthode définie pour le service. Dans ActionScript, vous pouvez appliquer la méthode suivante à un composant WebService pour renvoyer l'opération du nom donné :

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

Lecture de documents WSDL

Vous pouvez afficher un document WSDL dans un navigateur Web, un éditeur de texte simple, un éditeur XML ou un environnement de développement tel qu'Adobe Dreamweaver, qui intègre un utilitaire pour l'affichage de documents WSDL dans un format en facilitant la lecture.

Les documents WSDL contiennent les balises décrites dans le tableau suivant.

Balise	Description
`<binding>`	Spécifie le protocole que les clients (les applications créées avec Flex, par exemple) utilisent pour communiquer avec un service Web. Il existe des liaisons pour SOAP, HTTP GET, HTTP POST et MIME. Flex ne prend en charge que la liaison SOAP.
`<fault>`	Spécifie la valeur d'une erreur renvoyée suite à un problème de traitement d'un message.
`<input>`	Spécifie le message qu'un client (une application créée avec Flex, par exemple) envoie à un service Web.

Balise	Description
`<message>`	Définit les données transférées par une opération WebService.
`<operation>`	Définit une combinaison des balises `<input>`, `<output>` et `<fault>`.
`<output>`	Spécifie le message envoyé par le service Web à un client de service Web (une application créée avec Flex, par exemple).
`<port>`	Spécifie un point de terminaison de service Web, définissant l'association entre une liaison et une adresse de réseau.
`<portType>`	Définit la ou les opérations fournies par un service Web.
`<service>`	Définit une collection de balises `<port>`. Chaque service se mappe à une balise `<portType>` et spécifie différentes manières d'accéder aux opérations dans cette balise `<portType>`.
`<types>`	Définit les types de données utilisés par les messages d'un service Web.

Opérations orientées RPC et opérations orientées document

Un fichier WSDL peut spécifier des opérations orientées RPC ou des opérations orientées document (littéral de document). Flex prend en charge les deux styles d'opérations.

Lorsqu'elle appelle une opération orientée RPC, une application envoie un message SOAP spécifiant une opération et ses paramètres. Lorsqu'elle appelle une opération orientée document, une application envoie un message SOAP contenant un document XML.

Dans un document WSDL, chaque balise `<port>` comporte une propriété `binding` qui spécifie le nom d'une balise `<soap:binding>` spécifique, comme l'illustre l'exemple suivant :

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

La propriété `style` de la balise `<soap:binding>` associée détermine le style d'opération. Dans cet exemple, le style est `document`.

Toute opération dans un service peut spécifier le même style ou remplacer le style spécifié pour le port associé au service, comme l'illustre l'exemple suivant :

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

Services Web avec état

Flex fait appel à des sessions de serveur Java pour conserver l'état des points de terminaison de service Web qui utilisent des cookies pour enregistrer des informations de session. Cette fonction fait office d'intermédiaire entre les applications et les services Web. Elle ajoute une identité de point de terminaison à tout élément que le point de terminaison transmet à l'application. Si le point de terminaison envoie des informations de session, l'application les reçoit. Cette fonction ne nécessite aucune configuration ; elle n'est pas prise en charge pour les destinations qui font appel au canal RTMP lorsque le service proxy est utilisé.

Utilisation d'en-têtes SOAP

Un en-tête SOAP est une balise facultative dans une enveloppe SOAP qui contient généralement des informations spécifiques aux applications (des informations d'authentification, par exemple).

Ajout d'en-têtes SOAP à des demandes de services Web

Certains services Web requièrent la transmission d'un en-tête SOAP lors de l'appel d'une opération.

Vous pouvez ajouter un en-tête SOAP à toutes les opérations de services Web ou à des opérations individuelles en appelant la méthode `addHeader()` ou `addSimpleHeader()` d'un objet WebService ou Operation dans une fonction d'écouteur d'événement.

Avant d'utiliser la méthode `addHeader()`, vous devez créer séparément les objets SOAPHeader et QName. La méthode `addHeader()` présente la signature suivante :

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

Pour créer un objet SOAPHeader, utilisez le constructeur suivant :

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

Pour créer l'objet QName dans le premier paramètre de la méthode `SOAPHeader()`, utilisez le constructeur suivant :

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

Le paramètre `content` du constructeur `SOAPHeader()` est un ensemble de paires nom-valeur basées sur le format suivant :

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

La méthode `addSimpleHeader()` constitue un raccourci pour un en-tête SOAP nom-valeur unique. Lorsque vous utilisez la méthode `addSimpleHeader()`, vous créez les objets SOAPHeader et QName dans les paramètres de la méthode. La méthode `addSimpleHeader()` présente la signature suivante :

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

La méthode `addSimpleHeader()` utilise les paramètres suivants :

- *`qnameLocal`* est le nom local de l'en-tête QName.
- *`qnameNamespace`* est l'espace de noms de l'en-tête QName.
- *`headerName`* est le nom de l'en-tête.
- *`headerValue`* est la valeur de l'en-tête. Il peut s'agir d'une chaîne dans le cas d'une valeur simple, d'un objet auquel un codage XML de base sera appliqué ou de données XML si vous souhaitez spécifier les données XML de l'en-tête vous-même.

  Dans l'exemple suivant, le code indique comment utiliser les méthodes `addHeader()` et `addSimpleHeader()` pour ajouter un en-tête SOAP. Les méthodes sont appelées dans une fonction d'écouteur d'événement nommée `headers` et l'écouteur d'événement est assigné dans la propriété `load` d'une balise `<mx:WebService>` :

```
<?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>
```

Suppression d'en-têtes SOAP

Utilisez la méthode `clearHeaders()` de l'objet WebService ou Operation pour supprimer les en-têtes SOAP ajoutés à l'objet, ainsi que l'illustre l'exemple suivant pour un objet WebService. Vous devez appeler la méthode `clearHeaders()` au niveau (WebService ou Operation) auquel l'en-tête a été ajouté.

```
<?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>
```

Réorientation d'un service Web vers une URL différente

Certains services Web nécessitent le passage à une URL de point de terminaison différente après avoir traité le WSDL et l'exécution d'un appel initial au service Web. Vous voulez par exemple utiliser un service Web nécessitant la transmission d'informations d'identification de sécurité. Le service Web appelé pour l'envoi des informations d'identification de connexion accepte les informations d'identification et retourne l'URL de point de terminaison réelle requise pour utiliser les opérations métier du service. Avant d'appeler les opérations métier, vous devez modifier la propriété `endpointURI` du composant WebService.

L'exemple suivant présente un écouteur d'événement result qui stocke dans une variable l'URL de point de terminaison retournée par un service Web, puis transmet cette variable à une fonction afin de modifier l'URL de point de terminaison pour les demandes ultérieures :

```
...
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;

}
...
```

Un service Web nécessitant la transmission d'informations d'identification de sécurité peut également renvoyer un identifiant que vous devrez attacher à un en-tête SOAP pour toute demande ultérieure. Pour plus d'informations, voir « Utilisation d'en-têtes SOAP » à la page 89.

Sérialisation des données de service Web

Codage des données ActionScript

Le tableau suivant présente les mappages de codage des types ActionScript 3.0 en types complexes de schéma XML.

Définition du schéma XML	Types ActionScript 3.0 pris en charge	Remarques
Eléments de niveau supérieur		
xsd:element nillable == true	Object	Si la valeur d'entrée est `null`, la sortie codée est définie avec l'attribut `xsi:nil`.
xsd:element fixed != null	Object	La valeur d'entrée est ignorée et remplacée par la valeur fixe.
xsd:element default != null	Object	Si la valeur d'entrée est `null`, elle est remplacée par cette valeur par défaut.
Eléments locaux		
xsd:element maxOccurs == 0	Object	La valeur d'entrée est ignorée et omise de la sortie codée.
xsd:element maxOccurs == 1	Object	La valeur d'entrée est traitée comme une entité unique. Si le type associé est un tableau codé SOAP, les tableaux et les implémentations `mx.collection.IList` sont transmis intacts pour être traités par le codeur SOAP comme des cas spéciaux pour ce type.
xsd:element maxOccurs > 1	Object	La valeur d'entrée doit être itérable (un tableau ou une implémentation `mx.collections.IList`, par exemple), bien que les valeurs non itérables soient enveloppées avant traitement. Les éléments individuels sont codés en tant qu'entités distinctes conformément à la définition.
xsd:element minOccurs == 0	Object	Si la valeur d'entrée n'est pas définie ou est `null`, la sortie codée est omise.

