Sales & Technical Enablement Workshop

Developing XPages - Part II

Lotus Channel Technical Sales

Matthias Schneider
IBM Corporation
March 2010

Table of Contents

1 Time Estimates

The IT Professional should be able to complete this whole lab in 90 minutes.

2 Purpose and Description

You will learn how page flow works in XPages and how to code actions behind action buttons. We will demonstrate how to provide data to the different controls on the XPage and how to enable type-ahead functionality in XPages. You will also learn how to show data from a Notes view in XPages. how to use the XPages scope contexts with advanced data binding and how to propagate data from one form to another.

Further, you will learn how to show a subset of view data only, how to add data validation to the forms. You will also use the Java perspective to add a page icon to your page.

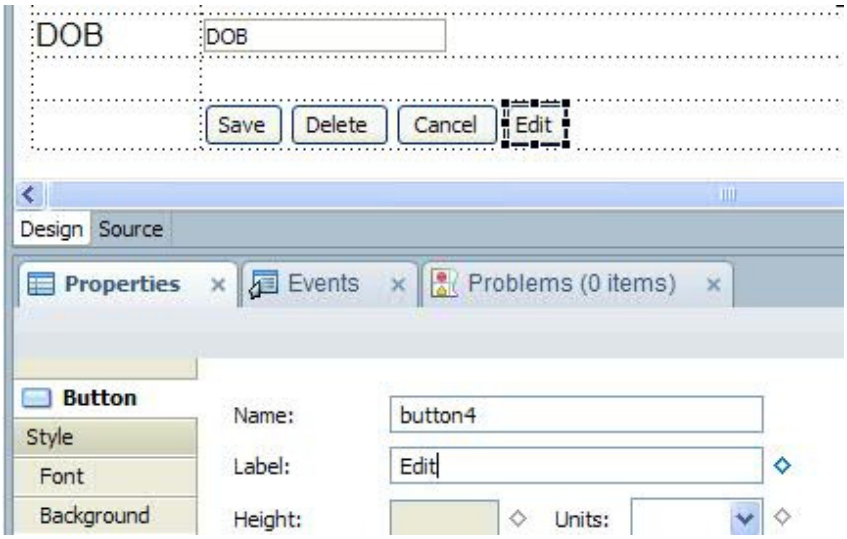
You will complete the custom controls from last exercise, provide static and dynamic lookups for the possible values for these controls, complete the search page and show search results in the result page.

3 Detailed Steps

Implement Page Navigation

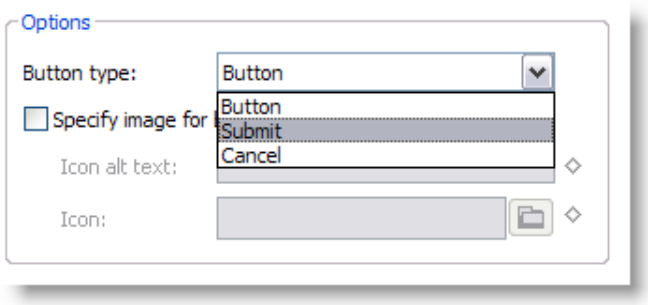
1. Add the **styles.css** to the **profileForm** Custom Control (**Hint**: **Style** tab in the **Properties** dialog).

2. Add 4 button controls into the second column of the last table row and label them as:
   - Save
   - Delete
   - Cancel
   - Edit



For the Save button:

3. Change the button type to **Submit.**

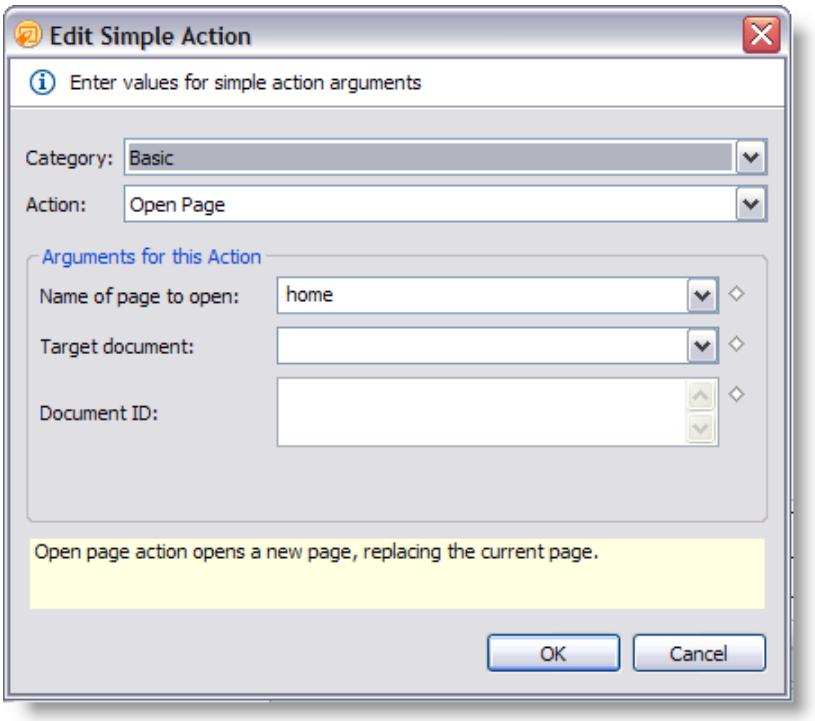


4. Set the class to **profilesButtonSubmit** on the Style tab. Create a computed “Visible” property with this formula:

```
document1.isEditable()
```

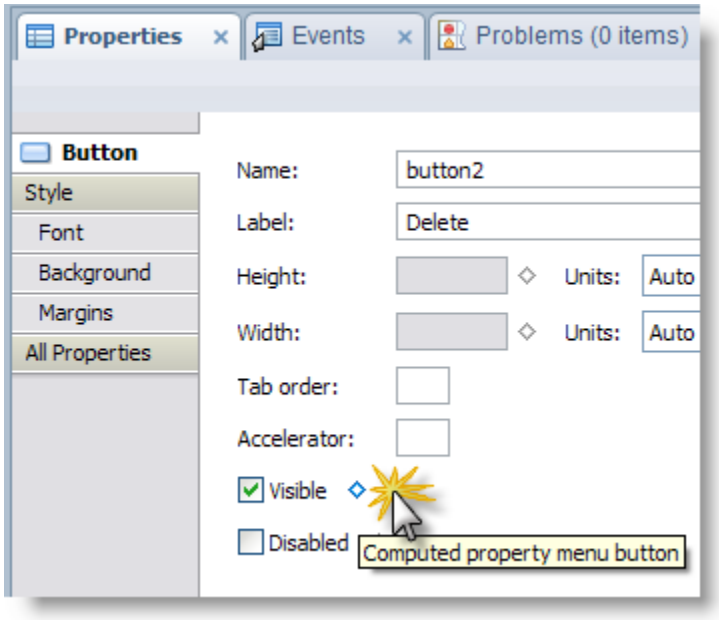
5. Add a Simple Action for the Mouse **onclick** event on the **Events** tab.

6. Set the Category to **Basic**, the Action to **Open Page** and select the **home** page as the target.



For the Delete button:

7. Add a computed **Visible** property.

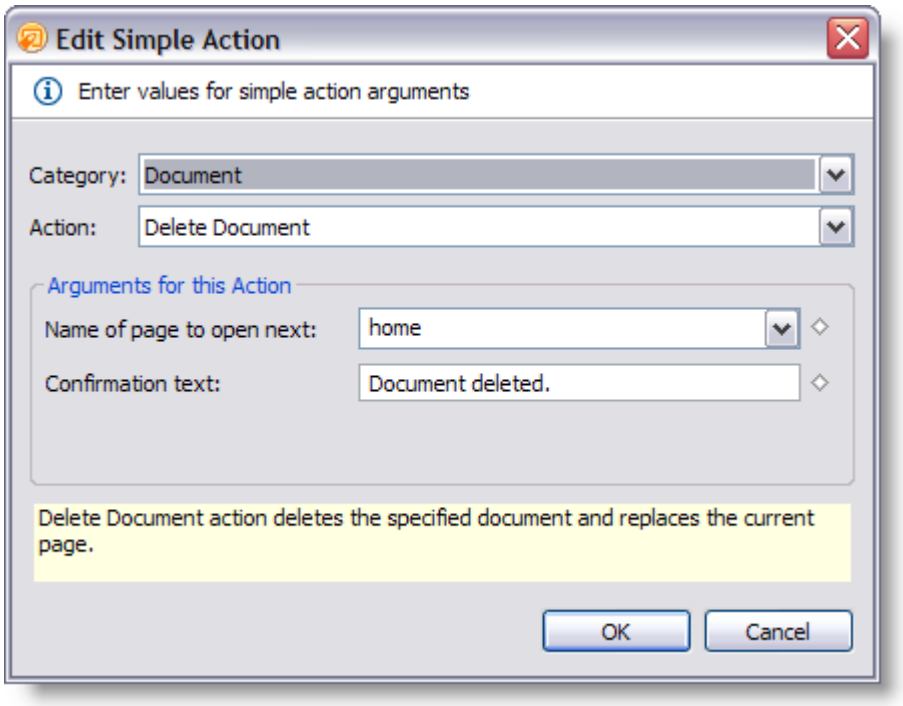


8. In the JavaScript Editor add this formula:

```
!document1.isNewNote()
```

9. Set the class to **profilesButtonCommand.**

10. Add a Simple Action to the Mouse **onclick** event that deletes the document and returns to the **home** page with some custom confirmation text.



For the Cancel button:

11. Change the button type to **Cancel.**

12. Change the class to **profilesButtonCancel.**

13. Add a Simple Action to the Mouse onclick event that opens the **home** page.

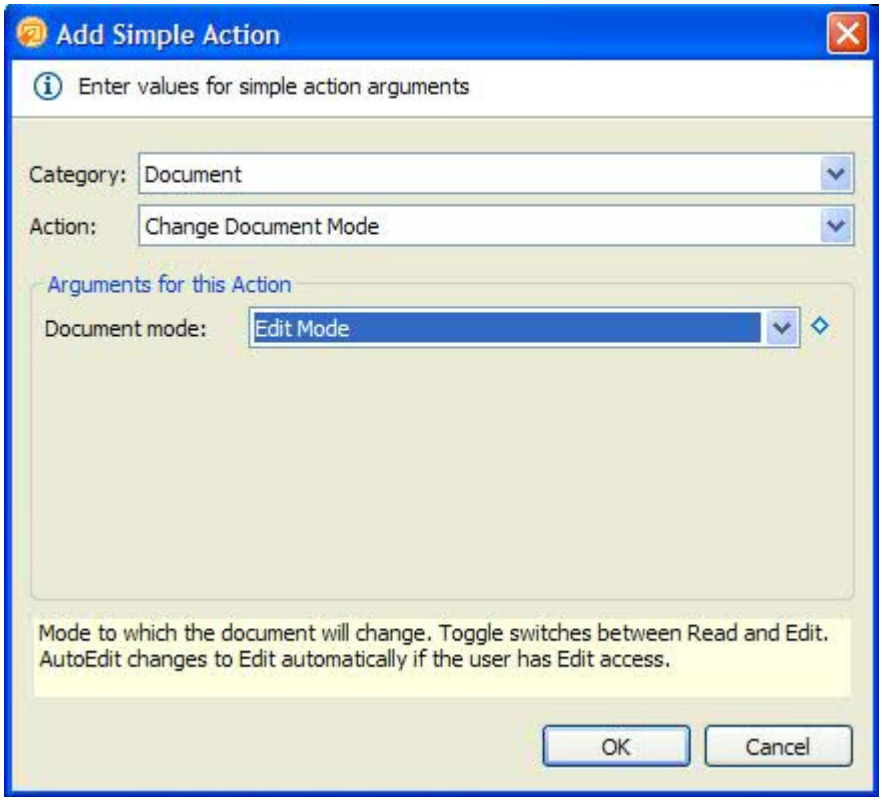
For the Edit button:

