Building Service Clients with Adobe AIR and Flex

A Tutorial from Technical Evangelism @ Adobe (contributions from others). MAJOR thanks to Sujit Reddy Gurrala, Ben Forta and

Synopsis:

This course covers how to build clients to various server side technologies and using multiple protocols available via the Flash Builder 4, Flex 4 and AIR 2.0 technologies. The course is updated to take advantages of the new Flash Builder 4 IDE and the Spark, Halo and FXG component libraries. This document contains all the necessary materials to prepare someone to deliver this course. You will need to download some additional resources as noted herein.

SECTION ONE: PREPARATION 3

Forward: 3

Audience Assumptions 3

Pre-Requisites and Downloads 3

Install Checklist 4

1. Adobe Flash Builder 4 4
2. Java ™ SDK 1.5 or higher 4
3. BlazeDS "Vancouver" build 5

4 MAMP/WAMP/LAMP 11

5. Cold Fusion 17

SECTION TWO: THE LABS 18

PROJECT 1: HTML CLIENT 18

Project 1 Solution Code 21

PROJECT 2: REST STYLE HTTPSERVICE 21

Project 2 Solution Code 31

PROJECT 3: WEB SERVICE INTROSPECTION AND CONSUMPTION 32

Project 3 Solution Code: 42

PROJECT 4: FLEX REMOTING PROJECT 43

Project 4 Solution Code 48

PROJECT 5: WORKING WITH PHP AND DATA PAGING 49

Project 5 Solution Code 68

PROJECT 6: - FLASH BUILDER 4 TALKS TO COLDFUSION 69

Project 6 Solution Code 74

If you wish to contribute revisions to this document, please contact the editor dnickull at adobe dot com.

Section One: Preparation

Forward:

This ***Building Service Clients with Adobe Flex and AIR*** course was written for Adobe MAX 2009. This course has been put together in hopes to provide developers a boot camp to learn all the basics of how to build multiple types of service clients and control various aspects of each service. Although we tried hard to cover everything, it was simply not possible in such a short time. Our overall goal is to provide you with an introduction to this topic so you can make your own decision if you want to pursue this exciting new application development technology in the future. If you do, we will be providing additional references where you can continue learning and become part of the larger community after this course is over.

This course is written in such a manner that you should be able to take this alone as a self paced tutorial, although during MAX, we will be there to lead you through it and help in the event you encounter any problems.

IMPORTANT: There is a high probability we will not get through the entire course during the 90 minutes time allotted at MAX. This is by design. The course reflects a best case scenario whereby everyone covers the materials quickly. We felt it better to have extra rather than not enough content. If we do not get through the entire course, you can take the remaining labs by yourself as this instructional handout has sufficient notes to complete everything.

We hope you enjoy this course as much as we enjoyed putting it together. Remember – we are here for you. Don't hesitate to ask any questions during the event and afterwards.

Audience Assumptions

.
- Attendees are familiar with XML
- Attendees understand the basics of network architecture and various protocols

Pre-Requisites and Downloads

Before you can take this course, you must have the following software installed on your computer:

- Adobe Flash Builder 4 or later
- Adobe AIR 1.5 runtime or later (AIR 2.0 preferred)

- Java ™ SDK 1.5 or higher
- BlazeDS Vancouver build
- MAMP (Windows), LAMP (Linux) or WAMP (Windows)
- ColdFusion 8 (9 preferred)

Additionally, you must have the CD of courseware that includes the two directories and a soft copy of this handout.

Install Checklist

1. Adobe Flash Builder 4

– from http://labs.adobe.com/technologies/flashbuilder4/

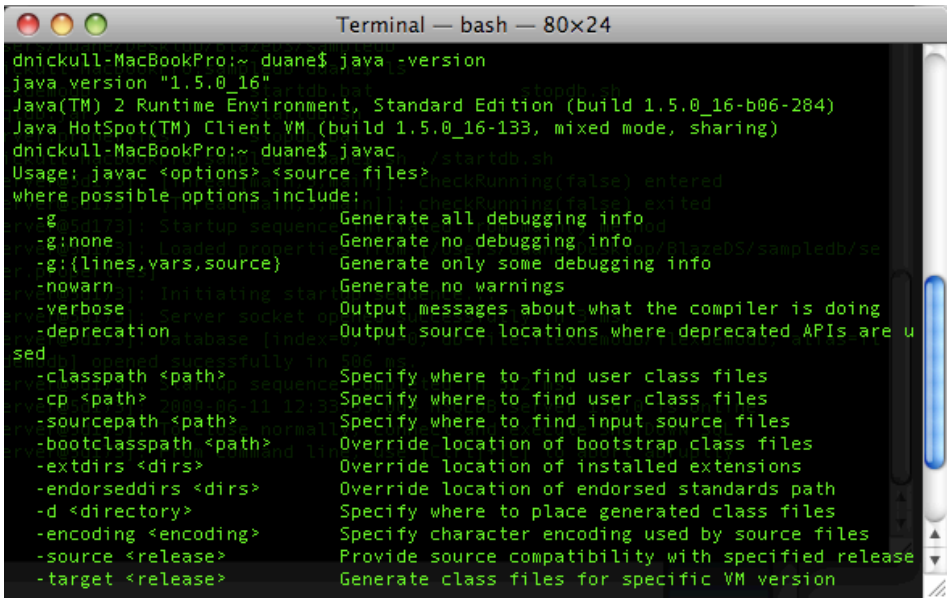
- follow the instructions for installing on your particular operating system.
- This will also install the AIR runtime. If it does not, get AIR from http://www.adobe.com/products/air/

2. Java ™ SDK 1.5 or higher

On most Macintosh and Linux systems, this will be preinstalled and configured for you. In order to run the BlazeDS, you must have the Java SDK installed and both the JAVA_HOME and PATH environmental variables set properly. To test this, open a terminal (command window) and type in the following commands

```
javac <enter>

java – version <enter>
```

NOTE: Do not type "<enter>", this means to hit the enter key. You should see similar to the following:



If you do not see this, please refer to the lab setup guide and install java by downloading and installing it as per the instructions for your particular operating system. Download and install Java JDK 1.5.0_16. (NOTE: it is possible that build 16 will be superceded by another build.

Subsequent builds of JDK 1.5.0 should work). Java is available from http://java.sun.com/javase/downloads/index_jdk5.jsp. Follow the instructions for your particular operating system as detailed in the instructions and set both your PATH and JAVA_HOME environmental variables.

When done properly, you should be able to type in "java -version" and "javac" in a command window (PC) or terminal prompt (Unix, Linux, Mac OSX) and see the results above.

If you do not have your PATH and JAVA_HOME set properly, see the documentation at http://www.java.com/en/download/help/path.xml

3. BlazeDS "Vancouver" build

The BLazeDS Vancouver build is available from http://www.web2open.org/downloads/BlazeDS.zip.

When you get it:

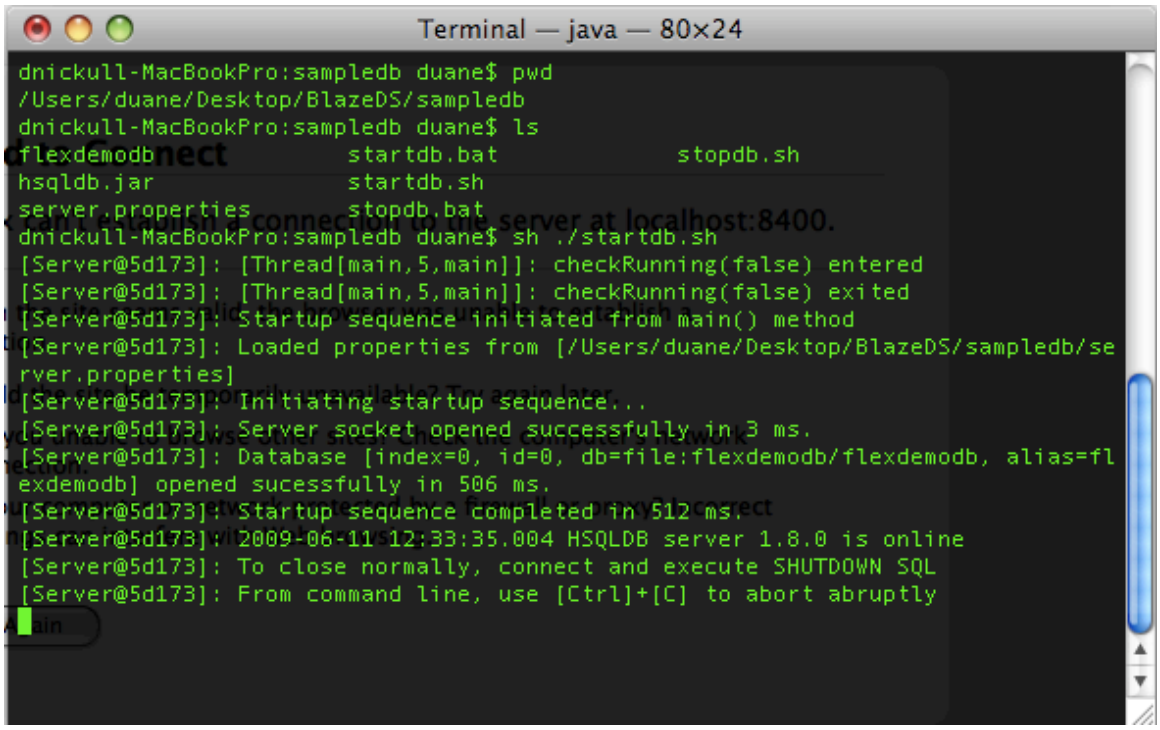
1. Unzip it to your hard drive somewhere. The root directory where you install it will be referred to as <BlazeDS_Root>.

2. Next, start up the BlazeDS server. To do this, open a Command Window (or shell) and navigate to **<BlazeDS_Root>\BlazeDS\sampledb** (if on OSX, use forward slashes instead of back for path).

3. On Windows, type in "**startdb**". This should start the database.

```
C:\Documents and
Settings\Administrator\Desktop\MAX2008_BuildingServiceClients\s
ampledb>startdb

C:\Documents and
Settings\Administrator\Desktop\MAX2008_BuildingServiceClients\s
ampledb>java -cp hsqldb.jar org.hsqldb.Server
[Server@1d58aae]: [Thread[main,5,main]]: checkRunning(false)
entered
[Server@1d58aae]: [Thread[main,5,main]]: checkRunning(false)
exited
[Server@1d58aae]: Startup sequence initiated from main() method
[Server@1d58aae]: Loaded properties from [C:\Documents and
Settings\Administrato
r\Desktop\MAX2008_BuildingServiceClients\sampledb\server.propertie
s]
[Server@1d58aae]: Initiating startup sequence...
[Server@1d58aae]: Server socket opened successfully in 40 ms.
[Server@1d58aae]: Database [index=0, id=0,
db=file:flexdemodb/flexdemodb, alias=
flexdemodb] opened sucessfully in 1512 ms.
[Server@1d58aae]: Startup sequence completed in 1552 ms.
[Server@1d58aae]: 2008-11-25 12:29:14.087 HSQLDB server 1.8.0 is
online
[Server@1d58aae]: To close normally, connect and execute SHUTDOWN
SQL
[Server@1d58aae]: From command line, use [Ctrl]+[C] to abort
abruptly
```

   a. On Mac, type **sudo sh ./startdb.sh**



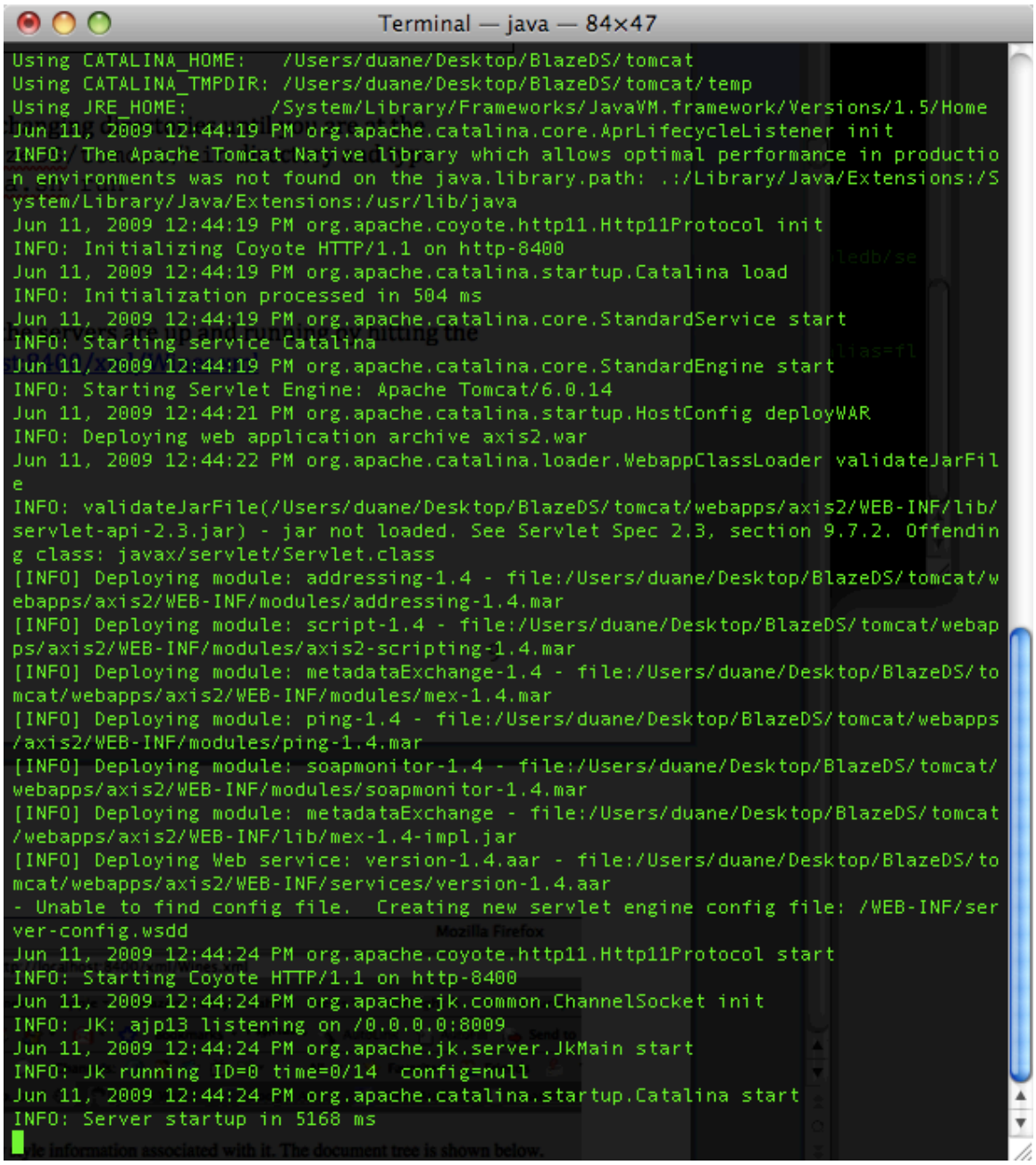
4. Now start the servers.

    a. On Windows, do this by changing directories until you are at the **<BlazeDS_Root>\BlazeDS\tomcat\bin** directory (note: use forward slashes on Unix based Systems) and type **catalina run**

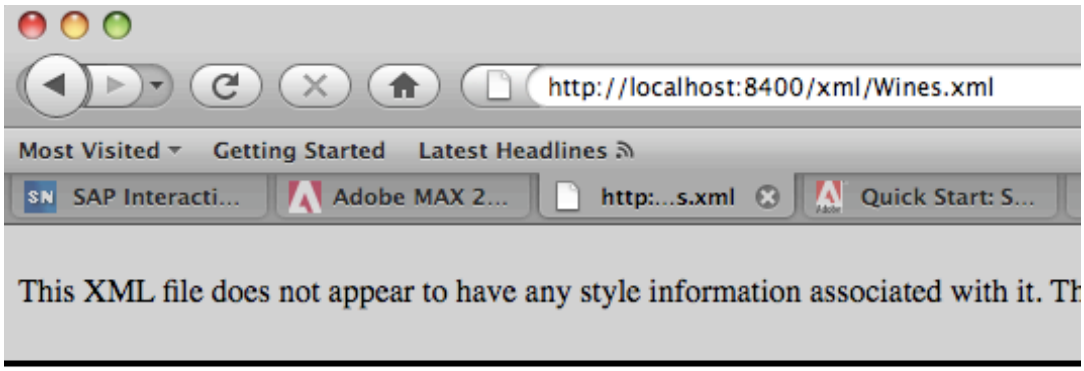
```
C:\Documents and
Settings\Administrator\Desktop\MAX2008_BuildingServiceClients\t
omcat\bin>catalina run

Using CATALINA_BASE:   C:\Documents and
Settings\Administrator\Desktop\MAX2008_B
uildingServiceClients\tomcat
Using CATALINA_HOME:   C:\Documents and
Settings\Administrator\Desktop\MAX2008_B
uildingServiceClients\tomcat
Using CATALINA_TMPDIR: C:\Documents and
Settings\Administrator\Desktop\MAX2008_B
uildingServiceClients\tomcat\temp
Using JRE_HOME:        C:\Program Files\Java\jdk1.6.0_10
Nov 25, 2008 12:34:20 PM org.apache.catalina.core.AprLifecycleListener init
INFO: The Apache Tomcat Native library which allows optimal performance in
produ
ction environments was not found on the java.library.path: C:\Program
Files\Java
\jdk1.6.0_10\bin;.;C:\WINDOWS\Sun\Java\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\CF
u
sionMX7\verity\k2\_nti40\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32
```

```
\
Wbem;C:\Program Files\ATI Technologies\ATI Control Panel;C:\Program Files\PC-
Doc
tor for Windows\services;C:\Program Files\ATI Technologies\Fire GL 3D Studio
Max
;C:\WINDOWS\Downloaded Program Files;C:\Program Files\Common
Files\MXI;C:\Progra
m Files\Common Files\Adobe\AGL;C:\Program Files\QuickTime\QTSystem\
Nov 25, 2008 12:34:20 PM org.apache.coyote.http11.Http11Protocol init
INFO: Initializing Coyote HTTP/1.1 on http-8400
Nov 25, 2008 12:34:20 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 432 ms
Nov 25, 2008 12:34:20 PM org.apache.catalina.core.StandardService start
INFO: Starting service Catalina
Nov 25, 2008 12:34:20 PM org.apache.catalina.core.StandardEngine start
INFO: Starting Servlet Engine: Apache Tomcat/6.0.14
Nov 25, 2008 12:34:24 PM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8400
Nov 25, 2008 12:34:24 PM org.apache.jk.common.ChannelSocket init
INFO: JK: ajp13 listening on /0.0.0.0:8009
Nov 25, 2008 12:34:24 PM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/20  config=null
Nov 25, 2008 12:34:24 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 3988 ms
```

b. On Mac OSX, do this by changing directories until you are at the **<Course_Root>/BlazeDS/tomcat/bin** directory and type
sudo sh ./catalina.sh run



5. You should be able to validate the servers are up and running by hitting the following URL: http://localhost:8400/xml/Wines.xml




```
– <Wines>
  – <Vintage>
      <Vintner>Gaja</Vintner>
      <Name>Conteisa</Name>
      <Vintage>2001</Vintage>
      <ParkerNotation>94</ParkerNotation>
      <Price>$324.00</Price>
    </Vintage>
  + <Vintage></Vintage>
  + <Vintage></Vintage>
  + <Vintage></Vintage>
  – <Vintage>
      <Vintner>Mouton Rothschild</Vintner>
      <Name>Reserve</Name>
      <Vintage>1985</Vintage>
      <ParkerNotation>93</ParkerNotation>
      <Price>$1,324.00</Price>
    </Vintage>
  – <Vintage>
      <Vintner>Screaming Eagle</Vintner>
      <Name>Reserve</Name>
      <Vintage>1996</Vintage>
      <ParkerNotation>99</ParkerNotation>
      <Price>$2,453.00</Price>
    </Vintage>
  + <Vintage></Vintage>
  – <Vintage>
      <Vintner>Vincorp</Vintner>
      <Name>Osoyoos Larose</Name>
      <Vintage>2002</Vintage>
      <ParkerNotation>94</ParkerNotation>
      <Price>$324.00</Price>
    </Vintage>
  </Wines>
```

Please then test the following URL's to make sure they are there:

http://localhost:8400/axis – to make sure Apache Axis 1.4 is up and running fine

http://localhost:8400/axis2 – to make sure Apache Axis 1.4 is up and running fine

http://localhost:8400/xml/Wines.xml

http://localhost:8400/xml/crossdomain.xml

4 MAMP/WAMP/LAMP

MAMP may be downloaded from:
http://www.mamp.info/en/download.html

WAMP may be downloaded from:
http://www.wampserver.com/en/download.php

To augment the WAMP or MAMP stack, you must create a Database in the MySQL instance by using the MySQL here:

```
CREATE DATABASE /*!32312 IF NOT EXISTS*/ evangelistdashboard;
USE evangelistdashboard;
--
-- Table structure for table `evangelistdashboard`.`customers`
--

DROP TABLE IF EXISTS `customers`;
CREATE TABLE `customers` (

 `customer_id` int(11) NOT NULL auto_increment,

 `customer_name` varchar(225) default NULL,

 `customer_address` blob,

 `customer_type` varchar(100) default NULL,

 `entry_created_user` int(11) default NULL,

 `entry_edited_user` int(11) default NULL,

 `entry_modified_date` datetime default NULL,

 PRIMARY KEY  (`customer_id`),
```

```
 UNIQUE KEY `customer_name` (`customer_name`)

) ENGINE=InnoDB AUTO_INCREMENT=82 DEFAULT CHARSET=utf8;

--
-- Dumping data for table `evangelistdashboard`.`customers`
--

/*!40000 ALTER TABLE `customers` DISABLE KEYS */;

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (1,'Customer 01',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-09
15:57:05'),

 (2,'Customer 2',0x656C656374726F6E696320636974792C2042616E67616C6F7265,'Company',2,2,'2009-05-
28 14:29:50'),

 (3,'Customer 3',0x42544D2C2042616E67616C6F7265,'Training institute',2,2,'2009-05-28 14:29:50'),

 (5,'Customer 04',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:50'),

 (6,'Customer 5',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (7,'Customer 6',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (8,'Customer 7',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (9,'Customer 8',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (10,'Customer 9',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (11,'Customer 10',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (12,'Customer 11',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (13,'Customer 12',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (14,'Customer 13',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (15,'Customer 14',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (16,'Customer 15',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
```

```
10:18:55'),

 (17,'Customer 16',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (18,'Customer 17',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (19,'Customer 18',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (20,'Customer 19',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (21,'Customer 20',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (22,'Customer 21',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (23,'Customer 22',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (24,'Customer 23',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (25,'Customer 24',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (26,'Customer 25',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (27,'Customer 26',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (28,'Customer 27',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (29,'Customer 28',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (30,'Customer 29',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (31,'Customer 30',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (32,'Customer 31',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (33,'Customer 32',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),
```

```
 (34,'Customer 33',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (35,'Customer 34',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (36,'Customer 35',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (37,'Customer 36',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (38,'Customer 37',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (39,'Customer 38',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (40,'Customer 39',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (41,'Customer 40',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (42,'Customer 41',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (43,'Customer 42',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (44,'Customer 43',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (45,'Customer 44',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (46,'Customer 45',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (47,'Customer 46',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (48,'Customer 47',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (49,'Customer 48',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (50,'Customer 49',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (51,'Customer 50',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),
```

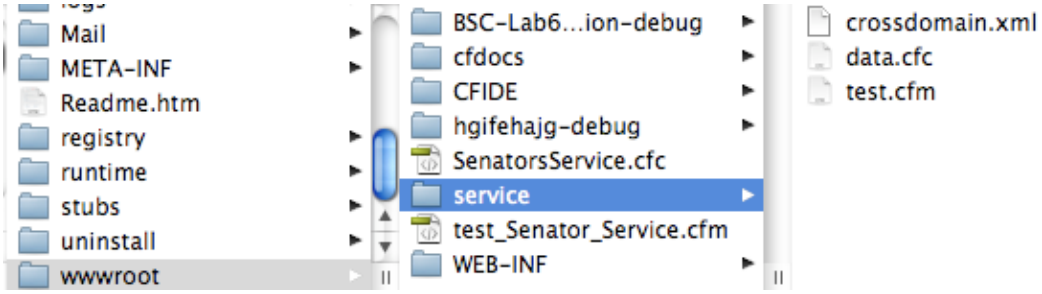
```
 (52,'Customer 51',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (53,'Customer 52',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (54,'Customer 53',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (55,'Customer 54',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (56,'Customer 55',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (57,'Customer 56',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (58,'Customer 57',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:55'),

 (59,'Customer 58',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (60,'Customer 59',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (61,'Customer 60',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (62,'Customer 61',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (63,'Customer 62',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (64,'Customer 63',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (65,'Customer 64',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (66,'Customer 65',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (67,'Customer 66',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (68,'Customer 67',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (69,'Customer 68',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
```

```
10:18:56'),

 (70,'Customer 69',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (71,'Customer 70',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (72,'Customer 71',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (73,'Customer 72',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56');

INSERT INTO `customers`
(`customer_id`,`customer_name`,`customer_address`,`customer_type`,`entry_created_user`,`entry_edited_user`,`e
ntry_modified_date`) VALUES

 (74,'Customer 73',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (75,'Customer 74',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (76,'Customer 75',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (77,'Customer 76',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (78,'Customer 77',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (79,'Customer 78',0x4C616E67666F726420726F61642C2042616E67616C6F7265,'Company',2,2,'2009-06-03
10:18:56'),

 (81,'added customer 2',0x536F6D652061646472657373,'Test',2,2,'2009-06-03 20:06:32');

/*!40000 ALTER TABLE `customers` ENABLE KEYS */;



/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;

/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
```

5. Cold Fusion

A trial ColdFusion may be downloaded from:
http://www.adobe.com/products/coldfusion/

After installing your ColdFusion server, create a new directory under the /wwwroot folder called “service” and add the three files shown below.