Le tableau suivant présente les mappages de codage de types ActionScript 3.0 en types intégrés de schéma XML.

Type de schéma XML	Types ActionScript 3.0 pris en charge	Remarques
`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` est utilisé (sans retour à la ligne).
`xsd:boolean`	Boolean Number Object	Toujours codé en tant que `true` ou `false`. `Number == 1` alors `true`, dans le cas contraire, `false`. `Object.toString()` == « true » ou « 1 », alors `true` ; dans le cas contraire, `false`.
`xsd:byte` `xsd:unsignedByte`	Number String	Chaîne d'abord convertie en nombre.
`xsd:date`	Date Number String	Les méthodes d'accesseur Date.UTC sont utilisées. Nombre utilisé pour définir Date.time. Chaîne considérée préformatée et codée telle quelle.
`xsd:dateTime`	Date Number String	Les méthodes d'accesseur Date.UTC sont utilisées. Nombre utilisé pour définir Date.time. Chaîne considérée préformatée et codée telle quelle.
`xsd:decimal`	Number String	`Number.toString()` est utilisé. Infinity, -Infinity et NaN ne sont pas valides pour ce type. Chaîne d'abord convertie en nombre.
`xsd:double`	Number String	Limité à la plage de Number. Chaîne d'abord convertie en nombre.
`xsd:duration`	Object	`Object.toString()` est appelé.
`xsd:float`	Number String	Limité à la plage de Number. Chaîne d'abord convertie en nombre.
`xsd:gDay`	Date Number String	`Date.getUTCDate()` est utilisé. Nombre directement utilisé pour le jour. Chaîne analysée en tant que nombre du jour.
`xsd:gMonth`	Date Number String	`Date.getUTCMonth()` est utilisé. Nombre directement utilisé pour le mois. Chaîne analysée en tant que nombre du mois.
`xsd:gMonthDay`	Date String	`Date.getUTCMonth()` et `Date.getUTCDate()` sont utilisés. Chaîne analysée pour les portions de mois et de jour.

Type de schéma XML	Types ActionScript 3.0 pris en charge	Remarques
`xsd:gYear`	Date Number String	`Date.getUTCFullYear()` est utilisé. Nombre directement utilisé pour l'année. Chaîne analysée en tant que nombre de l'année.
`xsd:gYearMonth`	Date String	`Date.getUTCFullYear()` et `Date.getUTCMonth()` sont utilisés. Chaîne analysée pour les portions d'année et de mois.
`xsd:hexBinary`	flash.utils.ByteArray	mx.utils.HexEncoder est utilisé.

`xsd:integer`
`et dérivés :`
`xsd:negativeInteger`
`xsd:nonNegativeInteger`
`xsd:positiveInteger`
`xsd:nonPositiveInteger`
	Number
	String
		Limité à la plage de Number.
		Chaîne d'abord convertie en nombre.

`xsd:int` `xsd:unsignedInt`	Number String	Chaîne d'abord convertie en nombre.
`xsd:long` `xsd:unsignedLong`	Number String	Chaîne d'abord convertie en nombre.
`xsd:short` `xsd:unsignedShort`	Number String	Chaîne d'abord convertie en nombre.

Type de schéma XML	Types ActionScript 3.0 pris en charge	Remarques

`xsd:string`
`et dérivés :`
`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()` est invoqué.

`xsd:time`	Date Number String	Les méthodes d'accesseur Date.UTC sont utilisées. Nombre utilisé pour définir Date.time. Chaîne considérée préformatée et codée telle quelle.
`xsi:nil`	null	Si la définition d'élément de schéma XML correspondante comporte minOccurs > 0, une valeur `null` est codée à l'aide de xsi:nil ; dans le cas contraire, l'élément est entièrement omis.

Le tableau suivant présente le mappage de types ActionScript 3.0 en types codés SOAP.

Type SOAPENC	Types ActionScript 3.0 pris en charge	Remarques
soapenc:Array	Array mx.collections.IList	Les tableaux codés SOAP sont traités comme des cas spéciaux et ne sont pris en charge qu'avec les services Web de style codés RPC.
soapenc:base64	flash.utils.ByteArray	Codé de la même manière que `xsd:base64Binary`.
soapenc:*	Object	Tout autre type codé SOAP est traité comme s'il figurait dans l'espace de noms XSD en fonction de la propriété `localName` de l'objet QName du type.

Décodage du schéma XML et de SOAP vers ActionScript 3.0

Le tableau suivant présente les mappages de décodage de types intégrés de schéma XML en types ActionScript 3.0.

Type de schéma XML	Types ActionScript 3.0 décodés	Remarques
`xsd:anyType` `xsd:anySimpleType`	String Boolean Number	Si le contenu est vide -> `xsd:string`. Si le contenu est projeté sur Number et la valeur est NaN ; ou si le contenu commence par « 0 » ou « -0 », ou si le contenu se termine par « E » : alors, si le contenu est « true » ou « false » -> `xsd:boolean` dans le cas contraire -> `xsd:string`. Dans le cas contraire, le contenu est un nombre valide et donc -> `xsd:double`.
`xsd:base64Binary`	`flash.utils.ByteArray`	`mx.utils.Base64Decoder` est utilisé.
`xsd:boolean`	Boolean	Si le contenu est « true » ou « 1 », alors `true` ; dans le cas contraire, `false`.
`xsd:date`	Date	S'il n'existe aucune information de fuseau horaire, l'heure locale est utilisée.
`xsd:dateTime`	Date	S'il n'existe aucune information de fuseau horaire, l'heure locale est utilisée.
`xsd:decimal`	Number	Le contenu est créé via `Number(content)` et est donc limité à la plage de Number.
`xsd:double`	Number	Le contenu est créé via `Number(content)` et est donc limité à la plage de Number.
`xsd:duration`	String	Le contenu est renvoyé avec réduction des espaces.
`xsd:float`	Number	Le contenu est converti via `Number(content)` et est donc limité à la plage de Number.
`xsd:gDay`	uint	Le contenu est converti via `uint(content)`.
`xsd:gMonth`	uint	Le contenu est converti via `uint(content)`.
`xsd:gMonthDay`	String	Le contenu est renvoyé avec réduction des espaces.
`xsd:gYear`	uint	Le contenu est converti via `uint(content)`.
`xsd:gYearMonth`	String	Le contenu est renvoyé avec réduction des espaces.
`xsd:hexBinary`	`flash.utils.ByteArray`	`mx.utils.HexDecoder` est utilisé.

Type de schéma XML	Types ActionScript 3.0 décodés	Remarques

`xsd:integer`
`et dérivés :`
`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	Le contenu est décodé via `parseInt()`.

`xsd:string`
et dérivés :
`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	Le contenu brut est simplement renvoyé en tant que chaîne.

`xsd:time`	Date	S'il n'existe aucune information de fuseau horaire, l'heure locale est utilisée.
`xsi:nil`	null	

Le tableau suivant présente les mappages de décodage de types codés SOAP en types ActionScript 3.0.

Type SOAPENC	Type ActionScript décodé	Remarques
`soapenc:Array`	Array `mx.collections.ArrayCollection`	Les tableaux codés SOAP sont traités comme des cas spéciaux. Si `makeObjectsBindable` présente la valeur `true`, le résultat est enveloppé dans une classe ArrayCollection ; dans le cas contraire, un tableau simple est renvoyé.
`soapenc:base64`	`flash.utils.ByteArray`	Décodé de la même manière que `xsd:base64Binary`.
`soapenc:*`	Object	Tout autre type codé SOAP est traité comme s'il figurait dans l'espace de noms XSD en fonction de la propriété `localName` de l'objet QName du type.

Le tableau suivant présente les mappages de décodage de types de données personnalisés en types de données ActionScript 3.0.

