

# Using Remote Object to Send Data to the Server

## Install ColdFusion 8

1. Browse to the url <http://www.adobe.com/products/coldfusion/> and click the **Download free trial** link.
2. Download ColdFusion 8 Developer Edition.
3. Install ColdFusion 8 by running `coldfusion-801-win.exe` on Windows or ColdFusion 8 Installer on Mac OSX.
  1. At the Serial Number screen, leave the serial number blank and select the 'Developer Edition' checkbox.
  2. At the Installer Configuration screen, ensure the 'Server' configuration is selected.
  3. At the Subcomponent Installation screen, select the components that you wish to install (none of the subcomponents are required for this tutorial).
  4. At the Configure Web Servers/Websites screen, select the 'Built-in web server (Development use only)' radio button.
  5. At the Enable RDS screen, click the 'Enable RDS' checkbox and enter your password.
  6. Complete the installation.

## Installing the Server Files

4. Unzip **ex9\_CF.zip** to a convenient location (referred to as `<zipfolder>` below).
5. Copy the folder 'reservations' from `<zipfolder>/wwwroot/` to `<cfinstall>/ColdFusion8/wwwroot/`.

6. Copy the folder 'reservationsdb' from <zipfolder>/db/ to <cfinstall>/ColdFusion8/db/.

## Starting and testing the Installation

7. Browse to this url: <http://localhost:8500/CFIDE/administrator/>.
8. Enter the administrator password you selected during the ColdFusion installation.
9. Select **Data Sources** in the ColdFusion administration page (under the **Data & Services** section in the left-hand menu).
10. Under **Add New Data Source** enter `reservations` as the **Data Source Name** and select `Apache Derby Embedded` as the **Driver**, then click **Add**.
11. Click **Browse Server** and browse to the `reservationsdb` folder you created in Step 6, then click **Submit** to add a new data source.
12. Browse to this url: <http://localhost:8500/reservations/verify.cfm>

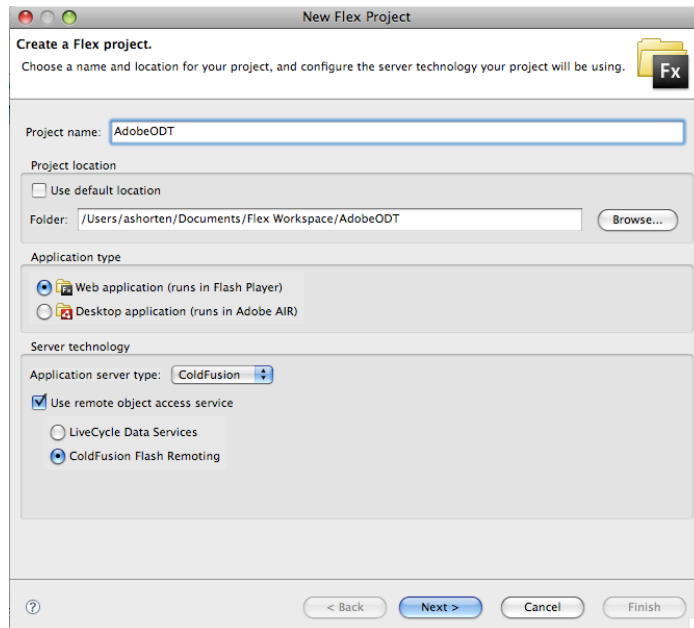
You should see a table listing some example reservations.

## Recreate the project using ColdFusion

14. In Flex Builder **Navigator** view, right click on the **AdobeODT** project and select **Delete**. Make sure the **Do not delete contents** is selected.
15. Select **File > New > Flex Project**.
16. The project name is `AdobeODT`.
17. Uncheck the **Use default location** checkbox.

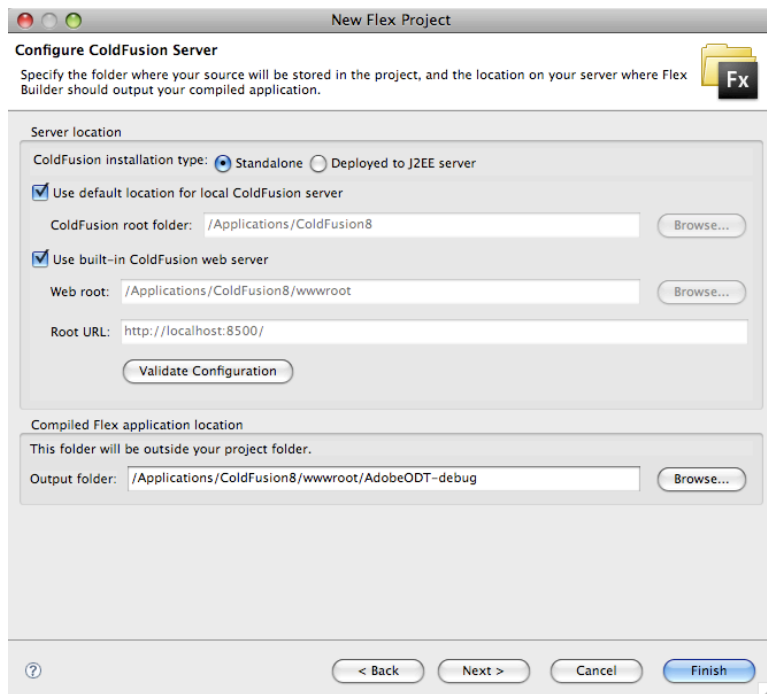
18. Select the project location used in previous tutorials.
19. The application type is Web Application.
20. Under **Server Technology**, select ColdFusion as the application server type.
21. Check **Use remote object access service** and ColdFusion Flash Remoting should be selected.

The dialog window should appear as follows:



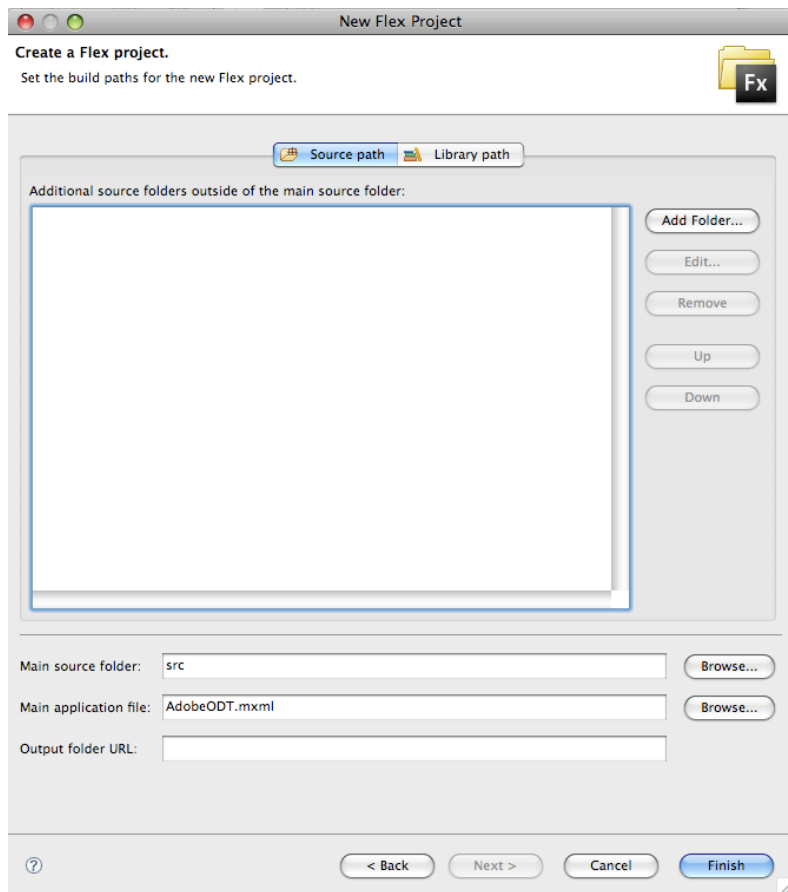
22. Click **Next**.
23. The default values should work with the standalone ColdFusion 8 server.
24. Use the **Validate Configuration** button to validate your set up.
25. Under Compile Flex application location, the output folder should be <install>ColdFusion8/wwwroot/AdobeODT-debug.

The dialog box should appear as follows:



26. Click **Next**.
27. The main source folder is `src`.
28. The main application file is `AdobeODT.mxml`.

The dialog box should appear as follows:



29. Click **Finish**.
30. Run **AdobeODT.mxml**.
31. The browser url should display <http://localhost:8500/odt/AdobeODT-debug/AdobeODT.html>. This indicates ColdFusion 8 server is being used.

## Create value object class

32. In the **Navigator** view, right-click on the **src** folder of the **AdobeODT** project and select **New > Folder**.
33. The folder name is `vo`.
34. Click **Finish**,
35. Right-click on the **vo** folder and select **New > ActionScript Class**.
36. The name of the class is `Reservation`.
37. Keep all other defaults and click **Finish**.
38. Before the class definition, add a `Bindable` metadata tag.
39. 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;
```
40. Create another public variable named `fullName` datatyped as `String` having no value.  

```
public var fullName:String = "";
```
41. 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()
{
}
}
```

42. Save the file.

## Populate value object

43. In **ReservationForm.mxml**, locate the **Script** block.
44. After the last import statement, import the **Reservation** class.

```
import vo.Reservation;
```

45. After the last function in the Script block, create a new private function named **clearForm** that takes no parameters and returns void.

46. Within the function set the `text` property of the following controls to an empty string: `fullName`, `address`, `city`, `state`, `postalcode`, `phone` and `dateNeeded`.
47. 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();
```

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

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

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

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

51. **Assign** `fullName.text` to `dataObj.fullName`.
52. **Assign** `address.text` to `dataObj.adress`.
53. **Assign** `city.text` to `dataObj.city`.
54. **Assign** `state.text` to `dataObj.state`.
55. **Assign** `postalcode.text` to `dataObj.postalCode`.
56. **Assign** `phone.text` to `dataObj.phone`.
57. **Assign** `dateNeeded.text` to `dataObj.dateNeeded`.
58. **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

59. After the **Script** block, create a `RemoteObject` call with an `id` property having a value of `hs`, a `destination` property having a value of `ColdFusion` and a `source` property having a value of `reservations.Reservation`.
60. Add a `fault` property with a value of `faultHandler` and pass `event` as the only parameter.

```
<mx:RemoteObject id="hs"
destination="ColdFusion" source="reservations.ReservationService"
fault="faultHandler(event)">
</mx:RemoteObject>
```

61. 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="ColdFusion" source="reservations.ReservationService"
fault="faultHandler(event)">
<mx:method name="doCreate"
result="resultHandler(event)"/>
</mx:RemoteObject>
```

62. Locate the **Script** block.

63. After the last import statement, import the `ResultEvent` class.
64. Import the `FaultEvent` class.
65. Import the `Alert` class.

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

66. 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`.
67. Within the function invoke the `show()` method of the `Alert` class and pass two parameters:
  1. Your form was sent successfully.
  2. Confirmation

Your code should appear like this:

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

68. 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`.

69. Within the function invoke the `show()` method of the `Alert` class and pass two parameters:

3. Your form was not sent successfully.

4. Error

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

70. Locate the **validateForm()** function.

71. 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);
```

72. Call the `clearForm()` function.

```
clearForm();
```

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

74. 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")]
```

- . . .
75. Save the file.
  76. 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.