Crossdomain.xml

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<!--NOTE: This is a simplified crossdomain policy file for testing and
demonstrating purposes only.  Please ensure you fully understand the implication
of this file before using it in a production environment-->

 <cross-domain-policy>
<allow-access-from domain="*" />
 <site-control permitted-cross-domain-policies="master-only"/>
</cross-domain-policy>
```

Data.cfc

```
<cfcomponent output="false">
  <cffunction name="getArtists" access="remote" returntype="query">
    <cfset var  result="">
      <cfquery datasource="cfartgallery" name="results">
          SELECT * FROM artists ORDER BY lastname, firstname  </cfquery>
    <cfreturn results>
  </cffunction>
</cfcomponent>
```

```
```

Test.cfm

This file is to use to test to make sure the database and CF are configured properly.

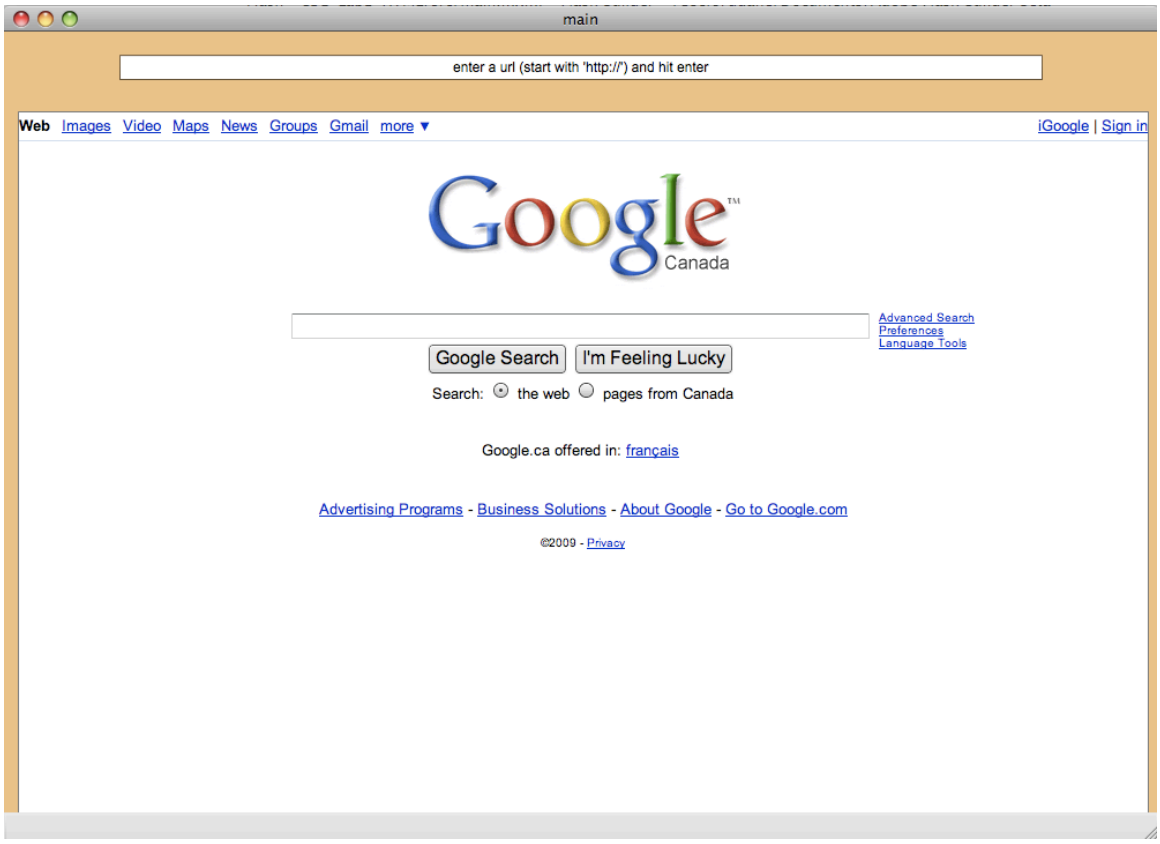
```
<cfobject type="component" component="data" name="dObj"> <cfdump
var="#dObj.getArtists()#">
```

SECTION TWO: THE LABS

Project 1: HTML Client

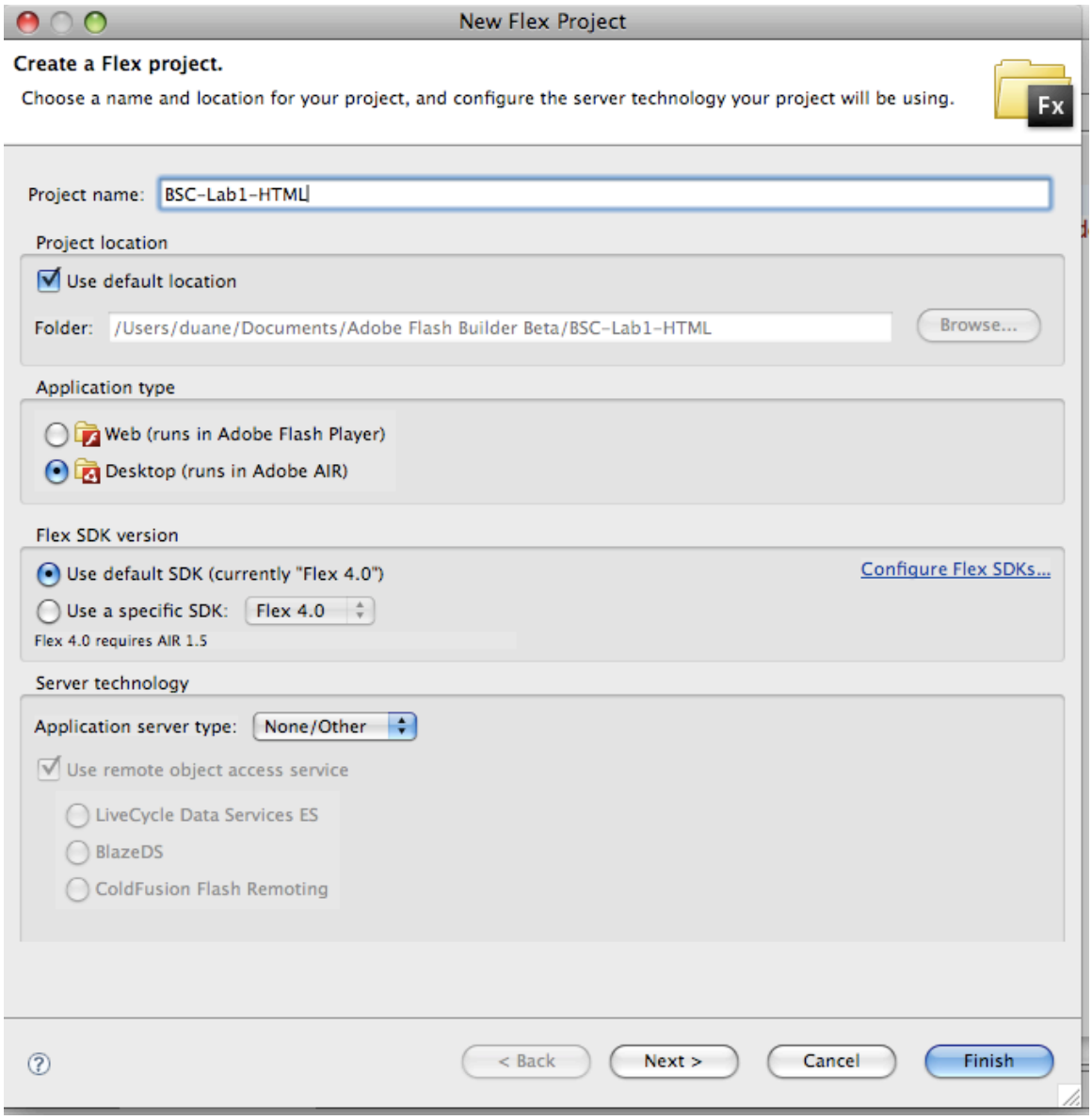
The simplest of all clients is a simple HTML client. Adobe AIR uses the Webkit HTML engine which gives you a powerful set of capabilities including AJAX, CSS and HTML controls.

Demonstrates the WebKit engine including a discussion on how much is available from Webkit in AIR. Attendees will write this project from scratch and learn how to
set the URL, how it handles international characters, CSS and AJAX.



Instructions:

Step 1: Build this from Scratch (new -> Project -> AIR…). This will not work as a Flex application as the HTML component is not part of the Flex framework. Use the settings as shown in the image below:



Step 2:

Add a vertical layout manager to your project by adding the following lines of code:

```
<s:layout>
        <s:BasicLayout/>
</s:layout>
```

Project 1 Solution Code

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/halo"
                       backgroundColor="#eeb470" minHeight="700"
                       minWidth="1000">
      <s:layout>
          <s:BasicLayout/>
      </s:layout>

      <s:TextInput horizontalCenter="true" id="myTI" top="20"
          left="100" right="100" enter="myHTML.location=myTI.text”
          textAlign="center" enabled="true" baseColor="#850909"
          text="enter a url (start with 'http://') and hit enter" />

      <mx:HTML location="http://www.google.com" id="myHTML" left="12"
               right="8" height="614"  y="69" width="972"/>
</s:WindowedApplication>
```

Project 2: REST Style HTTPService

Representational State Transfer refers to a collection of network architecture principles that outline how resources are defined and addressed (as opposed to HTTP, which is strictly limited to one abstract interaction pattern with minor variations).

A REST interface describes any service that transmits domain-specific data over HTTP without an additional messaging layer such as SOAP, or session tracking via HTTP cookies.

REST is worth reading about in further detail as it contains many subtle nuances. Claims have been made (as noted at http://en.wikipedia.org/wiki/Representational_State_Transfer) that it is possible to:

1. Design a service in accordance with the REST architectural style without using HTTP, and

2. That it is possible to design simple XML + HTTP interfaces that do not conform to REST principles
http://en.wikipedia.org/wiki/Representational_State_Transfer

Instructions:

1. Grab a web browser and navigate to the following URL:
   http://www.nickull.net/xml/Wines.xml
   Or optionally (BlazeDS must be started and running) to:
   http://localhost:8400/xml/Wines.xml

2. You should see the following XML:

```
- <Wines>
  - <Vintage>
      <Vintner>Gaja</Vintner>
      <Name>Conteisa</Name>
      <Vintage>2001</Vintage>
      <ParkerNotation>94</ParkerNotation>
      <Price>$324.00</Price>
    </Vintage>
  - <Vintage>
      <Vintner>Gaja</Vintner>
      <Name>Sori Tildin Nebbiolo</Name>
      <Vintage>2005</Vintage>
      <ParkerNotation>N/A</ParkerNotation>
      <Price>$299.00</Price>
    </Vintage>
  - <Vintage>
      <Vintner>Moueix</Vintner>
      <Name>Petrus</Name>
      <Vintage>2001</Vintage>
      <ParkerNotation>97</ParkerNotation>
      <Price>$2,988.00</Price>
    </Vintage>
  + <Vintage></Vintage>
  + <Vintage></Vintage>
  + <Vintage></Vintage>
  + <Vintage></Vintage>
  + <Vintage></Vintage>
  </Wines>
```