Type personnalisé	Type ActionScript 3.0 décodé	Remarques
Apache Map `http://xml.apache.org/xml-soap:Map`	Object	La représentation SOAP de `java.util.Map. Keys` doit être représentable sous forme de chaînes.
Apache Rowset `http://xml.apache.org/xml-soap:Rowset`	Tableau d'objets	
ColdFusion QueryBean `http://rpc.xml.coldfusion:QueryBean`	Tableau d'objets `mx.collections.ArrayCollection` d'objets	Si `makeObjectsBindable` présente la valeur true, le tableau résultant est enveloppé dans une classe ArrayCollection.

Prise en charge de l'élément de schéma XML

Les structures ou attributs de structures de schéma XML suivants ne sont que partiellement implémentés dans Flex 4 :

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

Les structures ou attributs de structures de schéma XML suivants sont ignorés et ne sont pas pris en charge dans Flex 4 :

```
<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>
```

Personnalisation du mappage de type de service Web

Lorsqu'il utilise des données d'une invocation de service Web, Flex crée généralement des objets ActionScript anonymes non typés et imite la structure XML dans le corps du message SOAP. Si vous voulez que Flex crée une instance d'une classe spécifique, vous pouvez utiliser un objet mx.rpc.xml.SchemaTypeRegistry et enregistrer un objet QName avec une classe ActionScript correspondante.

Vous disposez par exemple de la définition de classe suivante dans un fichier nommé User.as :

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

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

Vous voulez alors invoquer une opération `getUser` sur un service Web renvoyant les données XML suivantes :

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

A l'invocation de l'opération getUser, pour être certain d'obtenir une instance de la classe User, et non un objet générique, faites figurer le code ActionScript suivant dans une méthode de l'application :

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

`SchemaTypeRegistry.getInstance()` est une méthode statique qui renvoie l'instance par défaut du registre de type. Dans la plupart des cas, c'est tout ce dont vous avez besoin. Cette manière de procéder enregistre toutefois un objet QName donné avec la même classe ActionScript à travers toutes les opérations de service Web de l'application. Pour enregistrer différentes classes pour différentes opérations, faites figurer le code suivant dans une méthode de l'application :

```
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;
```

Utilisation de la sérialisation de service Web personnalisée

Deux méthodes permettent de prendre le contrôle intégral de la manière dont les objets ActionScript sont sérialisés dans XML et dont les messages de réponse XML sont désérialisés. La méthode recommandée est l'utilisation directe d'E4X.

Si vous transmettez une instance de XML comme seul paramètre à une opération de service Web, elle est transmise intacte en tant qu'enfant du nœud `<SOAP:Body>` dans la demande sérialisée. Utilisez cette stratégie pour contrôler intégralement le message SOAP. De même, lorsque vous désérialisez une réponse de service Web, vous pouvez définir la propriété `resultFormat` de l'opération sur e4x. Cela renvoie un message de réponse contenant l'objet XMLList avec les enfants du nœud `<SOAP:Body>`. Vous pouvez ensuite implémenter la logique personnalisée requise pour créer les objets ActionScript appropriés.

Plus fastidieuse, la seconde méthode consiste à fournir vos propres implémentations de mx.rpc.soap.ISOAPDecoder et mx.rpc.soap.ISOAPEncoder. Si vous avez par exemple écrit une classe nommée MyDecoder implémentant ISOAPDecoder, le code suivant pourra figurer dans une méthode de l'application :

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

Lorsque vous appelez `someOperation`, Flex appelle la méthode `decodeResponse()` de la classe MyDecoder. Il incombe ensuite à l'implémentation personnalisée de traiter le message SOAP intégral et de produire les objets ActionScript attendus.

Utilisation de composants RemoteObject

Vous pouvez utiliser un composant Flex RemoteObject pour appeler des méthodes sur un composant ColdFusion ou une classe Java.

Vous pouvez également utiliser des composants RemoteObject contenant des objets PHP et .NET avec des logiciels tiers, tels que les projets « open source » AMFPHP et SabreAMF, ainsi que Midnight Coders WebORB. Pour plus d'informations, voir les sites Web suivants :

- Zend Framework http://framework.zend.com/
- AMFPHP http://amfphp.sourceforge.net/
- SabreAMF http://www.osflash.org/sabreamf
- Midnight Coders WebORB http://www.themidnightcoders.com/

Les composants RemoteObject utilisent le protocole AMF pour envoyer et recevoir des données, tandis que les composants WebService et HTTPService font appel au protocole HTTP. AMF est considérablement plus rapide que HTTP. Le codage et la configuration côté serveur sont toutefois généralement plus complexes.

Flash Builder pour PHP est un outil de développement créé en partenariat avec Zend Technologies qui inclue une copie intégrée de Zend Studio. Pour plus d'informations, voir le site Web d'Adobe.

De même que les composants HTTPService et WebService, les composants RemoteObject peuvent être utilisés pour afficher le résultat d'une requête de base de données dans une application. Vous pouvez également utiliser les composants pour insérer, mettre à jour et supprimer les données d'une base de données. Une fois le résultat de la requête renvoyé à l'application, vous pouvez l'afficher dans un ou plusieurs contrôles de l'interface utilisateur.

Pour obtenir des informations de référence API sur le composant RemoteObject, voir mx.rpc.remoting.mxml.RemoteObject.

Exemple d'application RemoteObject

Code MXML

Dans l'exemple suivant, l'application utilise un composant RemoteObject pour appeler un composant ColdFusion. Le composant ColdFusion interroge une table de base de données MySQL appelée users. Il renvoie le résultat de l'interrogation à l'application, dans laquelle ce résultat est lié à la propriété `dataProvider` d'un contrôle DataGrid et affiché dans ce contrôle DataGrid. L'application envoie également le nom d'utilisateur et l'adresse électronique des nouveaux utilisateurs au composant ColdFusion, qui les ajoute à la table de la base de données des utilisateurs.

```
<?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>
```

Dans cette application, la propriété `destination` du composant RemoteObject est définie sur `Coldfusion` et la propriété `source` est définie sur le nom complet du composant ColdFusion.

En revanche, lorsque vous utilisez LiveCycle Data Services ou BlazeDS, vous spécifiez un nom de classe complet dans la propriété `source` d'une destination de service distant dans un fichier de configuration (par défaut, remoting-config.xml). Indiquez le nom de la destination dans la propriété `destination` du composant RemoteObject. La classe de destination doit également comporter un constructeur no-args. Avec ColdFusion, vous pouvez procéder de cette manière pour configurer une destination au lieu d'appliquer la propriété `source` au composant RemoteObject.

Composant ColdFusion

L'application appelle le composant ColdFusion suivant. Ce code ColdFusion effectue des insertions et des interrogations de base de données SQL, puis renvoie les résultats de l'interrogation à l'application. La page ColdFusion utilise la balise `cfquery` pour insérer des données dans la base de données et pour l'interroger. Elle fait appel à la balise `cfreturn` pour formater les résultats de l'interrogation en tant qu'objet d'interrogation ColdFusion.

```
<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>
```

Appel de composants RemoteObject dans ActionScript

Dans l'exemple ActionScript suivant, l'appel de la méthode `useRemoteObject()` déclare le service, définit la destination, configure des écouteurs d'événement result et fault et procède à l'appel de la méthode `getList()` du service.

```
<?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>
```

Accès à des objets Java dans le chemin source

Le composant RemoteObject vous permet d'accéder à des objets Java sans état et avec état figurant dans le chemin source des applications Web LiveCycle Data Services, BlazeDS ou ColdFusion. Vous pouvez placer des fichiers de classes autonomes dans le répertoire WEB-INF/classes et des classes contenues dans des fichiers JAR (Java Archive) dans le répertoire WEB-INF/lib de l'application Web pour les ajouter au chemin source. Vous pouvez spécifier le nom de classe complet dans la propriété `source` d'une destination de service distant dans le fichier services-config.xml de LiveCycle Data Services, BlazeDS ou ColdFusion ou dans un fichier que le nom de classe complet inclut par référence (remoting-config.xml, par exemple). La classe doit également comporter un constructeur no-args. Pour ColdFusion, vous pouvez si vous le souhaitez définir la propriété `destination` du composant RemoteObject sur `Coldfusion` et la propriété `source` sur le nom complet d'un composant ColdFusion ou d'une classe Java.

