

Using Remote Object to Send Data to the Server

Install LiveCycle Data Services ES

1. Browse to the url `http://www.adobe.com/products/livecycle/dataservices/` and click the **Download free trial** link.
2. Download LiveCycle Data Services ES Express.
3. Install LiveCycle Data Services ES by running `lcds251-win.exe`.
 - a. At the Serial Number screen, leave the serial number blank to install the Express edition.
 - b. At the Installation Location screen, accept the installation path of `c:\lcds`.
 - c. At the Installer Options screen, select LiveCycle Data Services with integrated JRun.
 - d. Complete the installation.

Installing the Server Files

4. Unzip **adobeODTServer.zip** to `C:\lcds\jrun4\servers\default`.

Starting and testing the Installation

5. From the Windows Start menu, select All Programs > Adobe > LiveCycle Data Services 2.51 ES > Start Integrated LiveCycle Data Services ES Server.

6. Browse to this url: <http://localhost:8700/odt>.

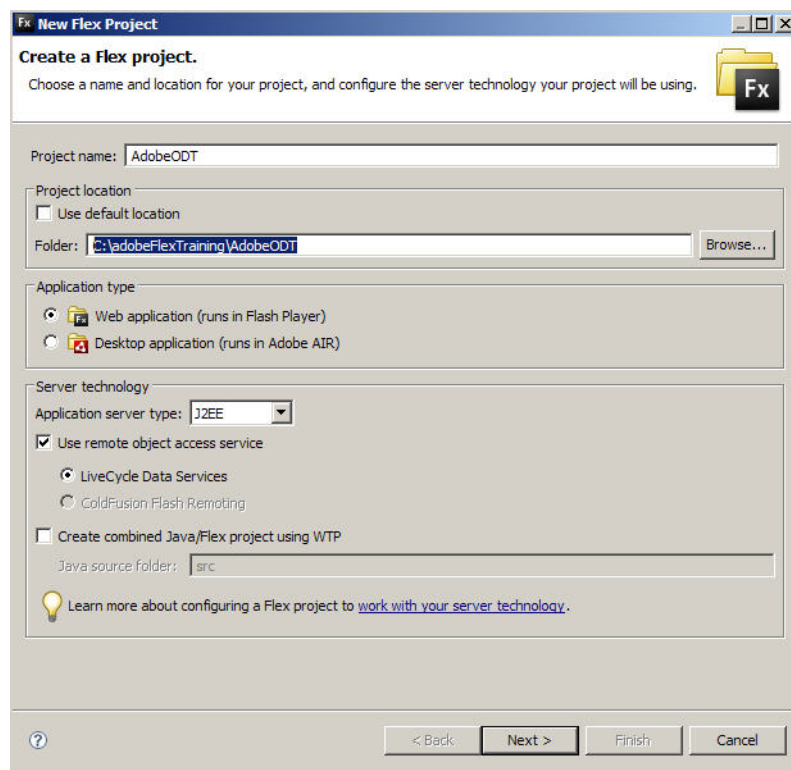
You should see the LiveCycle Data Services welcome page.

Recreate the project using LCDS

7. In Flex Builder **Navigator** view, right click on the **AdobeODT** project and select **Delete**. Make sure the **Do not delete contents** is selected.

8. Select **File > New > Flex Project**.
9. The project name is AdobeODT.
10. Uncheck the **Use default location** checkbox.
11. The project location folder is C:\adobeFlexTraining\AdobeODT
12. The application type is Web Application.
13. Under **Server Technology**, select J2EE as the application server type.
14. Check **Use remote object access service** and LiveCycle Data Services should be selected.

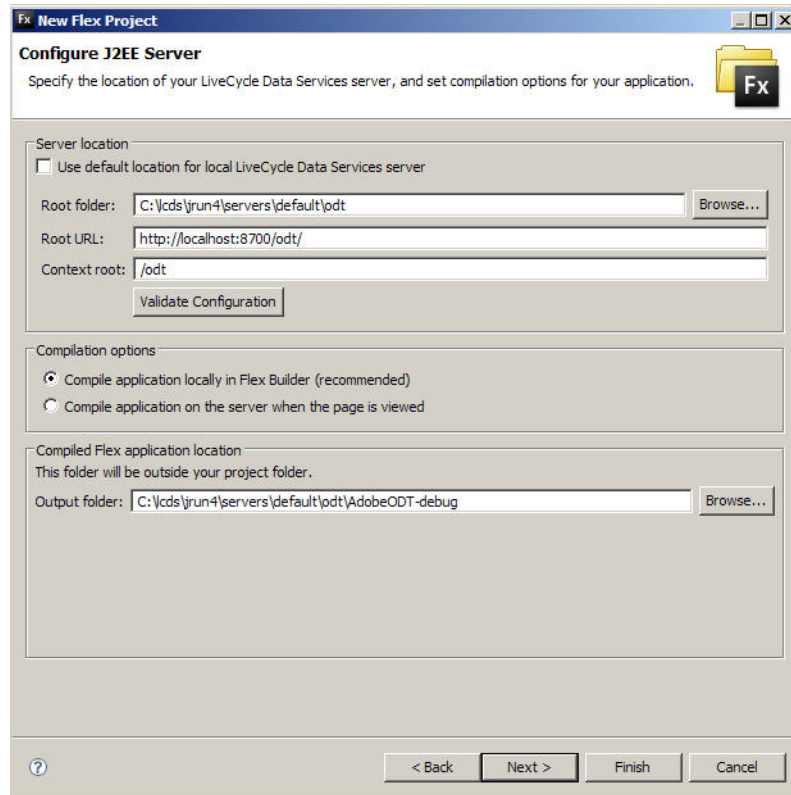
The dialog window should appear as follows:



15. Click **Next**.
16. Under Server location, uncheck Use default location for local LiveCycle Data Service server.
17. The root folder is C:\l\cds\jrun4\servers\default\odt.
18. The root url is <http://localhost:8700/odt/>.

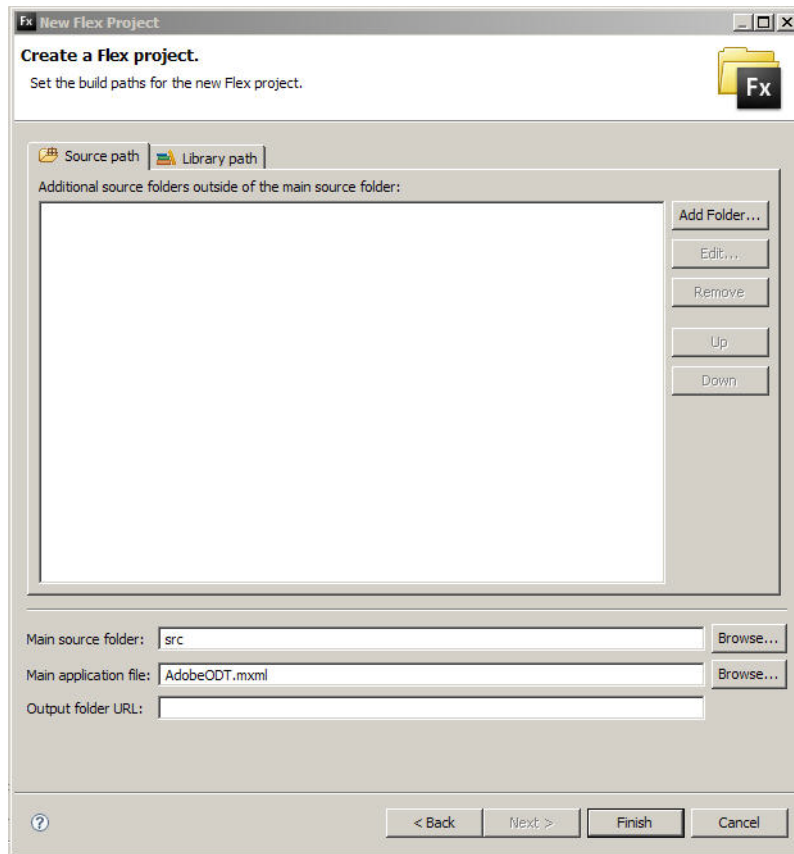
19. The context root is /odt.
20. Use the **Validate Configuration** button to validate your set up.
21. Under Compilation options, make sure **Compile application locally in Flex Builder** is selected.
22. Under Compile Flex application location, the output folder should be C:\l cds\jrun4\servers\default\odt\AdobeODT-debug.

The dialog box should appear as follows:



23. Click **Next**.
24. The main source folder is src.
25. The main application file is AdobeODT.xml.

The dialog box should appear as follows:



26. Click **Finish**.
27. Run **AdobeODT.mxml**.
28. The browser url should display <http://localhost:8700/odt/AdobeODT-debug/AdobeODT.html>. This indicates the LiveCycle Data server is being used.

Create value object class

29. In the **Navigator** view, right-click on the **src** folder of the **AdobeODT** project and select **New > Folder**.
30. The folder name is **vo**.
31. Click **Finish**,
32. Right-click on the **vo** folder and select **New > ActionScript Class**.
33. The name of the class is **Reservation**.

34. Keep all other defaults and click **Finish**.
35. Before the class definition, add a `Bindable` metadata tag.
36. After the class definition but before the constructor, create a public variable named `reservationId` datatyped as `int` having a value of 0 (zero);
`public var reservationId:int = 0;`
37. Create another public variable named `fullName` datatyped as `String` having no value.
`public var fullName:String = "";`
38. Create the following variable as public datatyped as `String` and having no value: `address`, `city`, `state`, `postalCode`, `phone`, `dateNeeded` and `options`.

Your class definition code should appear like the following:

```
[Bindable]
public class Reservation
{
    public var reservationId:int = 0;
    public var fullName:String = "";
    public var address:String = "";
    public var city:String = "";
    public var state:String = "";
    public var postalCode:String = "";
    public var phone:String = "";
    public var dateNeeded:String = "";
    public var options:String = "";

    public function Reservation()
    {
    }
}
```

39. Save the file.

Populate value object

40. In **ReservationForm.mxml**, locate the **Script** block.
41. After the last import statement, import the `Reservation` class.

```
import vo.Reservation;
```
42. After the last function in the `Script` block, create a new private function named `clearForm` that takes no parameters and returns `void`.
43. Within the function set the `text` property of the following controls to an empty string: `fullName`, `address`, `city`, `state`, `postalCode`, `phone` and `dateNeeded`.
44. Use the `removeAll()` method of the selectedOptions **ArrayCollection**.

Your code should appear as follows:

```
fullName.text = "";
address.text = "";
city.text = "";
state.text = "";
postalCode.text = "";
phone.text = "";
dateNeeded.text = "";
selectedOptions.removeAll();
```

45. Locate the **validateForm()** method.
46. After the code within the function, create a conditional to test if `vals` is equivalent to zero.

```
if (vals.length == 0){
}

```
47. If the conditional tests true, create and instantiate a local variable named `dataObj` datatyped as `Reservation`.

```
var dataObj:Reservation = new Reservation();
```

48. Assign `fullname.text` to `dataObj.fullName`.
49. Assign `address.text` to `dataObj.adress`.
50. Assign `city.text` to `dataObj.city`.
51. Assign `state.text` to `dataObj.state`.
52. Assign `postalcode.text` to `dataObj.postalCode`.
53. Assign `phone.text` to `dataObj.phone`.
54. Assign `dateNeeded.text` to `dataObj.dateNeeded`.
55. Use the `toString()` method of `selectedOptions` and assign it to `dataObj.options`.

Your code should appear as follows:

```
dataObj.reservationId = 0;
dataObj.fullName = fullname.text;
dataObj.address = address.text;
dataObj.city = city.text;
dataObj.state = state.text;
dataObj.postalCode = postalcode.text;
dataObj.phone = phone.text;
dataObj.dateNeeded = dateNeeded.text;
dataObj.options = selectedOptions.toString();
```

Send data by RemoteObject

56. After the **Script** block, create a `RemoteObject` call with an `id` property having a value of `hs` and a `destination` property having a value of `reservationService`.
57. Add a `fault` property with a value of `faultHandler` and pass `event` as the only parameter.

```
<mx:RemoteObject id="hs"
    destination="reservationService"
    fault="faultHandler(event)">
```

```
</mx:RemoteObject>
```

58. Between the **RemoteObject** tags create a method tag with a name property having a value of `doCreate` and a result property having a value of `resultHandler`. Pass `event` as the only parameter.

```
<mx:RemoteObject id="hs"
    destination="reservationService"
    fault="faultHandler(event)">

    <mx:method name="doCreate"
        result="resultHandler(event)"/>
</mx:RemoteObject>
```

59. Locate the **Script** block.
60. After the last import statement, import the `ResultEvent` class.
61. Import the `FaultEvent` class.
62. Import the `Alert` class.

```
import mx.rpc.events.ResultEvent;
import mx.rpc.events.FaultEvent;
import vo.Reservation;
import mx.controls.Alert;
```

63. Before the end of the **Script** block, create a private function named `resultHandler` that takes one parameter named `event` datatyped as `ResultEvent`. The function returns `void`.
64. Within the function invoke the `show()` method of the `Alert` class and pass two parameters:
- Your form was sent successfully.
 - Confirmation

Your code should appear like this:

```
private function resultHandler(event:ResultEvent):void{
    Alert.show("Your form was sent
successfully.", "Confirmation");
}
```

65. Before the end of the **Script** block, create a private function named `faultHandler` that takes one parameter named `event` datatyped as `FaultEvent`. The function returns `void`.
66. Within the function invoke the `show()` method of the `Alert` class and pass two parameters:
 - c. Your form was not sent successfully.
 - d. Error

```
private function faultHandler(event:FaultEvent):void{
    Alert.show("Your form was not sent
successfully.", "Error");
}
```

67. Locate the **validateForm()** function.
68. As the last line within the conditional statement, invoke the `doCreate()` method of `hs` call and pass `dataObj` as the only parameter.

```
hs.doCreate(dataObj);
```

69. Call the `clearForm()` function.

```
clearForm();
```

70. Locate and open the **Reservation.as** file in the **vo** folder.

71. After the package code within the curly brace, add a `RemoteClass` metadata tag. In parenthesis add an `alias` property having a value of `reservations.Reservation`.

```
package vo
{
    [RemoteClass(alias="reservations.Reservation")]
```

. . .

72. Save the file.

73. Return to **AdobeODT.mxml** and run.

Enter data into each field and select two options then submit. You should get an alert message with the status of your submission.