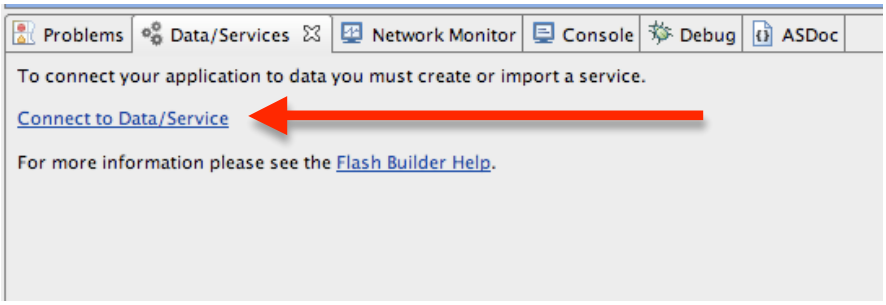
In order for this to work, a small file called crossdomain.xml is in the same directory as the Wines.xml on the server. This file looks like this:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM "http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
  <allow-access-from domain="*" />
  <site-control permitted-cross-domain-policies="master-only"/>
</cross-domain-policy>
```

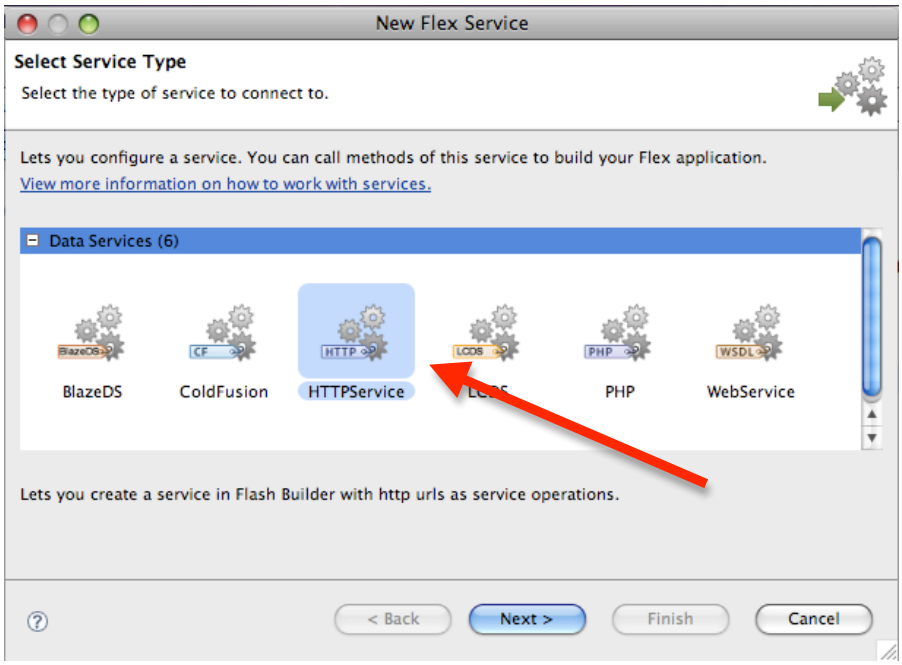
Describing what this file does is beyond the scope of this tutorial however more can be read at http://livedocs.adobe.com/flash/8/main/wwhelp/wwhimpl/common/html/wwhelp.htm?context=LiveDocs_Parts&file=00001621.html

3. With Flash Builder 4, start a new AIR Project called "BSC-Lab2-REST". Make it a web based project (Flex) and select no server type. Click "Finish".

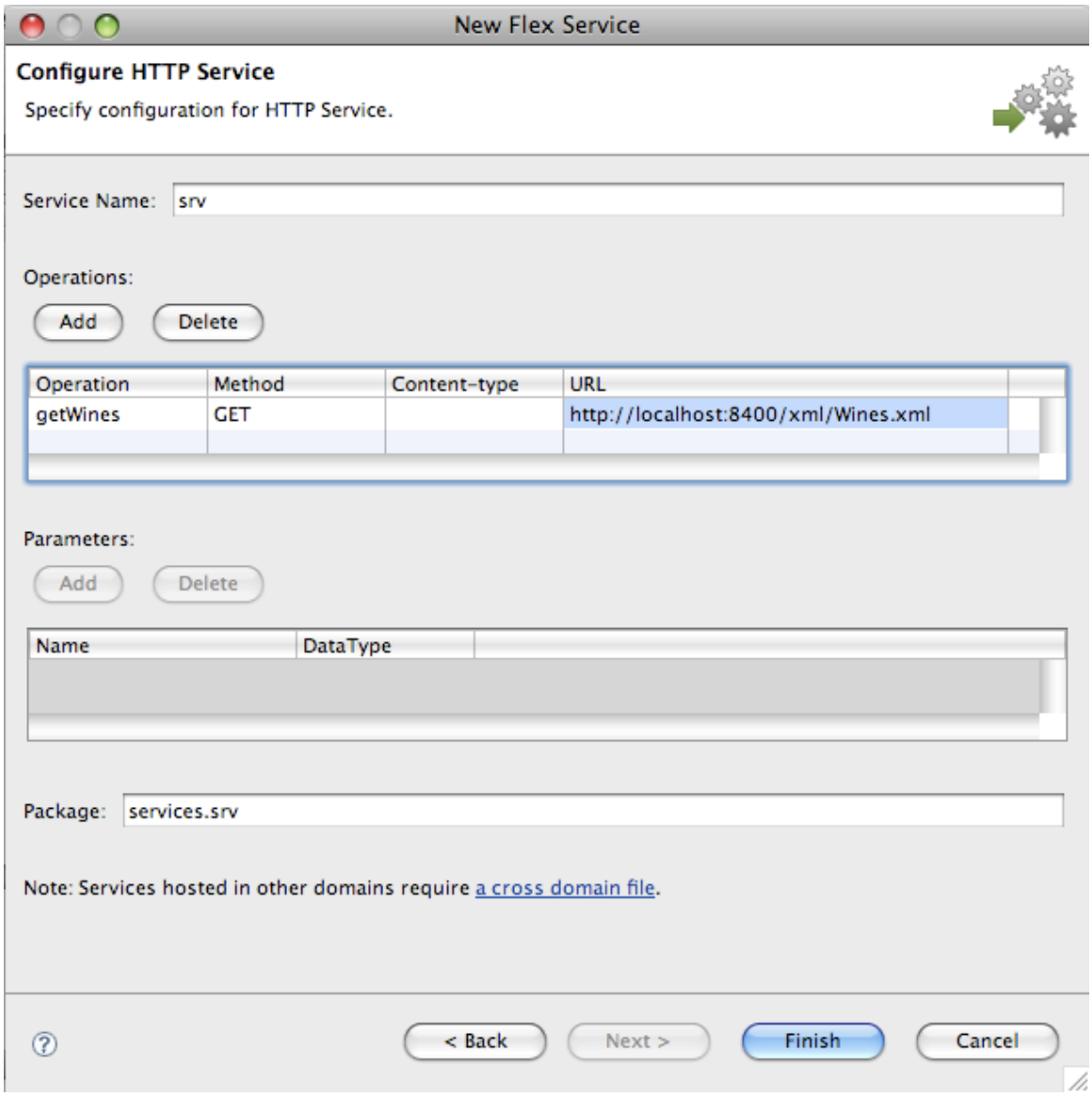
4. At the bottom of the Flash Builder IDE< look for and click on the hyperlink that says "Connect to Data Service" as shown below.



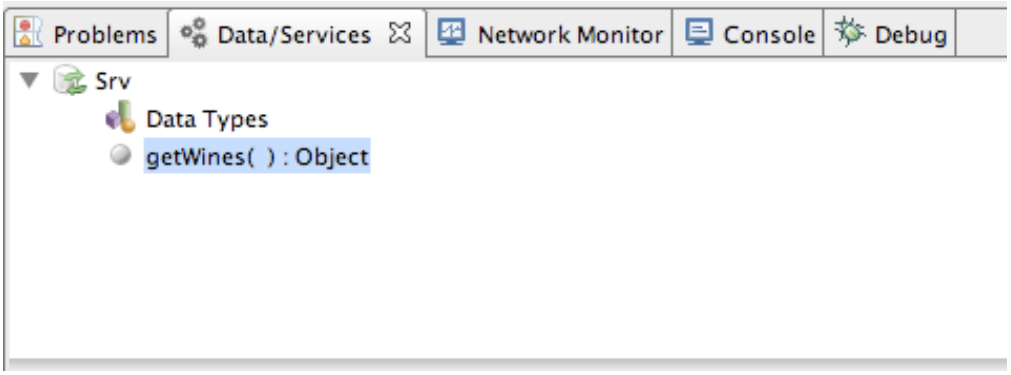
5. This will bring up the Dialog window with a number of choices for services. Select the HTTPService option as shown below:



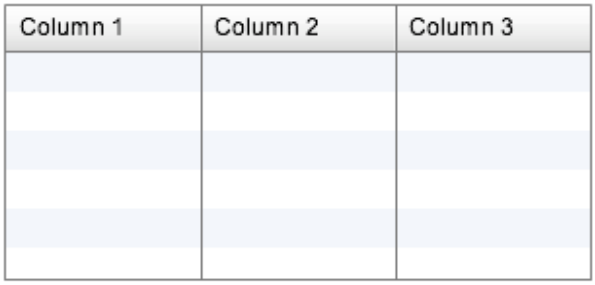
6. Click "Next" and then fill in the next screen by adding the name of the service ("srv" in this case) and add the URL.



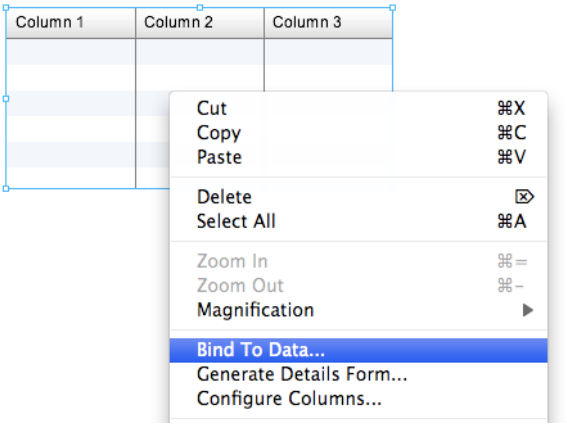
7. After completing this operation by clicking "Finish", you will see the data services in the panel at the bottom.



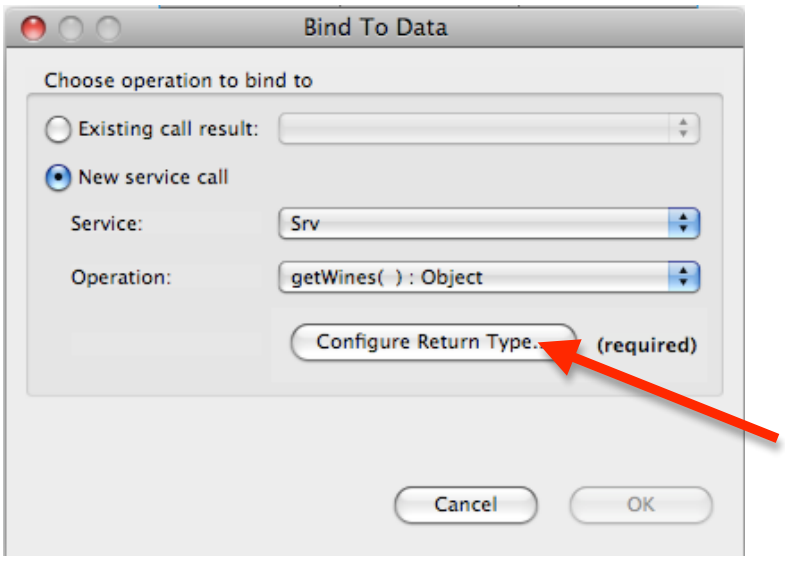
8. Next we need to add an object to bind our data to. Switch to the Design view and drag a <DataGrid> onto the middle of your application as shown below:



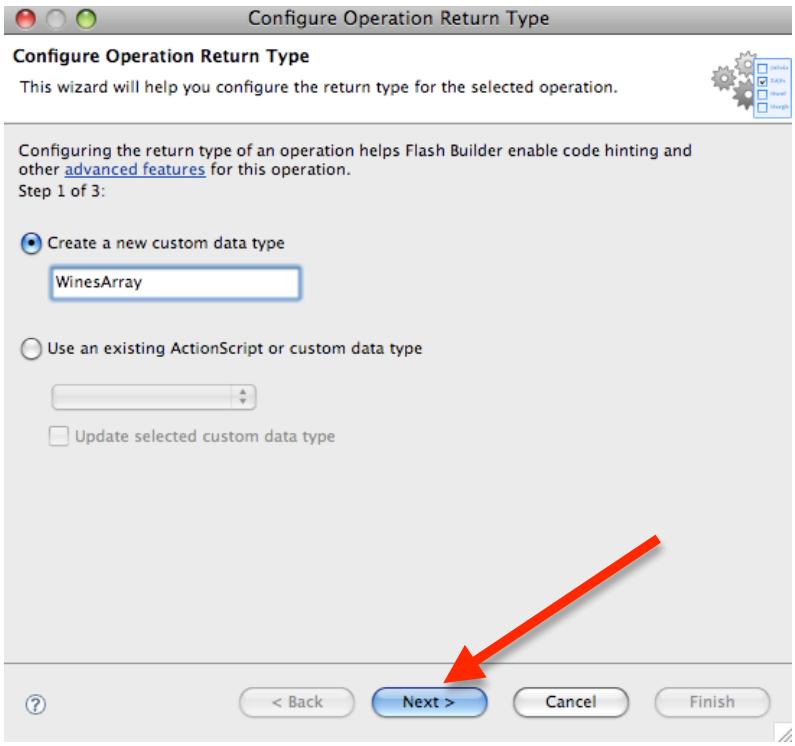
9. Highlight the DataGrid, then right click (Control Click on Mac) on the DataGrid object and select Bind Data from the list in the dialog box as shown below:



10. This will bring up the "Bind to Data" Dialog. Click on "Configure Return Type" to configure the datatype fo the returned object.



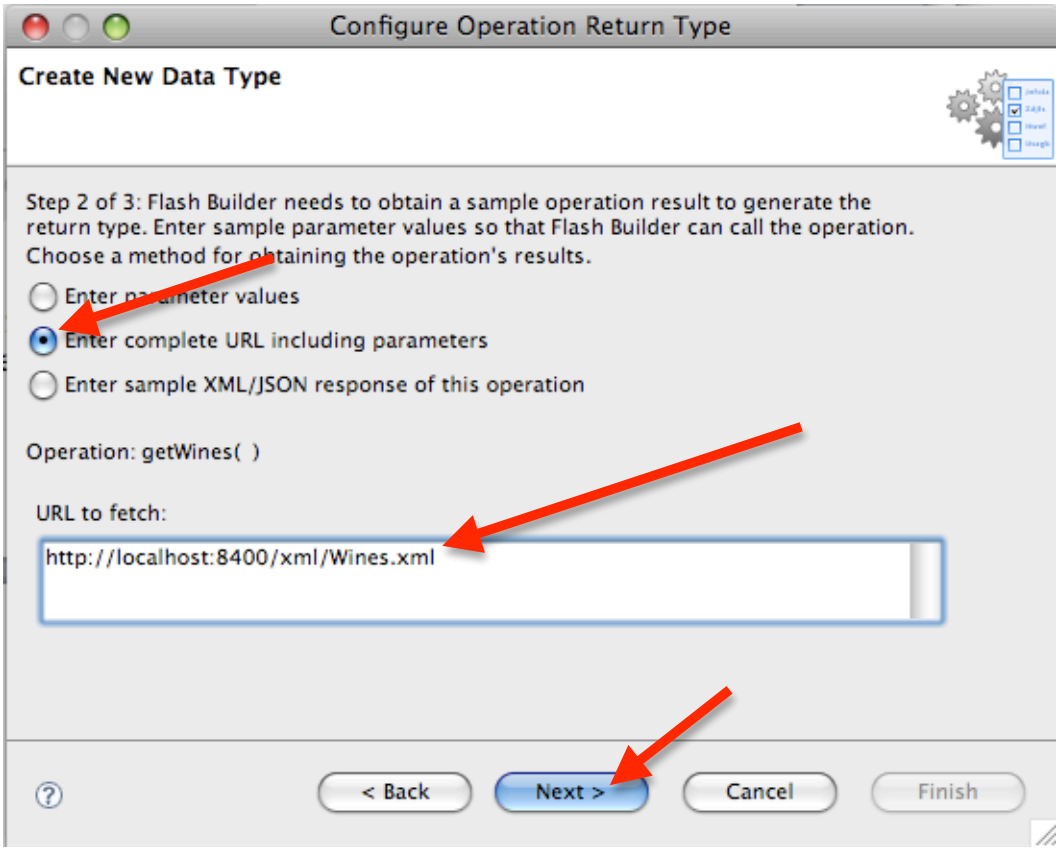
11. The next dialog will prompt you for a datatype. NOTE: on other projects, you might wish to use one of the default datatype however on this one, we are going to create our own object Type called "WinesArray" as shown below.



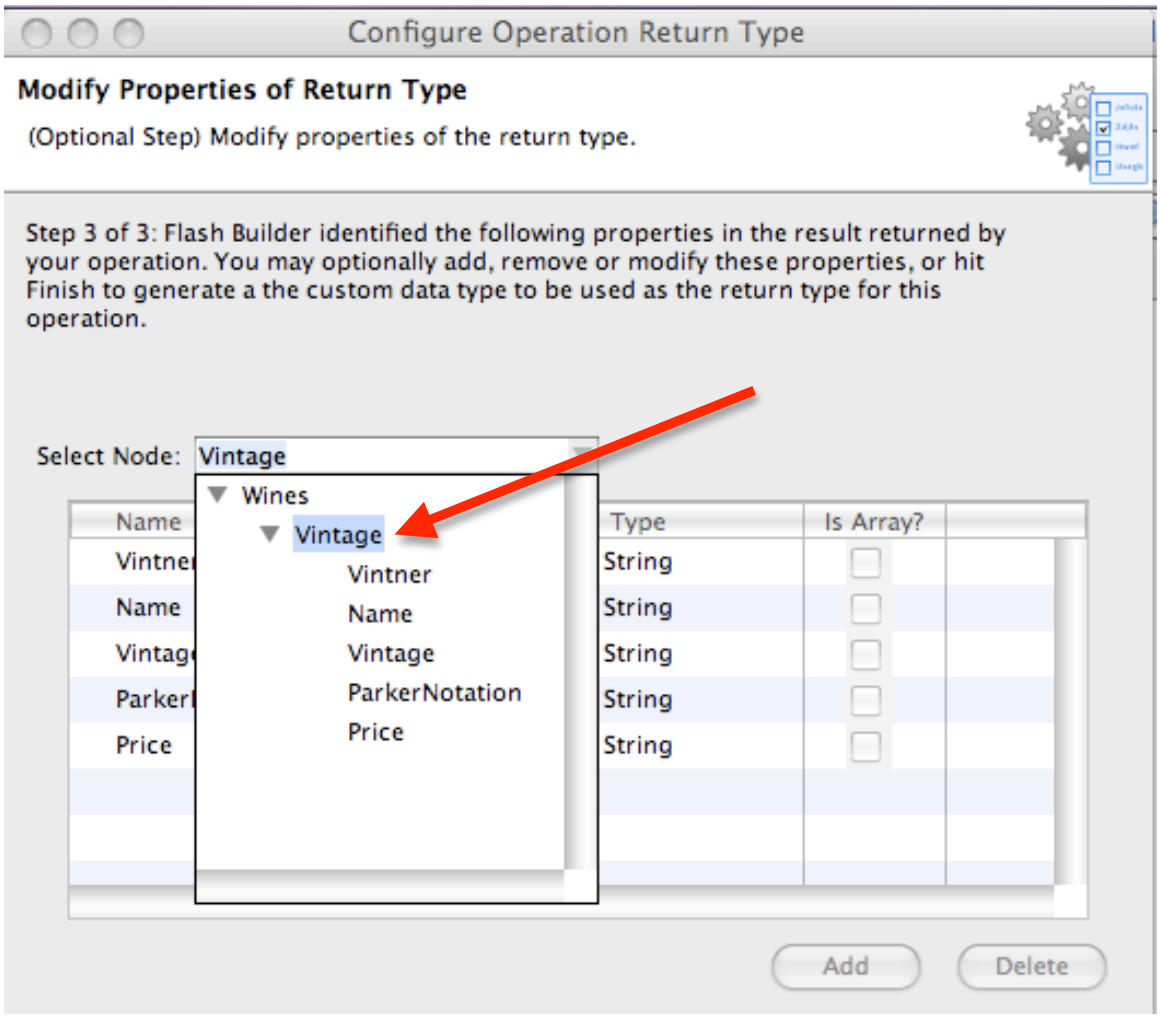
12. Click "Next"

13. The next dialog will invoke the service request to get a sample of the data to inspect. For this lab exercise, select the

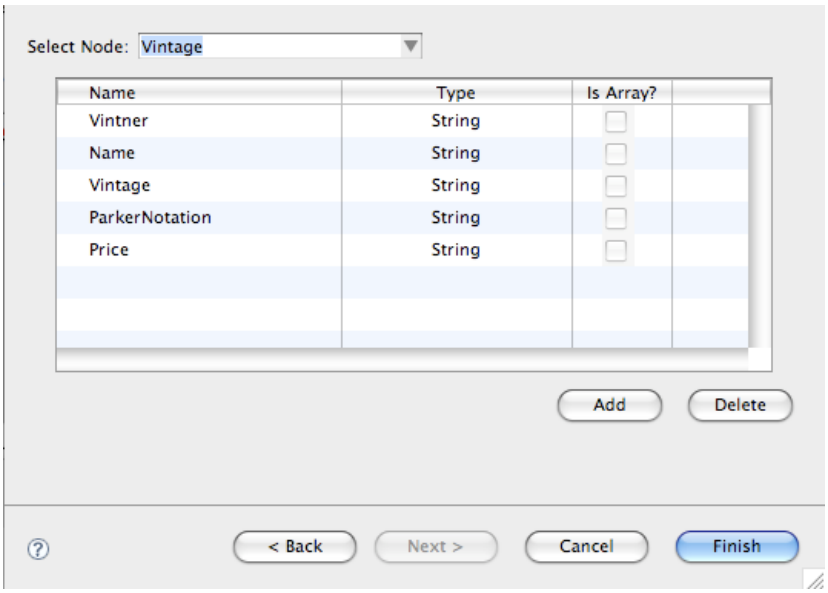
radio button entitled "Enter complete URL …" and enter the URL of the XML file in the "URL to fetch" box below and hit "Next".



14. Flash Builder will go to the URL and bring back a sample of data and break down the structure of it for you. In this case, it has selected that it may be an array. Rather than accept the first option as is, use the drop down list to select a context root of "Vintage" as shown below.



15. Now you will have the following



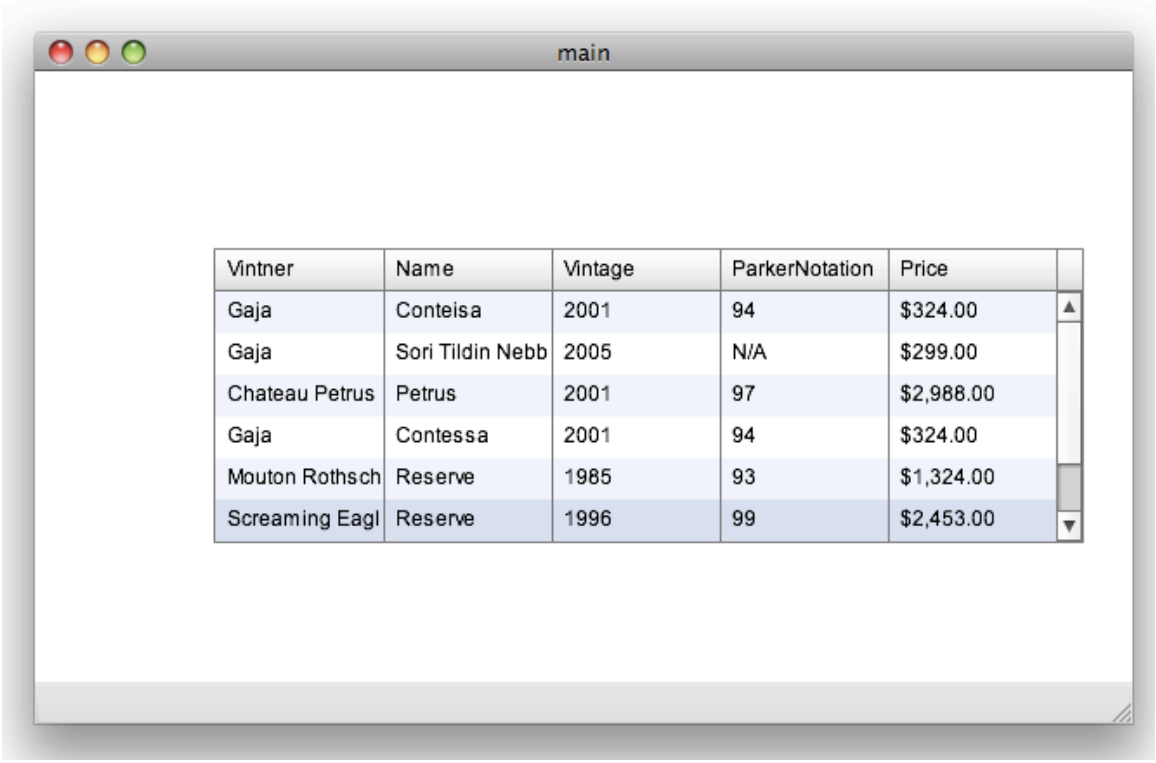
16. Click "Finish"

17. Close the last dialog by clicking "okay"

18. Your project should now reflect the DataGrid being bound to the data as shown below:

Vintner	Name	Vintage	ParkerNotation	Price

19. Run the project and you AIR application will build and display the XML from the server as shown below.



Project 2 Solution Code

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
xmlns:s="library://ns.adobe.com/flex/spark"
xmlns:mx="library://ns.adobe.com/flex/halo" xmlns:srv="services.srv.*">
	<fx:Script>
	<![CDATA[
	import mx.events.FlexEvent;
	import mx.controls.Alert;

	protected function
dataGrid_creationCompleteHandler(event:FlexEvent):void
	{
		getWinesResult.token = srv.getWines();
	}
	]]>
	</fx:Script>
	<fx:Declarations>
		<s:CallResponder id="getWinesResult"/>
```

```
            <srv:Srv id="srv" fault="Alert.show(event.fault.faultString)"
showBusyCursor="true"/>
      </fx:Declarations>
    <mx:DataGrid x="103" y="102" id="dataGrid"
creationComplete="dataGrid_creationCompleteHandler(event)"
dataProvider="{getWinesResult.lastResult}" editable="true">
      <mx:columns>
      <mx:DataGridColumn headerText="Vintner" dataField="Vintner"/>
      <mx:DataGridColumn headerText="Name" dataField="Name"/>
      <mx:DataGridColumn headerText="Vintage" dataField="Vintage"/>
      <mx:DataGridColumn headerText="ParkerNotation"
                         dataField="ParkerNotation"/>
      <mx:DataGridColumn headerText="Price" dataField="Price"/>
      </mx:columns>
   </mx:DataGrid>

</s:WindowedApplication>
```

Project 3: Web Service Introspection and Consumption

Background Information

Flex applications can interact with web services that define their interfaces in a Web Services Description Language 1.1 (WSDL 1.1) document, which is available as a URL. WSDL is a standard format for describing the messages that a web service understands, the format of its responses to those messages, the protocols that the web service supports, and where to send messages. The Flex web service API generally supports Simple Object Access Protocol (SOAP) 1.1, XML Schema 1.0 (versions 1999, 2000, and 2001), and WSDL 1.1 RPC-encoded, RPC-literal, and document-literal (bare and wrapped style parameters). The two most common types of web services use remote procedure call (RPC) encoded or document-literal SOAP bindings; the terms encoded and literal indicate the type of WSDL-to-SOAP mapping that a service uses.

Flex applications support web service requests and results that are formatted as SOAP messages. SOAP provides the definition of the XML-based format that you can use for exchanging structured and

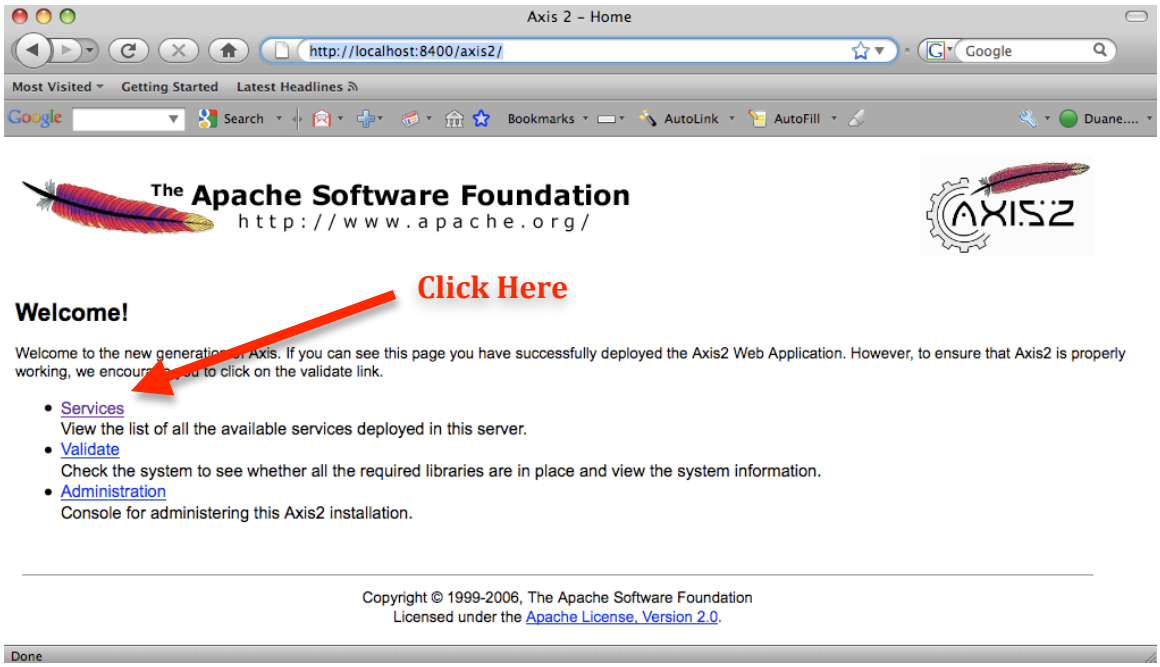
typed information between a web service client, such as a Flex application, and a web service.

Adobe® Flash® Player operates within a security sandbox that limits what Flex applications and other applications built with Flash can access over HTTP. Applications built with Flash are allowed HTTP access only to resources on the same domain and by the same protocol from which they were served. This presents a problem for web services, because they are typically accessed from remote locations. The proxy service, available in LiveCycle Data Services ES and Vega, intercepts requests to remote web services, redirects the requests, and then returns the responses to the client.

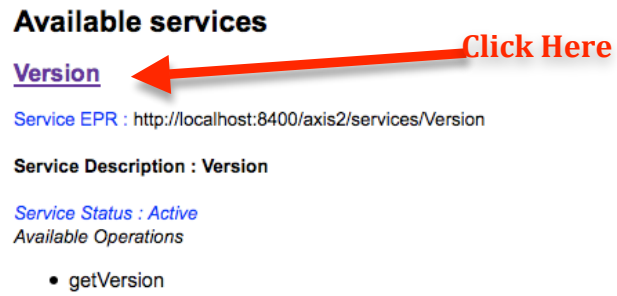
If you are not using LiveCycle Data Services ES or Vega, you can access web services in the same domain as your Flex application; or a crossdomain.xml (cross-domain policy) file that allows access from your application's domain must be installed on the web server hosting the RPC service.

The Lab

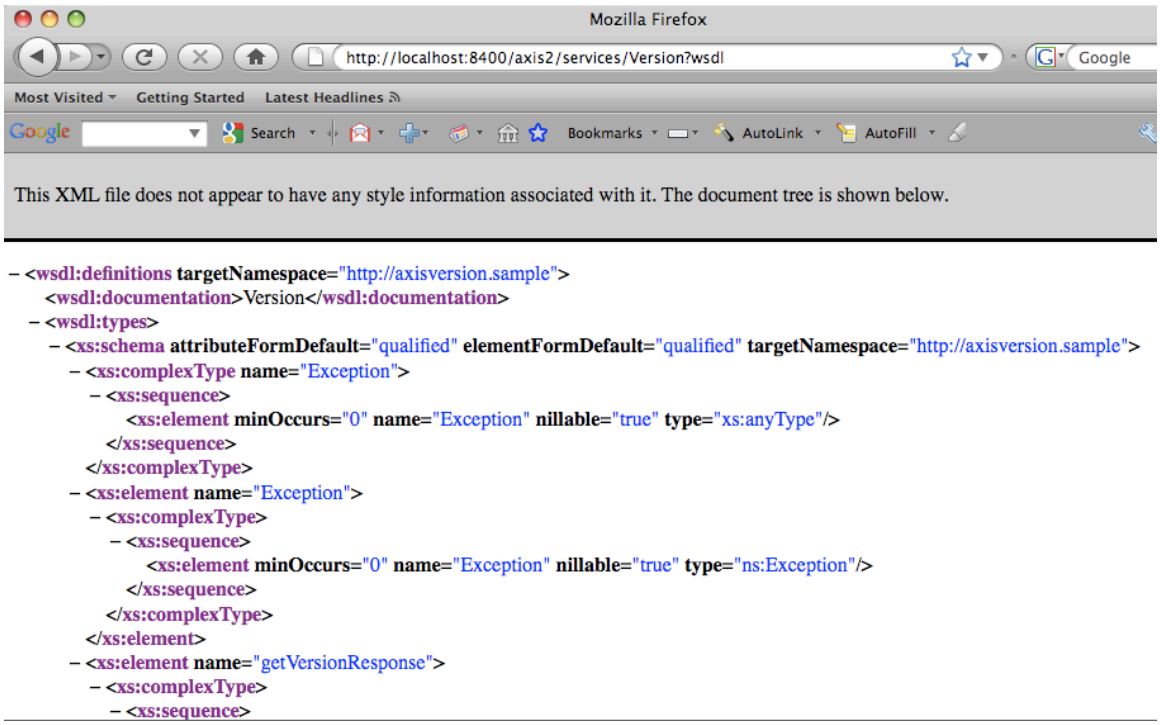
We are going to use a WSDL, import it and wire it up to make a remote call. This will demonstrate the WSDL wizard that is based on Apache Axis 2. To run this lab you will need to ensure that you have the special version of BLazeDS up and running for this lab. To try this, grab a browser and hit the URL http://localhost:8400/axis2/ You should see a screen like below:



Click on the "Services" hyperlink and you should see the getVersion() operation which returns the version of Apache Axis being run on the server.