Lorsque vous configurez une destination de service distant pour accéder à des objets sans état (étendue de la demande), Flex crée un objet différent pour chaque appel de méthode au lieu d'appeler les méthodes sur le même objet. Vous pouvez définir l'étendue d'un objet sur l'étendue de la demande (valeur par défaut), de l'application ou de la session. Les objets dans l'étendue de l'application sont accessibles à l'application Web qui contient l'objet. Les objets dans l'étendue de la session sont accessibles à toute la session client.

Lorsque vous configurez une destination d'objet distant pour accéder à des objets avec état, Flex crée l'objet une seule fois sur le serveur et maintient l'état entre les appels de méthode. Si le stockage de l'objet dans l'étendue de l'application ou de la session provoque des problèmes de mémoire, utilisez l'étendue de la demande.

Accès à des objets EJB et autres objets dans JNDI

Vous pouvez accéder à des objets EJB (Enterprise JavaBean) et autres objets stockés dans JNDI (Java Naming and Directory Interface) en appelant des méthodes sur une destination constituant une classe de façade de service qui recherche un objet dans JNDI et appelle sa méthode.

Vous pouvez utiliser des objets avec ou sans état pour appeler les méthodes d'objets Enterprise JavaBean et d'autres objets utilisant JNDI. Pour un objet EJB, vous pouvez appeler une classe de façade de service qui renvoie l'objet EJB de JNDI et appelle une méthode sur l'objet EJB.

Dans la classe Java, vous pouvez utiliser le schéma de codage Java standard, dans lequel vous créez un contexte initial et effectuez une recherche JNDI. Pour un objet EJB, utilisez également le schéma de codage standard dans lequel la classe contient des méthodes qui appellent la méthode `create()` de l'objet racine EJB et les méthodes métier de l'objet EJB résultant.

L'exemple suivant utilise une méthode appelée `getHelloData()` sur une destination de classe de façade :

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

Dans le côté Java, la méthode `getHelloData()` peut encapsuler tous les éléments nécessaires pour appeler une méthode métier sur un objet EJB. Dans l'exemple ci-après, la méthode Java effectue les actions suivantes :

- crée un nouveau contexte initial pour appeler l'objet EJB ;
- effectue une recherche JNDI qui obtient un objet racine EJB ;
- appelle la méthode `create()` de l'objet racine EJB ;
- appelle la méthode `sayHello()` de l'objet EJB.

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

Noms de méthodes réservés

La bibliothèque d'accès distant Flex utilise les noms de méthodes ci-dessous ; n'utilisez pas ces noms de méthodes pour vos propres méthodes :

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

De plus, ne commencez pas les noms de méthodes par le caractère de soulignement (_).

Pour accéder aux méthodes (opérations) RemoteObject, il suffit de les nommer d'après la variable de service. Des conflits de noms risquent toutefois de se produire si le nom d'une opération correspond à une méthode définie pour le service. Dans ActionScript, vous pouvez appliquer la méthode suivante à un composant RemoteObject pour renvoyer l'opération du nom donné :

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

Sérialisation entre ActionScript et Java

LiveCycle Data Services et BlazeDS sérialisent les données entre les types de données ActionScript (AMF 3), Java et ColdFusion dans les deux directions. Pour plus d'informations sur les types de données ColdFusion, voir la documentation ColdFusion.

Conversion des données d'ActionScript en Java

Le type des données envoyées d'une application à un objet Java par des paramètres de méthodes est automatiquement converti d'ActionScript en Java. Lorsque LiveCycle Data Services ou Blaze DS recherche une méthode appropriée sur l'objet Java, il utilise d'autres conversions, moins systématiques, pour trouver une correspondance.

Les types de données de base du client (les valeurs Boolean et String, par exemple) correspondent en général exactement à une API distante. Flex tente toutefois d'effectuer certaines conversions simples lorsqu'il recherche une méthode appropriée sur un objet Java.

Un tableau ActionScript peut indexer des entrées de deux manières. Un *tableau strict* est un tableau dans lequel tous les index sont des nombres. Un *tableau associatif* est un tableau dans lequel au moins un index est basé sur une chaîne. Il est important de savoir quel type de tableau vous envoyez au serveur, car ce type change le type de données des paramètres utilisés pour invoquer une méthode sur un objet Java. Un *tableau dense* est un tableau dans lequel tous les index numériques sont consécutifs, sans écart, commençant à 0 (zéro). Un tableau *très dense* est un tableau dans lequel il existe des écarts entre les index numériques ; le tableau est traité comme un objet et les index numériques deviennent des propriétés désérialisées dans un objet java.util.Map pour éviter l'envoi de nombreuses entrées nulles.

Le tableau suivant répertorie les conversions ActionScript (AMF 3) vers Java prises en charge pour les types de données simples.

Type ActionScript (AMF 3)	Désérialisation vers Java	Liaison de type Java prise en charge
Array (dense)	java.util.List	java.util.Collection, *Object*[ ] (tableau natif) Si le type est une interface, il est mappé sur les implémentations d'interface suivantes : • List devient ArrayList. • SortedSet devient TreeSet. • Set devient HashSet. • Collection devient ArrayList. Une nouvelle instance d'une implémentation Collection personnalisée est liée à ce type.
Array (sparse)	java.util.Map	java.util.Map
Boolean Chaîne `"true"` ou `"false"`	java.lang.Boolean	Boolean, boolean, String
flash.utils.ByteArray	byte []	
flash.utils.IExternalizable	java.io.Externalizable	
Date	java.util.Date (formaté pour UTC (Coordinated Universal Time))	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, types primitifs de double, long, float, int, short, byte
null	null	primitives
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 (zéro) Si `null` est envoyé, types primitifs de double, long, float, int, short, byte.
Object (generic)	java.util.Map	Si une interface Map est spécifiée, crée un objet java.util.HashMap pour java.util.Map et un nouvel objet java.util.TreeMap pour java.util.SortedMap.
String	java.lang.String	java.lang.String, java.lang.Boolean, java.lang.Number, java.math.BigInteger, java.math.BigDecimal, char[], tout type de nombre primitif
typed Object	typed Object Lorsque vous utilisez une balise de métadonnées `[RemoteClass]` qui spécifie un nom de classe distante. Le type Bean doit comporter un constructeur no-args public.	typed Object

Type ActionScript (AMF 3)	Désérialisation vers Java	Liaison de type Java prise en charge
non défini (undefined)	null	`null` pour Object, valeurs par défaut pour les primitives
XML	org.w3c.dom.Document	org.w3c.dom.Document
XMLDocument (type XML existant)	org.w3c.dom.Document	org.w3c.dom.Document Vous pouvez activer la prise en charge des données XML existantes pour le type XMLDocument sur tout canal défini dans le fichier services-config.xml. Ce paramètre n'intervient que dans l'envoi de données du serveur au client ; il détermine la manière dont les instances org.w3c.dom.Document sont envoyées à ActionScript. Pour plus d'informations, voir Configuration de la sérialisation AMF sur un canal.

Les valeurs primitives ne peuvent pas être définies sur `null` dans Java. Lors de la transmission de valeurs Boolean et Number du client vers un objet Java, Flex interprète les valeurs `null` comme valeurs par défaut des types primitifs ; par exemple, 0 pour double, float, long, int, short, byte, \u0000 pour char et `false` pour Boolean. Seuls les types Java primitifs obtiennent des valeurs par défaut.

LiveCycle Data Services et BlazeDS traitent les objets java.lang.Throwable comme tout autre objet typé. Ces objets sont traités avec des règles recherchant des champs publics et des propriétés bean. Des objets typés sont renvoyés au client. Ces règles sont identiques aux règles bean ordinaires, à ceci près qu'elles recherchent des getters pour les propriétés en lecture seule. La quantité d'informations obtenues d'une exception Java est ainsi plus importante. Afin de spécifier un comportement existant pour les objets Throwable, définissez la propriété `legacy-throwable` d'un canal sur `true`. Pour plus d'informations, voir Configuration de la sérialisation AMF sur un canal.