14. Apply the class **profilesButtonCommand.**

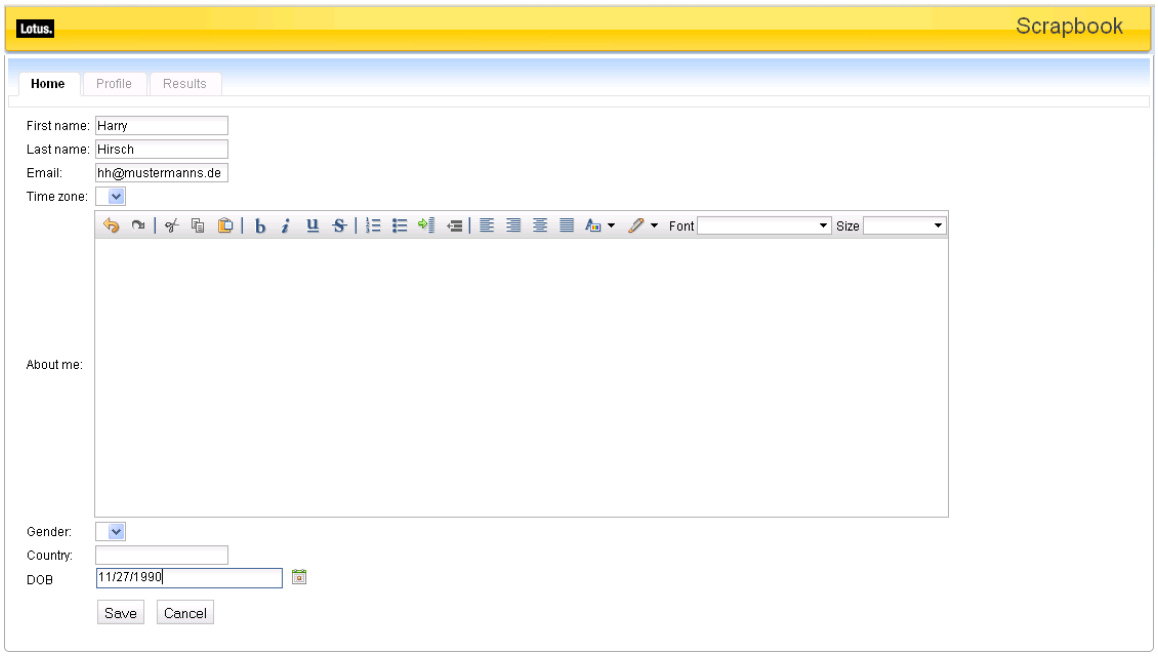
15. Create a computed “Visible” property with this formula:

```
!document1.isEditable()
```

16. Add a Simple Action to the Mouse **onclick** event to **Change Document Mode** to **Edit Mode**.



17. Save the custom control, preview the **profile** XPage in the browser, then enter and save a document.

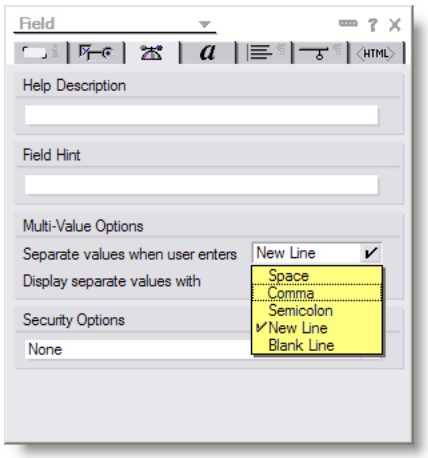


Implement Data Lookups

18. Create a Notes form named **keywords**.

19. Add the following fields:
    - Name (Text)
    - Values (Text, Allow Multiple Values)

20. Select “New Line” as the ONLY delimiter for both options below.



21. Create a Notes view called **keywordLookup.**
The selection formula will be:
SELECT form=”keywords”

Change the first column to show the **Name** field.
Make it sorted **Ascending.**

22. Preview the form in your Notes client and create a document with name set to **TimeZones.**

23. Add the text from **C:\MyLabFiles\XPages\codeSnippets\TimeZones.txt** to the Values field.

TimeZones

(GMT-12:00) International Date Line West
(GMT-11:00) Midway Island, Samoa
(GMT-10:00) Hawaii
(GMT-09:00) Alaska
(GMT-08:00) Pacific Time (US & Canada)
(GMT-07:00) Mountain Time (US & Canada)
(GMT-06:00) Central Time (US & Canada)

24. Save the document – you can hit **CTRL+S** for that.

25. Preview the form again and create another document with name set to **Countries**.

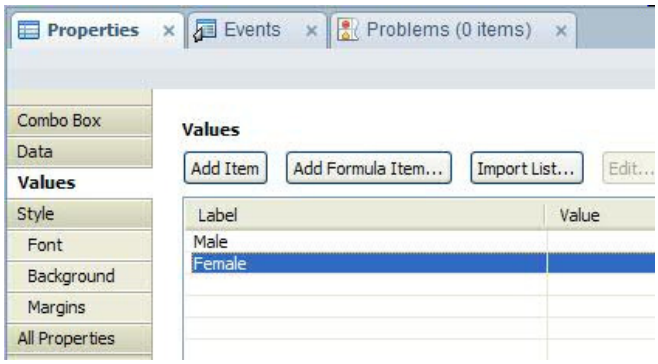
26. Add the text from **C:\MyLabFiles\XPages\codeSnippets\Countries.txt** to the Values field.

Name Countries
Values Afghanistan
Akrotiri
Albania
Algeria
American Samoa
Andorra
Angola
Anguilla
Antarctica
Antigua and Barbuda
Argentina

27. Save the document.

28. In the **profileForm** custom control select the **Gender** combo box.

29. In the **Values** tab click the **Add Item** button twice and amend the lines of text to **Male** and **Female.**

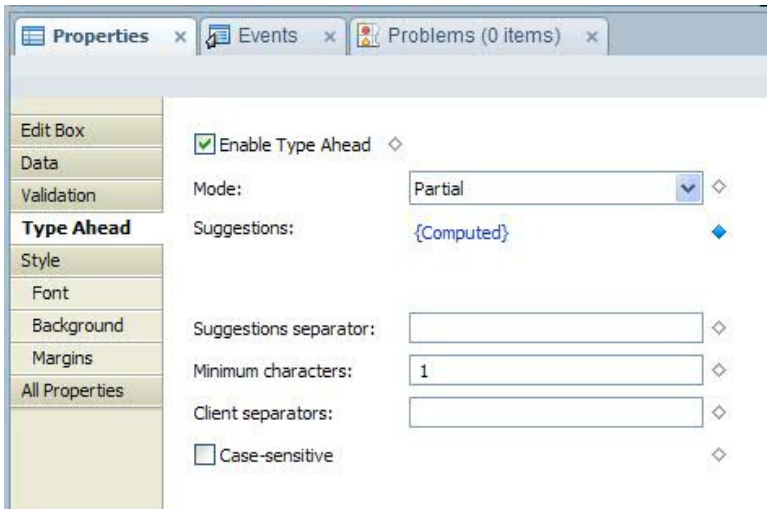


30. Select the **Country** edit box control. On the **Type Ahead** tab check the **Enable Type Ahead** box and make the Mode **Partial**.

31. Make the **Suggestions** a computed value and add the following formula (Server-side JavaScript):

```
@DbLookup(@DbName(), "keywordLookup", "Countries", "Values")
```

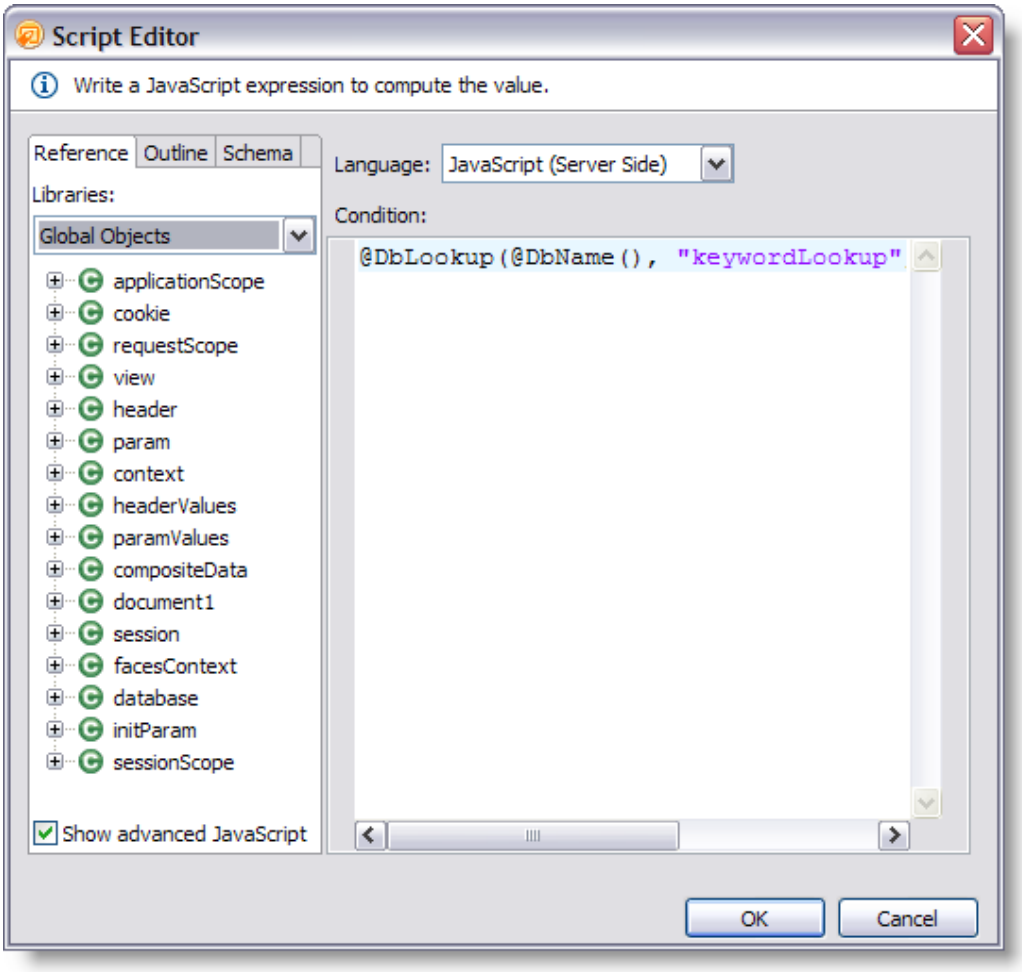
32. Uncheck the Case-sensitive checkbox.



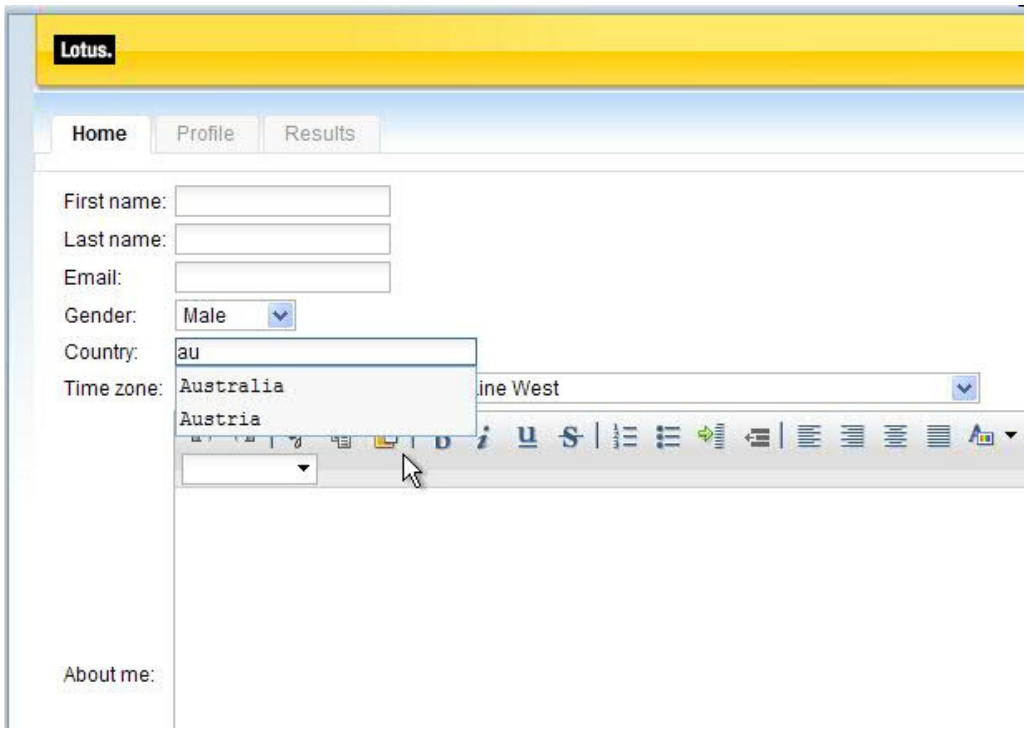
33. Select the **TimeZone** combo box control.