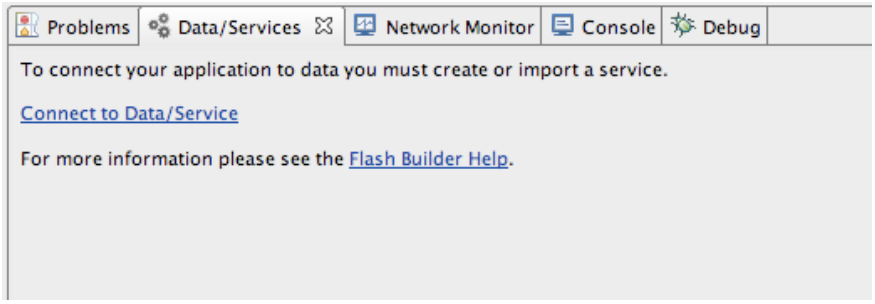
Click on the "Version" hyperlink and you should see the WSDL (Web Services Description Document) as seen below.



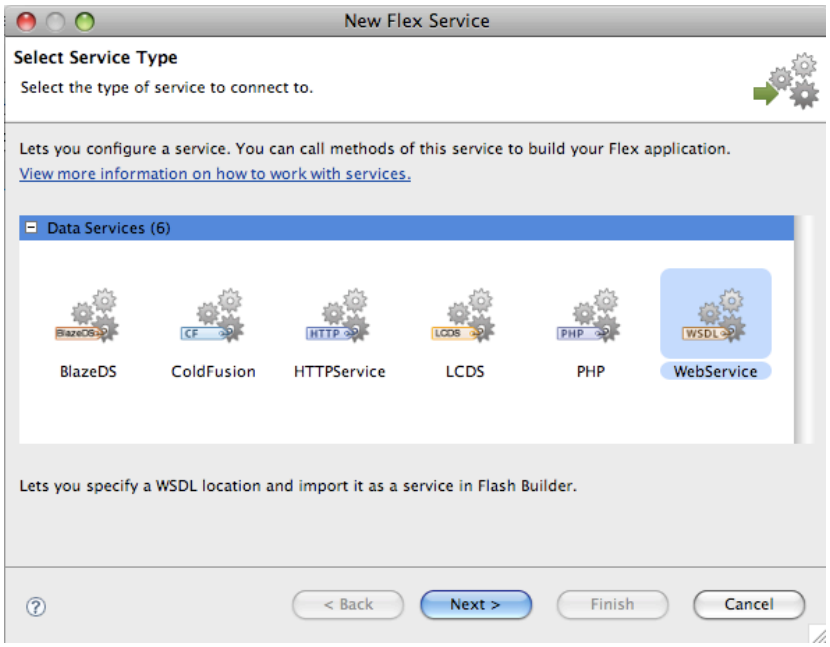
1. Make new project and name is BSC-Lab5_WSDL.

2. Make sure your local BlazeDS is still running.

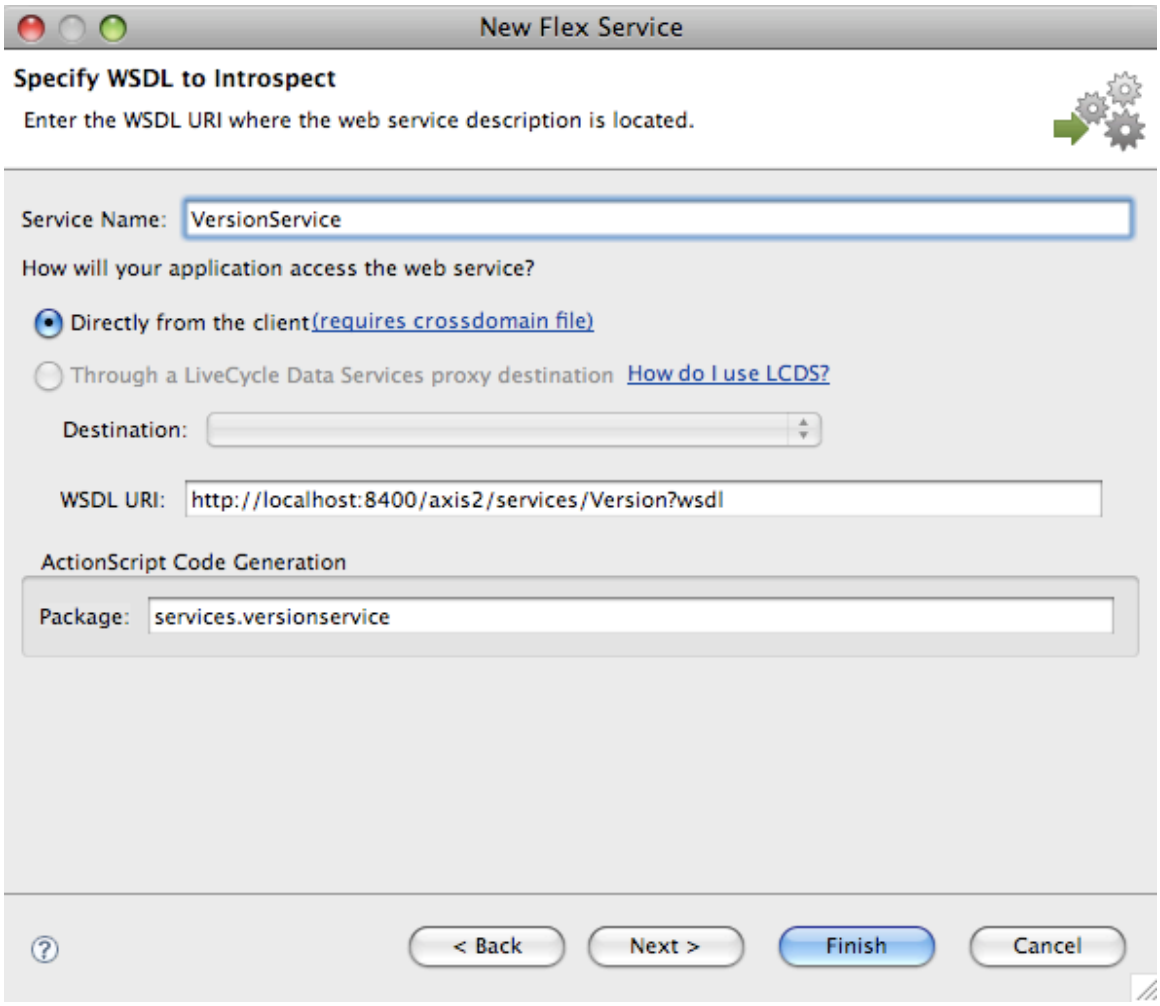
3. Select ***Data/Services*** > ***Connect to Data/Service*** from the bottom menu in Flash Builder 4. This will open a dialog (wizard).



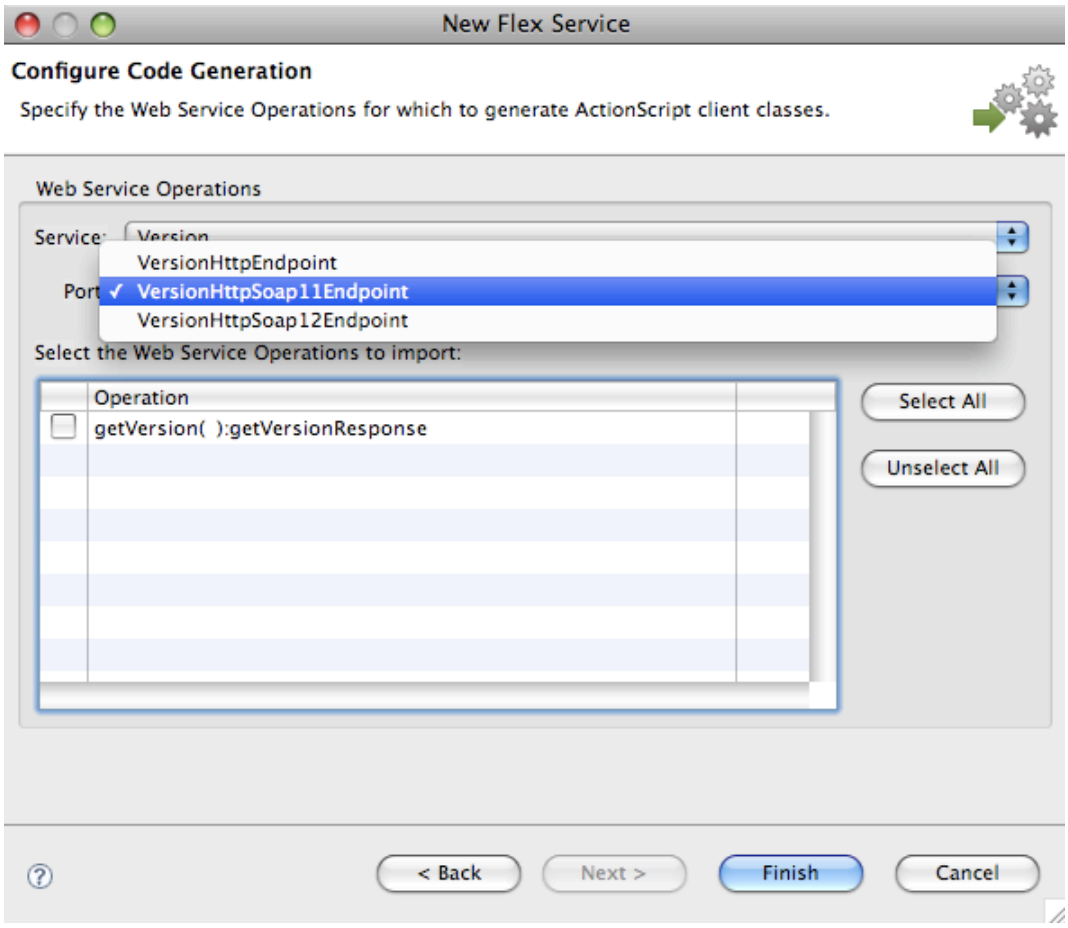
4. In the next dialog box, give your service a name and also enter the URL of the WSDL so Flash Builder 4 can inspect it as shown below. Hit Next.



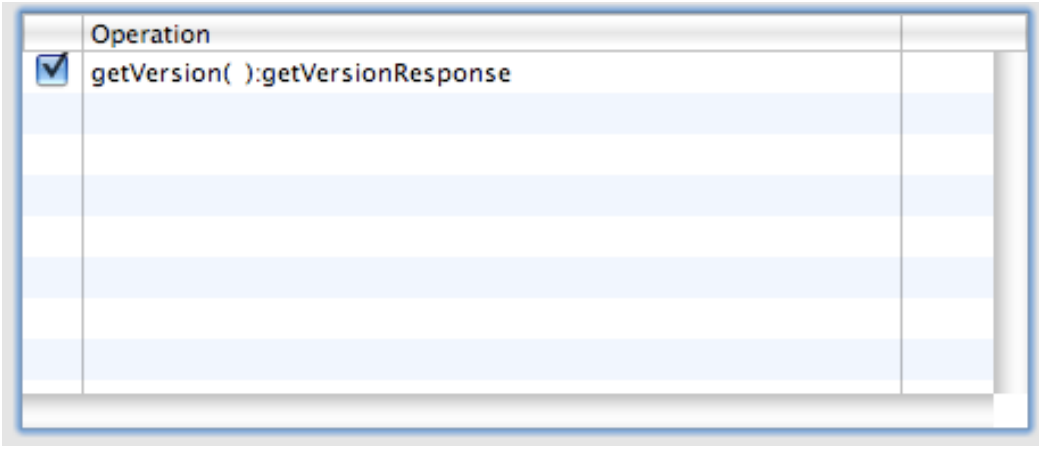
5. Cut and paste the URL of the WSDL into the "WSDL URI" box ass shown below. In the "Service Name" field, add a name for your service. In this instance I used "VersionService". Do not hit "Finish", hit "Next".



6. The next dialog will appear after Flash Builder has inspected the WSDL and will offer you a choice of ports. Flash Builder 4 will work with a SOAP 1.1. endpoint so select that option from the list as shown below.

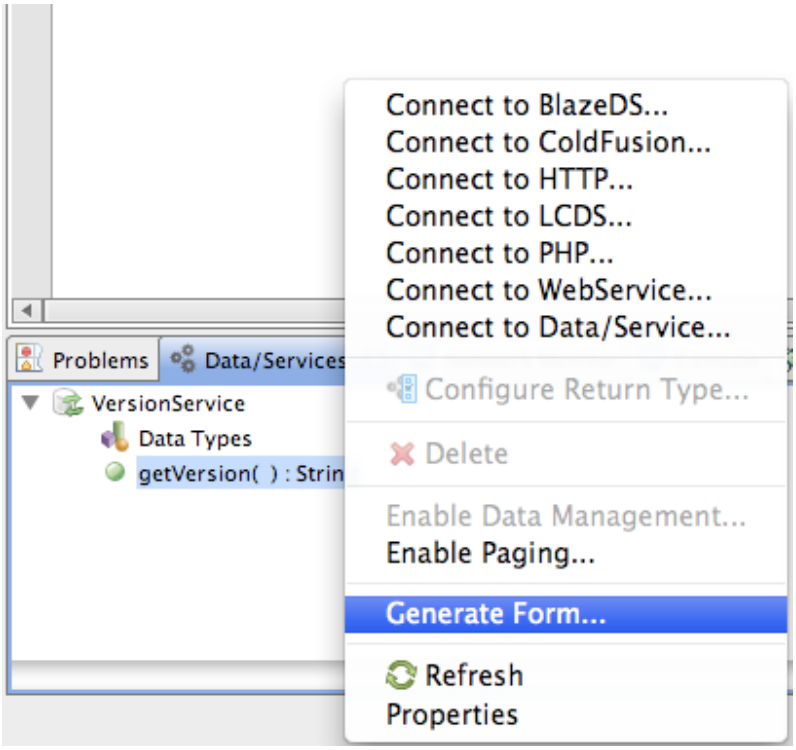


7. The getVersion() operation will now be available in the list below. Select the check box beside it and hit "Finish"

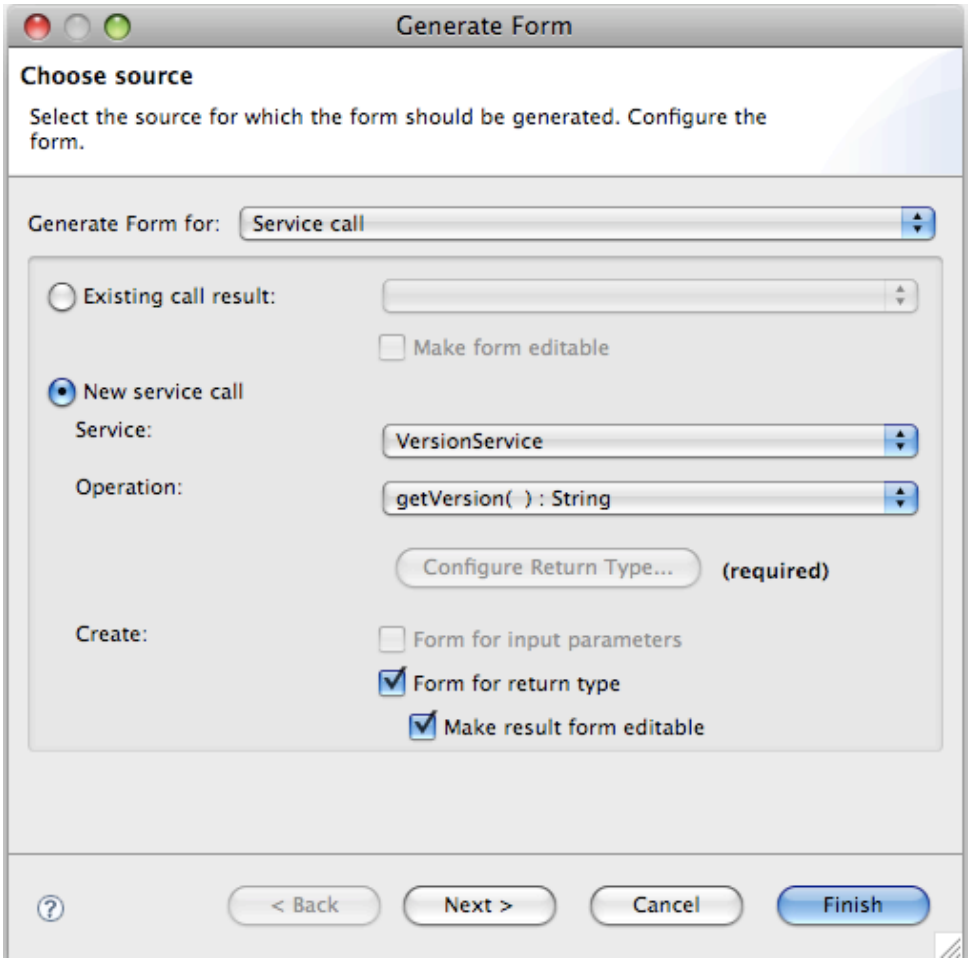


8. You will now see a dialog asking you to configure the return type. Since the return type is "string" we do not need to do anything else. Close the dialog with no action.

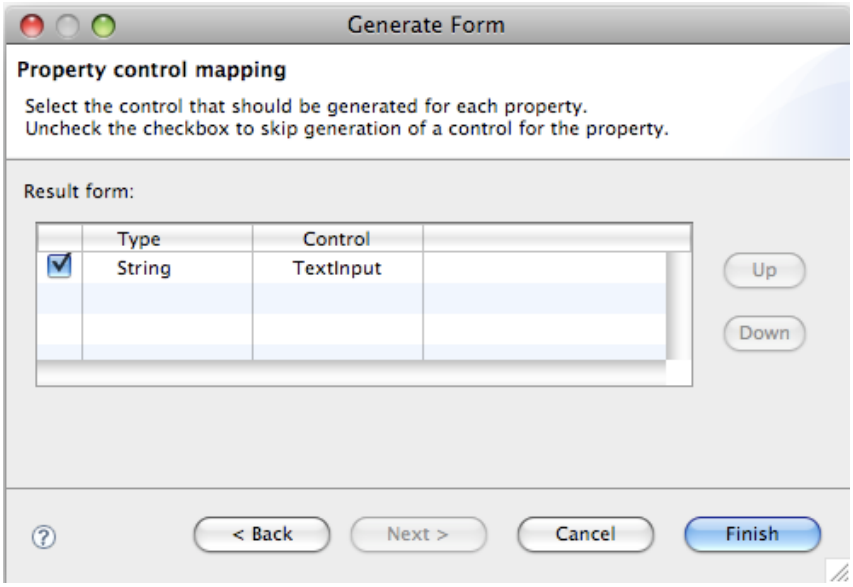
9. In the Data/Services tab at the bottom, right click (or Control Click for Mac) on the getVersion operation and select "Generate Form" as shown below:



10. A dialog box will pop up giving you some options. The options will allow you to select operation parameters and settings. Leave the defaults as shown below:



11. Hit "Next" and the next dialog allows you to control where each returned object maps to in terms of form components. In this case, we are simply allowing the String (the value returned from the call) to be boubnd to a TextInput. Select "Finish".



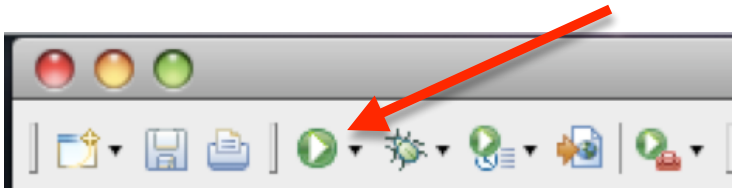
12. The last step is to perform a minor tweak to the textInput to ensure it is big enough to hold the string value. Switch to source code view and change this line:

```
<mx:FormItem label="GetVersion">
       <s:TextInput id="getVersionTextInput"
                    text="{getVersionResult.lastResult as String}"/>
</mx:FormItem>
```

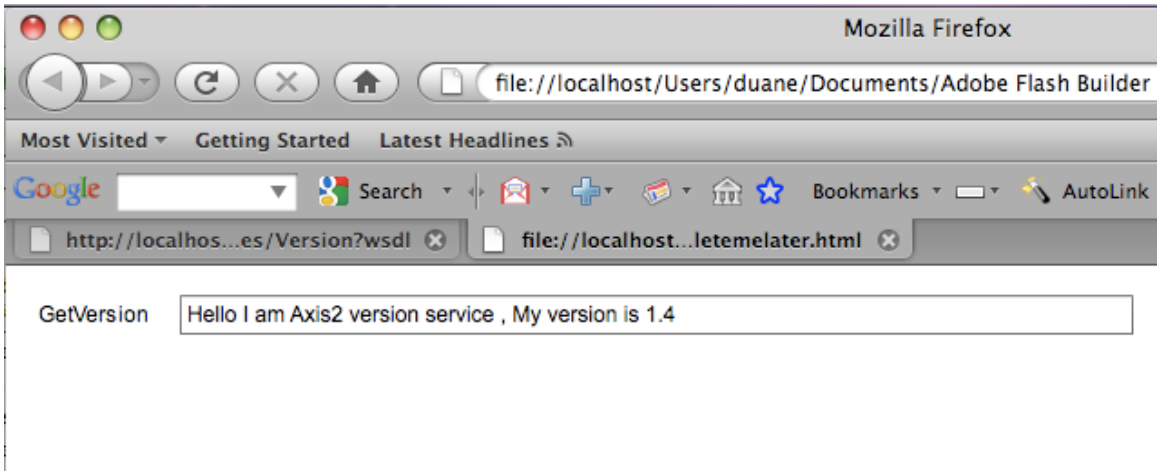
13. To add the width attributes as shown below:

```
<mx:Form id="form"
creationComplete="form_creationCompleteHandler(event)" width="651">
  <mx:FormItem label="GetVersion" width="607">
        <s:TextInput id="getVersionTextInput"
        text="{getVersionResult.lastResult as String}" width="526"/>
  </mx:FormItem>
</mx:Form>
```

14: Now run the program by clicking the green arrow icon.



15: You should see the web service result in your application.