Vous pouvez transmettre des tableaux stricts en tant que paramètres à des méthodes s'attendant à une implémentation des API java.util.Collection ou de tableau Java natif.

Une collection Java peut contenir n'importe quel nombre de types d'objets, alors qu'un tableau Java nécessite que les entrées soient du même type (java.lang.Object[ ] et int[ ], par exemple).

LiveCycle Data Services et BlazeDS convertissent aussi les tableaux stricts ActionScript en implémentations appropriées pour les interfaces API Collection courantes. Par exemple, si un tableau strict ActionScript est envoyé à la méthode d'objet Java `public void addProducts(java.util.Set products)`, LiveCycle Data Services et BlazeDS le convertissent en une instance java.util.HashSet avant de le transmettre en tant que paramètre, car HashSet est une implémentation appropriée pour l'interface java.util.Set. De même, LiveCycle Data Services et BlazeDS transmettent une instance de java.util.TreeSet aux paramètres typés avec l'interface java.util.SortedSet.

LiveCycle Data Services et BlazeDS transmettent une instance de java.util.ArrayList aux paramètres typés avec l'interface java.util.List et toute autre interface qui étend java.util.Collection. Ces types sont ensuite renvoyés au client en tant qu'instances mx.collections.ArrayCollection. Pour renvoyer au client les tableaux ActionScript normaux, vous devez définir l'élément `legacy-collection` sur `true` dans la section `serialization` des propriétés d'une définition de canal. Pour plus d'informations, voir Configuration de la sérialisation AMF sur un canal.

Mappage explicite d'objets ActionScript et Java

Pour les objets Java que LiveCycle Data Services et BlazeDS ne traitent pas implicitement, les valeurs trouvées dans les propriétés bean publiques avec les méthodes get/set et les variables publiques sont envoyées au client en tant que propriétés sur un objet. Les propriétés privées, les constantes, les propriétés statiques et les propriétés en lecture seule entre autres ne sont pas sérialisées. Pour les objets ActionScript, les propriétés publiques définies avec les accesseurs get/set et les variables publiques sont envoyées au serveur.

LiveCycle Data Services et BlazeDS utilisent la classe Java standard java.beans.Introspector afin d'obtenir des descripteurs de propriétés pour une classe JavaBean. Il fait en outre appel à la réflexion pour collecter des champs publics sur une classe et a recours de préférence à des propriétés bean plutôt qu'à des champs. Les noms des propriétés Java et ActionScript doivent correspondre. Le code Flash Player natif détermine la manière dont les classes ActionScript sont introspectées sur le client.

Dans la classe ActionScript, utilisez la balise de métadonnées `[RemoteClass(alias=" ")]` pour créer un objet ActionScript se mappant directement sur l'objet Java. La classe ActionScript vers laquelle les données sont converties doit être utilisée ou référencée dans le fichier MXML afin d'être liée dans le fichier SWF et disponible lors de l'exécution. Pour ce faire, une manière efficace consiste à projeter l'objet de résultat, comme l'illustre l'exemple suivant :

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

La classe elle-même doit utiliser des références fortement typées de sorte que ses dépendances soient aussi liées.

Les exemples suivants présentent le code source d'une classe ActionScript utilisant la balise de métadonnées `[RemoteClass(alias=" ")]` :

```
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;
    }
}
```

Vous pouvez utiliser la balise de métadonnées `[RemoteClass]` sans alias si vous n'effectuez pas de mappage à un objet Java du serveur, mais que vous renvoyez le type d'objet depuis le serveur. L'objet ActionScript est sérialisé vers un objet Map spécial lorsqu'il est envoyé au serveur. L'objet renvoyé du serveur aux clients est toutefois le type ActionScript d'origine.

Pour empêcher l'envoi d'une propriété au serveur à partir d'une classe ActionScript, utilisez la balise de métadonnées `[Transient]` située au-dessus de la déclaration de cette propriété dans la classe ActionScript.

Conversion des données de Java en ActionScript

Un objet renvoyé par une méthode Java est converti de Java en ActionScript. LiveCycle Data Services et BlazeDS traitent également les objets figurant dans des objets. LiveCycle Data Services traite implicitement les types de données Java répertoriés dans le tableau suivant.

Type Java	Type ActionScript (AMF 3)
java.lang.String	String
java.lang.Boolean, boolean	Boolean
java.lang.Integer, int	int Si la valeur est < 0xF0000000 || value > 0x0FFFFFFF, la valeur est élevée à Number en raison des conditions de codage AMF.
java.lang.Short, short	int Si i est < 0xF0000000 || i > 0x0FFFFFFF, la valeur est élevée à Number.
java.lang.Byte, byte[]	int Si i est < 0xF0000000 || i > 0x0FFFFFFF, la valeur est élevée à Number.
java.lang.Byte[]	flash.utils.ByteArray
java.lang.Double, double	Number
java.lang.Long, long	Number
java.lang.Float, float	Number
java.lang.Character, char	String
java.lang.Character[], char[]	String
java. math.BigInteger	String
java.math.BigDecimal	String
java.util.Calendar	Date Les dates sont envoyées dans le fuseau horaire UTC (Coordinated Universal Time). Les clients et les serveurs doivent régler l'heure en fonction des fuseaux horaires.
java.util.Date	Date Les dates sont envoyées dans le fuseau horaire UTC (Coordinated Universal Time). Les clients et les serveurs doivent régler l'heure en fonction des fuseaux horaires.
java.util.Collection (java.util.ArrayList, par exemple)	mx.collections.ArrayCollection
java.lang.Object[]	Array
java.util.Map	Objet (non typé). Par exemple, un objet java.util.Map[] est converti en tableau (d'objets).
java.util.Dictionary	Objet (non typé)
org.w3c.dom.Document	Objet XML
null	null
java.lang.Object (autre type que les types précédemment répertoriés)	Objet typé Les objets sont sérialisés à l'aide de règles d'introspection JavaBean et incluent également des champs publics. Les champs statiques, transitoires ou non publics ainsi que les propriétés bean non publiques ou statiques sont exclus.

Configuration de la sérialisation AMF sur un canal

Vous pouvez prendre en charge la sérialisation du type AMF existant utilisé dans les versions précédentes de Flex et configurer d'autres propriétés de sérialisation dans les définitions de canaux du fichier services-config.xml.

Le tableau suivant décrit les propriétés que vous pouvez définir dans l'élément `<serialization>` d'une définition de canal.

Propriété	Description
`<ignore-property-errors>true</ignore-property-errors>`	La valeur par défaut est `true`. Détermine si le point de terminaison doit générer une erreur lorsqu'un objet entrant du client comporte des propriétés non prévues ne pouvant pas être définies sur l'objet du serveur.
`<log-property-errors>false</log-property-errors>`	La valeur par défaut est `false`. La valeur `true` active la journalisation des erreurs relatives aux propriétés non prévues.
`<legacy-collection>false</legacy-collection>`	La valeur par défaut est `false`. La valeur `true` active le renvoi des instances de java.util.Collection en tant que tableaux ActionScript. La valeur `false` active le renvoi des instances de java.util.Collection en tant qu'instances mx.collections.ArrayCollection.
`<legacy-map>false</legacy-map>`	La valeur par défaut est `false`. La valeur `true` active la sérialisation des instances java.util.Map en tant que tableau ECMA ou tableau associatif, et non en tant qu'objet anonyme.
`<legacy-xml>false</legacy-xml>`	La valeur par défaut est `false`. La valeur `true` active la sérialisation des instances org.w3c.dom.Document en tant qu'instances flash.xml.XMLDocument et non en tant qu'instances XML (compatibles E4X) intrinsèques.
`<legacy-throwable>false</legacy-throwable>`	La valeur par défaut est `false`. La valeur `true` active la sérialisation des instances java.lang.Throwable en tant qu'objets d'informations de statut AMF (à la place de la sérialisation bean normale, incluant les propriétés en lecture seule).

