

Remoting Basics

Client-Server Architecture

Flex Client

Flex Framework

Data Services Components

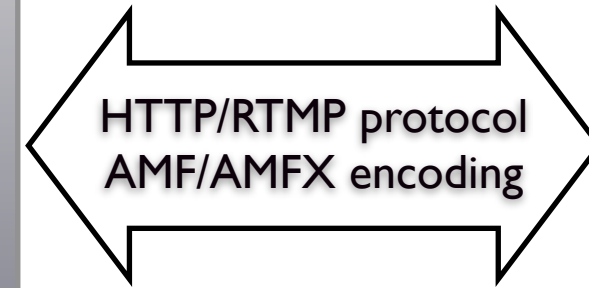
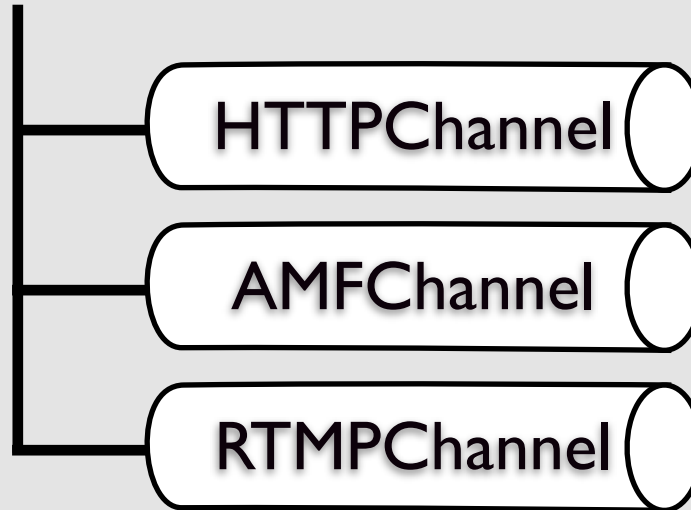
HTTPService/WebService

RemoteObject

Producer/Consumer

DataService

ChannelSet



Web Server

Java, PHP, .NET, etc.

Data Services
Server Components

Client Components

HTTPService

- the result format can be Object, E4X or plain text
- conversion to strongly-typed data needs to be done manually
- designed for the REST paradigm (REpresentation State Transfer)

WebService

- **designed for the RPC paradigm**
- **as a result of the dynamic processing of the WSDL it builds a dynamic proxy**
- **you can generate a static proxy with the *Web Service Import Wizard***

RemoteObject

- uses the AMF serialization protocol
- very efficient
- mapping between the client and server classes is done with the *[RemoteClass]* metadata tag

Processing the results

- the communication is asynchronous
- you can have an *IResponder*

```
var operation : AbstractOperation = service.getOperation(operationName);  
operation.arguments = arguments;
```

```
var token : AsyncToken = operation.send();  
token.addResponder(new AsyncResponder(resultHandler, faultHandler, token));
```

- data binding to *lastResult*

```
<mx:HTTPService id="srv" destination="catalog" useProxy="true"/>  
<mx:DataGrid dataProvider="{srv.lastResult.catalog.product}"/>  
<mx:Button label="Get Data" click="invokeService()"/>
```

- handling *result* and *fault* events

```
<mx:WebService id="srv" destination="ws-catalog"  
  result="trace('HANDLING RESULT...' + event.result.toString());"  
  fault="trace('SOMETHING WRONG HAPPENED')"/>
```

DataServices

- **synchronizing collections between the client and the server**
- **conflicts resolution**
- **data push**
- **paging and lazy loading of hierarchical data**
- **managing the changes on the client with the [Managed] metadata tag**
- **use fill() and commit() to perform synchronization with the server**
- **use createItem, getItem, deleteItem and releaseItem to manage data on an item level**
- **mapping between the client and server classes is done with the [RemoteClass] metadata tag**

Services & Destinations

- **services are the target of messages from the client-side Flex components**
- **think of destinations as instances of a service configured in a certain way**
- **you can configure services and their destinations in the services-config.xml, but the best practice is to use separate files:**
 - **remoting-config.xml**
 - **proxy-config.xml**
 - **messaging-config.xml**
 - **data-management-config.xml**

Summary

- **overview of the client-server architecture**
- **HTTPService**
- **WebService**
- **RemoteObject**
- **processing the results**