Project 3 Solution Code:

```
<?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/halo" minWidth="1024"
minHeight="768" xmlns:versionservice="services.versionservice.*">
	<fx:Script>
	<![CDATA[
	import mx.events.FlexEvent;
	import mx.controls.Alert;

	protected function
		form_creationCompleteHandler(event:FlexEvent):void
		{
		getVersionResult.token = versionService.getVersion();
		}
	]]>
	</fx:Script>
	<fx:Declarations>
		<versionservice:VersionService id="versionService"
fault="Alert.show(event.fault.faultString)" showBusyCursor="true"/>
		<s:CallResponder id="getVersionResult"/>
```

```
      </fx:Declarations>
      <mx:Form id="form"
creationComplete="form_creationCompleteHandler(event)" width="651">
      <mx:FormItem label="GetVersion" width="607">
            <s:TextInput id="getVersionTextInput"
text="{getVersionResult.lastResult as String}" width="526"/>
      </mx:FormItem>
      </mx:Form>
</s:Application>
```

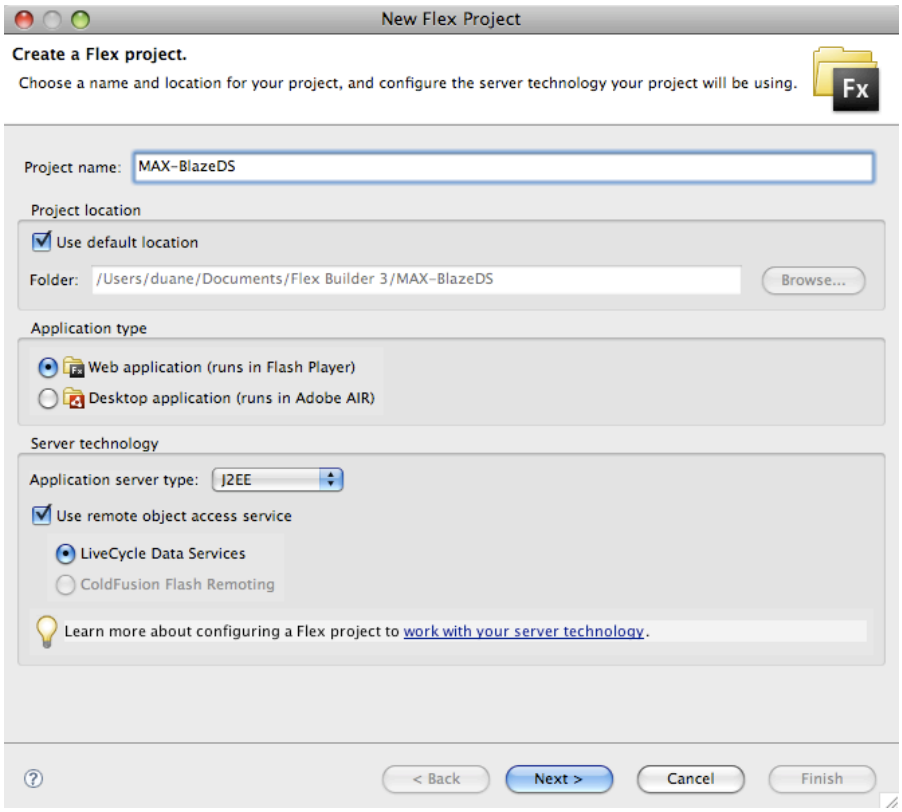
Project 4: Flex Remoting Project

Note: Make sure BlazeDS is still up and running for this lab. This can be done by checking http://localhost:8400/samples/

1. Start a new project and name it MAX-BlazeDS. Make this a Flex application and select the box for server technologies as shown below.

   Application Server Type -> J2EE
   Use Remote Object Access Service -> Checked
   LiveCycle Data Services -> Checked

   Click "Next"



2. As shown below, on the next screen you will need to validate your server configuration. You will need to fill in the proper values for Root Folder, Root URL and Context Root as shown. Use the "Browse" button to locate the

   `<BlazeDS_Root>\BlazeDS\tomcat\webapps\samples`

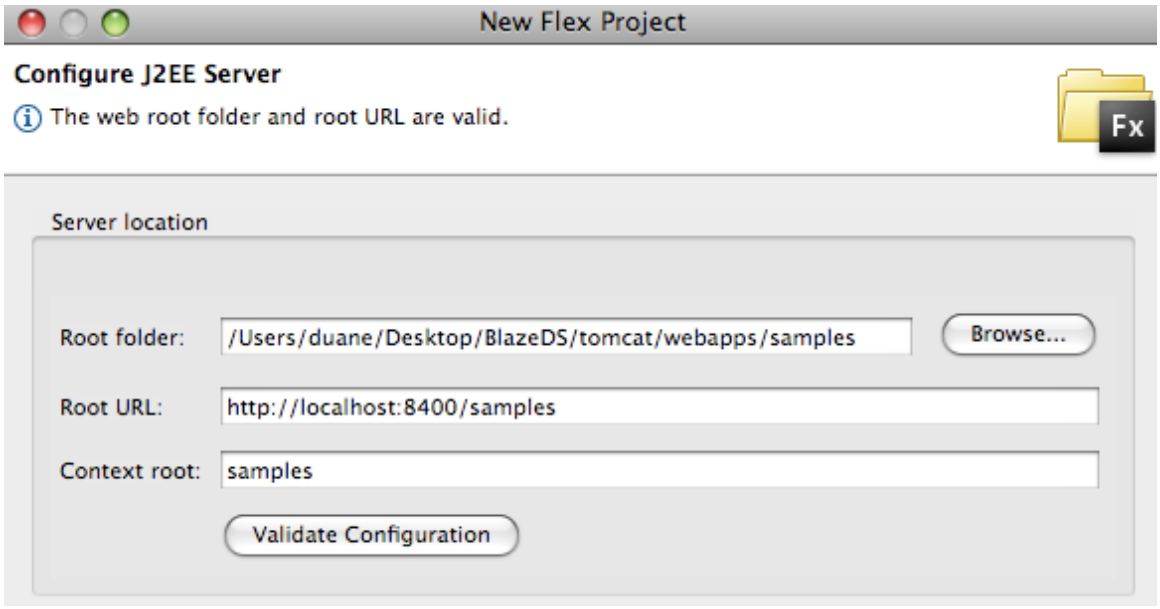
   directory.

   Set the other values as shown:

   Root URL -> http://localhost:8400/samples
   Context Root -> /samples

   The output folder should be filled out by default.

   NOTE: you must have permissions to write to this folder for the lab to work.

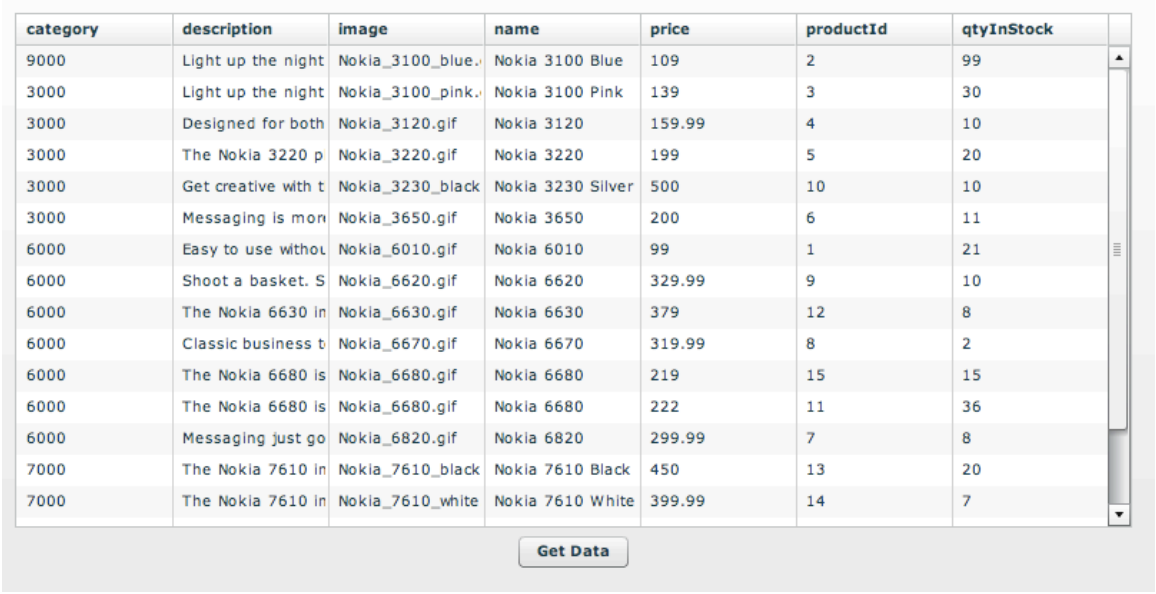


3. Click Finished and you will have a new project done.

4. In your Flex project, enter the following code:

```
<?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/halo"
minWidth="1024" minHeight="768">
      <s:layout>
          <s:BasicLayout/>
      </s:layout>
      <fx:Declarations>
            <mx:RemoteObject id="srv" destination="product"/>
      </fx:Declarations>
      <mx:DataGrid dataProvider="{srv.getProducts.lastResult}"
width="836" height="362" y="6" x="5"/>
      <s:Button label="Get Data" click="srv.getProducts()"  x="349"
y="423"/>

</s:Application>
```

5. Click "run" and your project should be able to connect to the locahost:8400 remote object and return the screen as follows:



category	description	image	name	price	productId	qtyInStock
9000	Light up the night	Nokia_3100_blue.·	Nokia 3100 Blue	109	2	99
3000	Light up the night	Nokia_3100_pink.·	Nokia 3100 Pink	139	3	30
3000	Designed for both	Nokia_3120.gif	Nokia 3120	159.99	4	10
3000	The Nokia 3220 p	Nokia_3220.gif	Nokia 3220	199	5	20
3000	Get creative with t	Nokia_3230_black	Nokia 3230 Silver	500	10	10
3000	Messaging is mor	Nokia_3650.gif	Nokia 3650	200	6	11
6000	Easy to use withou	Nokia_6010.gif	Nokia 6010	99	1	21
6000	Shoot a basket. S	Nokia_6620.gif	Nokia 6620	329.99	9	10
6000	The Nokia 6630 in	Nokia_6630.gif	Nokia 6630	379	12	8
6000	Classic business t	Nokia_6670.gif	Nokia 6670	319.99	8	2
6000	The Nokia 6680 is	Nokia_6680.gif	Nokia 6680	219	15	15
6000	The Nokia 6680 is	Nokia_6680.gif	Nokia 6680	222	11	36
6000	Messaging just go	Nokia_6820.gif	Nokia 6820	299.99	7	8
7000	The Nokia 7610 in	Nokia_7610_black	Nokia 7610 Black	450	13	20
7000	The Nokia 7610 in	Nokia_7610_white	Nokia 7610 White	399.99	14	7

Get Data

6. Now that we have done this, let's play with the server side code a bit. Navigate (use whatever software is typical for your OS) to the

   <BlazeDS_Root>/tomcat/webapps/samples/WEB-INF/src/flex/samples/products/

   directory and open up the file ProductService.java.

7. In the previous Flex sample, we wrote the following line of code:

```
<s:Button label="Get Data" click="srv.getProducts()"  x="349" y="423"/>
```

8. The srv.getProducts() corresponds to the Java method of the same name. These are case sensitive. Note line 13 of the java source says:

```
public List getProducts() throws DAOException {..}
```

9. If you have configured a Java SDK (JDK) on your system and have the JAVA_HOME environmental variable set properly, you will be able to modify the java code. First, in the file

   <BlazeDS_Root>/blazeds/tomcat/webapps/samples/WEB-INF/src/flex/samples/product/Product.java

   Modify the line of code that reads:

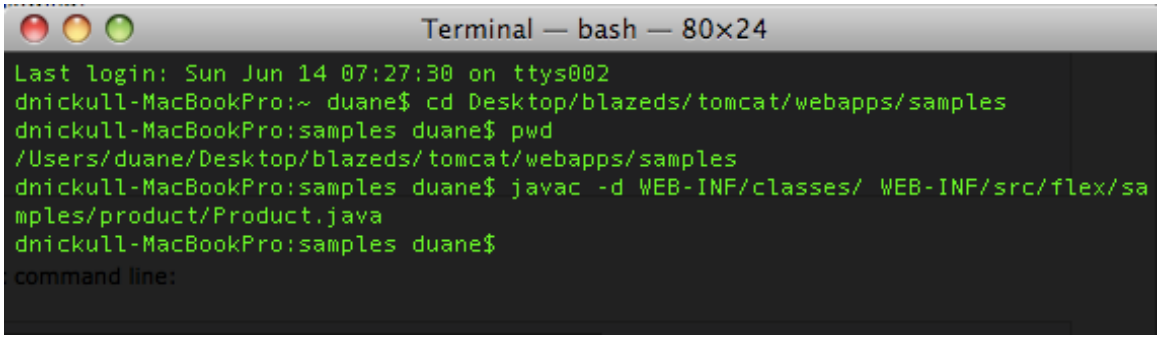
```
public String getDescription() {
    return description;
}
```

   So that it reads

```
public String getDescription() {
    return "BlazeDS Vancouver Build Rocks!";
}
```

   And save the file

10. Grab a terminal (Command Window) and change directory to **<BLazeDS_Root>/blazeds/tomcat/webapps/samples**

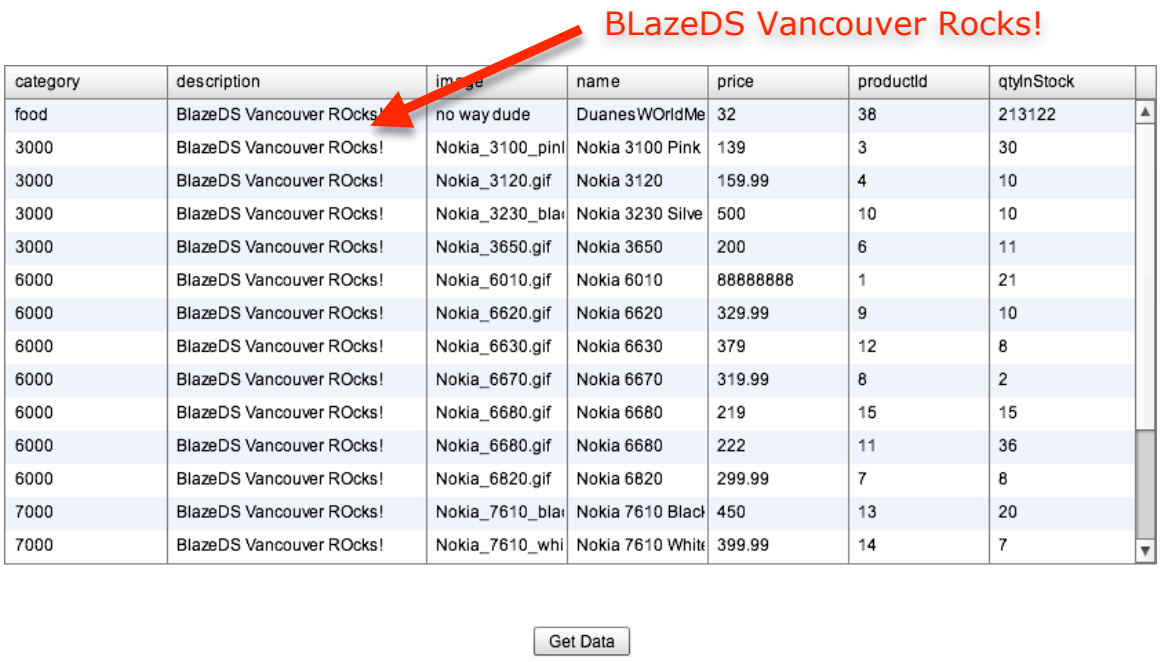


11. Recompile the Product.java class and place the resulting class file in the destination folder by typing in:

```
javac -d WEB-INF/classes/ WEB-
INF/src/flex/samples/product/Product.java
```

This command creates the file Product.class, and deploys it to the WEB-INF\classes\flex\samples\product directory.

Re-run the sample and you will see a different result in the column for products. Note it might take a few seconds for the server to pick up the new class.



Project 4 Solution Code

```
<?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/halo"
minWidth="1024" minHeight="768">

      <fx:Declarations>
            <mx:RemoteObject id="srv" destination="product"/>
      </fx:Declarations>
      <mx:DataGrid dataProvider="{srv.getProducts.lastResult}"
width="836" height="362"/>
      <s:Button label="Get Data" click="srv.getProducts()"  x="384"
```

```
y="407"/>

</s:Application>
```

Project 5: Working with PHP and Data Paging

Flash Builder 4 lets you enable data paging with just few clicks for any server technology.It can be enabled for any service type (Including HTTP Service).

In this sample, we will enable data paging for a PHP class. For data paging to work you just need to implement 2 functions on the server and leave it to the generated Flex code by Flash Builder to invoke these functions when data has to be fetched, for example when user scrolls the DataGrid scroll bar.

Here is a snippet of the PHP code:

```
<?php


//The full file is at the end of this tutorial in Appendix A

public function getItems_paged($startIndex, $numItems) {

                        $this->connect();
                        $startIndex = mysqli_real_escape_string($this->connection,
$startIndex);
                        $numItems = mysqli_real_escape_string($this->connection,
$numItems);
                        $sql = "SELECT * FROM customers LIMIT $startIndex,
$numItems";

                        $result = mysqli_query($this->connection, $sql) or die('Query
failed: ' . mysqli_error($this->connection));

                        $rows = array();
            while ($row = mysqli_fetch_object($result)) {
```

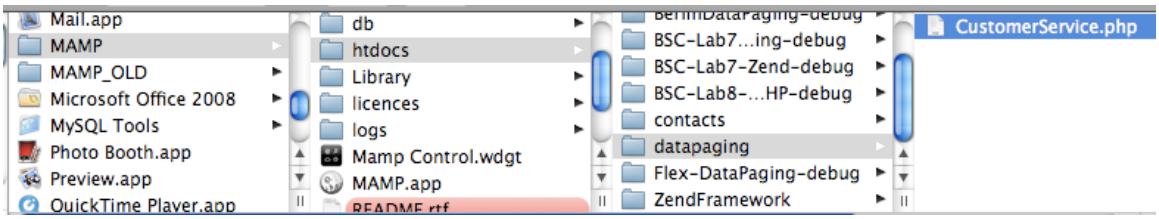
```
                        $rows[] = $row;
            }

            mysqli_free_result($result);
                        mysqli_close($this->connection);

               return $rows;

        }

}

?>
```

Two functions/methods/operations to implement on server

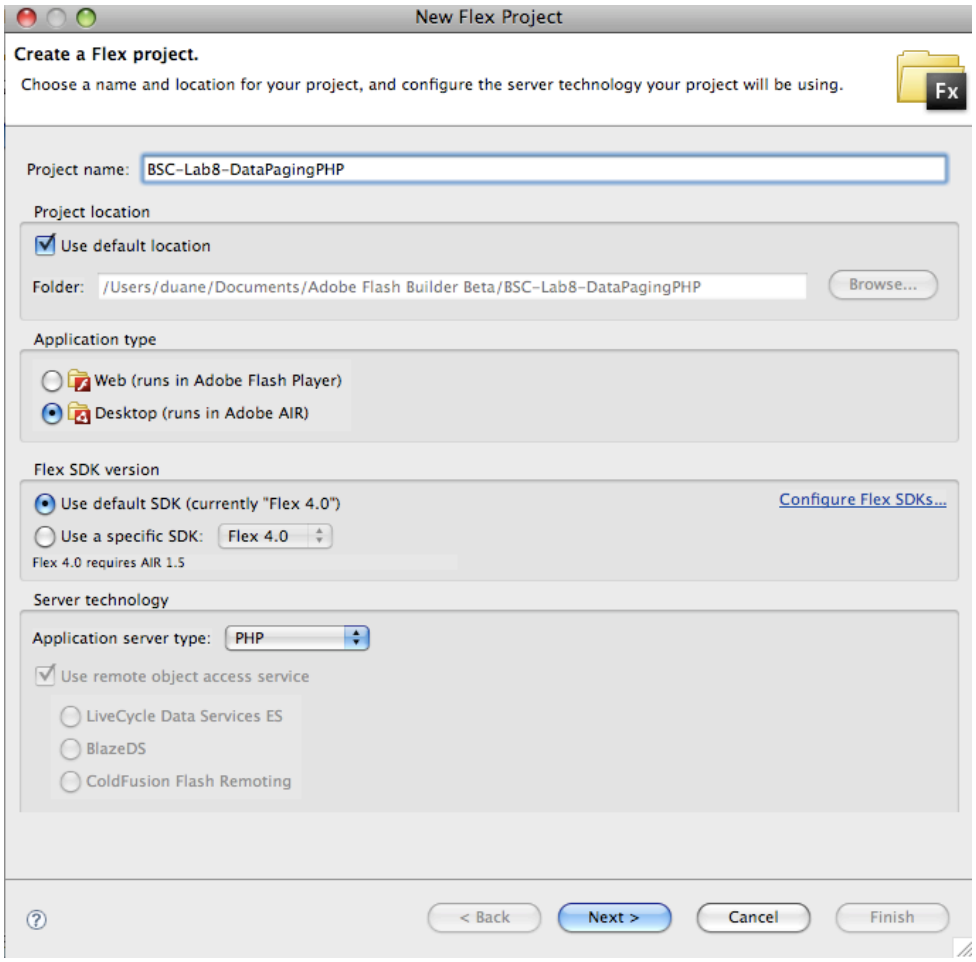
1. Returns number of rows in the database for example "count()"

2. Takes two input arguments and returns the data collection. The two input arguments are start index and number of items to return for example "getItems_paged($startIndex, $numItems)"

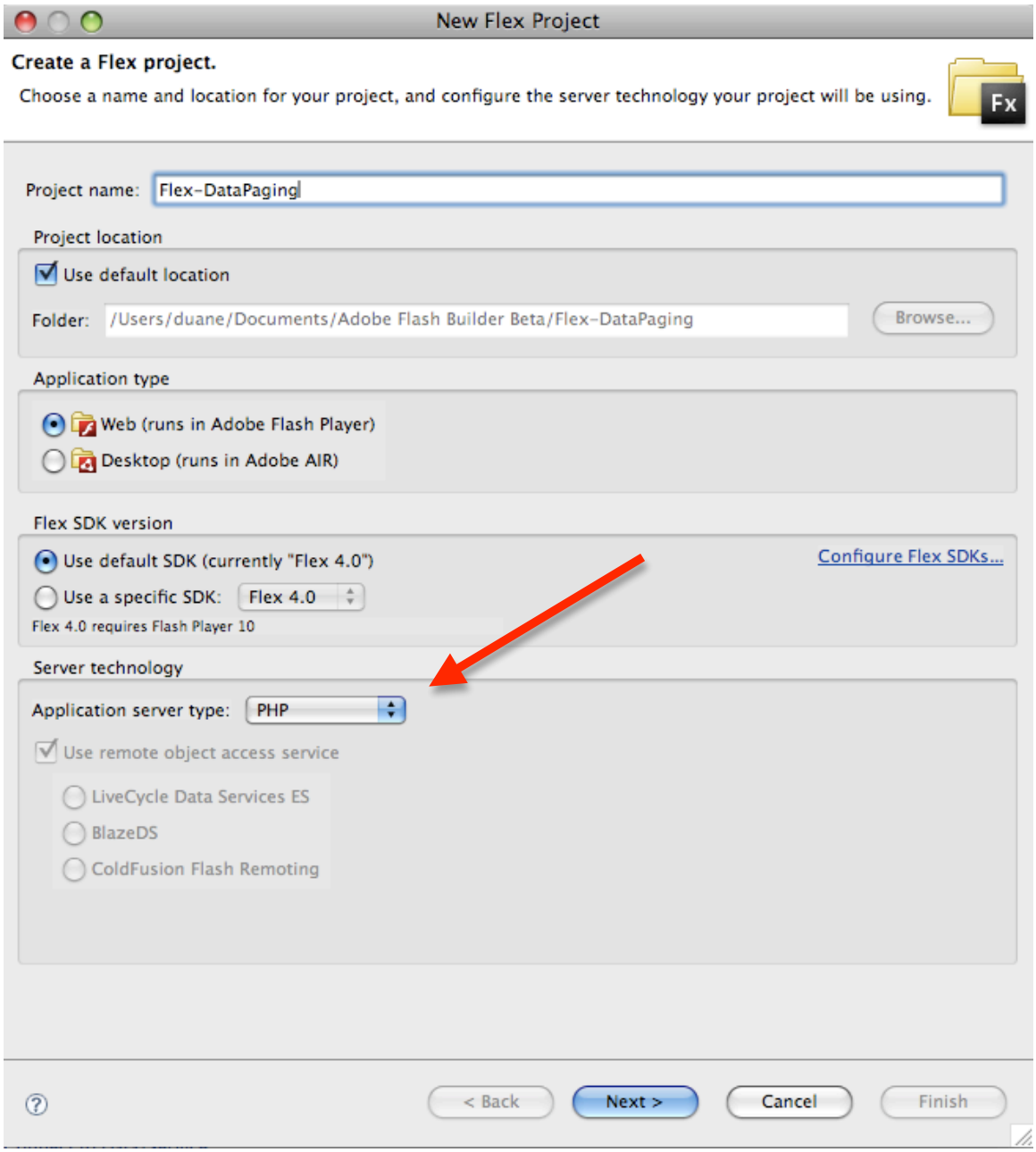
Before you begin, set up the CustomerService.php file under your MAMP htdocs root. Below you can see that I used the path htdocs/datapaging/CustomerService.php.