Propriété	Description
`<type-marshaller>className</type-marshaller>`	Spécifie une implémentation de flex.messaging.io.TypeMarshaller qui traduit un objet en instance de la classe désirée. Utilisé à l'invocation d'une méthode Java ou au remplissage d'une instance Java, lorsque le type de l'objet d'entrée de la désérialisation (un objet anonyme ActionScript par exemple est toujours désérialisé en tant qu'objet java.util.HashMap) ne correspond pas à l'API de destination (java.util.SortedMap, par exemple). Le type peut donc être converti en type souhaité.
`<restore-references>false</restore-references>`	La valeur par défaut est `false`. Commutateur avancé permettant à l'outil de désérialisation de conserver la trace des objets de références lorsqu'une traduction de type doit avoir lieu ; par exemple, lorsqu'un objet anonyme est envoyé pour une propriété de type java.util.SortedMap, l'objet est d'abord désérialisé en un objet java.util.Map en tant qu'objet normal, puis traduit en une implémentation appropriée de SortedMap (java.util.TreeMap, par exemple). Si d'autres objets pointaient vers le même objet anonyme d'un graphique d'objets, ce paramètre restaure ces références au lieu de créer des implémentations SortedMap. Notez que l'attribution de la valeur `true` à cette propriété peut conduire à un ralentissement considérable des performances en présence de larges volumes de données.
`<instantiate-types>true</instantiate-types>`	La valeur par défaut est `true`. Défini sur `false`, ce commutateur avancé empêche l'outil de désérialisation de créer des instances d'objets fortement typés, retient les informations de type et désérialise les propriétés brutes dans une implémentation Map, en particulier flex.messaging.io.ASObject. Les classes figurant dans les packagesflex.* sont toujours instanciées.

Utilisation de la sérialisation personnalisée

Si les mécanismes standard de sérialisation et de désérialisation de données entre ActionScript sur le client et Java sur le serveur ne répondent pas à vos besoins, vous pouvez écrire votre propre schéma de sérialisation. Vous implémentez l'interface flash.utils.IExternalizable ActionScript sur le client et l'interface java.io.Externalizable Java correspondante sur le serveur.

La sérialisation est souvent utilisée pour éviter de transmettre l'ensemble des propriétés de la représentation côté client ou côté serveur d'un objet à travers le niveau réseau. Lorsque vous implémentez la sérialisation personnalisée, vous pouvez coder vos classes afin que les propriétés spécifiques uniquement client ou uniquement serveur ne soient pas transmises via le réseau. Lorsque vous utilisez le schéma de sérialisation standard, toutes les propriétés publiques font l'aller-retour entre le client et le serveur.

Côté client, l'identité d'une classe qui implémente l'interface flash.utils.IExternalizable est écrite dans le flux de sérialisation. La classe sérialise et reconstruit l'état de ses instances. La classe implémente les méthodes `writeExternal()` et `readExternal()` de l'interface IExternalizable pour contrôler le contenu et le format du flux de sérialisation, mais pas le nom ou le type de classe, pour un objet et ses supertypes. Ces méthodes remplacent le comportement de sérialisation AMF natif. Elles doivent être symétriques à leur homologue distant pour enregistrer l'état de la classe.

Côté serveur, une classe Java qui implémente l'interface java.io.Externalizable exécute des fonctionnalités analogues à une classe ActionScript implémentant l'interface flash.utils.IExternalizable.

Remarque : *n'utilisez pas des types qui implémentent l'interface IExternalizable avec la classe HTTPChannel si une sérialisation par référence précise est requise. Cette opération conduit en effet à la perte des références entre les objets récurrents, qui apparaissent alors clonées au niveau du point de terminaison.*

L'exemple suivant présente le code source complet de la version (ActionScript) client d'une classe Product se mappant sur une classe Product Java sur le serveur. La classe Product client implémente l'interface IExternalizable et la classe Product serveur l'interface Externalizable.

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

La classe Product client utilise deux types de sérialisation. Elle utilise la sérialisation standard, compatible avec l'interface java.io.Externalizable, et la sérialisation AMF 3. L'exemple suivant présente la méthode `writeExternal()` de la classe Product client, qui utilise les deux types de sérialisation :

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

Ainsi que l'illustre l'exemple suivant, la méthode `writeExternal()` de la classe Product serveur est presque identique à la version client de cette méthode :

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

Dans la méthode `writeExternal()` de la classe Product client, la méthode `flash.utils.IDataOutput.writeFloat()` est un exemple de méthode de sérialisation standard qui répond aux spécifications des méthodes `java.io.DataInput.readFloat()` Java pour utiliser des types primitifs. Cette méthode envoie la propriété `price`, de type Float, à la classe Product serveur.

L'exemple de sérialisation AMF 3 dans la méthode `writeExternal()` de la classe Product client est l'appel à la méthode `flash.utils.IDataOutput.writeObject()`, qui se mappe à l'appel de la méthode `java.io.ObjectInput.readObject()` dans la méthode `readExternal()` de la classe Product serveur. La méthode `flash.utils.IDataOutput.writeObject()` envoie à la classe Product serveur la propriété `properties`, qui constitue un objet, et la propriété name, qui constitue une chaîne. Cette opération est possible car le point de terminaison AMFChannel comporte une implémentation de l'interface java.io.ObjectInput qui s'attend à ce que les données envoyées par la méthode `writeObject()` soient au format AMF 3.

Ensuite, lorsque la méthode `readObject()` est appelée dans la méthode `readExternal()` de la classe Product serveur, elle utilise la désérialisation AMF 3 ; c'est la raison pour laquelle la version ActionScript de la valeur `properties` est supposée être de type Map et la valeur `name` de type String.

L'exemple suivant présente la source complète de la classe Product serveur :

```
// 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;
        }
}
```

L'exemple suivant présente la méthode `readExternal()` de la classe Product serveur :

```
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));
    }
```

La méthode `writeExternal()` de la classe Product client n'envoie pas la propriété `id` au serveur pendant la sérialisation car elle n'est d'aucune utilité à la version serveur de l'objet Product. De même, la méthode `writeExternal()` de la classe Product serveur n'envoie pas la propriété `inventoryId` au client car il s'agit d'une propriété spécifique au serveur.

Notez que pendant la sérialisation, les noms des propriétés d'une classe Product ne sont envoyés ni dans un sens, ni dans l'autre. L'état de la classe étant fixe et gérable, les propriétés sont envoyées dans un ordre bien défini sans leur nom. La méthode `readExternal()` les lit dans l'ordre approprié.

Transmission de paramètres explicites et liaison de paramètres

L'appel des composants HTTPService, WebService et RemoteObject peut se faire de deux manières différentes : par la transmission de paramètres explicites et par la liaison de paramètres. La transmission de paramètres explicites consiste à fournir des informations à un service sous la forme de paramètres pour une fonction ActionScript. Cette manière d'appeler un service ressemble fortement à l'appel de méthodes sous Java. Les validateurs de données Flex ne peuvent pas être automatiquement utilisés avec la transmission de paramètres explicites.

La liaison de paramètres vous permet de copier des données de contrôles de l'interface utilisateur ou de modèles pour demander des paramètres. Elle n'est disponible que pour les composants d'accès aux données que vous déclarez dans MXML. Vous pouvez appliquer des validateurs aux valeurs des paramètres avant de soumettre des demandes aux services. Pour plus d'informations sur la liaison de données et les modèles de données, voir Data binding et Storing data. Pour plus d'informations sur la validation de données, voir Validating Data.

La liaison de paramètres consiste à déclarer des balises de paramètres de la méthode RemoteObject imbriquées dans une balise `<mx:arguments>` sous une balise `<mx:method>`, des balises de paramètres HTTPService imbriquées dans une balise `<mx:request>` ou des balises de paramètres d'opération WebService imbriquées dans une balise `<mx:request>` sous une balise `<mx:operation>`. La méthode `send()` permet d'envoyer la demande.

