How to write a generic LCDS enabled Java service

October 5, 2009
John Mattos

Summary

This document provides the reader with a quick step by step guide to setting up a generic LCDS service in a Java project

Introduction

This guide will cover off a very simple 'Hello World' style Java based RPC service. A companion article will describe how to setup a Flex project that can consume these services. In order to get started, you need to have Flex Builder and LiveCycle Data Services ES installed. If these are already installed, skip the 'Installation' section below and jump to 'Getting Started'

Installation

Installing Flex Builder 3

Download the Flex Builder trial from here (**http://www.adobe.com/go/tryflex**). Download the correct version for your platform. When downloaded, double click the installer and install with all the default options. If prompted for a serial number and you don't have one, just leave it blank. You will then be able to use Flex Builder as a free sixty day evaluation copy.

Installing LiveCycle Data Services ES

Download the LiveCycle Data services trial here (http://www.adobe.com/go/trylivecycle_dataservices). Download LiveCycle Data Services ES for Windows and run the installer. At the serial number screen, leave the serial number blank to use the single CPU license version of LCDS. At the installation location screen, accept the default installation path of `c:\lcds`. At the '**Installer Options**' screen, select '**LiveCycle Data Services with Tomcat**'. Your installation is now complete.

Viewing the LCDS Examples (highly recommended)

1. From the Windows start menu, select '**All Programs > Adobe > LiveCycle Data Services 2.6.1 ES > Start Samples Database**'.
2. From the Windows Start menu, select '**All Programs > Adobe > LiveCycle Data Services 2.6.1 ES > Start LiveCycle Data Services Server**' to start the Tomcat server.
3. Navigate to http://localhost:8400/lcds-samples

Getting started

There are three steps to using a Java based RPC service: Creating the Java service itself , configuring the remoting endpoint and lastly creating the Flex project to consume the service.

For this example I'll be leveraging the `/lcds` sample application that ships with LCDS. If you have done the default LCDS installation you can see this application running by navigating to **http://localhost:8400/lcds**

Create a Java project (the Java Service)

1. In Eclipse, create a new project by selecting '**File > New > Project > Java**' and name your project POJOProject
2. Create a new package named '`services`'
3. Create a new Java class named '`SimpleService.java`'
4. Copy the code shown below:

```
package services;

/*
 * Created on 3/31/2006 - modified 10/02/2009
 *
 * SimpleService class demonstrates service
 * Flex training app
 */

/**
 *@author David Gassner, extended from code by Robert Crooks, Matthew Boles
 *simplified by John Mattos
 *
 * simple service for Flex 3 client
 * can be called as remote object service
 */

import java.util.HashMap;
import java.util.Map;

public class SimpleService {

       // properties
       private String helloString="Hello from a Simple Service!";
       private Map[] menuInfo = new HashMap[2];

       // constructors
       public SimpleService() {}

       //methods

       /**
        *A method that returns a simple string
        */
       public String outMethod()
       {
            return helloString;
       }

       /**
        *A method that returns static complex data
        */
       public Map[] arrayOutMethod()
       {
            Map menu1=new HashMap();
            Map menu2=new HashMap();
```

```
            menu1.put("starter","Smoked Salmon");
            menu1.put("main","Filet");
            menu1.put("dessert","Key Lime Pie");
            menu2.put("starter","Calamari");
            menu2.put("main","Curried Lamb");
            menu2.put("dessert","Berry Pie");
            menuInfo[0]=menu1;
            menuInfo[1]=menu2;
            return menuInfo;
        }

        /**
         *A method that received 2 Strings and returns a concatenated String
         */
        public String multArgsMethod(String oneArg, String twoArg)
        {
            return ("The String arguments are: " + oneArg +
                     " and " + twoArg );
        }
}
```

5. Compile the `SimpleService` to `SimpleService.class`
6. Copy the services directory containing your class file to **C:\lcds\tomcat\webapps\lcds\WEB-INF\classes**
7. You could also use ANT to package up the Java service into a JAR file and deploy it under **C:\lcds\tomcat\webapps\lcds\WEB-INF\libs**. This would be a more appropriate solution for a more complex service.
8. The LCDS-Samples application available at **http://localhost:8400/lcds-samples** contain other great examples, as does Christophe Coenraets "Test Drive" application **http://coenraets.org/blog/2009/05/new-update-to-the-spring-blazeds-integration-test-drive/**