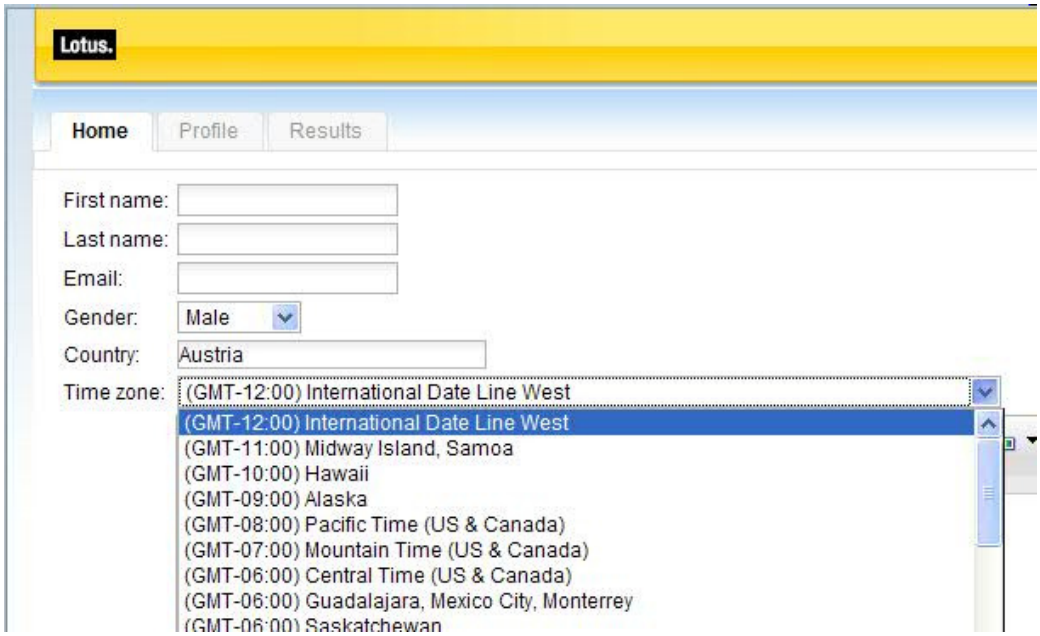
34. On the **Values** tab click on **Add Formula Item** and add this formula:

```
@DbLookup(@DbName(), "keywordLookup", "TimeZones", "Values")
```



35. Save the **profileForm** custom control and preview the **profile** XPage in the browser or Notes client.





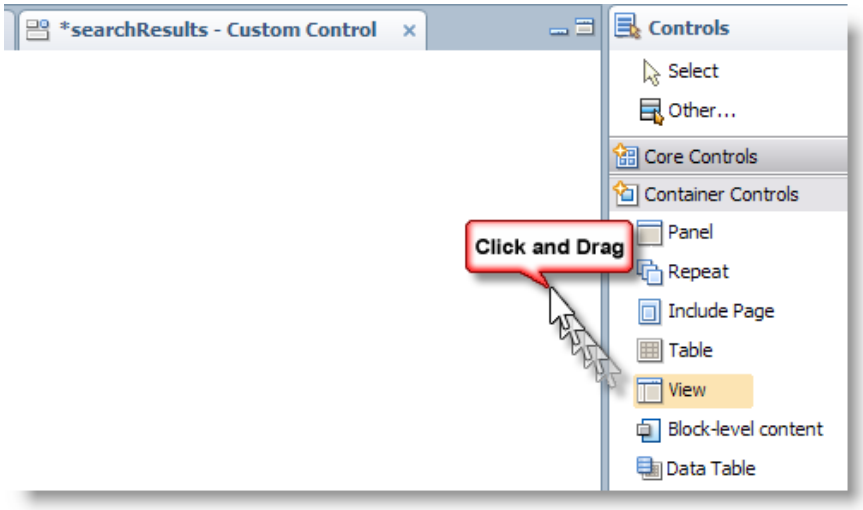
Work with Data Sources

34. Open the **searchResults** custom control and remove any temporary text that you added in a previous exercise.

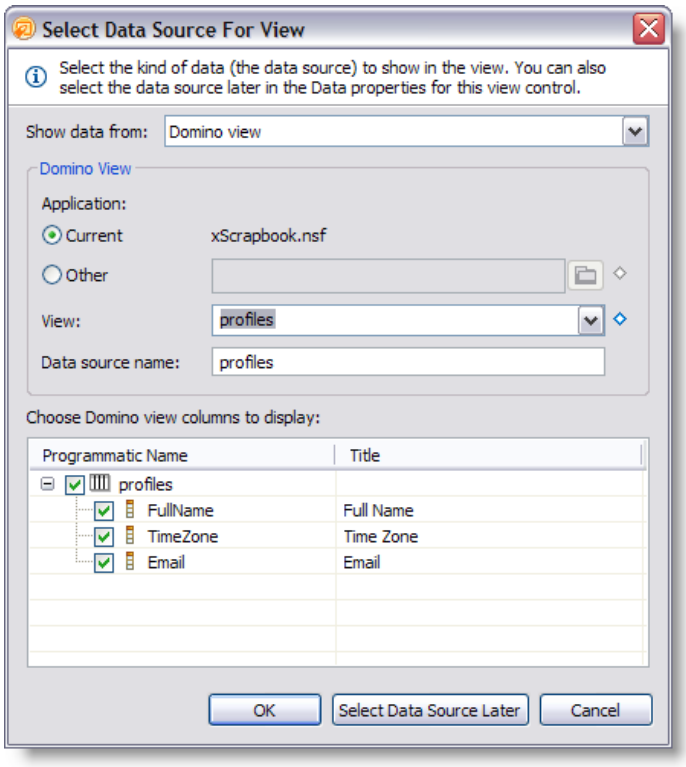
35. Add the **styles.css** resource to the custom control.

36. Press **Enter** three times to add some lines at the top of the control (we will use this area later).

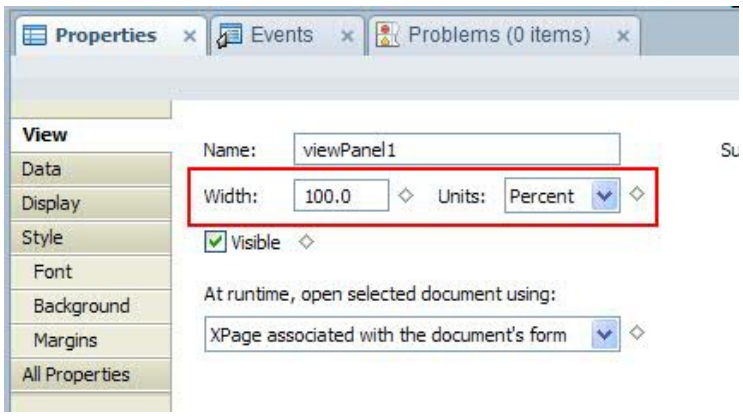
37. Drag a **View** control from the **Container Controls** palette onto the page canvas:



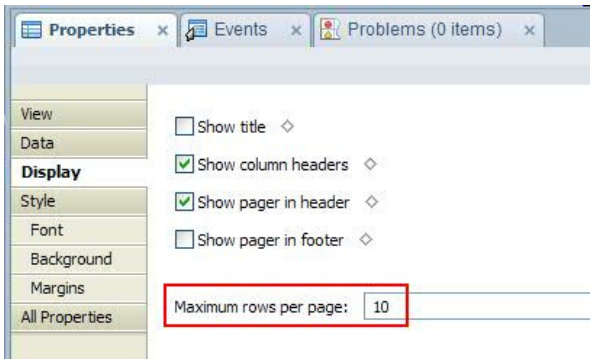
38. Select **profiles** as the view option in the helper dialog and click **OK.**



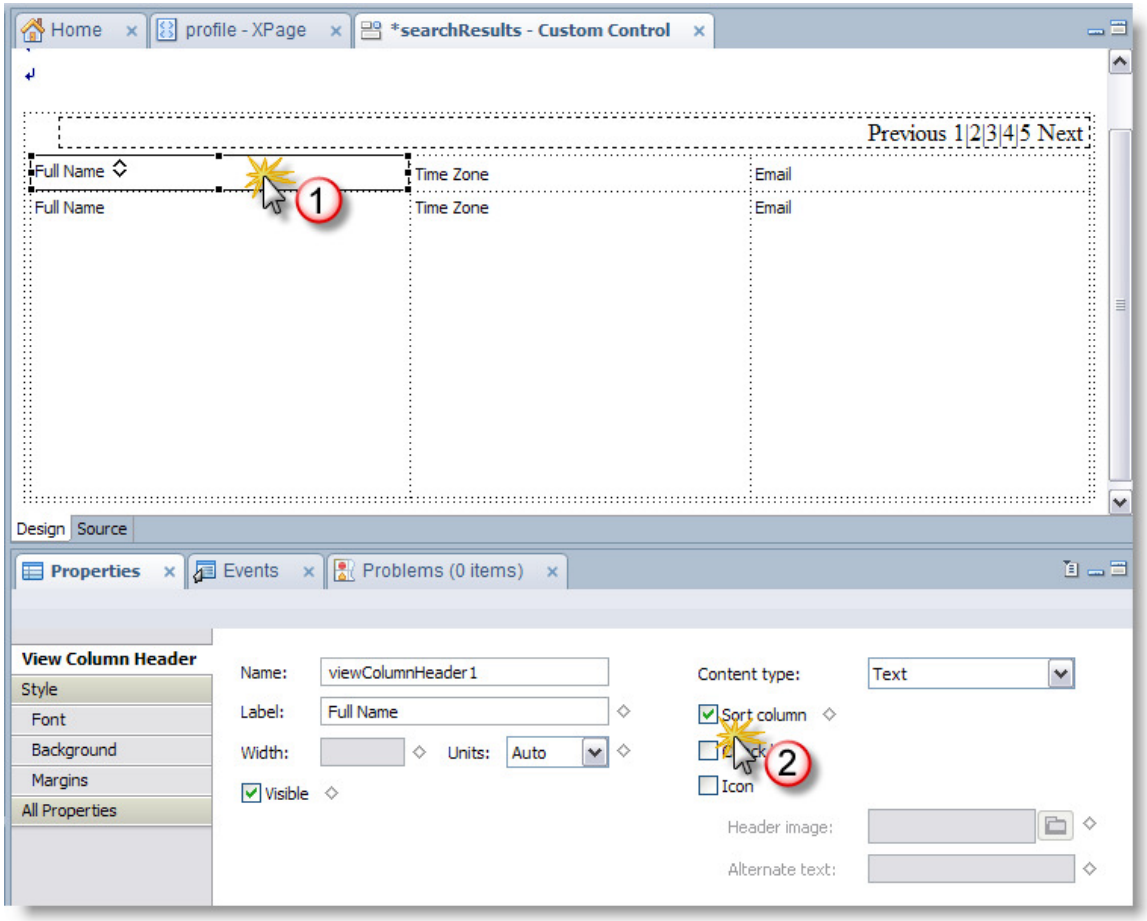
39. Click on the **View** tab in the Properties dialog and set the width to 100%:



40. On the **Display** tab set the maximum rows to 10.

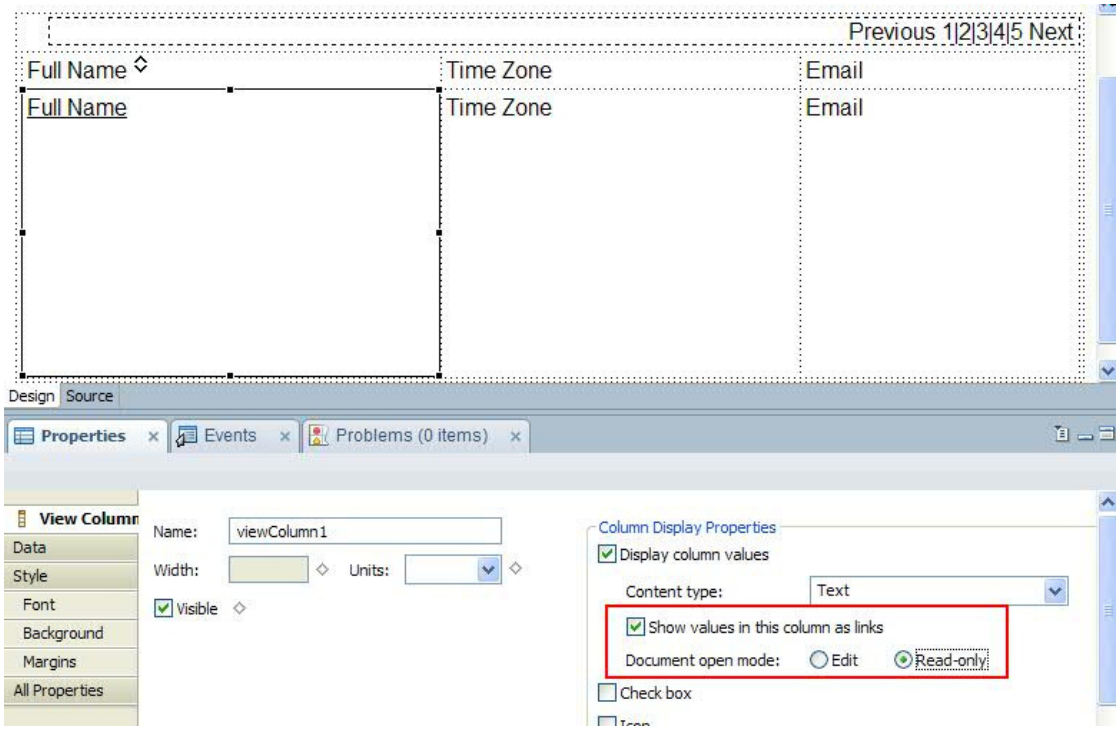


41. Click on the **FullName** column header. Check the **Sort column** option in the **View Column Header** tab.

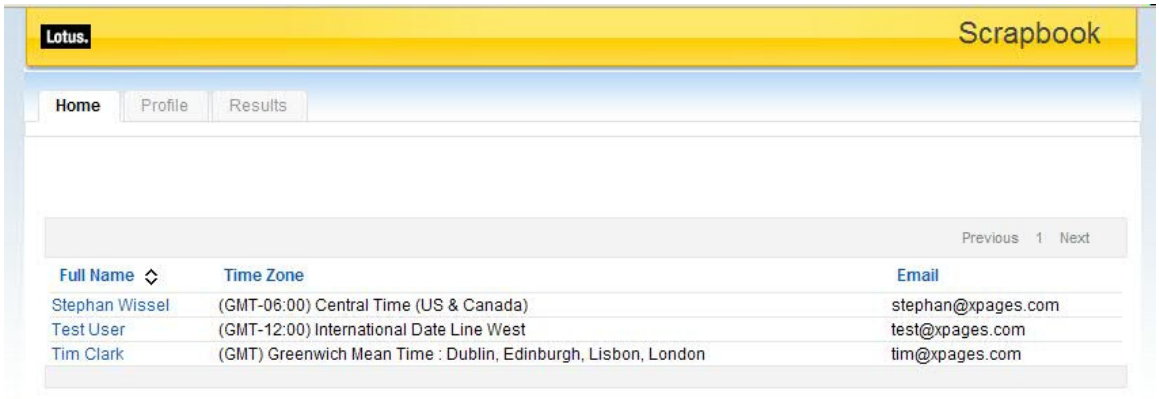


42. Click on the **FullName** column.
    Enable **Show value in this column as links**.

43. Select the **Read-only** option for the **Document open mode**.



44. Save the **searchResults** control and preview the **results** XPage in the browser or Notes client.

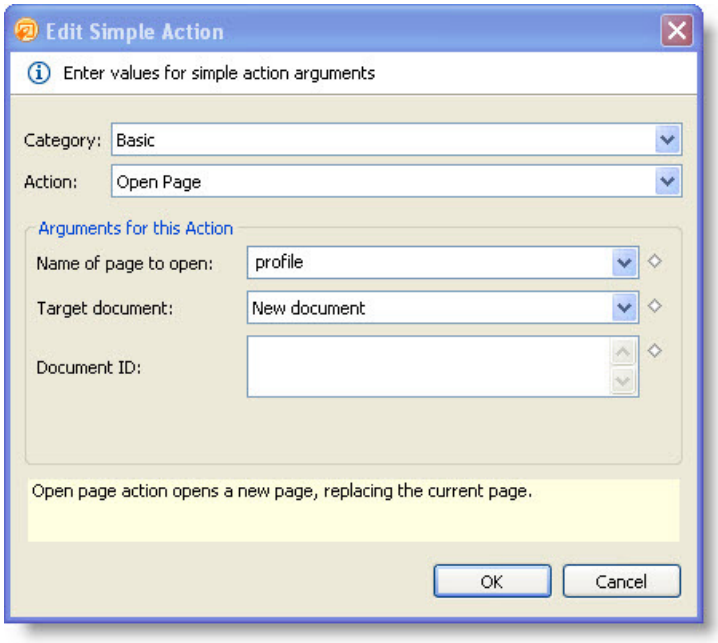


Make Use of Scoped Variables

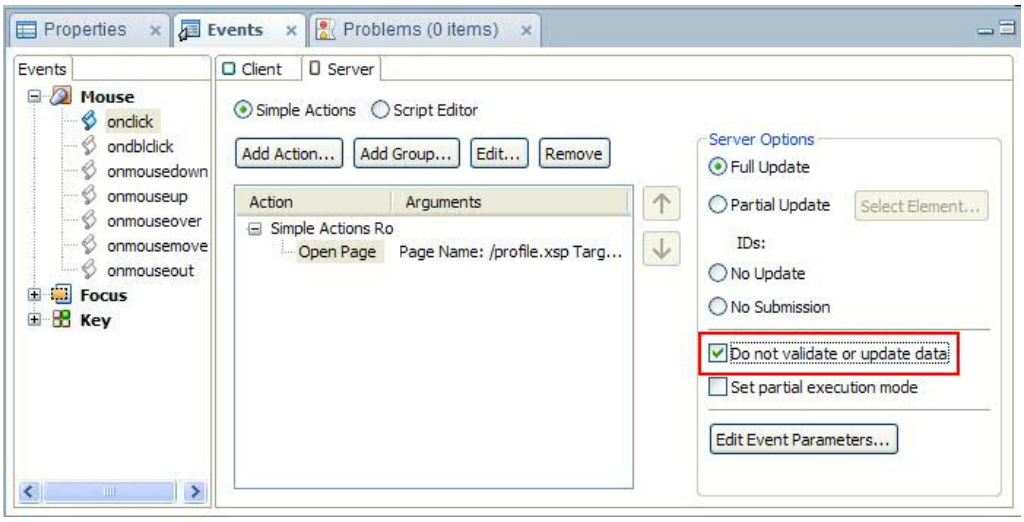
45. Open the **searchForm** custom control in the XPages editor and remove any temporary text you may have added in a previous exercise.

46. Add a button labeled **New Profile**, map it to the style **profilesButtonCommand**. (**Hint** – you will need to add the stylesheet styles.css to the page.)

47. Configure the **onclick** event of the button to open a new **profile** document.



48. Check the **Do not validate or update data** box in the **Server Options** pane.



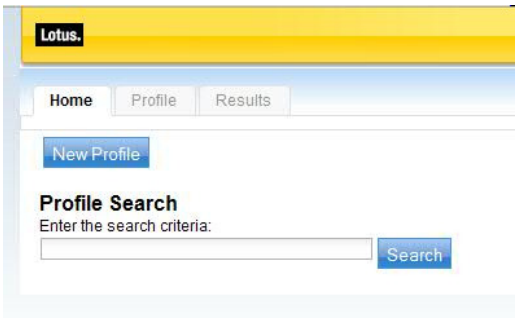
49. Add a label under the button and label it **Profile Search**.
    Make it bold and increase the font size.
    (**Hint:** Look in **Font** tab under **Style** in the **Properties** dialog.)

50. Add another label underneath the first and label it **Enter the search criteria:**.

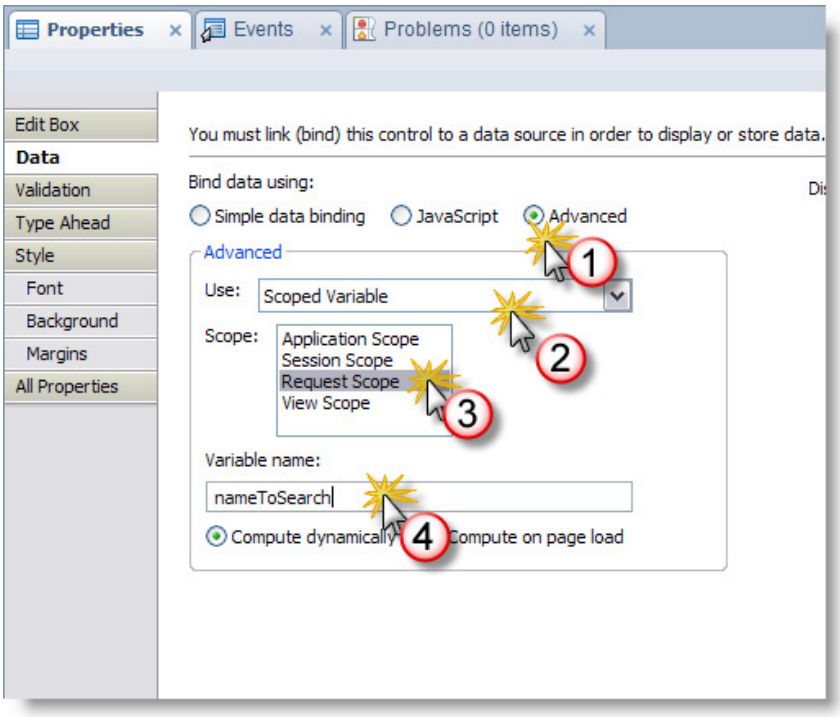
51. Add an edit box underneath the last label. Name it **searchBox** and stretch it in Designer to about **250px**.

52. Add a button labeled **Search** next to the edit box, and set the class to **profilesButtonCommand**.

53. Save the custom control and preview the **home** XPage in your browser or Notes client.



54. In the **searchForm** custom control, add an **Advanced** data binding to the edit box:
    - Scoped Variable: requestScope
    - Variable Name: nameToSearch



55. Select the edit box and change to the type-ahead tab.

56. Enable type-ahead, select **partial** mode and specify a computed suggestion list with the following formula:

```
@DbColumn(@DbName(), "profiles", 0)
```

57. Uncheck the **Case-sensitive** box.

58. Add an **Simple Action** to the **Search** button to **Execute Script** with this formula:

```
sessionScope.nameToSearch = requestScope.nameToSearch
```

59. Add a second **Simple Action** to **Open Page** and select **results.**

60. Save the custom control.

61. Open the **searchResults** custom control.

62. Add a **Computed Field** control (from Core Controls) at the top. Name it **resultsComputedField** and map it to the styles class **searchInfo**.

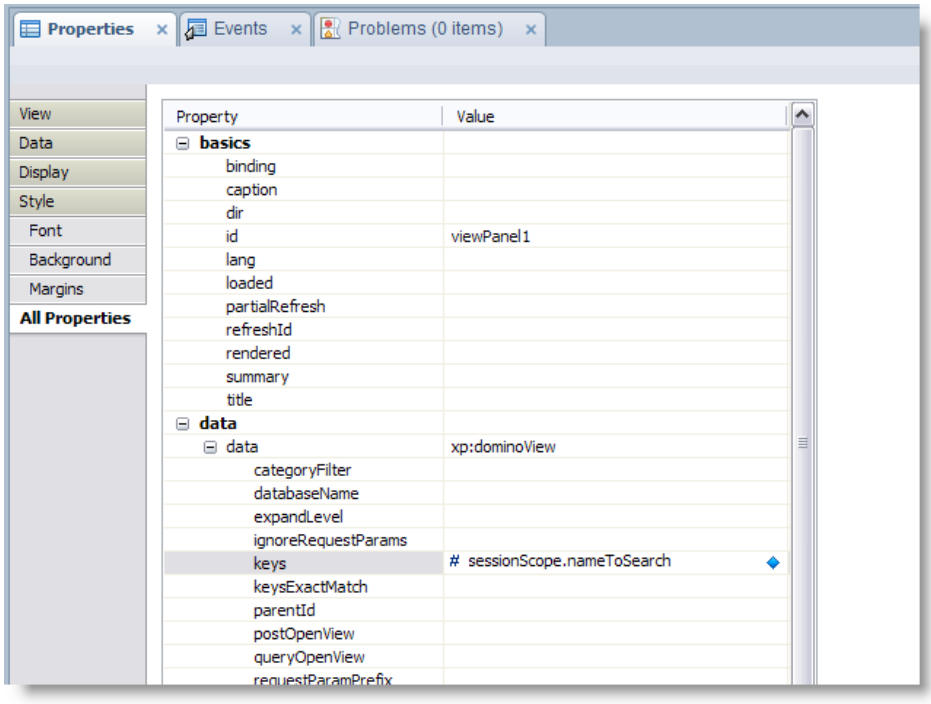
63. On the **Value** tab of the field, select **JavaScript** and add the following script fragment:

```
var criteria = sessionScope.nameToSearch;
var title = "Profile search results for: ";
if(null != criteria && criteria != ""){
  title += criteria;
}
else{
 title = "No search criteria provided.";
}
return title;
```

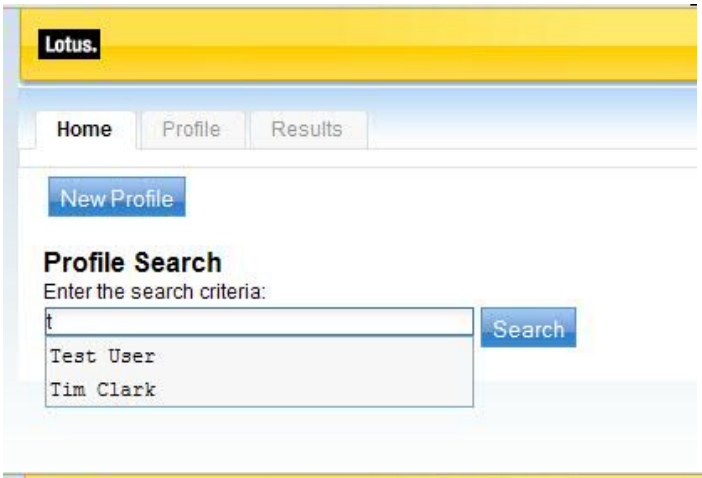
64. Select the **View** control (viewPanel1 - use Outline to ensure you have the correct element selected) and click on the **All Properties** tab.

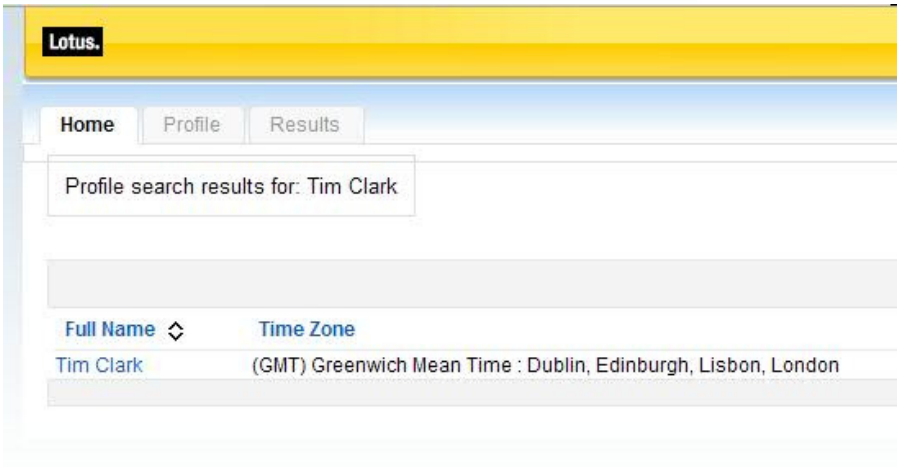
65. Expand the **data\data** entry and add the following computed expression for **keys**:

```
sessionScope.nameToSearch
```



66. Save the custom control and preview the home XPage.



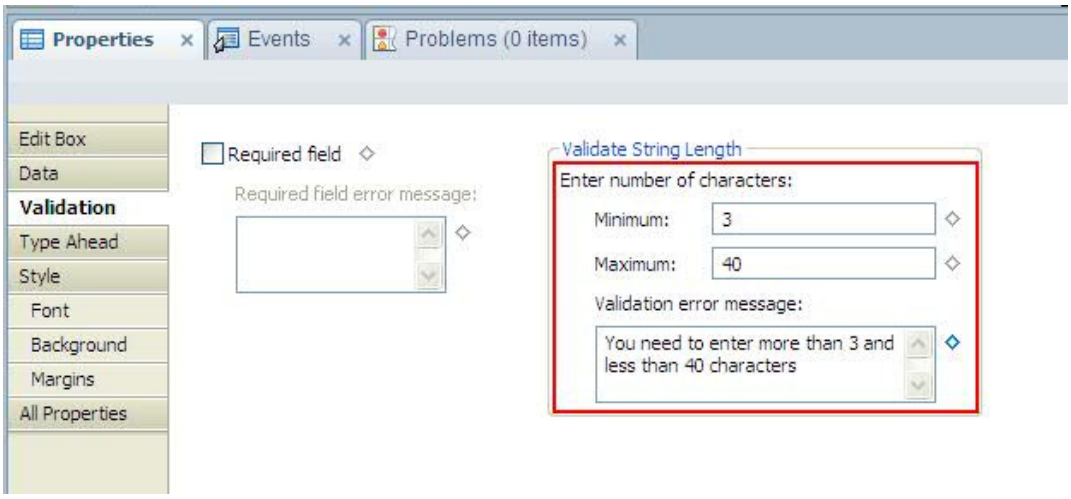


Implement Client-side Validation

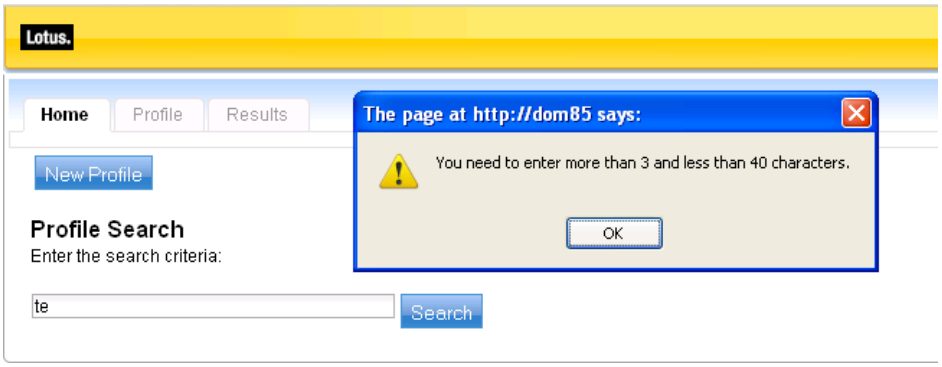
67. Open the **searchForm** custom control and select the **searchBox**.

68. On the **Validation** tab, specify the minimum (4) and maximum (41) values for the search field.

69. Specify an error message.



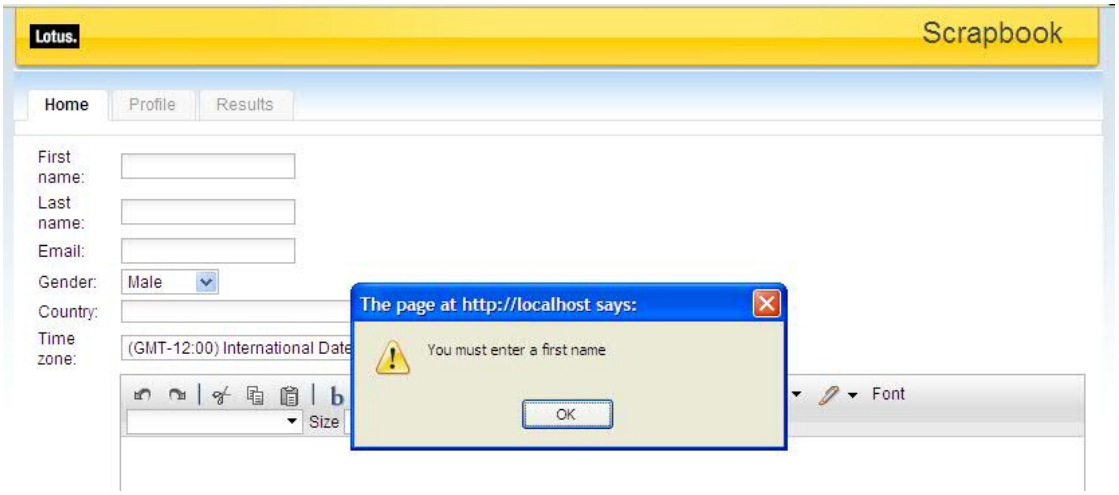
70. Save the custom control and preview the **home** XPage.



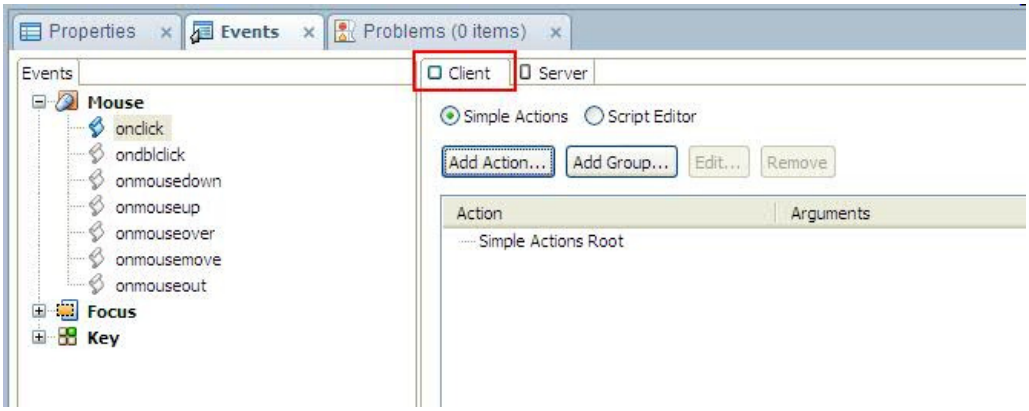
71. Open the **profileForm** custom control.

72. Make **FirstName** and **LastName** required fields (**Hint**: Check **Validation** tab in Properties dialog). Set the minimum number of characters for each field to 3.

73. Save the custom control and preview the **profile** XPage.



74. Add validation to the **LastName** field to check that it is not the same as **FirstName** by going to the **onclick** event of the **Save** button, then adding a **Client** script for the event.

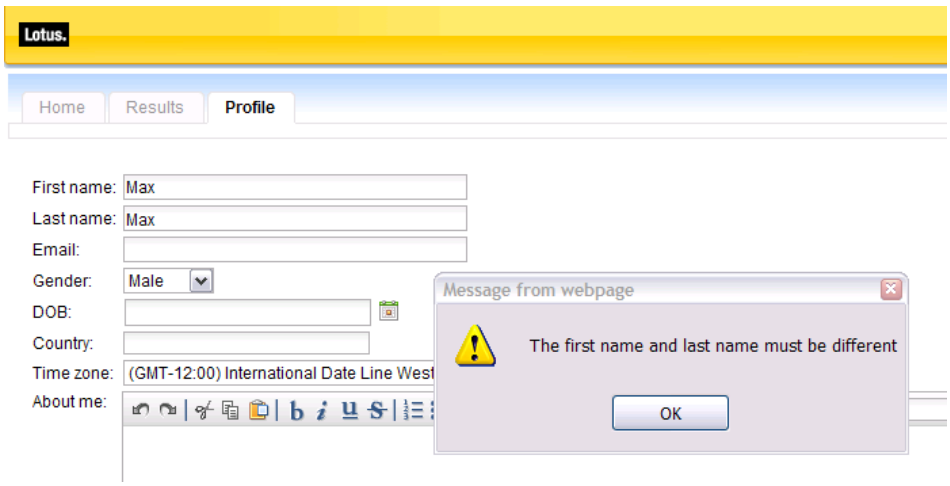


75. Enter the following script.

    Important: Make sure the field names entered for **FirstName** (id:firstName1) and **LastName** (id:lastName1) match the field names you have used!

```
var firstName = dojo.byId("#{id:firstName1}");
var lastName = dojo.byId("#{id:lastName1}");
if (firstName.value == lastName.value){
   alert("The first name and last name must be different");
   return false;
}else{
   return true;
}
```

76. Save the custom control and preview the **profile** XPage.

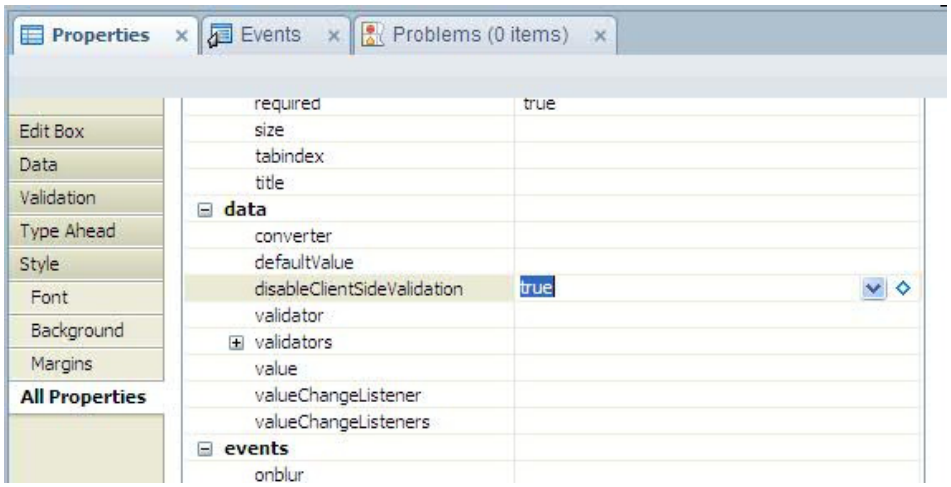


The following steps demonstrate another way to implement error handling. While we have previously used JavaScript, you will now add a Display Error control for that.

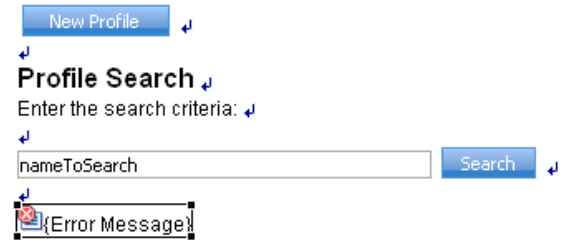
Implement Server-side Validation

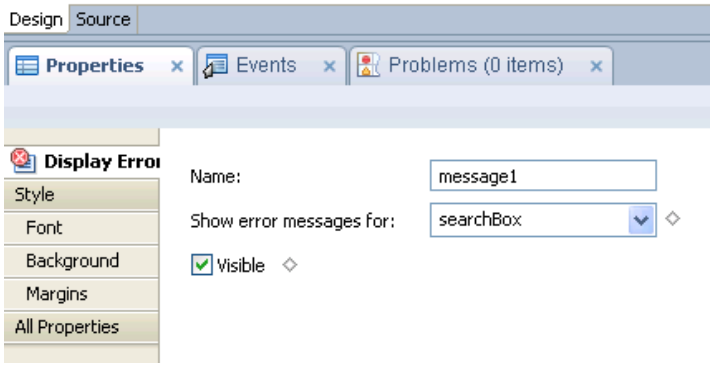
77. Navigate to the **searchForm** custom control. Select the **searchBox** element..

78. From the All Properties section, select **data** and set **disableClientSideValidation** to **true.**

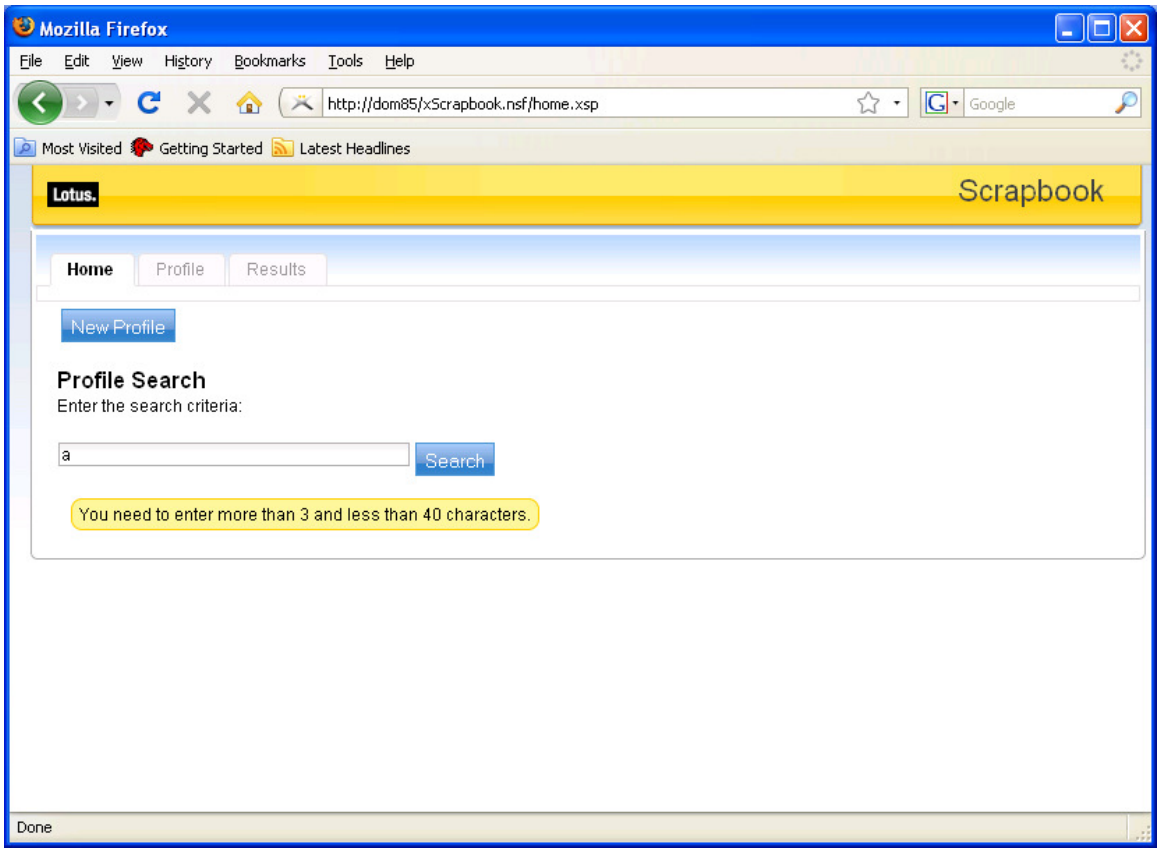


79. Add a **Display Error** control (from Core Controls) at the bottom of the page. Set its **Show error messages for** property to **searchBox**.

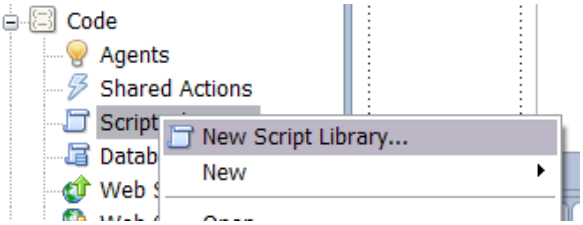




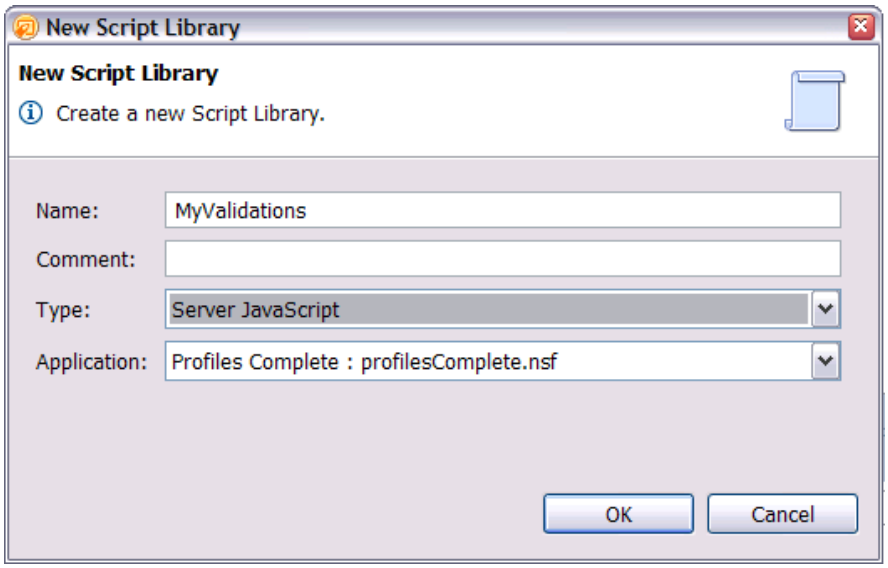
80. Save the XPage, then preview and test in browser.



81. From the left-hand navigator, expand the **Code** section. Right-click on **Script Libraries**, and select **New Script Library**:



82. Name the new library **MyValidations**. It will contain **Server**-side **JavaScript** code:



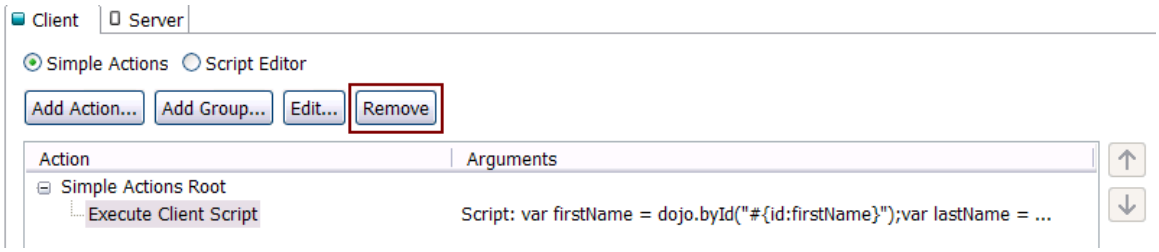
Add the following code there:

```
function checkForDifferentNameFields(firstName, lastName)
{
   if (firstName != lastName) return true;
   else return false;
}
```

83. **Save** the new library.

84. Open the **profileForm** custom control. **Add** the library to the **Resources.**

85. Navigate to the **Save** button and remove the **Client** Script.

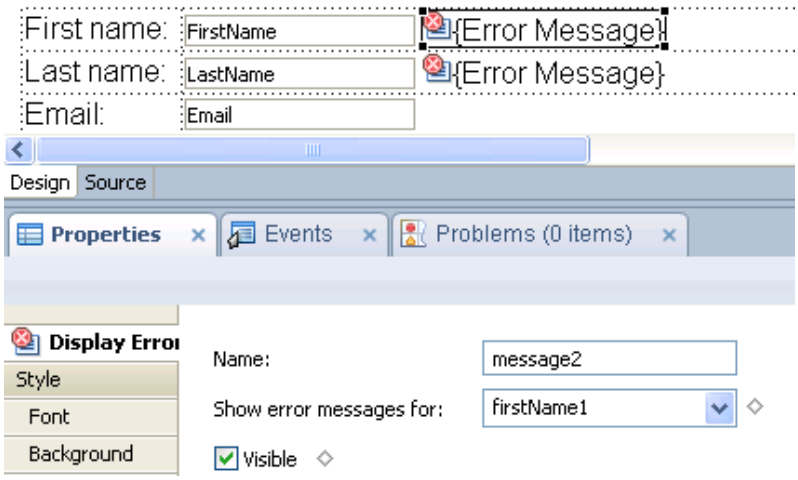


86. Select the **FirstName** element.

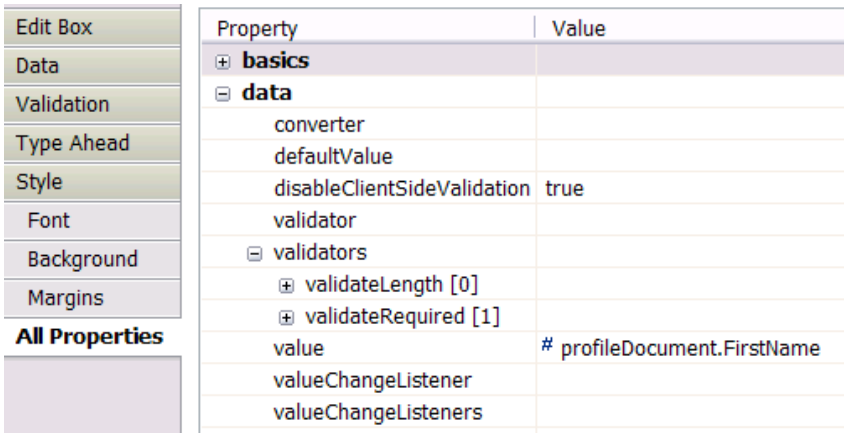
87. From the All Properties section, select **data** and set **disableClientSideValidation** to **true.**

88. Repeat the previous step for **LastName**.

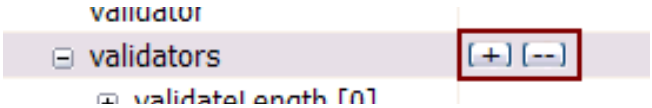
89. Add two **Display Error** controls for **FirstName** and **LastName**.



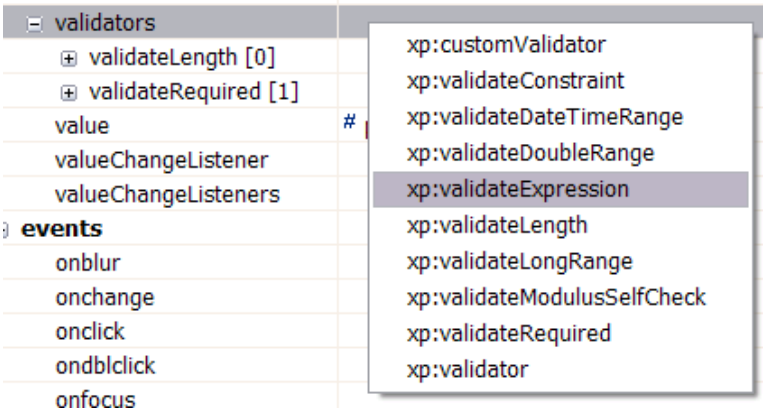
90. Open the **All Properties** tab for **FirstName**. Navigate to **data/validators**.



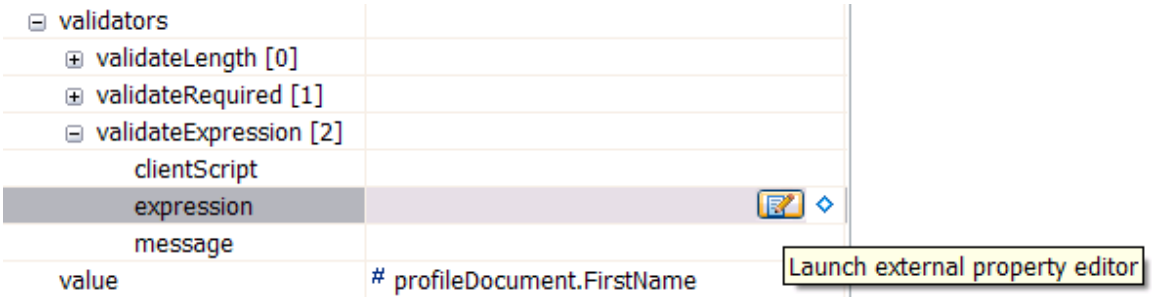
91. Click on the **validators** row. You could now specify additional validators for this field.



92. Click on the **Plus** icon to add another validator. Select **xp:validateExpression** from the list.



93. Click on **expression** and open the property editor by clicking the **Edit** icon.

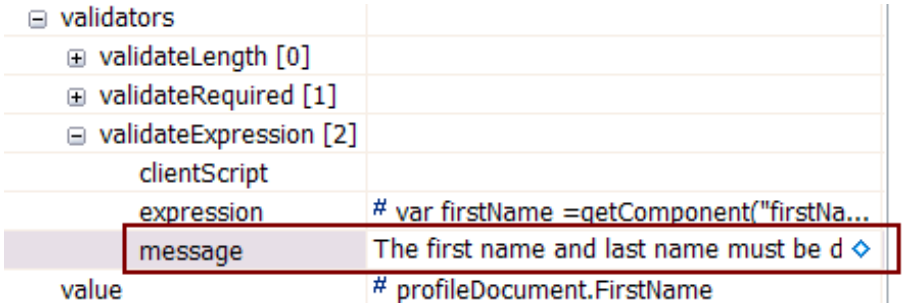


Add the following code there:

```
var firstName =getComponent("firstName1").getSubmittedValue();
var lastName = getComponent("lastName1").getSubmittedValue();
var res = checkForDifferentNameFields(firstName, lastName);
return res;
```


94. Click **OK**, then save the **profileForm** custom control.

95. Add a message to display in case of an error.



96. **Repeat** the steps 90 – 95 for the **LastName** field.

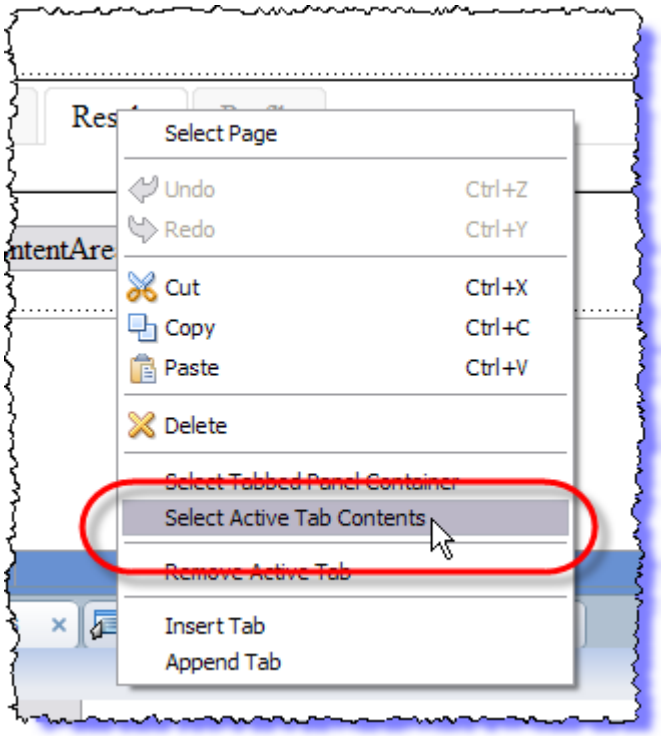
97. Test the **profile** XPage.

Get the Tab Navigation working

98. Open the **container** custom control.

99. Select the **resultsTab** tab (right click on it in the design editor).

100. Select the **Select Active Tab Contents** context menu option.



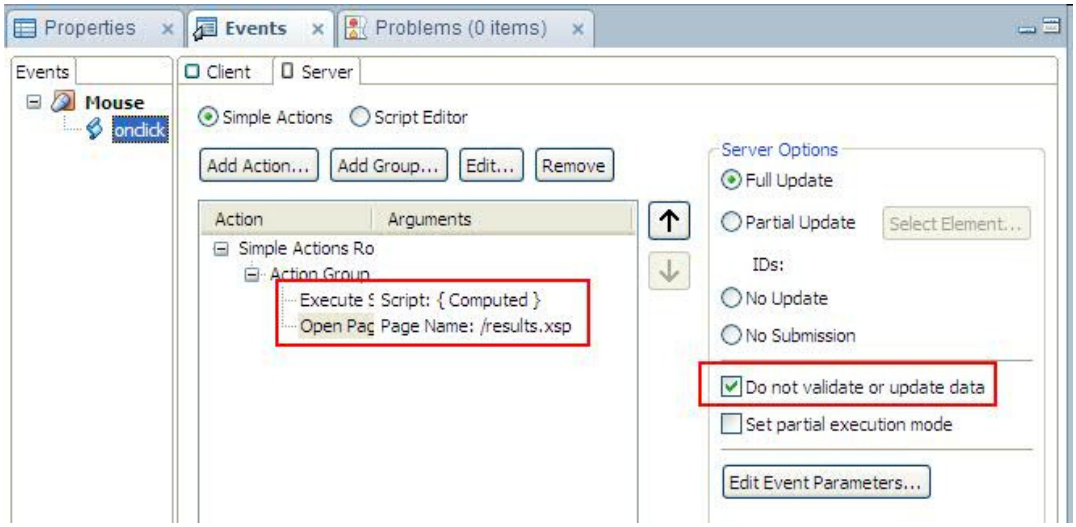
101. On the **Events** section for the **Results** tab add an **Execute Script** Simple Action with this formula:

```
sessionScope.nameToSearch = null
```

This clears any existing entered search term.

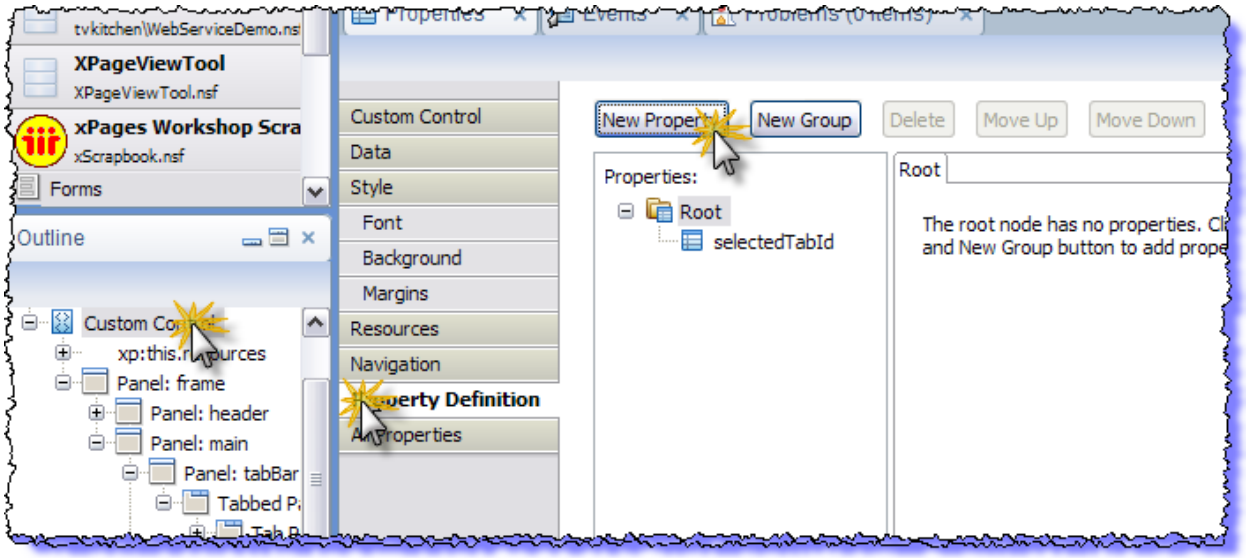
102. Add another Simple Action **Open Page** - specify **results** there.

103. Select **Do not validate or update data**.



104. Repeat step 100, 102 & 103 (but not 101) for the other 2 tabs, specifying the respective pages.

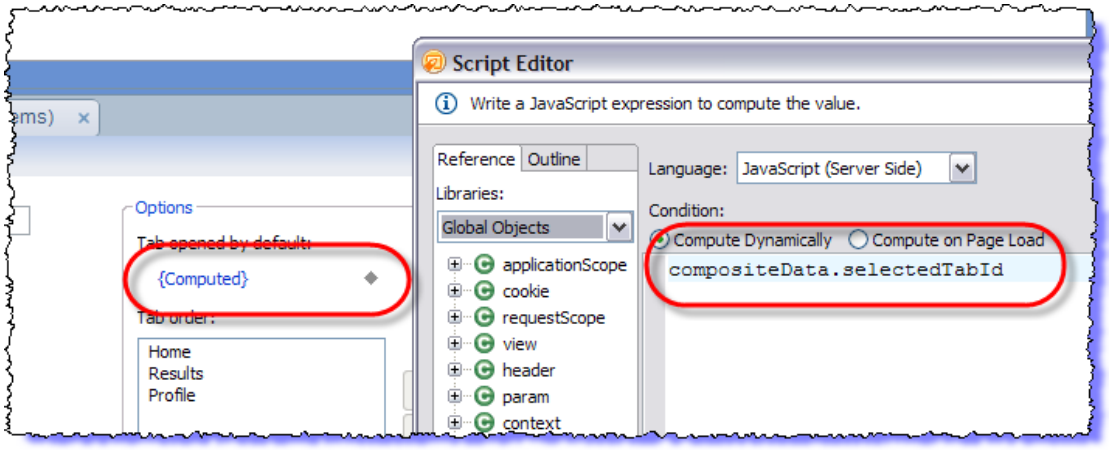
105. Click on the custom control (the **Outline** is a good place to do that) and then select the **Property Definition**.



106. Click **New Property**, name it **selectedTabId** of type **string** (You don't need to provide validation or visibility rules here).

107. Select the Tabbed Panel in the Outline and edit the **Tab opened by default** option as a computed option use this formula:

```
compositeData.selectedTabId
```



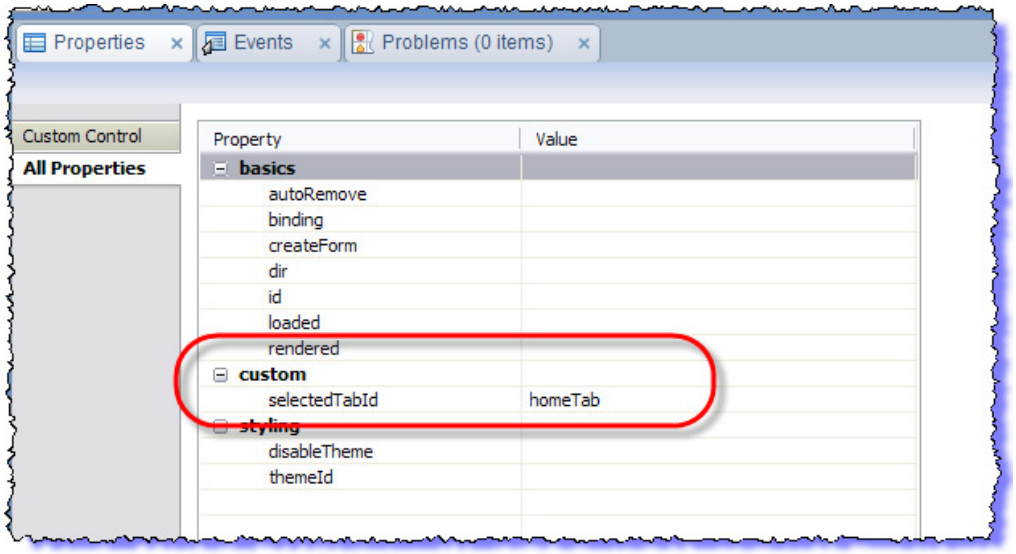
108. For each of the three XPages (home, profile, results), select the container control and click on **All Properties**.

109. Set the **selectedTabID** property to the tab for the respective page.

home – homeTab

profile – profileTab

results - resultsTab

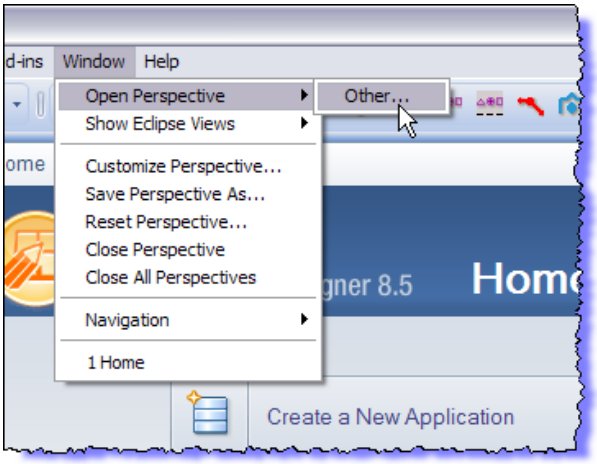


110. Save all XPages and preview the home page in the browser or Notes client. You should now be able to switch between the tabs.

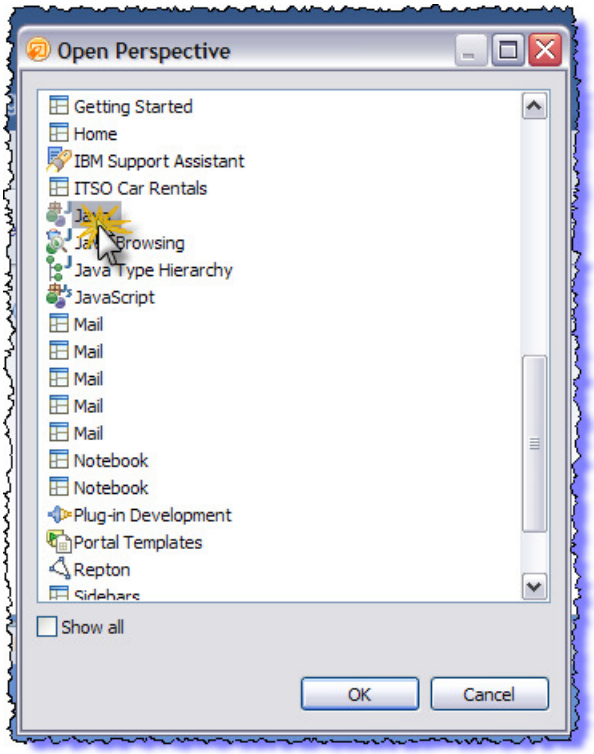
Note: One question typically asked is: Can't I just have one page and use partial page refresh to switch the tabs? The answer: yes of course. But then you lose the direct addressability of a page.

Customize the Application Icon

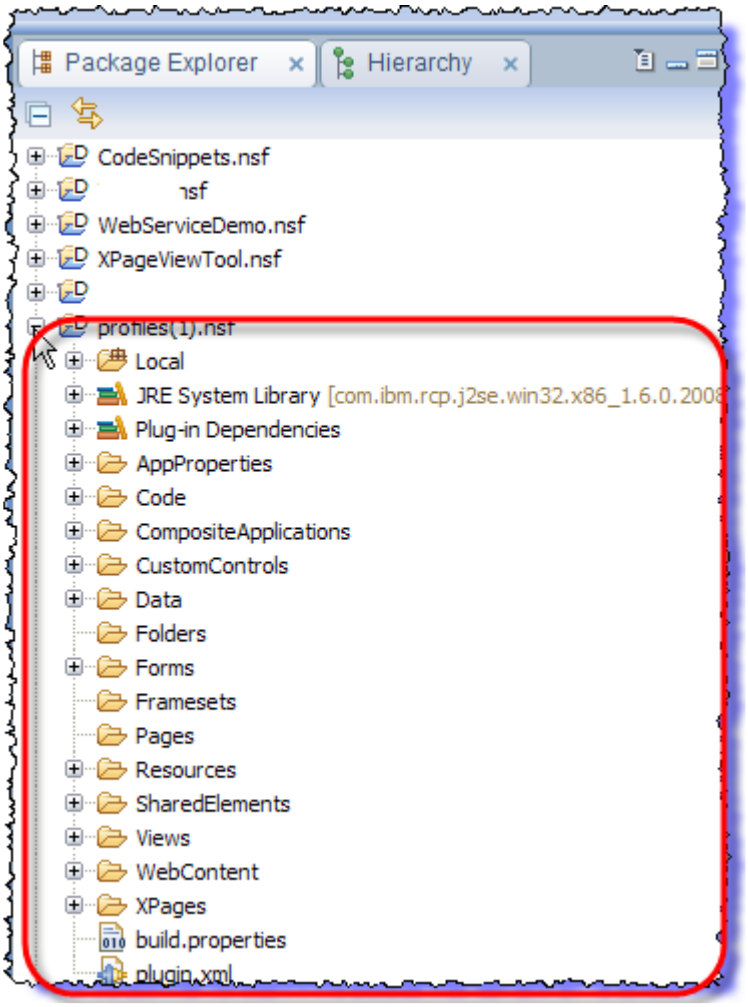
111. From the menu, select **Window > Open Perspective > Other.**



112. Select **Java.**

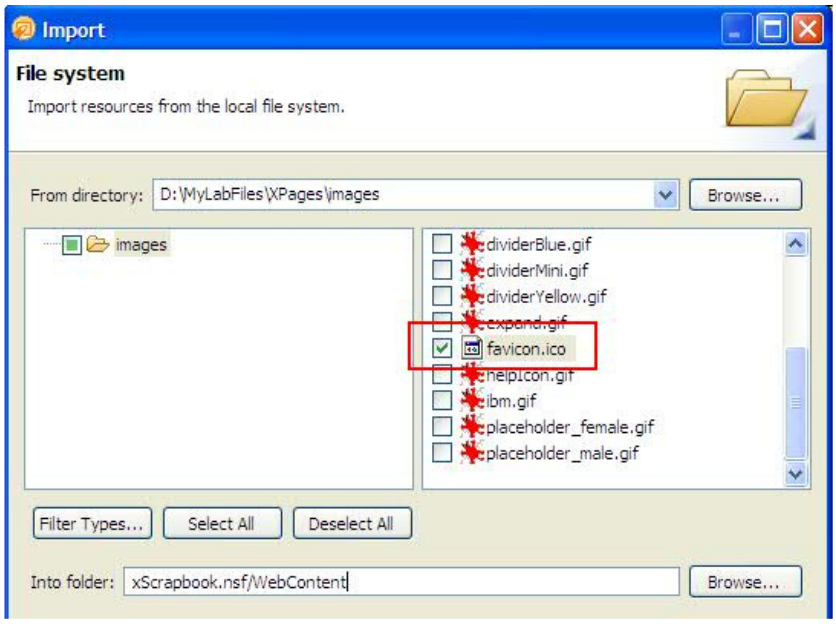


113. The navigator on the left shows the Notes database as a series of folders. Expand the folder for **xScrapbook.nsf.**



114. Navigate to the **WebContent** folder, right-click and select **Import.**

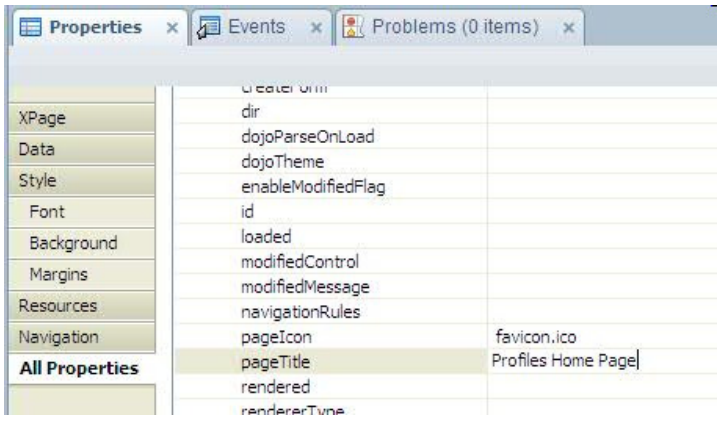
115. Select **General > FileSystem** from the Import dialog. Navigate to **C:\MyLabFiles\XPages\images, s**elect the **favicon.ico** file and click **Finish**.



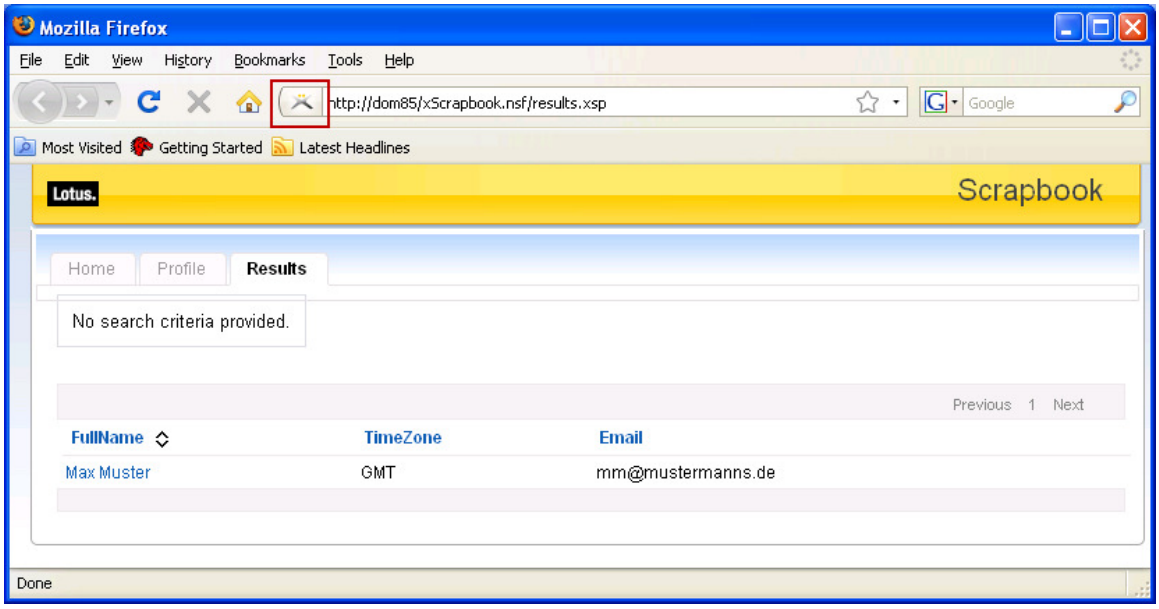
Note: The WebContent directory fulfills the same function as the WebContent directory in a J2EE application. You can introduce additional content here if you wish.

116. To switch back select: **Window > Open Perspective > Other > Domino Designer.**

117. Edit the page properties for your three XPages and set the **pageIcon** to **favicon.ico** and the **pageTitle** to something meaningful:



118. Save the pages and preview in the browser or Notes client



4 Summary

In this exercise you learned how page flow works in XPages and how to code actions behind action buttons. You provided data to the different controls on the XPage and enabled type-ahead functionality in XPages. You learned how to show data from a Notes view in XPages, how to use the XPages scope contexts with advanced data binding and how to propagate data from one form to another.
We also demonstrated how to show a subset of view data only and how to add data validation to the forms in the sample application. You completed the tab navigation and communicated state information.
Last but not least, you learned how to use the Java perspective to add a page icon to your page.