Transmission de paramètres explicites avec les composants RemoteObject et WebService

La transmission de paramètres explicites utilisée avec les composants RemoteObject est similaire à celle utilisée avec les composants WebService. L'exemple suivant illustre le code MXML utilisé pour déclarer un composant RemoteObject et appeler un service en utilisant la transmission de paramètres explicites dans l'écouteur d'événement click d'un contrôle Button. Un contrôle ComboBox fournit les données au service. Des écouteurs d'événement simple traitent les événements result et fault de niveau service.

```
<?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>
```

Transmission de paramètres explicites avec les composants HTTPService

La transmission de paramètres explicites avec les composants HTTPService diffère de la transmission de paramètres explicites avec les composants RemoteObject et WebService. L'appel d'un service se fait toujours avec la méthode `send()` d'un composant HTTPService. L'approche n'est donc pas la même qu'avec les composants RemoteObject et WebService, sur lesquels vous appelez des méthodes qui sont des versions côté client des méthodes ou opérations du service RPC.

La transmission de paramètres explicites vous permet de spécifier un objet contenant des paires nom-valeur comme paramètre de la méthode `send()`. Le paramètre de la méthode `send()` doit être un type de base simple ; vous ne pouvez pas utiliser d'objets imbriqués complexes car il n'existe aucune méthode générique permettant de les convertir en paires nom-valeur.

si vous ne spécifiez pas de paramètre de la méthode `send()`, le composant HTTPService utilise les paramètres d'interrogation spécifiés dans la balise `<mx:request>`.

Les exemples suivants illustrent deux manières d'appeler un service HTTP en utilisant la méthode `send()` avec un paramètre. Le second exemple indique également comment appeler la méthode `cancel()` pour annuler un appel de service HTTP.

```
<?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>
```

Liaison de paramètres avec les composants RemoteObject

Lorsque vous utilisez la liaison de paramètres avec des composants RemoteObject, vous déclarez toujours des méthodes dans la balise `<mx:method>` d'un objet RemoteObject.

Une balise `<mx:method>` peut contenir une balise `<mx:arguments>` contenant des balises enfant pour les paramètres de la méthode. La propriété name d'une balise `<mx:method>` doit correspondre au nom d'une méthode du service. L'ordre des balises d'argument doit correspondre à celui des paramètres de méthode du service. Vous pouvez nommer les balises d'argument de manière telle à ce que leurs noms correspondent le plus possible aux noms réels des paramètres de méthode correspondants, mais cela n'est pas nécessaire.

Remarque : *si les balises d'argument dans une balise `<mx:arguments>` portent le même nom, les appels de service échouent si la méthode distante n'attend pas un tableau comme unique source d'entrée. Aucun avertissement à ce sujet n'est affiché lorsque l'application est compilée.*

Vous pouvez lier des données aux paramètres de méthode d'un composant RemoteObject. Vous référencez les noms de balise des paramètres en vue de la liaison et de la validation de données.

L'exemple suivant illustre une méthode comportant deux paramètres liés à la propriété text des contrôles TextInput. Un validateur PhoneNumberValidator est assigné à `arg1`, qui est le nom de la première balise d'argument.

```
<?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 envoie les valeurs des balises d'argument à la méthode dans l'ordre spécifié par les balises MXML.

L'exemple suivant utilise la liaison de paramètres dans la balise `<mx:method>` d'un composant RemoteObject pour lier les données d'un élément ComboBox sélectionné à l'opération `employeeRO.getList` lorsque l'utilisateur clique sur un contrôle Button. La liaison de paramètres correspond à un appel de service par l'utilisation de la méthode `send()` sans paramètres.

```
<?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>
```

Si vous ignorez si le résultat d'un appel de service contient un tableau ou un objet individuel, vous pouvez utiliser la méthode `toArray()` de la classe mx.utils.ArrayUtil pour convertir le résultat en un tableau, comme l'indique cet exemple. Si vous transmettez la méthode `toArray()` à un objet individuel, il renvoie un tableau avec cet objet comme unique élément de tableau. Si vous transmettez un tableau à la méthode, elle renvoie le même tableau. Pour plus d'informations sur l'utilisation d'objets ArrayCollection, voir Data providers and collections.

Liaison de paramètres avec les composants HTTPService

Lorsqu'un service HTTP accepte des paramètres d'interrogation, vous pouvez les déclarer comme balises enfant d'une balise `<mx:request>`. Les noms des balises doivent correspondre aux noms des paramètres d'interrogation auxquels s'attend le service.

L'exemple suivant illustre la liaison de paramètres dans la balise `<mx:request>` d'un composant HTTPService pour lier les données d'un élément ComboBox sélectionné à la demande `employeeSrv` lorsque l'utilisateur clique sur un contrôle Button. La liaison de paramètres correspond à un appel de service par l'utilisation de la méthode `send()` sans paramètres. Cet exemple présente une propriété `url` sur le composant HTTPService. La manière dont vous appelez un service est toutefois la même, que vous vous connectiez au service directement ou passiez par une destination.

```
<?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>
```

Pour vérifier si le résultat d'un appel de service contient un tableau ou un objet individuel, vous pouvez utiliser la méthode `toArray()` de la classe mx.utils.ArrayUtil afin de convertir le résultat en tableau, comme l'illustre l'exemple précédent. Si vous transmettez la méthode `toArray()` à un objet individuel, il renvoie un tableau avec cet objet comme unique élément de tableau. Si vous transmettez un tableau à la méthode, elle renvoie le même tableau. Pour plus d'informations sur l'utilisation d'objets ArrayCollection, voir Data providers and collections.

Liaison de paramètres avec les composants WebService

Lorsque vous utilisez la liaison de paramètres avec un composant WebService, vous déclarez toujours les opérations dans les balises `<mx:operation>` du composant WebService. Une balise `<mx:operation>` peut contenir une balise `<mx:request>` comportant les nœuds XML auxquels s'attend l'opération. La propriété name d'une balise `<mx:operation>` doit correspondre à l'un des noms de l'opération WebService.

Vous pouvez lier des données aux paramètres des opérations WebService. Vous référencez les noms de balise des paramètres en vue de la liaison et de la validation de données.

L'exemple suivant illustre comment la liaison de paramètres est utilisée dans la balise `<mx:operation>` d'un composant WebService pour lier les données d'un élément ComboBox sélectionné à l'opération `employeeWS.getList` lorsque l'utilisateur clique sur un contrôle Button. La balise `<deptId>` correspond directement au paramètre `deptId` de l'opération getList. La liaison de paramètres correspond à un appel de service par l'utilisation de la méthode `send()` sans paramètre. Cet exemple présente une propriété destination sur le composant WebService. La manière dont vous appelez un service est toutefois la même, que vous vous connectiez au service directement ou passiez par une destination.

```
<?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>
```

Vous pouvez aussi spécifier manuellement un corps de demande SOAP entier dans XML avec toutes les informations d'espace de noms correctement définies dans une balise `<mx:request>`. Pour ce faire, attribuez la valeur `xml` à l'attribut `format` de la balise `<mx:request>`, comme l'illustre l'exemple suivant :

```
<?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>
```

Traitement des résultats des services

Après que le composant RPC a appelé un service, les données renvoyées par le service sont placées dans un objet lastResult. Par défaut, la valeur de la propriété `resultFormat` des composants HTTPService et des opérations du composant WebService est `object` et les données renvoyées sont représentées par une arborescence simple d'objets ActionScript. Flex interprète les données XML renvoyées par un service Web ou HTTP pour représenter de manière appropriée les types de base, tels que String, Number, Boolean et Date. Pour utiliser des objets fortement typés, remplissez ces objets en utilisant l'arborescence d'objets créée par Flex.

Les composants WebService et HTTPService renvoient chacun des objets et des tableaux anonymes constituant des types complexes. Si `makeObjectsBindable` a la valeur (par défaut) `true`, les objets sont enveloppés dans des instances mx.utils.ObjectProxy et les tableaux dans des instances mx.collections.ArrayCollection.

Remarque : *ColdFusion n'est pas sensible à la casse et applique en interne la mise en majuscule à toutes ses données. Gardez à l'esprit cette caractéristique lorsque vous utilisez un service Web ColdFusion.*

Traitement des événements result et fault

Lorsqu'un appel de service est terminé, la méthode RemoteObject, l'opération WebService ou le composant HTTPService distribue un événement result ou un événement fault. Un *événement result* indique que le résultat est disponible. Un *événement fault* indique qu'une erreur s'est produite. L'événement result fait office de déclencheur de la mise à jour des propriétés liées à lastResult. Vous pouvez traiter les événements fault et result explicitement en ajoutant des écouteurs d'événement aux méthodes RemoteObject ou aux opérations WebService. Pour un composant HTTPService, vous pouvez spécifier les écouteurs d'événement result et fault sur le composant lui-même. Le composant HTTPService ne dispose en effet pas de plusieurs méthodes ou opérations.

Lorsque vous ne spécifiez pas d'écouteurs pour les événements result ou fault sur une méthode RemoteObject ou une opération WebService, les événements sont transmis au niveau du composant. Vous pouvez spécifier des écouteurs pour les événements result et fault à niveau de composant.

Dans l'exemple MXML suivant, les événements `result` et `fault` d'une opération WebService spécifient les écouteurs d'événement ; l'événement `fault` du composant WebService spécifie également un écouteur d'événement :

```
<?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>
```

Dans l'exemple ActionScript suivant, un écouteur d'événement result est ajouté à une opération WebService. Un écouteur d'événement fault est ajouté au composant WebService :

```
<?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>
```

Vous pouvez également utiliser l'événement mx.rpc.events.InvokeEvent pour indiquer lorsqu'une requête de composant d'accès aux données a été invoquée. Cela est utile si les opérations sont placées en file d'attente et invoquées ultérieurement.

Traitement de résultats en tant que données XML avec le format de résultat E4X

Vous pouvez définir la propriété `resultFormat` des composants HTTPService et des opérations WebService sur la valeur e4x pour créer une propriété lastResult de type XML. Vous pouvez accéder à la propriété `lastResult` en utilisant ECMAScript pour les expressions XML (E4X). Vous n'incluez pas le nœud racine de la structure XML dans la notation par point lorsque vous utilisez un objet XML `E4X` dans une expression de liaison. Cela diffère de la syntaxe d'une propriété `lastResult` définie sur object pour lequel vous devez inclure le nœud racine de la structure XML dans la notation par point. Par exemple, lorsque la propriété lastResult est définie sur `e4x`, vous utilisez `{srv.lastResult.product}` ; lorsque la propriété lastResult est définie sur object, vous utilisez `{srv.lastResult.products.product}`.

L'utilisation du format de résultat `e4x` est à privilégier pour travailler directement dans XML. Vous pouvez toutefois également définir la propriété `resultFormat` sur `xml` pour créer un objet `lastResult` de type flash.xml.XMLNode, qui est l'objet existant pour utiliser XML. De plus, vous pouvez définir la propriété `resultFormat` des composants HTTPService sur `flashvars` ou `text` pour créer des résultats en tant qu'objets ActionScript contenant des paires nom-valeur ou en tant que texte brut, respectivement.

Remarque : *pour appliquer la syntaxe E4X aux résultats de service, vous devez définir la propriété `resultFormat` du composant HTTPService ou WebService sur `e4x`. La valeur par défaut est `object`.*

Lorsque vous définissez la propriété `resultFormat` d'un composant HTTPService ou d'une opération WebService sur `e4x`, vous devrez éventuellement traiter les informations d'espace de noms contenues dans le code XML renvoyé. Pour un composant WebService, les informations d'espace de noms sont contenues dans le corps de l'enveloppe SOAP renvoyée par le service Web. L'exemple suivant présente une partie d'un corps SOAP contenant des informations d'espace de noms. Ces données ont été renvoyées par un service Web qui obtient des cotes de titres. Les informations d'espace de noms sont en gras.

```
...
<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>
...
```

Comme le corps soap:Body contient des informations d'espace de noms, si vous définissez la propriété `resultFormat` de l'opération WebService sur `e4x`, créez un objet d'espace de noms pour l'espace de noms http://ws.invesbot.com/. L'exemple suivant présente une application effectuant cette opération :

```
<?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>
```

Si vous le souhaitez, vous pouvez créer une variable pour un espace de noms et y accéder dans une liaison au résultat de service, comme l'illustre l'exemple suivant :

```
<?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>
```

Utilisez la syntaxe E4X pour accéder aux éléments et aux attributs des données XML renvoyées dans un objet lastResult. La syntaxe varie en fonction de la présence ou non d'un ou de plusieurs espaces de noms déclarés dans les données XML.

Aucun espace de noms

L'exemple suivant indique comment obtenir une valeur d'élément ou d'attribut lorsqu'aucun espace de noms n'est spécifié sur l'élément ou l'attribut :

```
var attributes:XMLList = XML(event.result).Description.value;
```

Le code précédent renvoie xxx pour le document XML suivant :

```
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <Description>
        <value>xxx</value>
    </Description>
</RDF>
```

Tout espace de noms

L'exemple suivant indique comment obtenir une valeur d'élément ou d'attribut lorsqu'un espace de noms quelconque est spécifié sur l'élément ou l'attribut :

```
var attributes:XMLList = XML(event.result).*::Description.*::value;
```

Le code précédent renvoie xxx pour l'un des documents XML suivants :

Premier document XML :

```
<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>
```

Deuxième document XML :

```
<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>
```

Espace de noms spécifique

L'exemple suivant indique comment obtenir une valeur d'élément ou d'attribut lorsque l'espace de noms rdf déclaré est spécifié sur l'élément ou l'attribut :

```
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;
```

Le code précédent renvoie `xxx` pour le document XML suivant :

```
<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>
```

L'exemple suivant présente une autre manière d'obtenir une valeur d'élément ou d'attribut lorsque l'espace de noms rdf déclaré est spécifié sur un élément ou un attribut :

```
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;
```

Le code précédent renvoie également `xxx` pour le document XML suivant :

```
<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>
```

Traitement de résultats de service Web contenant des objets DataSet ou DataTable .NET

Les services Web écrits avec la structure Microsoft .NET peuvent renvoyer au client des objets DataSet ou DataTable .NET spéciaux. Un service Web .NET fournit un document WSDL basique sans informations sur le type de données qu'il manipule. Lorsque le service Web renvoie un objet DataSet ou DataTable, les informations sur le type de données sont incorporées dans un élément de schéma XML dans le message SOAP, qui indique comment le reste du message doit être traité. Pour traiter au mieux les résultats de ce type de service Web, définissez la propriété `resultFormat` d'une opération WebService Flex sur `object`. Vous pouvez également définir la propriété `resultFormat` de l'opération WebService sur `e4x`. Les formats XML et e4x ne sont toutefois pas adaptés à la structure inhabituelle de la réponse et la liaison des données (par exemple à un contrôle DataGrid) nécessite l'implémentation de solutions de contournement.

La définition de la propriété `resultFormat` d'une opération WebService Flex sur `object` entraîne la conversion automatique des DataTable ou DataSet renvoyés par un service Web .NET en objets comportant une propriété Tables, qui contient le mappage d'un ou de plusieurs objets DataTable. Chaque objet DataTable du mappage Tables contient deux propriétés : Columns et Rows. La propriété Rows contient les données. L'objet event.result obtient les propriétés suivantes correspondant aux propriétés DataSet et DataTable dans .NET. Les tableaux d'objets DataSet ou DataTable comportent les mêmes structures que celles décrites ici, mais sont imbriqués dans un tableau de niveau supérieur sur l'objet de résultat.

Propriété	**Description**
`result.Tables`	Mappage des noms de tableaux à des objets contenant des données de tableaux.
`result.Tables["someTable"].Columns`	Tableau des noms des colonnes suivant l'ordre spécifié dans le schéma DataSet ou DataTable pour le tableau.
`result.Tables["someTable"].Rows`	Tableau d'objets représentant les données de chaque ligne du tableau, par exemple {columnName1:value, columnName2:value, columnName3:value}.

L'application MXML suivante renseigne un contrôle DataGrid avec les données DataTable renvoyées par un service Web .NET.

```
<?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>
```

L'exemple suivant illustre la classe .NET C#. Cette classe est l'implémentation back-end du service Web appelée par l'application et utilise la base de données modèle Microsoft SQL Server Northwind :

:

```
<%@ 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;
    }
}
```