Create a remoting endpoint

1. Destinations in the **remoting-config.xml** file specify remote services that can be called from flex, therefore we need to add the destination to **C:\lcds\tomcat\webapps\lcds\WEB-INF\flex\remoting-config**, as follows (after the `</default-channels>`

Note that C:\lcds\tomcat\webapps\lcds is a template you can use for creating your own LCDS applications

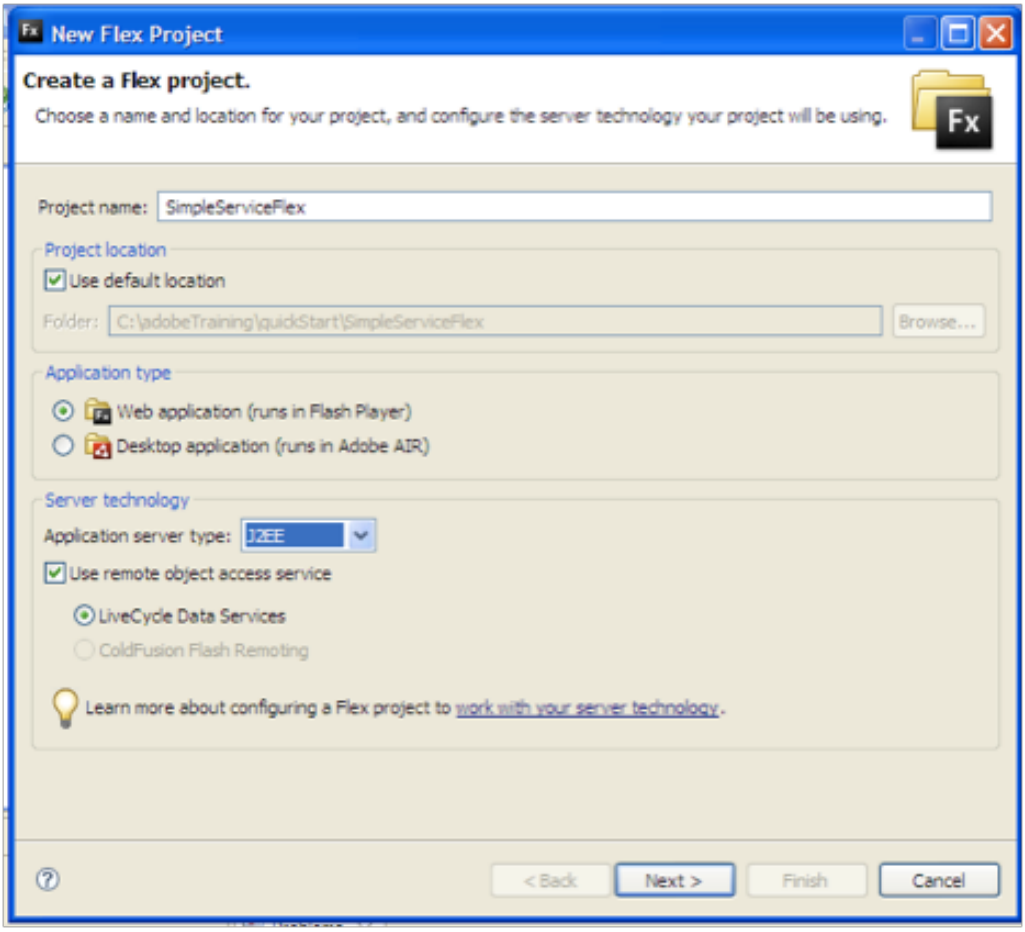
```
<destination id="simpleService">
        <properties>
            <source>services.SimpleService</source>
            <scope>application</scope>
        </properties>
</destination>
```

2. Once the java remote service is defined in **remoting-config.xml** we can proceed to creating the flex code that can call these services.