1. Create a new Flex project



2. Create a new Flex project and enter project details as shown in the image above. Click on next to enter the PHP server details, you will see a window launched as shown in the image below.



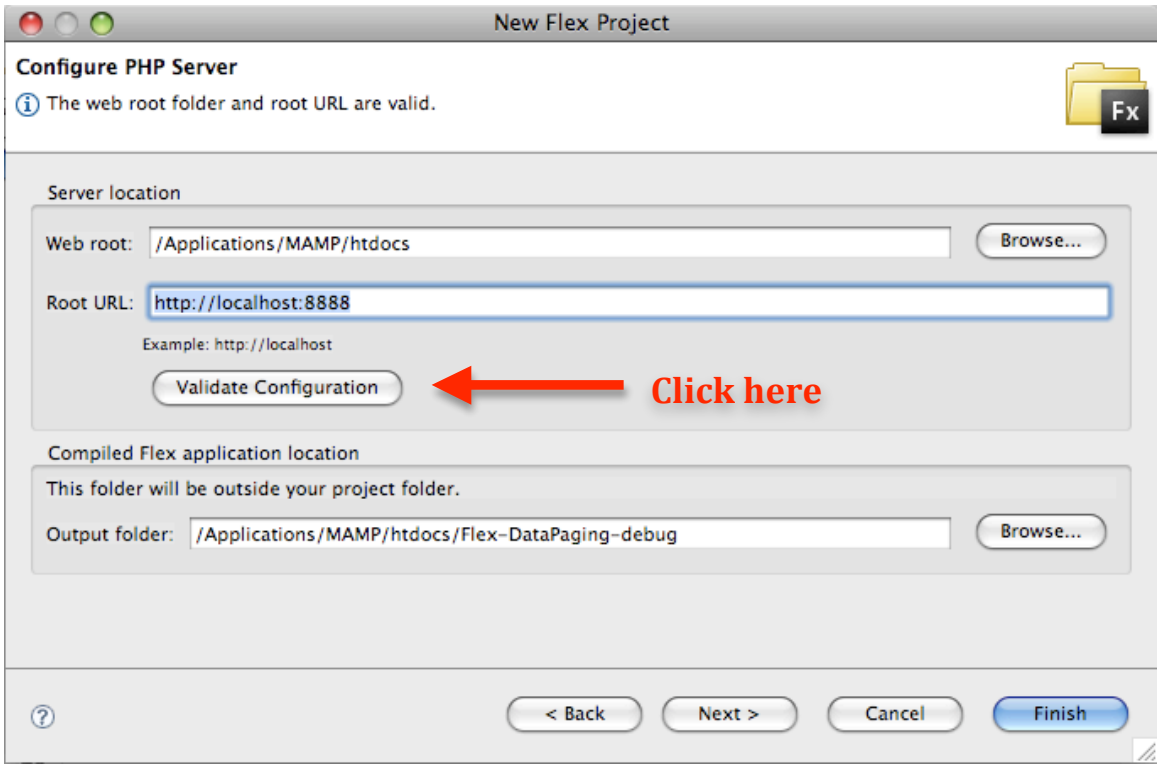
3. In this next screen, validate your projects configuration. Set the Web root to the root folder of your PHP server. In the sample below, it is **`/Applications/MAMP/htdocs`**.

4. Set the root URL to the root URL of your PHP server. Its **http://localhost:8888** in this case

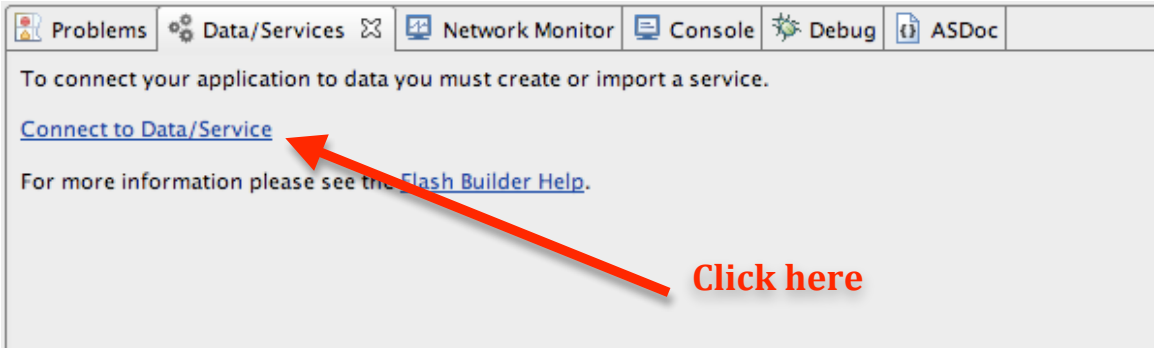
5. Just leave the output folder to default

6. Click the “Validate Configuration” to validate the server details entered

5. Click on finish to continue

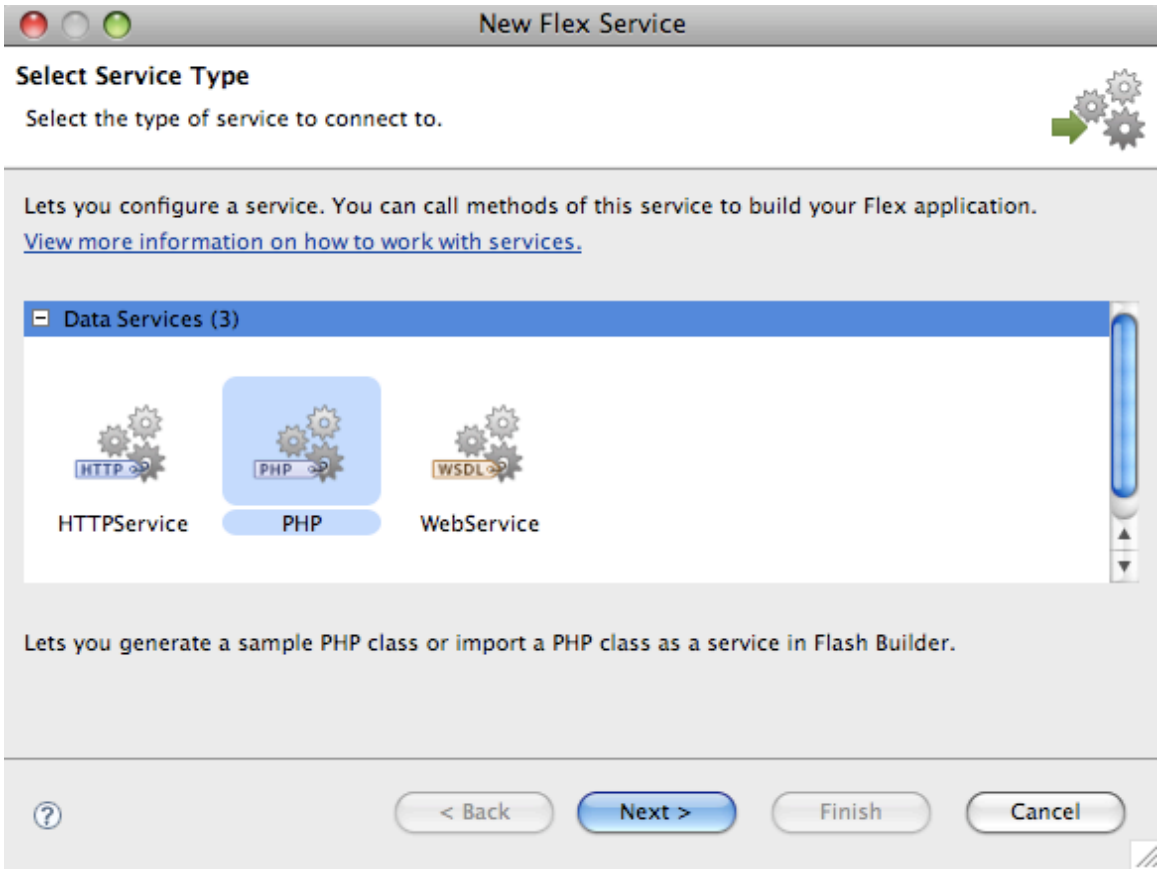


6. Create a service using the service Wizard as shown below.

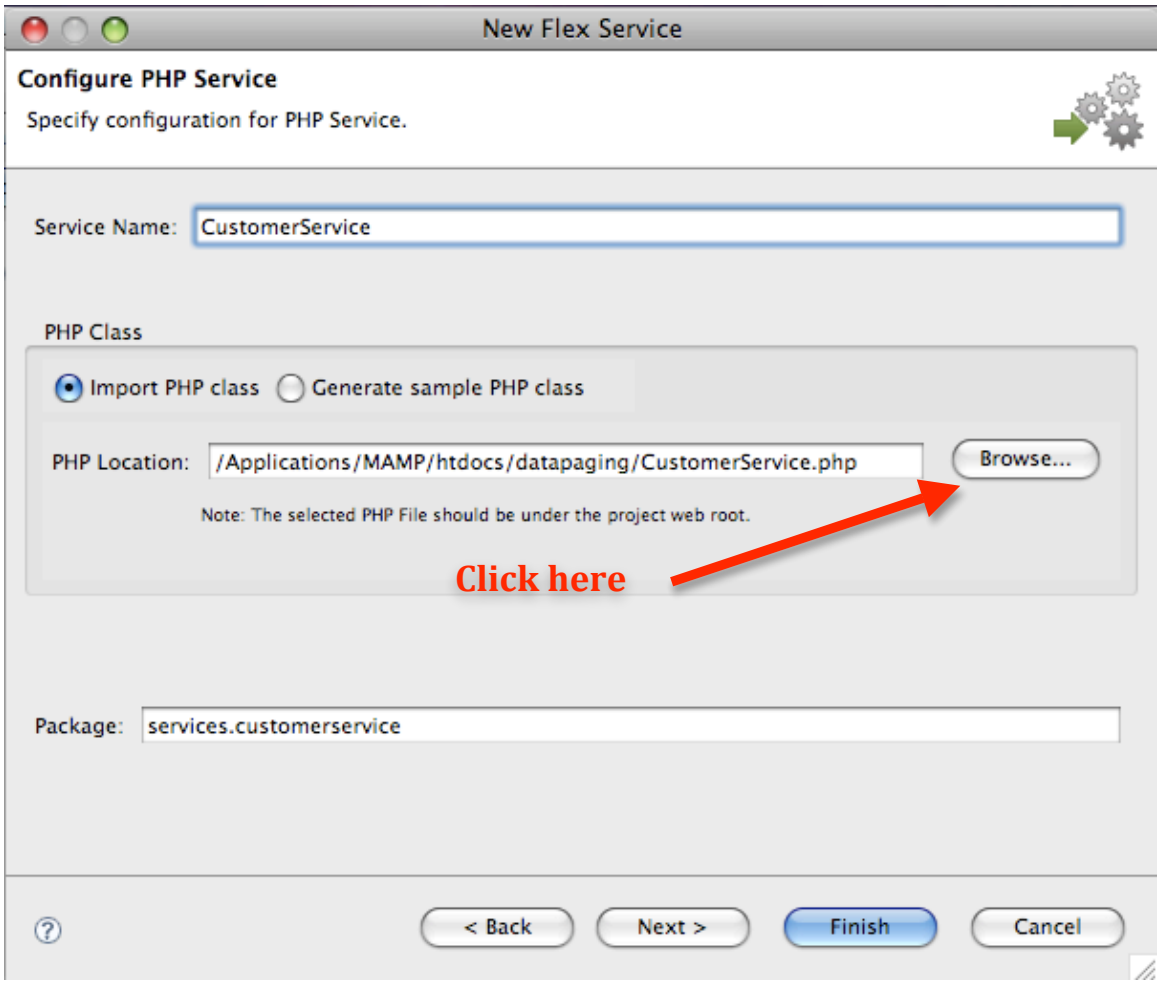


7. You can see Data/Service window, which is called services explorer. Click on the "***Connect to Data/Service***" link to create a new service, you can see a window launched as shown in the image below.

8. Set the service type to PHP.



9. In this window we can select the type of service we want to create. Since we want to communicate with the PHP we created, select PHP and click on ***next*** button.



10. Name the service as **CustomerService** (note that this happens automatically if you first browse for the file

`<MAMP_HTDOCS_ROOT>/path/CustomerService.php.`

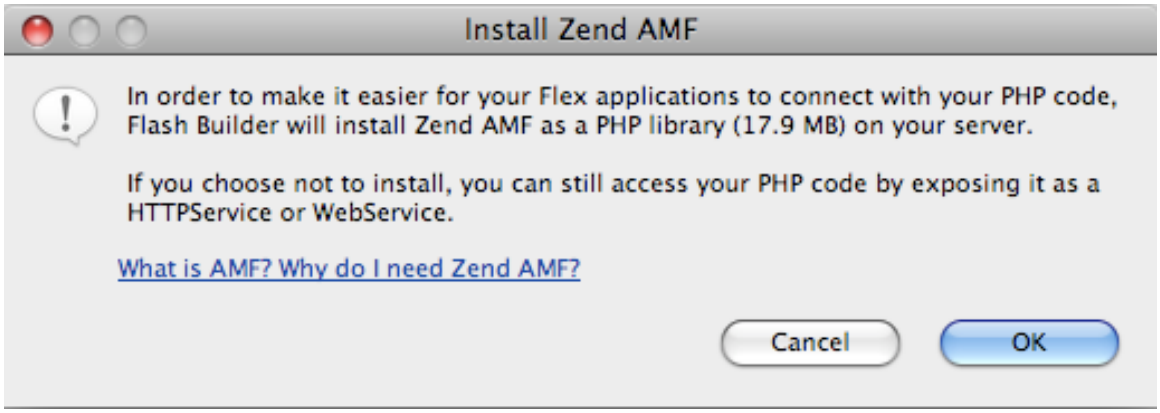
11. Make sure "Import PHP class" radio button is selected, because we are trying reuse the PHP class created.

12. Click on ***next*** button to continue.

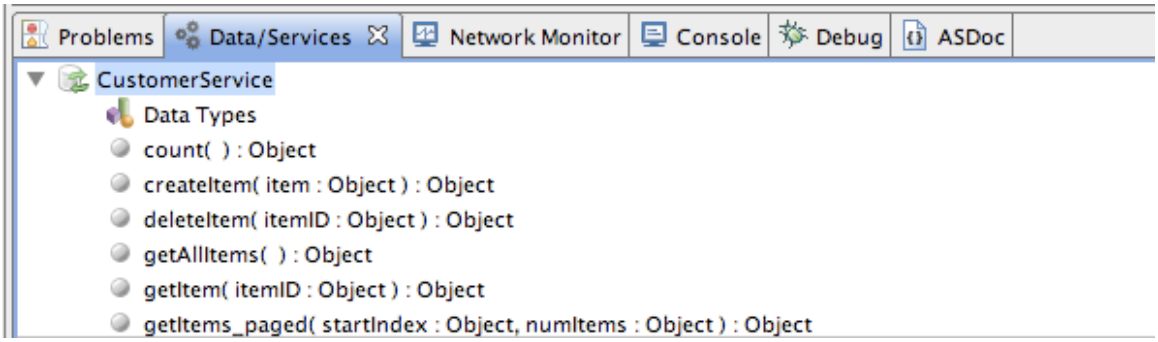
Optional:

NOTE: You can copy the CustomerService.php from the end of this courseware in Appendix A.

You might see a window displayed, saying Zend AMF library will be installed. Just click on OK and continue will the set up.



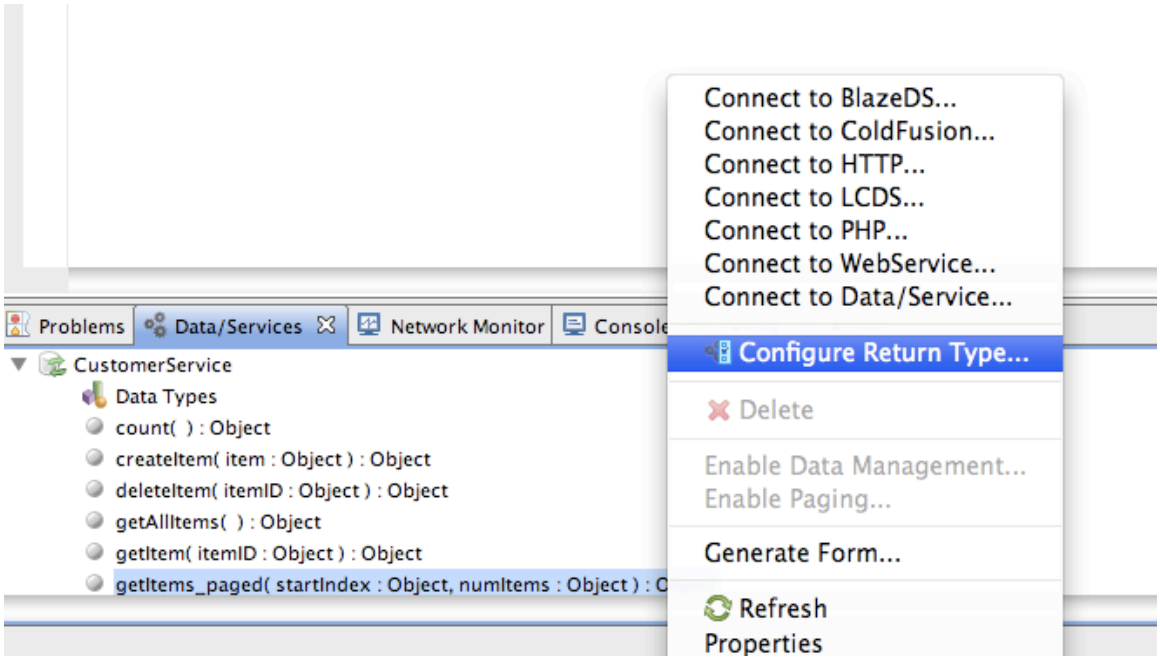
13. Flash Builder will deploy the Zend AMF on your PHP server. Once the installation completes, Flash Builder will display a message. Clicking OK will display the functions in the PHP class. Functions are referred as "operation". Image below shows the window with operations exposed.



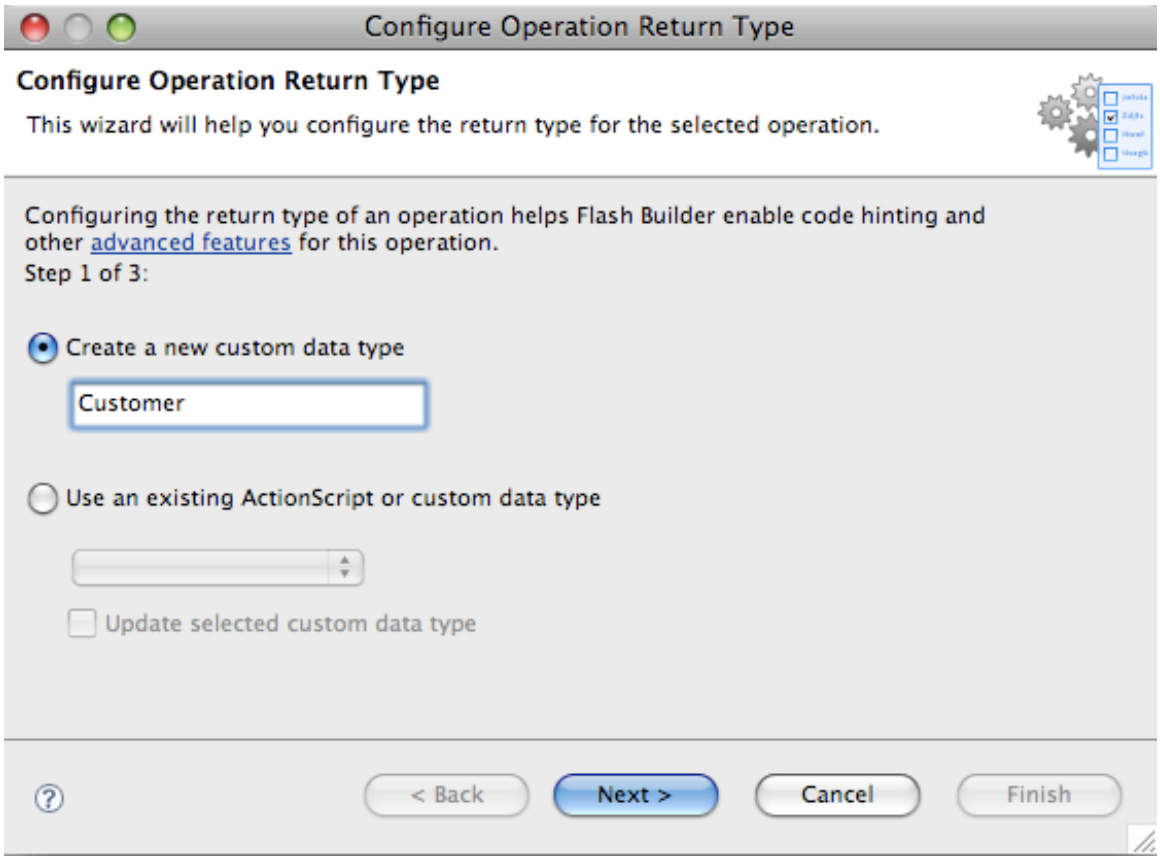
14. Click on finish to continue. A message window displaying next logical steps will be displayed as shown in the image below. You can also see the service created above listed in the services explorer. Just click on the ok button in the message window.

15. In this step we will configure the return type and test the operation and configure the return type on the client i.e. we will specify what type of object to create with the response from the server. This will make it easy for us to develop, since it is easier to deal with strong typed objects.

Right click (Control Click on Mac) on the "getItems_paged()" operation in the services explorer and select "Configure return type" as shown in the image below.

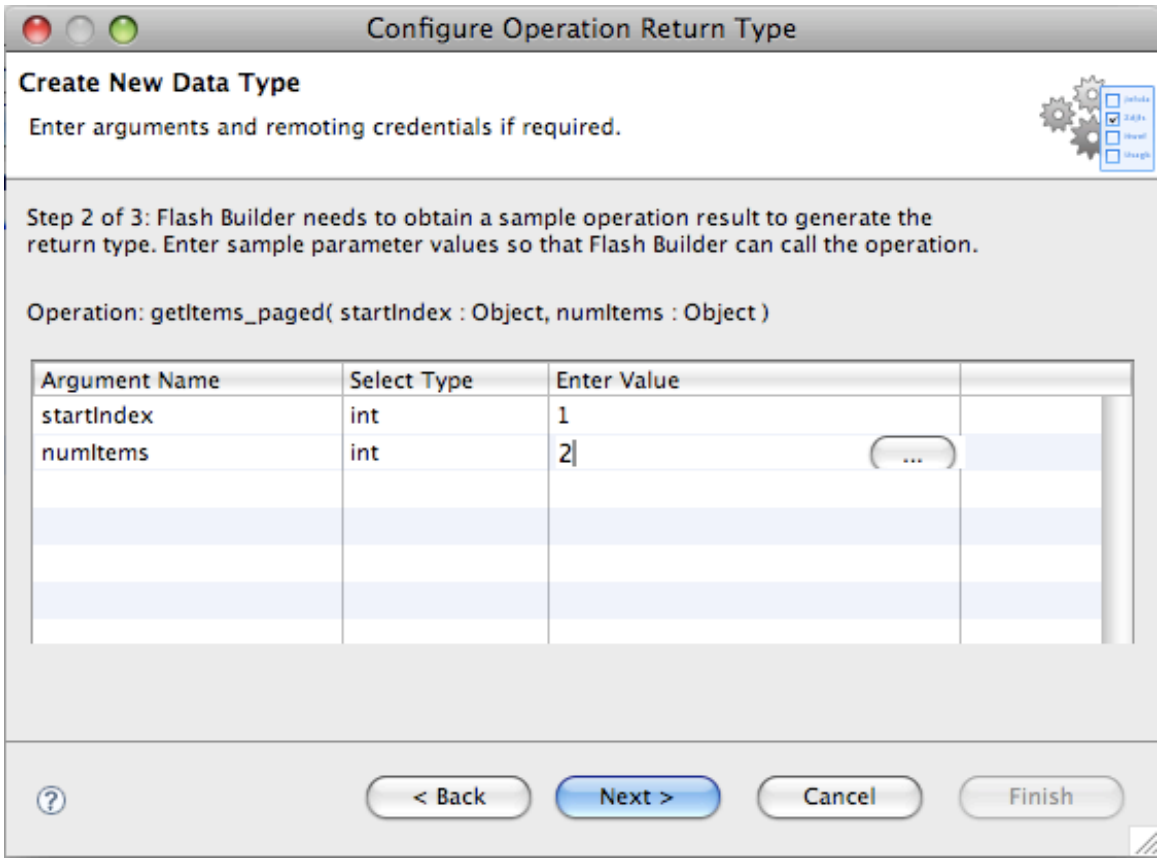


16. A window as shown in the image below will be launched with options to configure the return type. We can chose to create a new data type based on the response from the server. We will configure to create a class named "Customer" with the response.

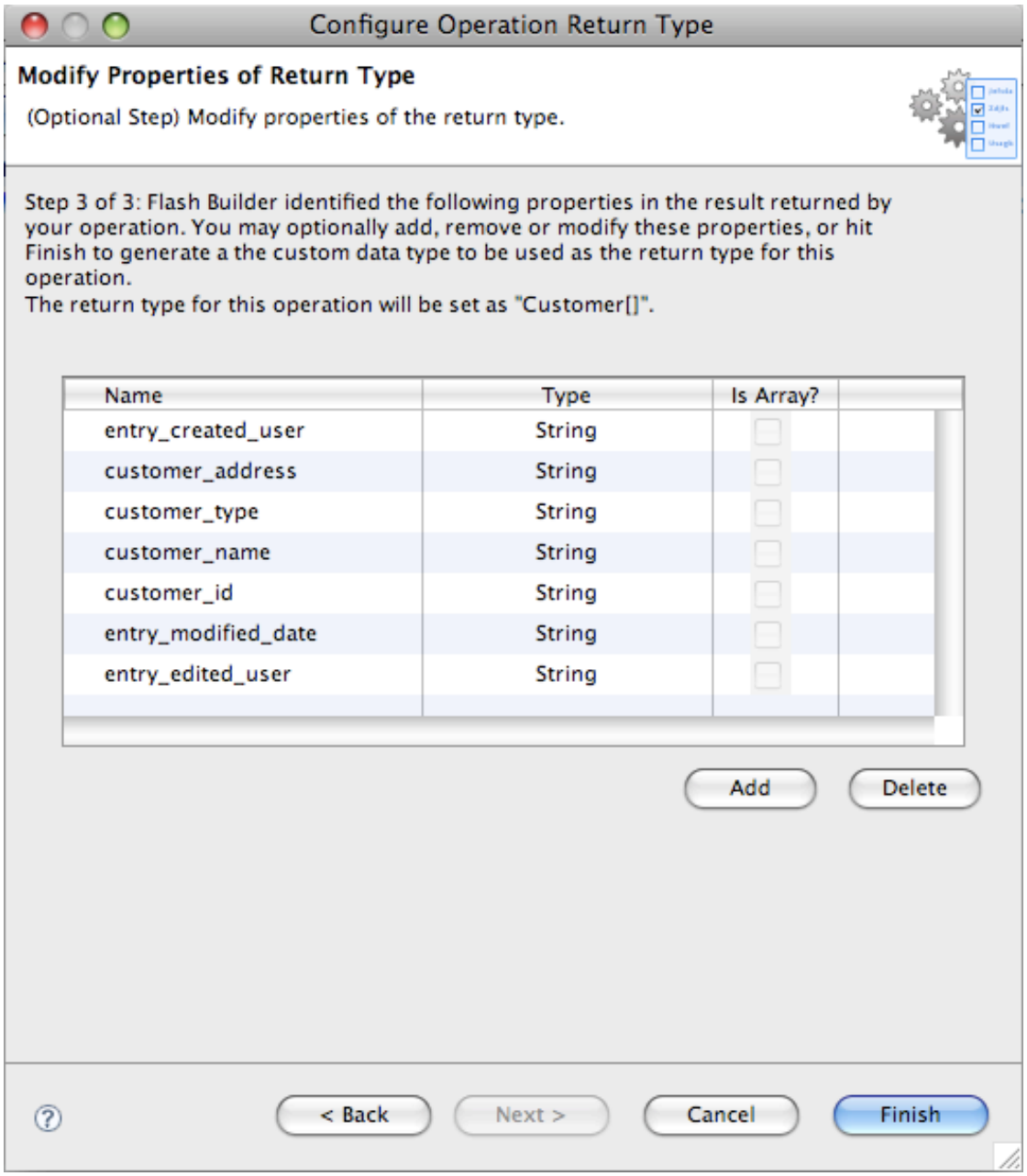


17. Enter the name of the class as shown in the image below and click on the next. Next invoke the operation and introspect the response. We can do this by sending a request to the server and see the response to configure the "Customer" class type.

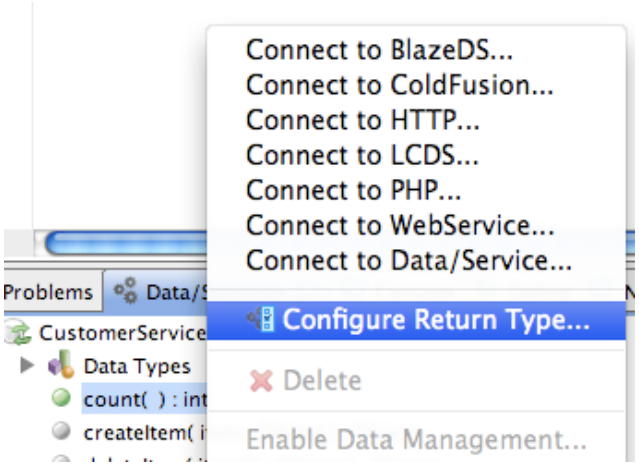
18. Operation requires two integer values as input arguments. Enter those two values as shown in the image above. Since our service doesn't require authentication, select "No" and click on next to continue. You can see the response from the server in the window as shown in the image below.



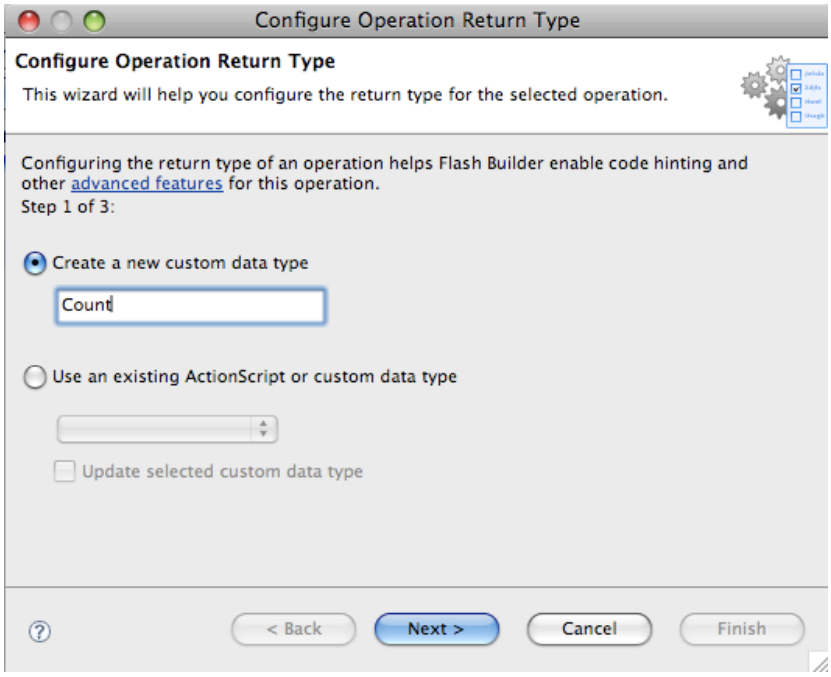
19. Just click on "Next". You can see the return type of the operation changed as shown in the image below. When we invoke the "getItems_paged()" operation, response will be object of the type "Customer[]" (note that this is different than "Customer"), in our case ArrayCollection containing Customer objects.



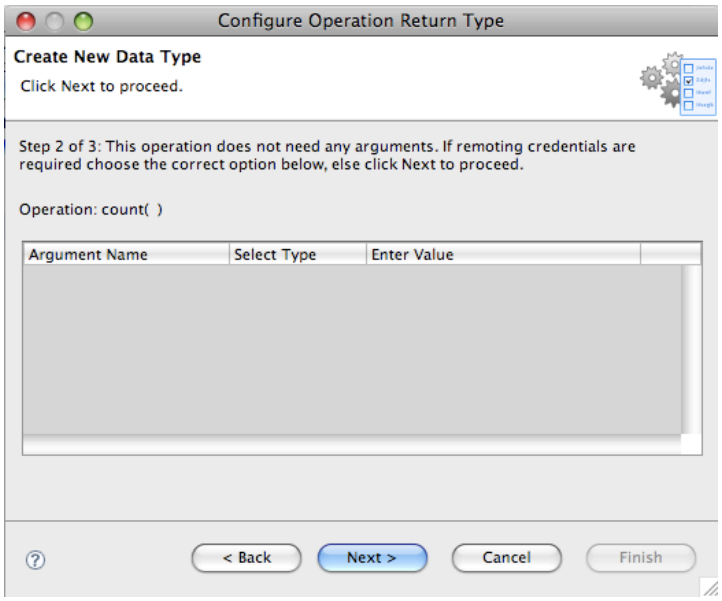
20. Similarly configure return type for “count()” operation to “int”.



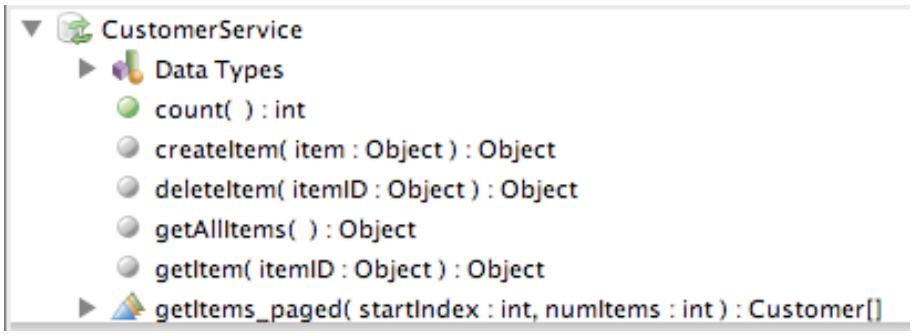
then enter "Count"



Note that the Count return configuration does not require any arguments as the previous example did:

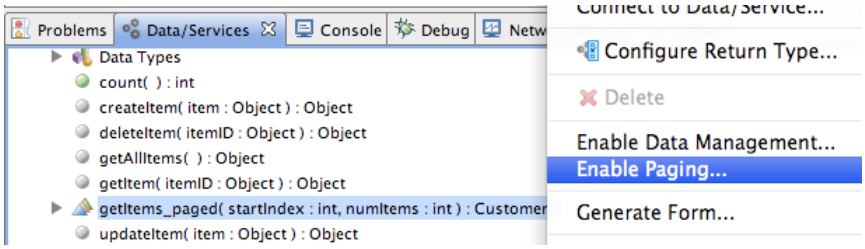


Voila~



Note that the lower "getItems_paged() operation has two parameters in it's signature. These correspond to the start index and the number of items returned.

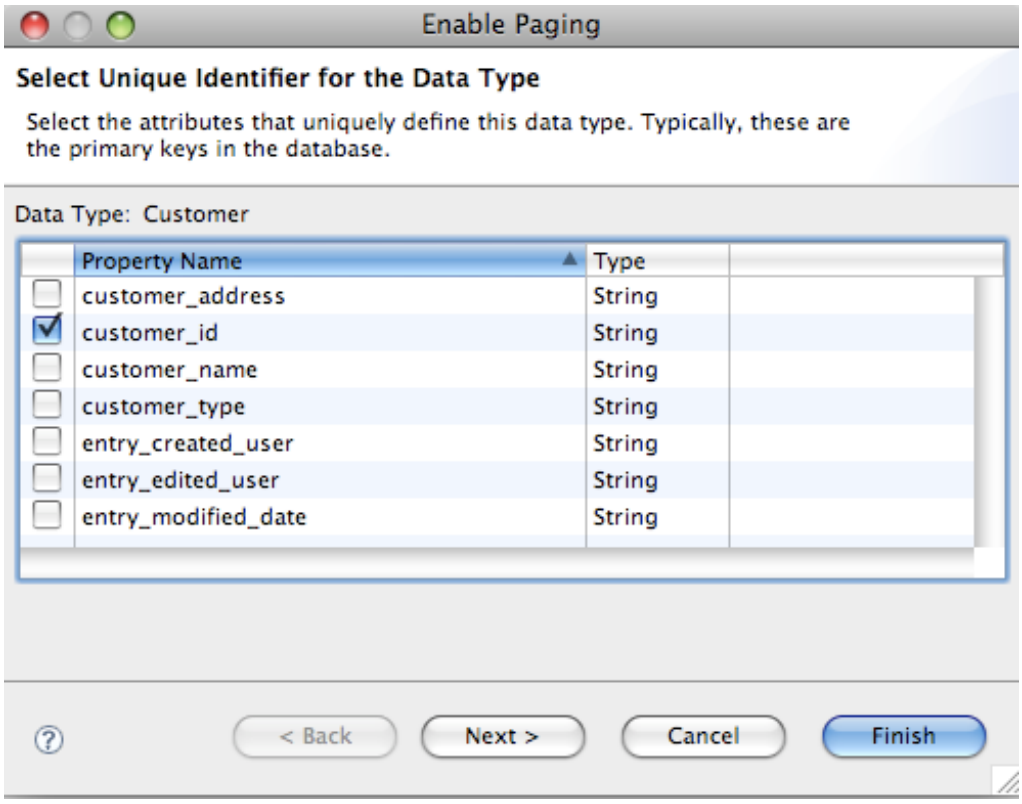
22. To enable paging right click on "getItems_paged()" operation and select "Enable paging" as shown in the image below.



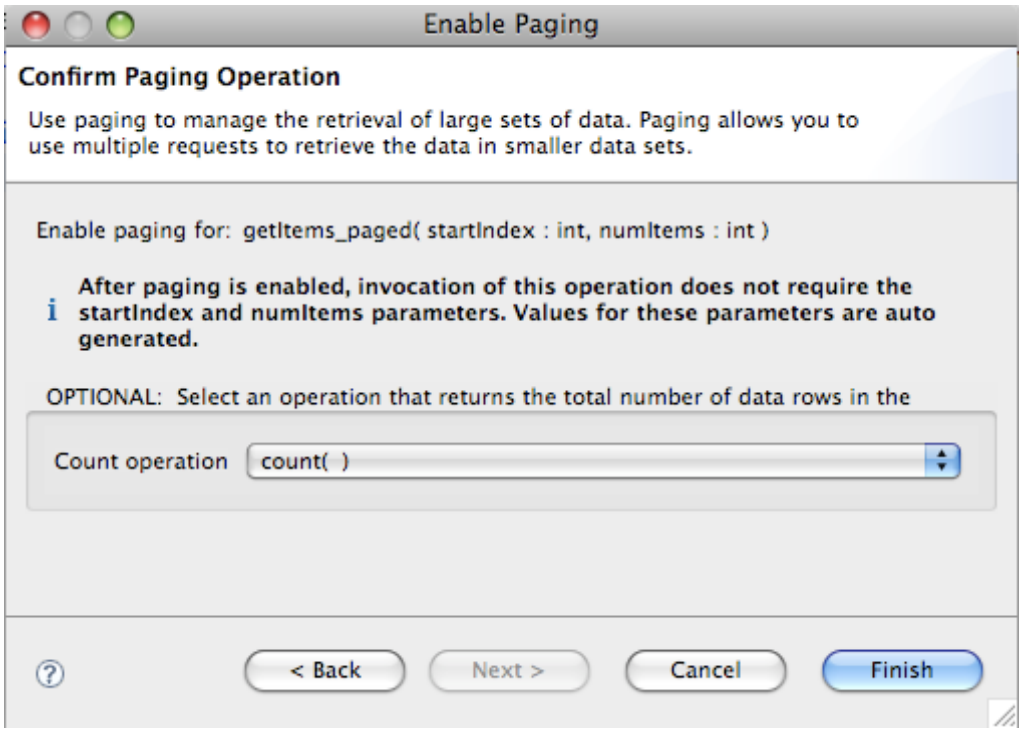
23. You will get a window as shown in the image below, where you need to select the primary key for the Customer class. If you recall the SQL used to create the database as shown below (Bold red font), the primary key was customer_id.

```
DROP TABLE IF EXISTS `customers`;
CREATE TABLE `customers` (
 `customer_id` int(11) NOT NULL auto_increment,
 `customer_name` varchar(225) default NULL,
 `customer_address` blob,
 `customer_type` varchar(100) default NULL,
 `entry_created_user` int(11) default NULL,
 `entry_edited_user` int(11) default NULL,
 `entry_modified_date` datetime default NULL,

 PRIMARY KEY (`customer_id`),

 UNIQUE KEY `customer_name` (`customer_name`)

) ENGINE=InnoDB AUTO_INCREMENT=82 DEFAULT CHARSET=utf8;

...
```

24. Use the primary key and check off the box beside it as shown below and hit "Next" (not "Finish").

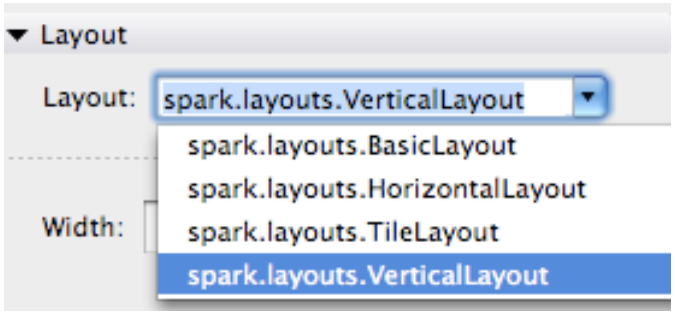


25. The dialog screen will come up and prompt you to select the operation to use for getting total count of records. Flex 4 can page small sets of data and does not require parameters be sent with subsequent paging requests. Count() is used to track how many records are left to use count() for this setting as shown below.



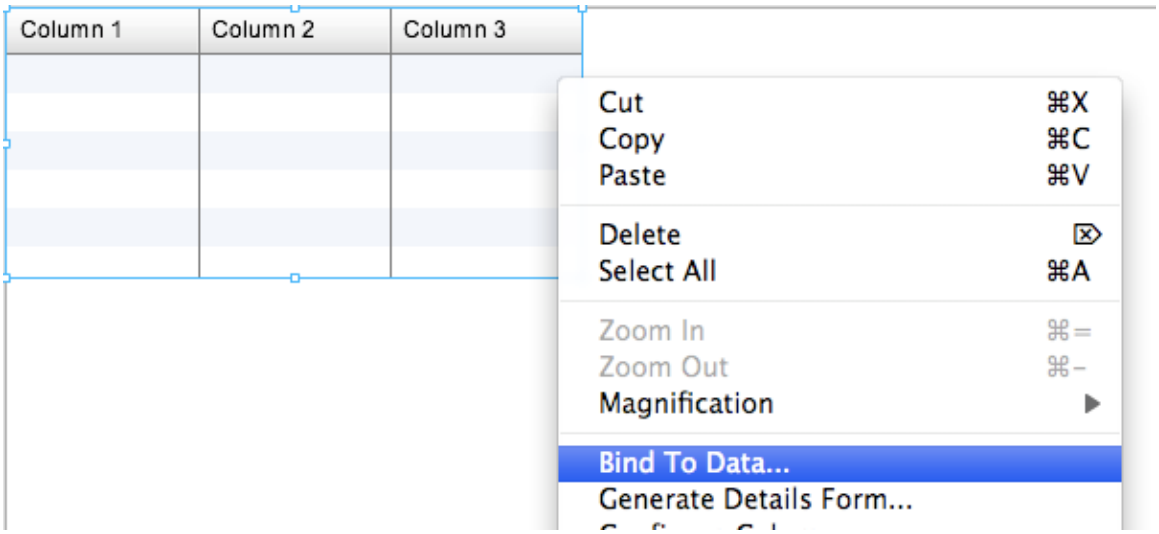
26. All that is left to do it to build a graphical user interface to control how we fetch data. We have enabled data paging and can demonstrate how data is fetched on demand automatically

27. Switch to design view as shown in the image below and change the Application layout to "vertical".

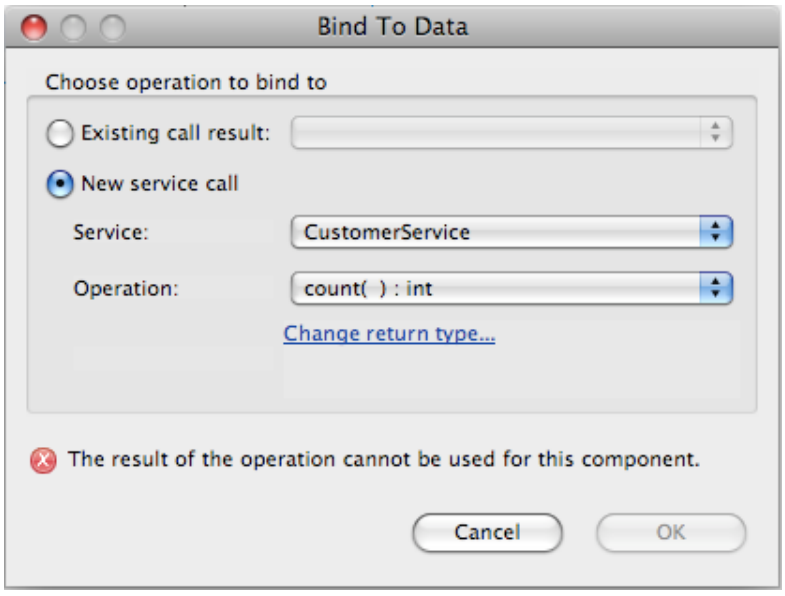


28. Drag and drop a DataGrid from the components panel to the design view and make sure it is selected (the outer lines turn blue).

29. Right click on the DataGrid and select “Bind To Data”



30. Note the red X at the bottom warning you that the result of the operation cannot be used for this component. This is because count() is chosen by default. Change the Operation to getItems_pages():Customer[].



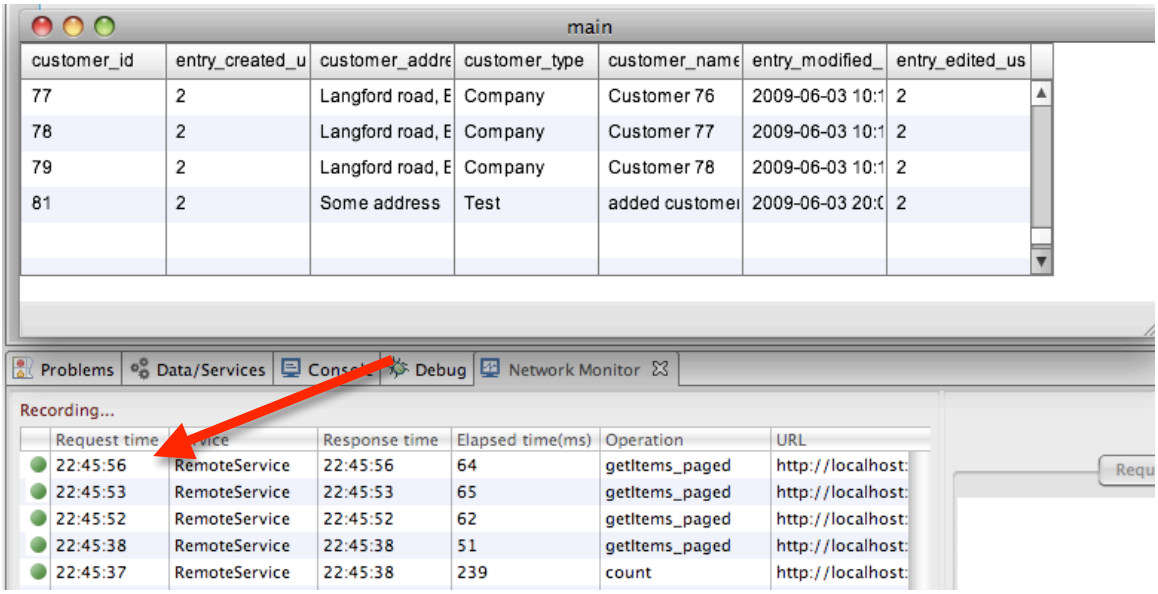
In this screen, select "Customer Service" and "getItems_paged()" operation and click ok and continue.

Save the application and run. Paging is enabled and the data will be retrieved only when it is required, for example when you scroll the DataGrid to a row whose data is not retrieved, the data gets loaded from server automatically and displayed. You can change the paging size by adding following lines of code as shown in the image below.

```
var dm:DataManager = customerService.getDataManager(
customerService.DATA_MANAGER_CUSTOMER);
dm.pageSize = 25;
```

WHEN YOU RUN:

To demonstrate the data paging, make sure the network monitor is viewable so you can see additional calls as the datagrid is scrolled down:



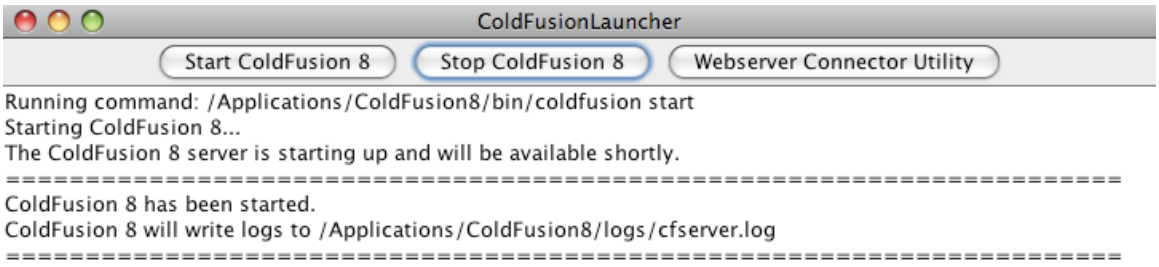
Project 5 Solution Code

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
xmlns:s="library://ns.adobe.com/flex/spark"
xmlns:mx="library://ns.adobe.com/flex/halo"
xmlns:customerservice="services.customerservice.*">
	<fx:Script>
		<![CDATA[
			import services.customerservice.Customer;
			import mx.rpc.AsyncToken;
			import mx.data.DataManager;
			import mx.events.FlexEvent;
			import mx.controls.Alert;


			protected function
dataGrid_creationCompleteHandler(event:FlexEvent):void
			{
				getItems_pagedResult.token =
customerService.getItems_paged();
			}
		]]>
	</fx:Script>
	<mx:DataGrid id="dataGrid"
creationComplete="dataGrid_creationCompleteHandler(event)"
dataProvider="{getItems_pagedResult.lastResult}" editable="true"
height="157" width="696">
		<mx:columns>
			<mx:DataGridColumn headerText="customer_id"
dataField="customer_id" editable="false"/>
			<mx:DataGridColumn headerText="entry_created_user"
dataField="entry_created_user"/>
			<mx:DataGridColumn headerText="customer_address"
dataField="customer_address"/>
			<mx:DataGridColumn headerText="customer_type"
dataField="customer_type"/>
			<mx:DataGridColumn headerText="customer_name"
dataField="customer_name"/>
			<mx:DataGridColumn headerText="entry_modified_date"
dataField="entry_modified_date"/>
			<mx:DataGridColumn headerText="entry_edited_user"
dataField="entry_edited_user"/>
		</mx:columns>
	</mx:DataGrid>

	<s:layout>
```

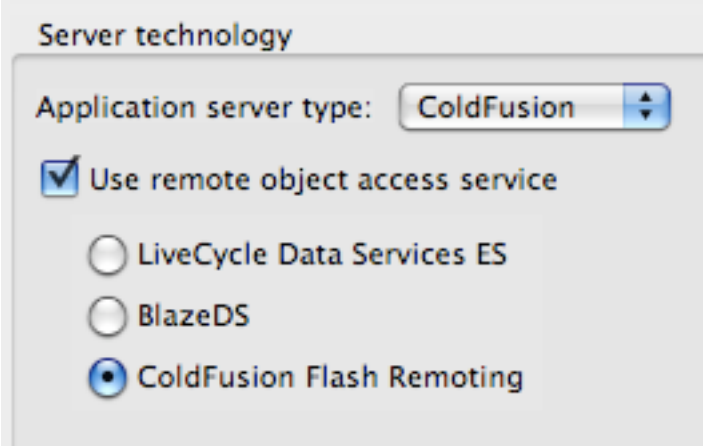
```
            <s:VerticalLayout/>
      </s:layout>
      <fx:Declarations>
            <s:CallResponder id="getItems_pagedResult"/>
            <customerservice:CustomerService id="customerService"
destination="CustomerService" endpoint="http://localhost:8888/BSC-Lab8-
DataPagingPHP-debug/gateway.php"
fault="Alert.show(event.fault.faultString)" showBusyCursor="true"
source="CustomerService"/>
      </fx:Declarations>

</s:WindowedApplication>
```

Project 6: - Flash Builder 4 talks to ColdFusion

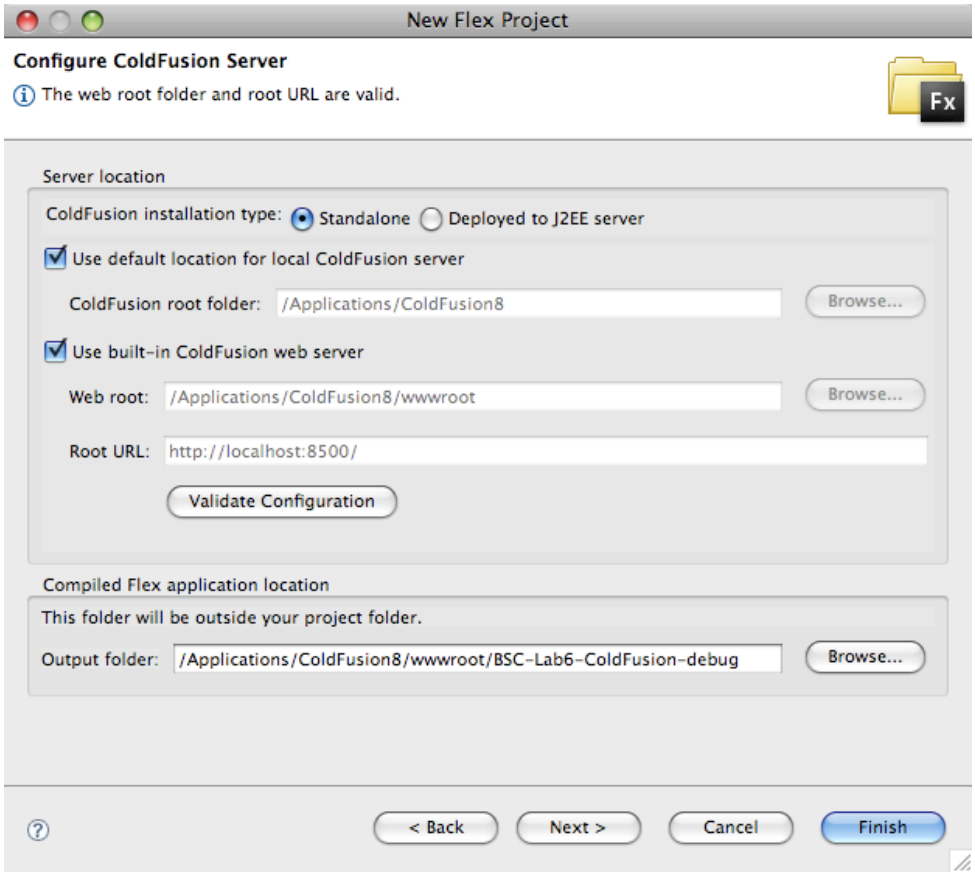
1. A start up your ColdFusion instance.



2. Start a new Flash Builder 4 project and choose "server Type = ColdFusion"



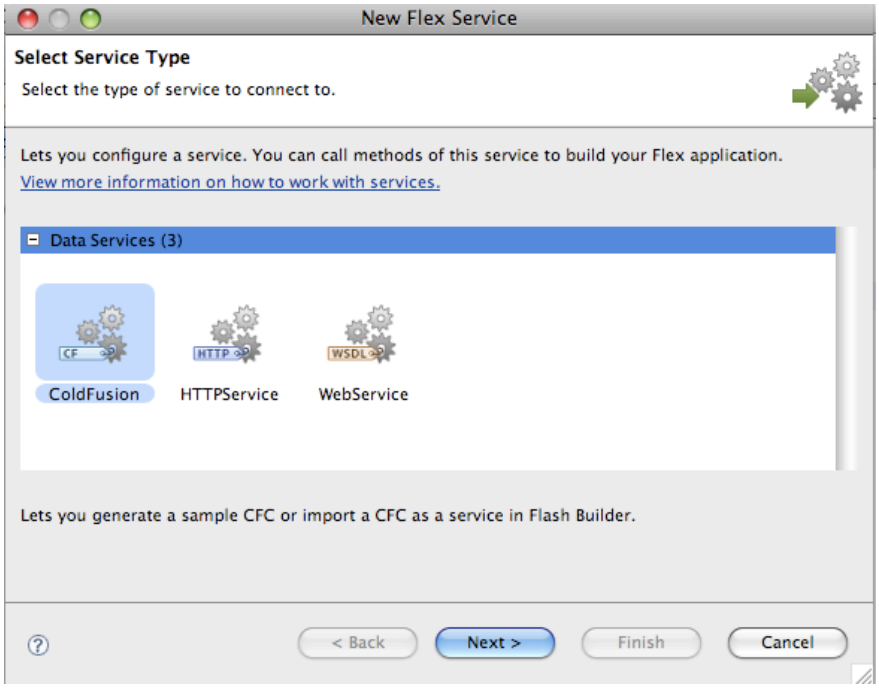
3. ON the next page set the connection properties and validate as shown below



4. Click Finish

5. In the data/services tab at the bottom – select "Connect to Data/Service"

6. Select the type of ColdFusion from the next dialog as shown below

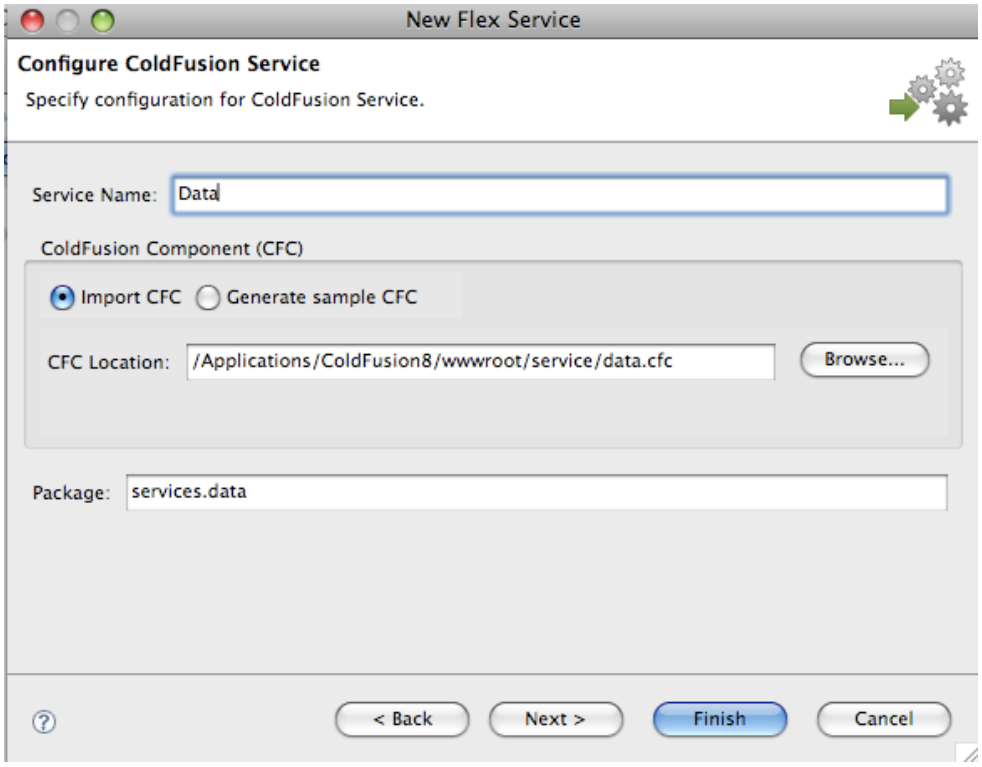


7. Click Next. Flash Builder 4 will offer that you can either point at an existing CFC (Cold Fusion Component) or generate a new one. Click the Browse button and aim it at the

8. **<COLD_FUSION_ROOT_FOLDER>/wwwroot/service/data.cfc** file. The Service name and packages will be done for you.

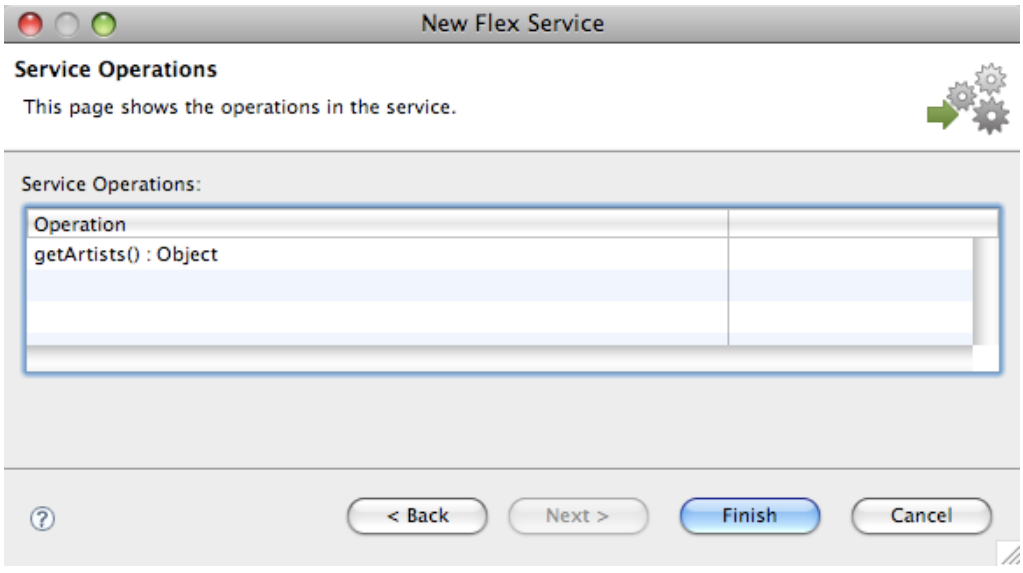
FILE: data.cdc

```
<cfcomponent output="false">
<cffunction name="getArtists" access="remote"
    returntype="query">
     <cfset var  result="">
        <cfquery datasource="cfartgallery" name="results">
</cfquery>
<cfreturn results>
</cffunction>
</cfcomponent>
```

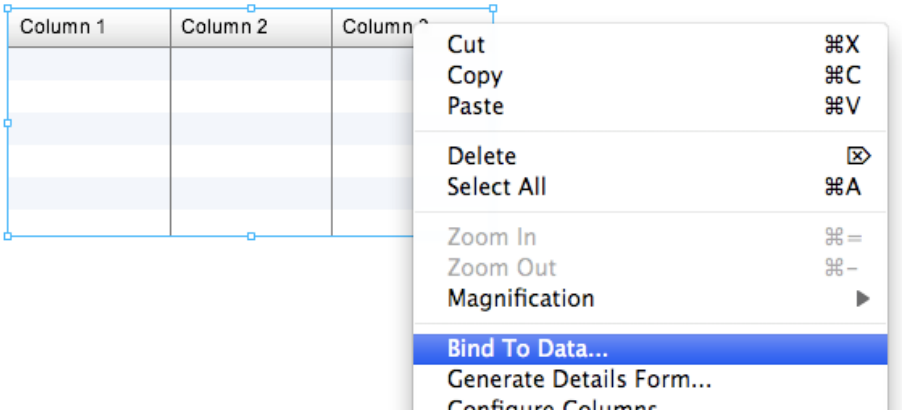


9. Click "Next". A dialog will pop up asking for your admin username and password. In this lab, the values are "Administrator/password".

10. Flash builder will inspect the CFC and bring back all available operations as shown below. Click Finish.

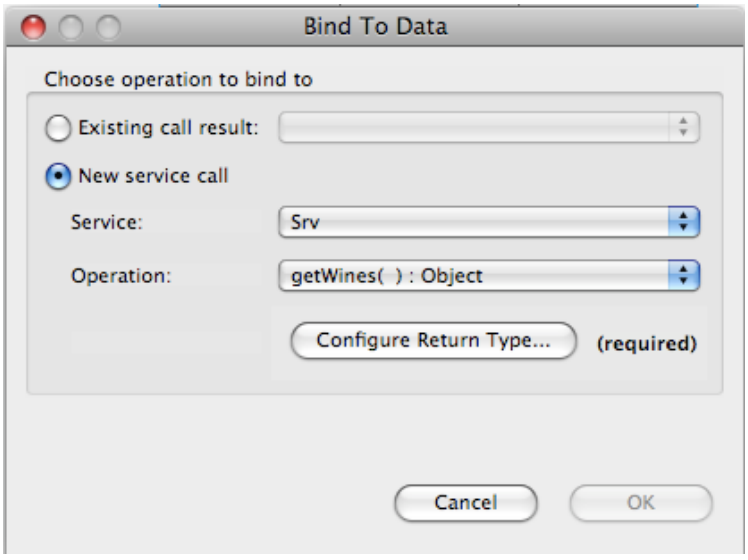


11. Switch to Design View and add a DataGrid. Select the dataGrid so it is highlited then right click (Control Click on Mac) and select "Bind Data" as shown below:

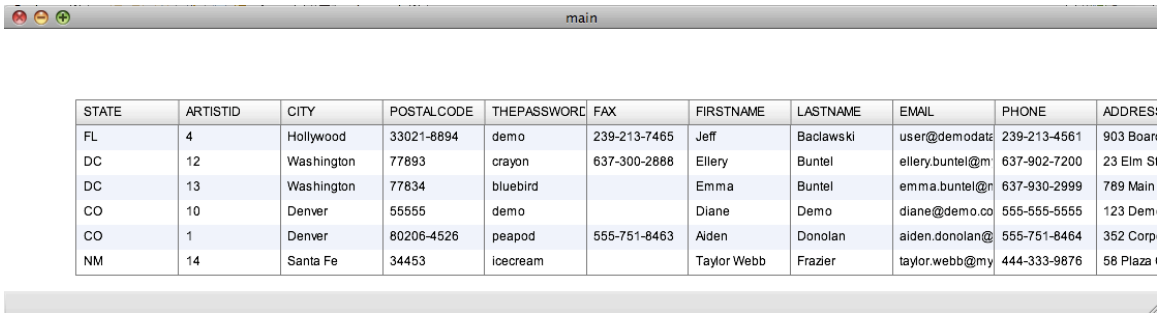


12. In the next dialog select Configure Return Type (note that the service and operation names below are not right. For this project they will be getArtists().

13. Accept the defaults and click finish



14. Run the application.



Project 6 Solution Code

```
<?xml version="1.0" encoding="utf-8"?>
<s:WindowedApplication xmlns:fx="http://ns.adobe.com/mxml/2009"
                       xmlns:s="library://ns.adobe.com/flex/spark"
                       xmlns:mx="library://ns.adobe.com/flex/halo"
xmlns:data="services.data.*">
      <mx:DataGrid x="69" y="68" id="dataGrid"
creationComplete="dataGrid_creationCompleteHandler(event)"
dataProvider="{getArtistsResult2.lastResult}" editable="true">
            <mx:columns>
                  <mx:DataGridColumn headerText="STATE"
dataField="STATE"/>
                  <mx:DataGridColumn headerText="ARTISTID"
dataField="ARTISTID"/>
                  <mx:DataGridColumn headerText="CITY"
dataField="CITY"/>
                  <mx:DataGridColumn headerText="POSTALCODE"
dataField="POSTALCODE"/>
                  <mx:DataGridColumn headerText="THEPASSWORD"
dataField="THEPASSWORD"/>
                  <mx:DataGridColumn headerText="FAX" dataField="FAX"/>
                  <mx:DataGridColumn headerText="FIRSTNAME"
dataField="FIRSTNAME"/>
                  <mx:DataGridColumn headerText="LASTNAME"
dataField="LASTNAME"/>
                  <mx:DataGridColumn headerText="EMAIL"
dataField="EMAIL"/>
                  <mx:DataGridColumn headerText="PHONE"
dataField="PHONE"/>
                  <mx:DataGridColumn headerText="ADDRESS"
dataField="ADDRESS"/>
            </mx:columns>
      </mx:DataGrid>
```

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

			protected function
dataGrid_creationCompleteHandler(event:FlexEvent):void
			{
				getArtistsResult.token = data.getArtists();
				getArtistsResult2.token = data.getArtists();
			}

		]]>
	</fx:Script>
	<fx:Declarations>
		<s:CallResponder id="getArtistsResult"/>
		<data:Data id="data" destination="ColdFusion"
endpoint="http://localhost:8500/flex2gateway/"
fault="Alert.show(event.fault.faultString)" showBusyCursor="true"
source="service.data"/>
		<s:CallResponder id="getArtistsResult2"/>
	</fx:Declarations>
</s:WindowedApplication>
```

About the teachers

Duane Nickull is a Senior Technical Evangelist for Adobe Systems and host of the Adobe TV series *"Duane's World".* Duane has written or participated in most of the larger SOA standards work in the past decade. He currently chairs the OASIS Service Oriented Architecture Reference Model Technical Committee (SOA-RM TC) which has just delivered a Reference Model for Service Oriented Architecture as a full OASIS standard. He served as a Vice Chair of the United Nations Centre for Facilitation of Commerce and Trade (UN/CEFACT) between 2003 and 2006. Within the United Nations, he oversaw the UN's Electronic Business strategy and Service Oriented Architecture and modeling efforts. He has served as the project team lead of the United Nations (UN/CEFACT) Electronic Business Architecture Group (SOA) and a specially appointed liaison between the W3C, UN and OASIS standards consortiums. Additionally, Duane has served as the chair and lead system architect for the United Nation's Electronic Business Working Group, a direct sub-group of CEFACT TMG and on the CEFACT TMG Steering Committee. He also has served as the Co-chair of the ebXML Technical Architecture group as well as co-editor of that specification

starting in 1999, largely recognized as the first post-internet and post XML SOA. Mr. Nickull has written and contributed many technical articles and books on these subjects Mr. Nickull has been called Mr. SOA by his peers during introductions to speak on the subject due to his overwhelming experience writing and contributing to the major Service Oriented Architectures (SOA's). Between 1995 and 2006, he spoke at over 500 venues in various countries around the world Duane has recently renewed his work in the theoretical field of computational intelligence and has recently spoken several times to various audiences via the Ontolog Forum on event-causality aware inference engines coupled to a query-able ontology. In the field of semantic reconciliation, Duane was a co-inventor of the first Context-sensitive XML Search Engine (www.goxml.com) and the first web based XML E-Commerce ASP. He is named on pending patents pertaining to XML indexing and retrieval covering 51 unique points. He also served as Technical Director for XSLT.com during the 1990's until as recently as 2002. He lives in Vancouver, Canada with his wife and three children, plays in a rock band, actively snowboards, races Porsche 911's and mountain bikes. Duane made his living as a professional musician for several years.

Duanes Website: http://technoracle.blogspot.com