

Using Remote Object to Send Data to the Server

Install ColdFusion 9

1. Browse to the url <http://www.adobe.com/products/coldfusion/> and click the **Get the Trial** link.
2. Download ColdFusion 9 Developer Edition.
3. Install ColdFusion 8 by running `coldfusion_9_WWE_win.exe` on Windows or ColdFusion 9 Installer on Mac OS X.
 - At the Serial Number screen, leave the serial number blank and select the 'Developer Edition' checkbox.
 - At the Installer Configuration screen, ensure the 'Server' configuration is selected.
 - At the Subcomponent Installation screen, select the components that you wish to install (none of the subcomponents are required for this tutorial).
 - At the Configure Web Servers/Websites screen, select the 'Built-in web server (Development use only)' radio button.
 - At the Enable RDS screen, click the 'Enable RDS' checkbox and enter your password.
 - Complete the installation.

Installing the Server Files

1. Unzip **ex9_CF.zip** to a convenient location (referred to as <zipfolder> below).
2. Copy the folder 'reservations' from <zipfolder>/wwwroot/ to <cfinstall>/ColdFusion9/wwwroot/.
3. Copy the folder 'reservationsdb' from <zipfolder>/db/ to <cfinstall>/ColdFusion8/db/.

Starting and testing the Installation

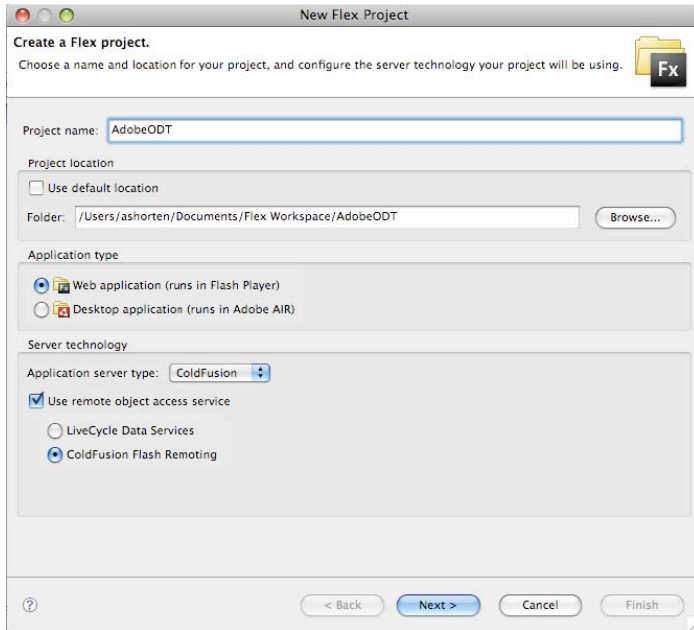
1. Browse to this url: `http://localhost:8500/CFIDE/administrator/`.
2. Enter the administrator password you selected during the ColdFusion installation.
3. Select `Data Sources` in the ColdFusion administration page (under the **Data & Services** section in the left-hand menu).
4. Under **Add New Data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

You should see a table listing some example reservations.

Recreate the project using ColdFusion

1. In Flex Builder **Navigator** view, right click on the **AdobeODT** project and select **Delete**. Make sure the **Do not delete contents** is selected.
2. Select **File > New > Flex Project**.
3. The project name is `AdobeODT`.
4. Clear the **Use default location** check box.
5. Select the project location used in previous tutorials.
6. The application type is `Web Application`.
7. Under **Server Technology**, select `ColdFusion` as the application server type.
8. Check **Use remote object access service** and `ColdFusion Flash Remoting` should be selected.

The dialog window should appear as follows:



9. Click **Next**.

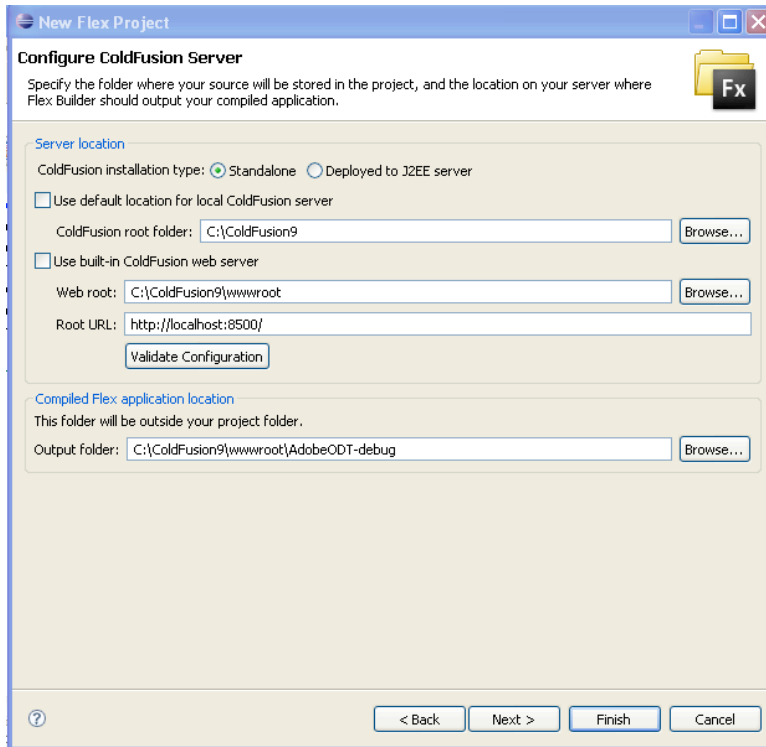
10. Change the settings on the Configure ColdFusion Server dialog window as follows.:

- **ColdFusion root folder:** <install>ColdFusion9
- **Web root:** <install>ColdFusion9\wwwroot
- **Root url:** http://localhost:8500

11. Use the **Validate Configuration** button to validate your set up.

12. Under Compile Flex application location, the output folder should be <install>ColdFusion9/wwwroot/AdobeODT-debug.

The dialog box should appear as follows:

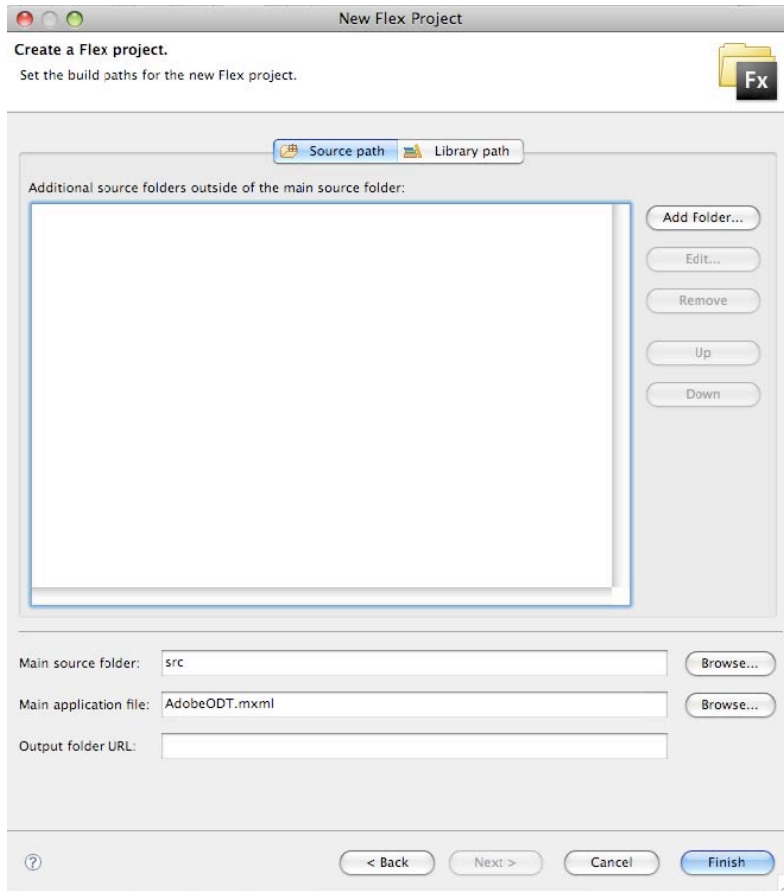


13. Click **Next**.

14. The main source folder is `src`.

15. The main application file is `AdobeODT.mxml`.

The dialog box should appear as follows:



16. Click **Finish**.

17. Run **AdobeODT.mxml**.

18. The browser URL should display `http://localhost:8500/odt/AdobeODT-debug/AdobeODT.html`. This indicates ColdFusion 9 server is being used.

Create value object class

1. In the **Navigator** view, right-click on the **src** folder of the **AdobeODT** project and select **New > Folder**.
2. The folder name is `vo`.
3. Click **Finish**,
4. Right-click on the **vo** folder and select **New > ActionScript Class**.
5. The name of the class is `Reservation`.
6. Keep all other defaults and click **Finish**.

7. Before the class definition, add a `Bindable` metadata tag.
8. 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;
```
9. Create another public variable named `fullName` datatyped as `String` having no value.

```
public var fullName:String = "";
```
10. 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()
    {
    }
}
```

11. Save the file.

Populate value object

1. In the **ReservationForm.mxml** component locate the **Script** block.
2. After the last `import` statement, import the `Reservation` class.

```
import vo.Reservation;
```

3. After the last function in the Script block, create a new private function named `clearForm` that takes no parameters and returns void.
4. Within the function set the `text` property of the following controls to an empty string: `fullname`, `address`, `city`, `state`, `postalcode`, `phone` and `dateNeeded`.
5. 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();
```

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

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

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

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

9. Assign the following:

- `fullname.text` to `dataObj.fullName`.
- `address.text` to `dataObj.adress`.
- `city.text` to `dataObj.city`.
- `state.text` to `dataObj.state`.

- `postalcode.text` to `dataObj.postalCode`.
- `phone.text` to `dataObj.phone`.
- `dateNeeded.text` to `dataObj.dateNeeded`.

10. 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

1. 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`.
2. 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>
```

3. 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>
```

4. Locate the **Script** block.
5. After the last import statement, import the `ResultEvent` class.
6. Import the `FaultEvent` class.
7. Import the `Alert` class.

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

8. 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`.
9. 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");
}
```

10. 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`.

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

- Your form was not sent successfully.
- Error

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

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

13. 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);
```

14. Call the `clearForm()` function.

```
clearForm();
```

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

16. 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")]  
  
. . .
```

17. Save the file.

18. 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.

Contact Information

Full Name:

Address:

City:

State:

Phone:

Date needed:

Confirmation

Your form was sent successfully

Selected Options

Selected Options: Microphone
 Dance Floor