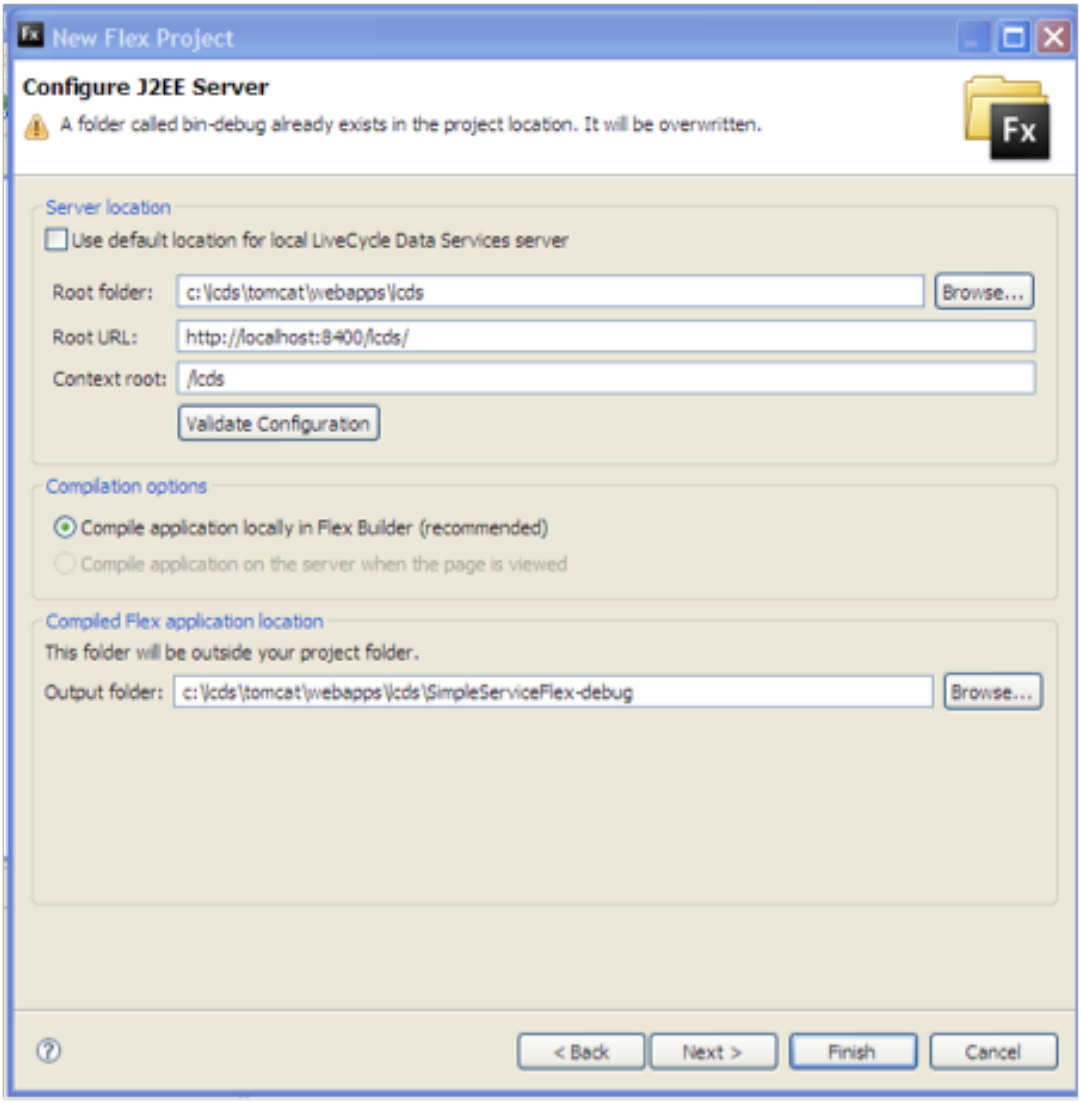
Create the Flex application

1. Click '**New > Flex Project**' within Flex Builder

2. Name the project 'SimpleServiceFlex' and choose 'J2EE' as the application server type:



3. Click the 'Next' button and point the application you're creating to the LCDS application on the default:

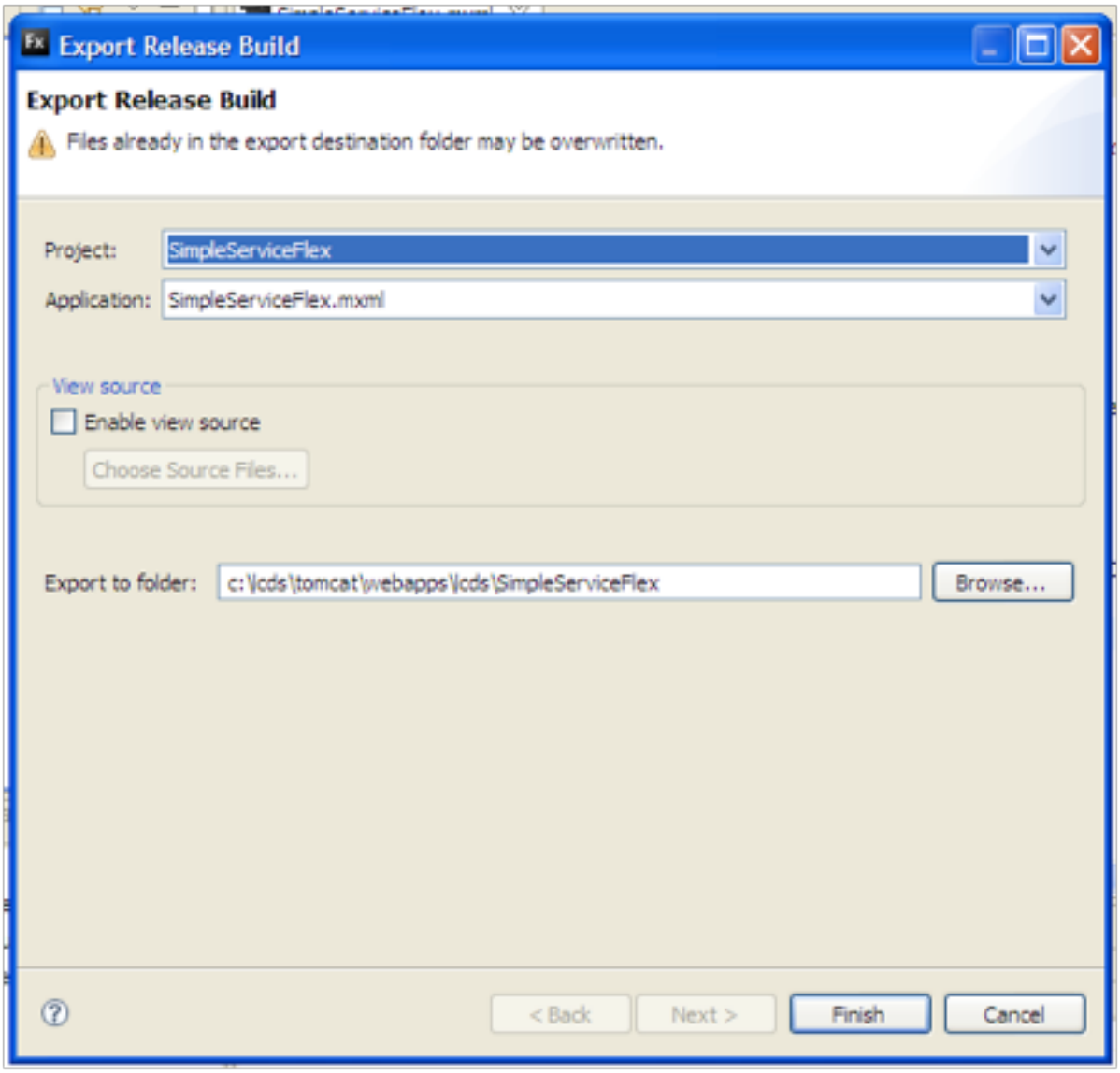


4. Flex will create a default application file - '**SimpleServiceFlex.mxml**'. OPen this up and paste in the following code:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml"
       layout="vertical"
       creationComplete="myRemoteObject.outMethod()">

       <mx:RemoteObject id="myRemoteObject"
            destination="simpleService"
            showBusyCursor="true"/>

       <mx:Label text="{myRemoteObject.outMethod.lastResult}"/>
```

```
        <mx:Button label="Get Data from Service"
            click="myRemoteObject.arrayOutMethod()"/>

        <mx:DataGrid dataProvider="{myRemoteObject.arrayOutMethod.lastResult}"/>

</mx:Application>
```

5. You can also export a 'release build' to the Tomcat server by clicking '**project > export release build**':



6. Run the application: