Lotus Domino Designer XPages Tutorial

Contents

XPages tutorial

This tutorial demonstrates the ease with which you can create a Web application in Domino Designer using XPage elements. You will create and preview three XPages. You will place controls on the XPages for accessing and viewing data, navigating to other pages, refreshing the same page, and performing other actions. You will even get into some JavaScript. A few design elements and graphics that are outside the scope of what you are learning are preloaded.

Familiarity with Domino Designer and "traditional" design elements is helpful but not essential. For those that are new to Eclipse, "Navigation in IBM Lotus Domino Designer" on page 19 provides a good introduction.

The demo application, called `Site Finder`, is like the tool on many Web sites to find a nearby location. You submit a zip code or address and the server returns matching locations. In this tutorial, you will develop the Web page for making the submission, as well as other pages for creating and editing the site documents.

For simplicity access is not restricted. Anyone can create and edit documents. This lets you preview the XPage features without having to set up a server. For a real application, you want to restrict access which is easily done through the access control list (ACL). From the menu, click **File** → **Application** → **Access Control**.

The search mechanism in the demo application is very simple, based only on exact matches. In a real application, you would want a more sophisticated mechanism.

Learning objectives

- Set up and view an application containing XPage elements
- Create an XPage
- Add controls to an XPage
- Bind data to an XPage
- Add navigation to an XPage
- Create a custom control
- Use advanced binding and scripting

Time required

This tutorial should take approximately 90 minutes to finish.

Setting up the sample application

During the course of the tutorial, you will create three XPages named `Site`, `SiteFinder`, and `SiteList`. When you first open the sample application, you will see existing XPages with similar names. Three of them are final solutions and are named `SiteFinal`, `SiteFinderFinal`, and `SiteListFinal`. Two of them are intermediate solutions and are named `Site1` and `SiteList1`.

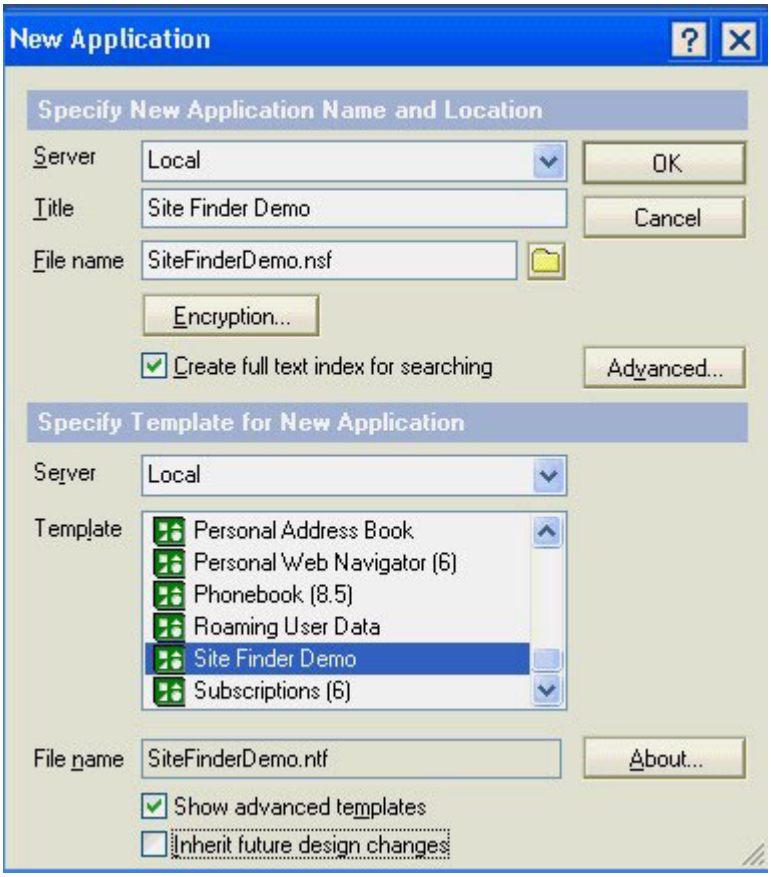
Launch Domino Designer and do the following:

1. From the menu, click **File** → **Application** → **New** (**Ctrl+N**).
2. In the `New Application` box, do the following:

a. Leave the Server selection as `Local`.
b. Type a title, for example, `Site Finder Demo`.
c. Accept or modify the filename, for example, `SiteFinderDemo.nsf`.
d. (Optional) Click **Encryption**. Select `Do not locally encrypt this database` and click **OK**.

   This allows other Notes users to access the application locally, for example, if you mail it to someone. In general, you should encrypt real applications to secure them. Local encryption does not prevent the application from being shared through replication to a server.
e. Check `Create a full text index for searching`. You will need this later.
f. Leave the template server selection as `Local`.
g. Check `Show advanced templates`.
h. In the template list, select `Site Finder Demo`.
i. Uncheck `Inherit future design changes`.

Here is what the `New Application` box should look like.



3. Click **OK**.

   Domino Designer creates the new application and its elements appear under `Applications` on the left side of the screen. Expand the elements to explore the design of this application.
4. Expand **Code** and double-click **Agents**. In the list of agents, right-click **Create sample data** → **Run**.

   This agent creates several sample records which you will use to test the application. Respond to any dialog boxes. Click the **X** in the upper right corner to close the pane containing the list of agents.

5. Expand **XPages** and double-click **SiteFinderFinal** (the final solution for `SiteFinder`).
6. Click **Design → Preview in Web Browser → Default System Web Browser** (or choose a specific browser from among those listed).

   A mini Web server running on your computer opens a browser, generates HTML for the XPage, and sends the page to the browser using `localhost` for the server name.

   If your computer is not set up to use `localhost`, the preview URL will fail. Open a browser and use `127.0.0.1` or the actual IP address of your computer in the URL, for example, `http://127.0.0.1/SiteLocator2.nsf/SiteFinder.xsp`.
7. Fill in the `Zip code` field with a value (for example, `02108`) and click the **Search** button.

   This submits the page to the server. The server processes the information from the page and sends another page to the client.
8. Close the browser when you are done.
9. In Designer, close the XPage element by clicking the **X** on its tab.

   The appearance varies. If a pane has more than one tab, you see an **X** on a tab. If a pane has just one tab, it appears as a window with the **X** in the upper right corner.

Feel free to explore the application by previewing the XPages with `Final` in the name. This is the final version of what you'll be designing as you go through the tutorial.

Creating an XPage

This lesson steps you through the basics for creating an XPage.

This lesson uses one form and one view, and does not take you through the exercise of creating them. The form contains the fields for storing the site information and the search criteria, and the view is used for displaying the list of sites and search results. You should take a quick look at them. For documentation about forms and views, look under "Lotus Domino Designer Basic User Guide and Reference" in the help.

- Forms are used by XPages to define what data is stored in a document, not how it is displayed. This is similar to the use of hidden fields on display forms to organize and calculate data. The `Site` form contains data fields for the creation of documents, and some explanatory text. The user will never see this so it does not have to be fancy.
- Similarly views are used by XPages to define collections of documents and their ordering. Much like you might use a view purely to index some data so that it can be addressed programmatically, the `SiteList` view serves as an index to your data documents. It lists the contents of all the documents sorted by site name. Views are a powerful way to organize documents, and it's common to have several views showing different subsets and sort orders.

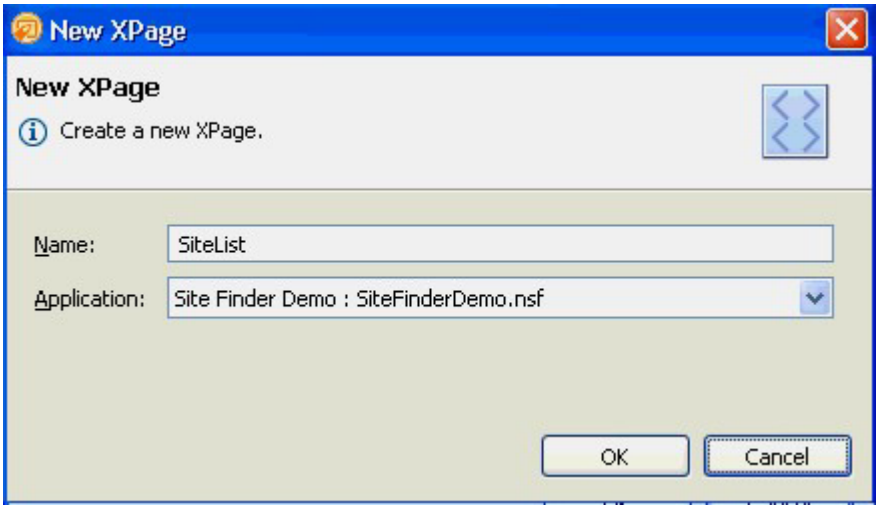
Make sure the application is expanded in the left pane and do the following:

1. Double-click **XPages**.

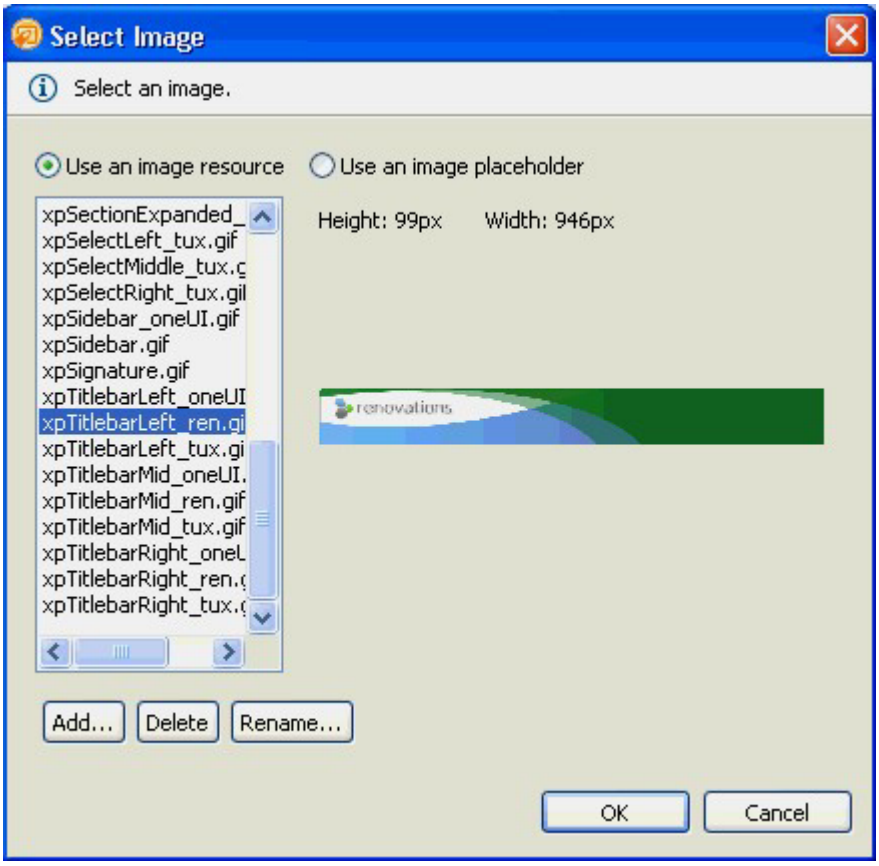
   This brings up a list of XPage elements in the center pane.
2. Click **New XPage** above the list.
3. Name your new XPage `SiteList` (`.xsp` is assumed).

   You can use spaces in the name but names are usually clearer if you don't.

The New XPage box should look like this.



4. Click **OK**. An empty XPage design element is created in the center pane.
5. Take the following steps to put header graphics on the page:
   a. From Core Controls in the right pane, select and drag **Image** onto the XPage. The Select Image box appears.
   b. In the Select Image box, select **xpTitlebarLeft_ren.gif**.
      The Select Image box should look like this.



   c. Click **OK**.

The list of images you see here were preloaded. Each image you see on the list is a resource in the current database. The **Add** button allows you to import additional images (and there are other ways to import images).

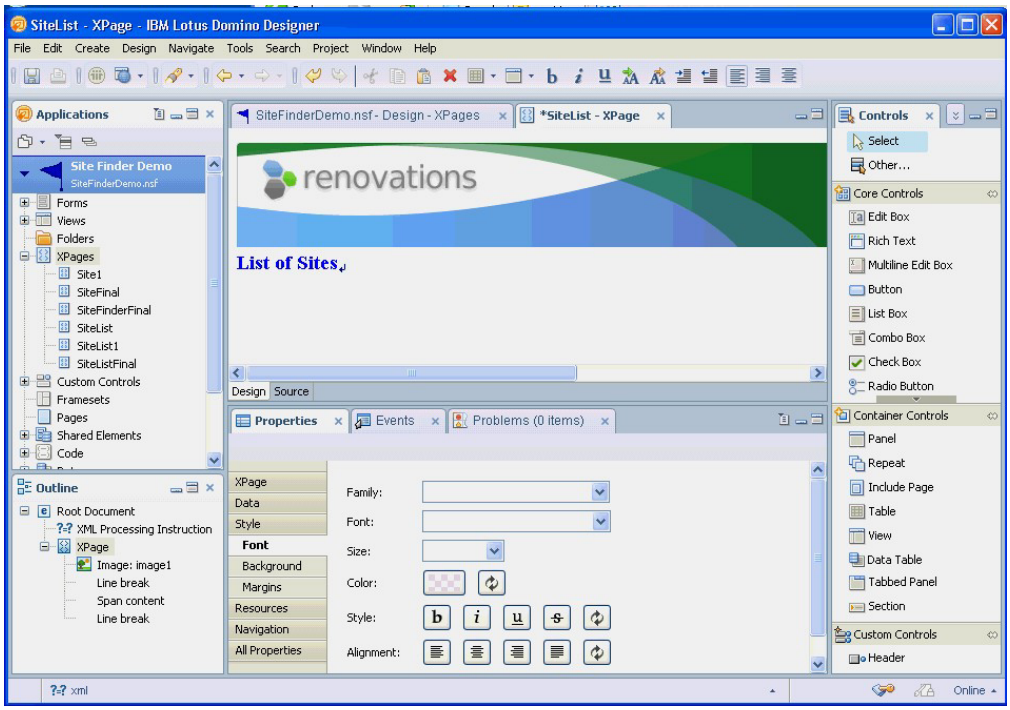
After you click **OK**, the image should appear as a header.

6. Click in the empty space to the right of the header image. You may have to scroll to the right if the space is not visible. Press **Enter** so the cursor goes to the next line.
7. Do the following to enter some header text:
   a. On the XPage, click on the empty line below the header.
   b. Type `List of Sites` and press **Enter**.
   c. Highlight the line of text.
   d. In the bottom pane, click **Font** and change the fonts as desired, for example, to bold 14 blue.

   Later you will add a stylesheet and apply it to this text as an alternate way to control formatting.

   You may be prompted about a `Disable default theme` checkbox. You can ignore the warning for this application. See "Designing with themes" on page 19 for more information.

The design for your XPage should look like this:



Experienced Web developers may want to use elements that structure the document, such as `H1` and `H2`. The design UI does not support this, but the source UI allows you to enter XML as desired. Click the **Source** tab to see and edit the page as XML. Click the **Design** tab to get back.

Creating an XPage for editing

Forms can also be used to design Web pages, but XPages allow more UI options and greater control of appearance with less effort. It's helpful to also have a form to provide a list of fields for binding data, and so documents can be opened from a Notes client. Also the form can be used to calculate computed fields with field formulas rather than putting code on the XPage.

In this lesson, you will learn how to add text to a page, add controls to a page, adjust the appearance of a page with style sheets, and bind data sources to a page. There are a lot of steps but just take them one at a time.

Do the following:

1. Above the list of XPages, click **New XPage**. Alternatively, you can right-click **XPages** → **New XPage** in the left pane.
2. Name your new XPage `Site` and click **OK**. The XPage and form names do not have to be the same but it's convenient.
3. Add a style sheet to the page as follows:
   a. In the bottom pane, click **Style**.
   b. Scroll down and click **Add style sheet to page**
   c. In the `Add Style Sheet to Page` box, select `styles.css`.
   d. Click **OK**.
4. Add a title for the page and style it as follows:
   a. On the XPage, type a title for the page, for example, `Site Description` and press **Enter**.
   b. Highlight the text.
   c. On the bottom pane, click **Styles** if you are not already there.
   d. In the list under `styles.css`, click **.title**.

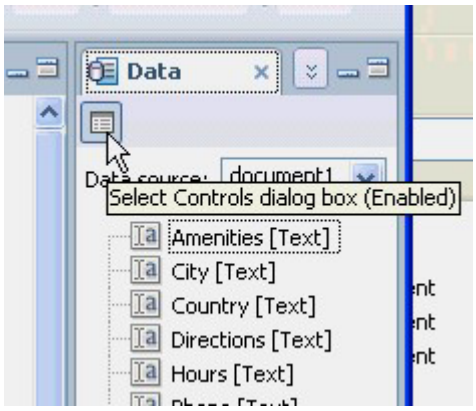
   The appearance of the text will change to conform to the selected style.
5. Associate (bind) the XPage with the Domino back-end data store as follows:
   a. Click outside the text on the XPage so focus is on the page itself.
   b. On the bottom pane under `Properties`, click **Data**.
   c. Click **Add** and select `Domino Document`.

      A data source named `document1` appears.
   d. On the right side of the pane, find `Form` and select `Site` from the drop-down list.

      This associates the XPage with the data definitions on the `Site` form.
6. Set up the data palette as follows:
   a. On the bottom left, click the **Data palette** link. A `Data` tab appears in the right pane.
   b. On the `Data` tab, select **document1** for `Data source`. You may have to place focus on the XPage before the new data source appears on the data palette.

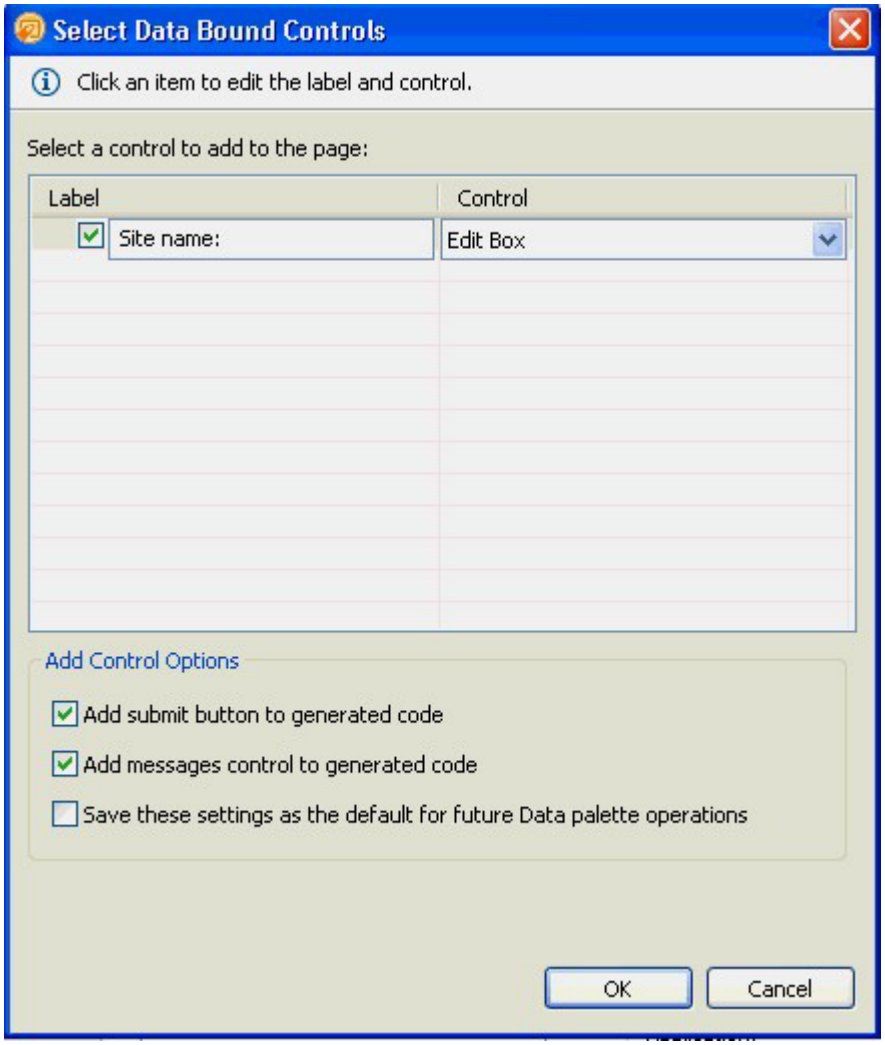
      The field names and types for the data source appear under the name of the data source.
   c. On the top left corner of the `Data` tab, click the icon for **Select Controls dialog box** to enable it.

      The icon for **Select Controls dialog box** looks like this:



7. Drag the `SiteName` field from the data palette onto the XPage below the title. The `Select Data Bound Controls` dialog box opens.
8. Fill in the dialog box as follows:
   a. Leave the checkbox checked to say you want to include this field.
   b. Leave `Site name` as the label text.
   c. Leave `Edit Box` selected as the control.
   d. Check `Add submit button to generated code`.
   e. Check `Add messages control to generated code`.

   The dialog box should like this.



9. Click **OK**.

   This creates a two-column table with the label and edit box on the first row, an area to display validation errors on the second row, and a `Submit` button on the third row.

10. One at a time, drag the SiteNumber and Phone fields into the empty cell to the left of the error display cell. For these fields, do not check for the submit button and messages control since you need only one set of these on the page. Just click **OK**.

    This inserts two rows above the error display for these fields.

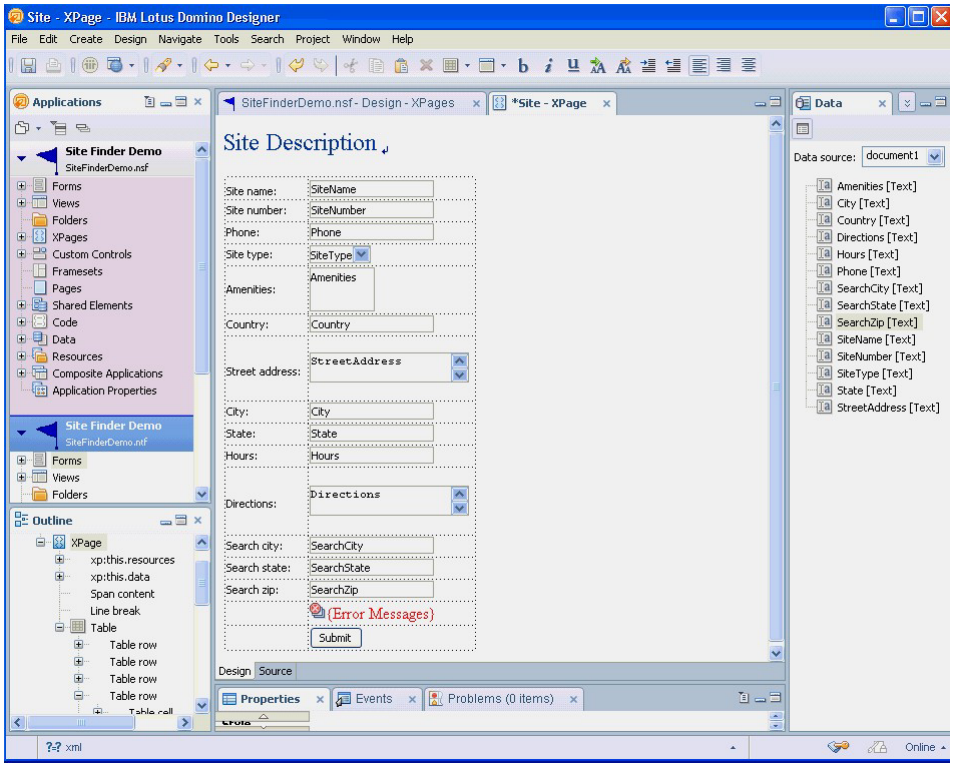
11. Drag over the `SiteType` field and adjust it as follows:
    a. Change the control from **Edit Box** to **Combo Box**, and click **OK**.
    b. Click on the control in column two to put focus on it. The default width is very narrow so use the drag handles on the field to increase the width.
    c. Click **Data** under `Properties` on the bottom pane and type `Office` for the default value.
    d. To add the other choices for the site types, click **Values** under `Properties`. Click **Add Items** three times. Select and change the labels for each item from `Untitled` to `Office`, `Retail`, and `Restaurant`.

       You need not enter anything for `Values` unless the stored value differs from the display value which is not the case here.

12. Drag over the Amenities field and adjust it as follows:
    a. Change the control from **Edit Box** to **List Box**, and click **OK**. You probably want a check box here rather than a list box. However, the design for a check box is more complicated and outside the scope of this tutorial.
    b. Click on the control in column two to put focus on it.
    c. Click **List Box** under Properties on the bottom pane and check Allow multiple selections
    d. Click **Values** under Properties. Click **Add Items** four times. Select and change the labels for each item from Untitled to Cafeteria, Fitness Center, Visitor Center, and Executive Briefing Center.
    e. Click on column 1 inside the cell but outside the label control ("Amenities:"). Click **Table Cell** under Properties and select **Top** for Cell vertical alignment.

       This aligns the label at the top of the cell.
13. Drag over the Country field making no adjustments.
14. Drag over the StreetAddress field and change it to a **Text Area** control. Drag it large enough for two lines.
15. Drag over the City, State, and Hours fields making no adjustments.
16. Drag over the Directions field and change it to a **Text Area** control. Drag it large enough for several lines.
17. Drag over the SearchCity, SearchState, and SearchZip fields making no adjustments. The reason there are both City and searchCity fields is to allow a different city for searching than the actual site names. For example, one of the sample sites is in Brighton, a suburb of Boston, so for searching we'll say it's in Boston. For country, however, we're going to assume that the actual country and the search country are the same.

    Your XPage should look like this.



At this point, your XPage should be similar to Site1 which you can open for comparison.

18. Adjust the table so the row containing the Submit button is hidden when the page is not editable. You will need a little JavaScript.
    a. Click on the table cell containing the Submit button (the cell, not the button).
    b. Under Properties in the bottom pane, click **Table Cell** if it is not already selected.
    c. Click the diamond next to **Visible**, then click **Compute value**. The script editor opens.
    d. In the script editor, double-click **document1** in the list of global variables on the left, then type a period.

       The global variable document1 is an object of type NotesXspDocument that represents the current document that the XPage is viewing. When you type the period, a list of methods for that object appears.
    e. In the list of methods, scroll down and double-click **isEditable()**.

       This returns true if the document is editable and false otherwise. The formula should look like this:

       ```
       document1.isEditable()
       ```
    f. Click **OK**.

       The script editor closes saving your code.

    This code hides the cell containing the button if the return value is false. You could also hide the button rather than the cell.
19. On the XPage, click outside any controls to put focus on the page and press **Ctrl+S** to save the page.
20. Click **Design** → **Preview in Web Browser** → **Default System Web Browser** (or select another browser). You can test your application by entering values and clicking Submit. This should enter a new document into the database. To see a list of the documents, use the Notes client. Close the browser when you are done.

    At this point, your XPage should be similar to SiteFinal which you can open for comparison.
21. Close the Site XPage by clicking the **X** at the upper right of the center pane.

Adding a view control

An XPage view control is used to display information from a defined view. The control you create in this lesson will let you display data from the SiteList view.

Make sure the SiteList XPage is open in the center pane and do the following:

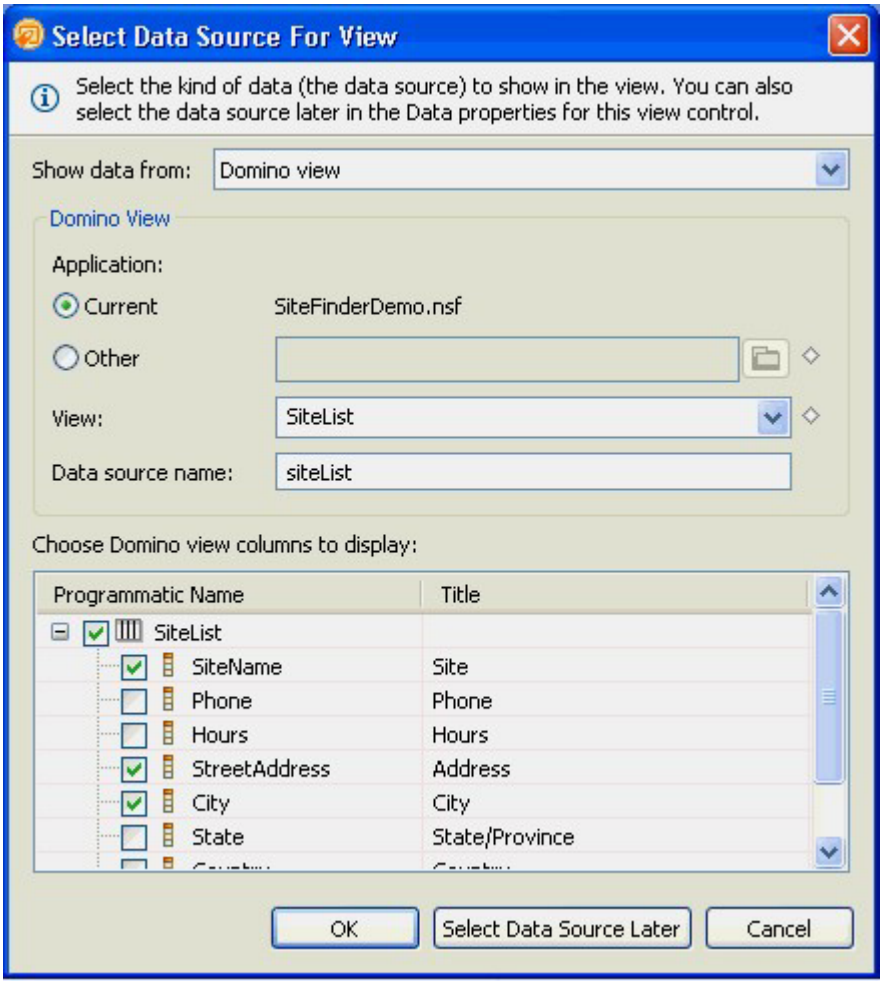
1. From Container Controls, select and drag **View** onto the XPage.
2. In the Select Data Source for View box:
   a. Leave Show data from set to **Domino view**.
   b. Leave Application set to **Current**.

      You can select any Domino application, but the view control won't work unless that application is available on the run-time server.
   c. In the View drop-down list, select **SiteList**.
   d. Accept siteList as the data source name.

      This name can be used in server-side JavaScript code. It does not have to match the view name.

e. In the list of columns at the bottom, leave checked `SiteName`, `StreetAddress`, `City`, and `SiteType`.

The box should look like this.

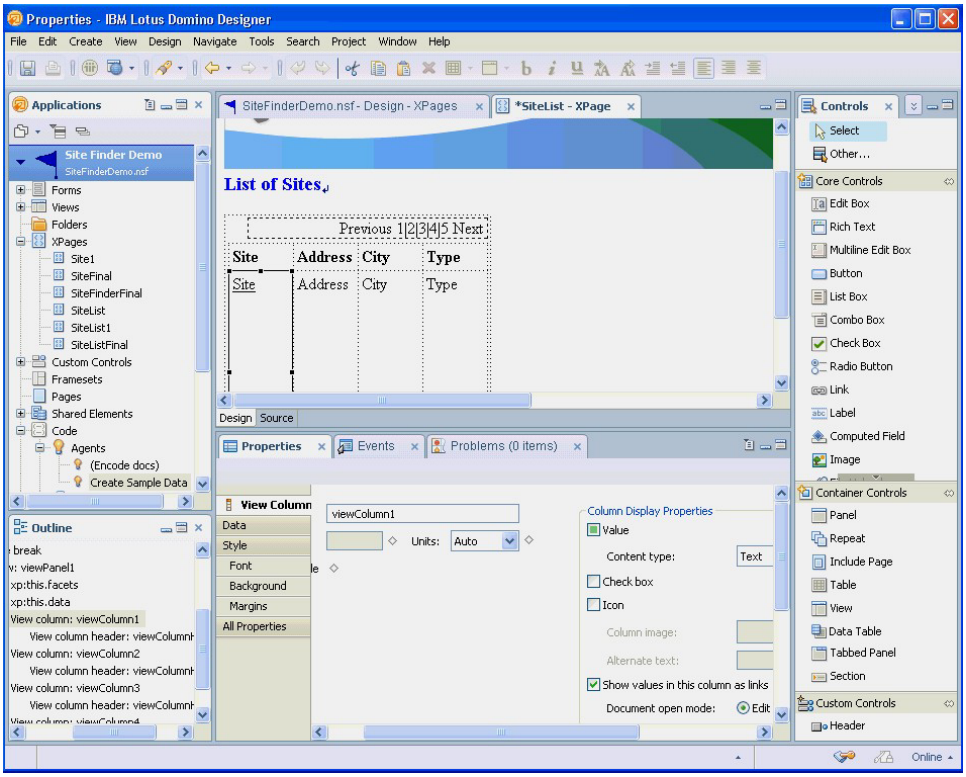


3. Click **OK**.

   A table representing the view appears. The four columns represent the view columns. The two rows represent the column titles and the column contents.

4. Adjust the view control as follows:
   a. To resize the columns, select both rows in one column (by clicking on the cell in the first row, then holding down the **Shift** key while clicking on the second cell) and drag a handle on either cell.
   b. Select the entire header row (by clicking on the first cell, then holding down the **Shift** key while clicking on the remaining cells in the row). Press **Ctrl+B** or use the font properties to make these cells bold.
5. Do the following to allow users to open a document by clicking it in the first column of the view control:
   a. Click inside the bottom left cell (`Site`).
   b. Under the `Properties` tab in the bottom pane, click **View column**.
   c. To the bottom right, check `Show values in this column as links`.

   The page should look like this.



6. Link the view control to the editing page as follows:
   a. Click in the upper left of the view control so that the entire view is in focus.
   b. Under `Properties` on the bottom pane, click **View**.
   c. From the drop-down list at the bottom (`At runtime, open selected document using`), select `Site`.
7. Save and preview the page as follows:
   a. Click on the XPage outside all controls so focus is on the page itself.

      Another way to put focus on the page is to click `XPage` under `Root Document` under `Outline` in the left pane. This puts focus on the page itself. You can put focus on specific design elements in the page by clicking on the corresponding element in the outline.
   b. Press **Ctrl+S** to save the page.
   c. Click **Design → Preview in Web Browser → Default System Web Browser** (or the browser of your choice). Click a document in the list. It should open with the `Site` XPage. You can make changes to the document from this screen and click `Submit` to save them. Submitting the page doesn't move you away from that page, and you can continue to edit and save the same document.

      There are alternative ways to save changes and refresh the browser:

      - A button of type `Submit`, which you use in this tutorial, updates the data sources and refreshes the page.
      - A button of type `Button` permits you to attach simple actions or JavaScript to an `onclick` event. Use this event to update the data sources and redirect the user to another page.
      - Page events permit you to attach simple actions or JavaScript to a `postSaveDocument` event. You can put a button of type `Submit` on the page and in the `postSaveDocument` event redirect the user to another page.

At this point, your XPage should be similar to `SiteList1` which you can open for comparison.

8. Close the browser.

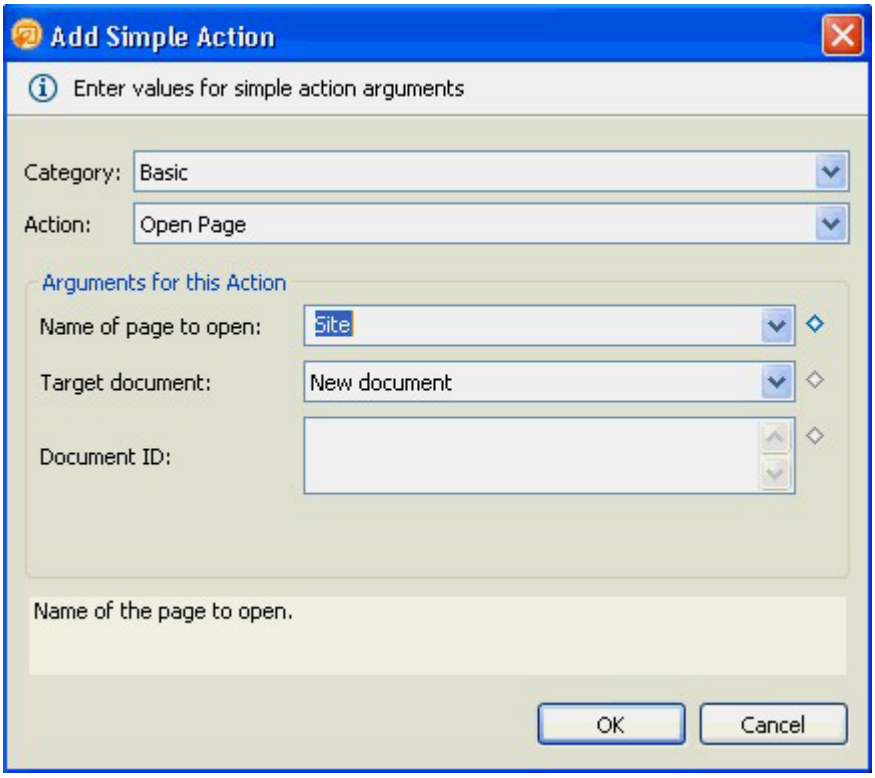
Creating controls to add and delete documents

The SiteList XPage lets you see and edit documents in the application, but we need to add controls to create and delete documents.

Do the following:

1. On the SiteList XPage, click inside the first column (the column, not the header) so it is in focus.
2. Under `Properties`, click **View Column**.
3. Check `Check box`. This puts a check box at the front of each row. The user can select a row by checking a box.
4. On the XPage, click at the end of the title and press **Enter** to put a blank line between the title and the view control.
5. If you don't see the list of controls in the right pane (for example, if the `Data` tab is there instead of the `Controls` tab), click the drop-down menu to the right of the `Data` tab (if you hover over the icon, it says `More windows`) and select **Controls**.
6. Drag a `Button` control onto the empty line under the title and do the following:
   a. Under `Properties` on the bottom pane, click **Button** if it is not already in focus.
   b. Change the label to `New Site`.
   c. Click the **Events** tab.
   d. Ensure that the `onClick` event is selected.
   e. Click **Add Action**. The `Add Simple Action` dialog box opens. The category should read **Basic** and the action **Open Page**. Adjust if this is not so.
   f. Select **Site** for the name of the page to open.
   g. Select **New Document** for the target document.

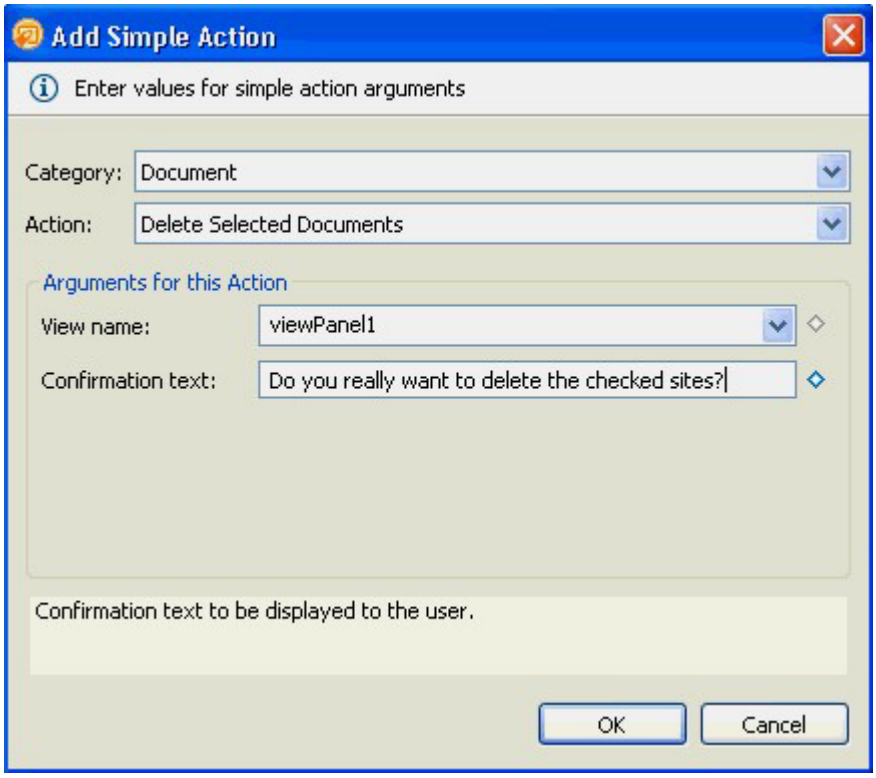
   The dialog box should look like this.



7. Click **OK**.
8. Drag a second `Button` control next to the first and do the following:
   a. On the bottom pane, click the **Properties** tab and click **Button** if it is not already in focus.
   b. Change the label to `Delete Selected`.
   c. Click the **Events** tab.
   d. Ensure that **onclick** is selected.
   e. Click **Add Action**.
   f. Select **Document** for the category.
   g. Select **Delete Selected Documents** for the action.
   h. Select **ViewPanel1** for the view name.

      If this is not the name of your view control, select your view control.
   i. Type **Do you really want to delete the checked sites?** for the confirmation text.

   The dialog box should look like this.



9. Click **OK**.
10. Press **Ctrl+S** to save the page.
11. Preview the page as desired.
12. Close the `SiteList` XPage by clicking the **X** at the upper right of the center pane.

At this point, your XPage should be similar to `SiteListFinal` which you can open for comparison.

Creating an XPage for searching

The user uses this page to specify search parameters and see the results. The search parameters are values for the city, state, country, and zip code. The user can fill in any of these parameters and click a button. The page is refreshed and results are displayed as a view. A search property for the view limits the display to documents that match the search parameters.

Do the following:

1. Above the list of XPages, click **New XPage**. Alternatively, you can right-click **XPages** → **New XPage** in the left pane.
2. Name your new XPage `SiteFinder` and click **OK**.
3. From under `Custom Controls` in the right pane, drag the `Header` custom control to the XPage.

   This is the same image you put on the `SiteList` XPage, but it has been put in a custom control for ease of maintenance. A custom control is similar to a subform.
4. Add a style sheet to the page as follows:
   a. Click on the page outside any controls so focus is on the page.

b. Under Properties in the bottom pane, click **Style**.
c. Scroll down and click **Add style sheet to page**
d. In the Add Style Sheet to Page box, select styles.css.
e. Click **OK**.

5. Add a title as follows:
   a. On the XPage, press **Enter** after the header, type a title for the page, for example, Site Finder, and press **Enter** again.
   b. Highlight the text.
   c. On the bottom pane, click **Style** if you are not already there.
   d. From the list under styles.css, click **.title**.

      These CSS files are stylesheet design elements included in the sample application.

   The appearance of the text will change to conform to the selected style.
6. Below the title text, enter instructions for the user, for example, Fill in any or all of these fields and click Search, and press **Enter**.
7. Create a table with four rows and two columns as follows:
   a. From Container Controls in the right pane, drag **Table** under the instructions on the XPage.

      The Insert Table dialog box appears.
   b. In the Insert Table dialog box, specify 4 for rows and 2 for columns.
   c. Click **OK**.
   d. Use the grab handles to adjust the width of the table.
8. Do the following for each cell in column 1:
   a. From Core Controls in the right pane, drag **Label** into the cell.
   b. In the bottom pane, change the label of the control to City (row 1), State (row 2), Country (row 3), or Zip/Postal code (row 4).
9. Do the following for each cell in column 2:
   a. From Core Controls in the right pane, drag **Edit Box** into the cell.
   b. Under Properties in the bottom pane, click **Edit Box**.
   c. Change the names to searchCity (row 1), searchState (row 2), searchCountry (row 3), and searchZip (row 4).
   d. Under Properties in the bottom pane, click **Data**.
   e. For Bind data using, click **Advanced**.
   f. From the first drop-down menu, select **Scoped Variable**.
   g. From the next drop-down menu, select **Session Scope**.
   h. Scroll down and type the variable name as searchCity (row 1), searchState (row 2), searchCountry (row 3), or searchZip (row 4).

   Binding these edit boxes to scoped variables will allow us to access their values in JavaScript code rather than in the data store. A session variable retains its value across pages for the duration of the user session on the server.
10. Do the following to create a button to submit and refresh the page:
    a. From Core Controls in the right pane, drag **Button** to the line below the table.
    b. In the bottom pane, change the label of the control to Search.
    c. From the drop-down list for the button type, select **Submit**.

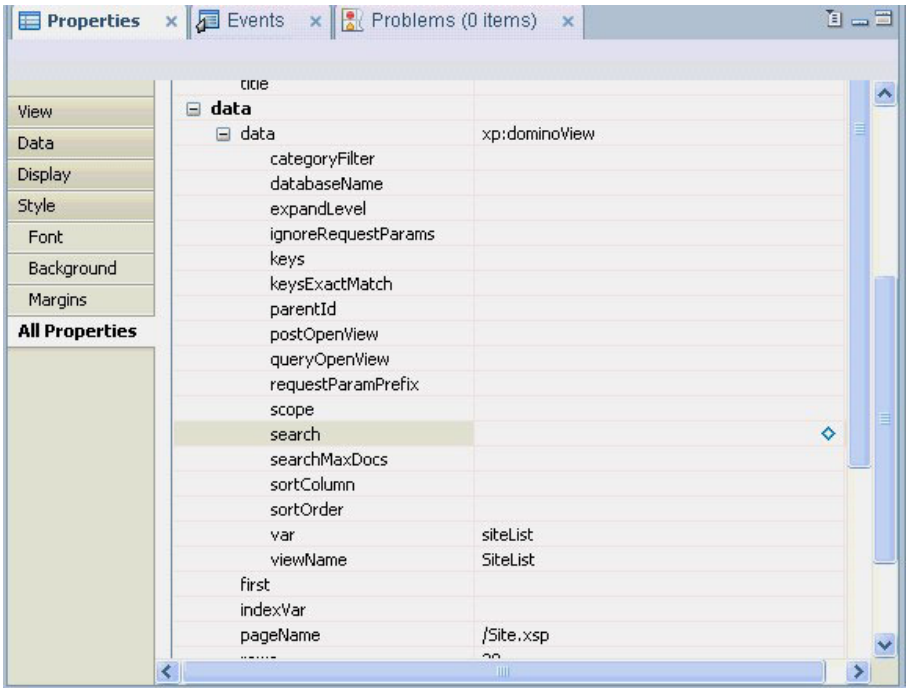
Clicking this button will submit the page to the server and refresh the contents of the page based on the values the user entered.

11. Do the following to display the query used to get results:
    a. On the next line on the XPage, type **Query string:** and a space.
    b. From `Core Controls` in the right pane, drag **Computed Field** onto the page.
    c. Under `Properties` in the bottom pane, click **Font**. Change the color to blue.
    d. Under `Properties` in the bottom pane, click **Value**.
    e. For `Bind data using`, click **Advanced**.
    f. From the first drop-down menu, select **Scoped Variable**.
    g. From the next drop-down menu, select **Session Scope**.
    h. Scroll down and type the variable name as `queryString`.

       The variable name is of your own choosing and has no special meaning. You will assign a value to it later.

    This allows us to display the query that we calculate later. The display is probably not necessary in a production application but is helpful for testing.
12. From `Container Controls` in the right pane, drag **View** onto the next line of the page.
    a. Leave `Show data from` set to **Domino view**.
    b. Leave `Application` set to **Current**.
    c. In the `View` drop-down list, select **SiteList**.
    d. Accept `siteList` as the data source name.
    e. In the list of columns at the bottom, uncheck everything except `SiteName`, `Phone`, `StreetAddress`, `City`, and `State`. You can change this and choose what columns to display, but `SiteName` must be included.
    f. Click **OK**.
13. Under `Properties` in the bottom pane, click **View**. For `At runtime, open selected document using`, select **Site**.
14. Under `Properties` in the bottom pane, click **Data**. Ensure that the view is set to **siteList**.
15. Now you set a search query to limit the view to those documents that satisfy what the user enters for the search fields. Do the following:
    a. Under `Properties` in the bottom pane, click **All Properties**.
    b. Scroll down. Expand **Data** then expand the second occurrence of **Data**. Select the **Search** property.

       Your screen should look like this.



c. Click the diamond to the right of the `Search` property and select **Compute value**.

   The script editor opens.

d. Enter the following code in the script editor. Make sure the language is server-side JavaScript and `Compute Dynamically` is set.

```
var tmpArray = new Array("");
var cTerms = 0;
if(sessionScope.searchZip != null & sessionScope.searchZip != "") {
 tmpArray[cTerms++] = "(FIELD SearchZip = \"" + sessionScope.searchZip + "\")";
}
if(sessionScope.searchCity != null & sessionScope.searchCity != "") {
 tmpArray[cTerms++] = "(FIELD SearchCity = \"" + sessionScope.searchCity + "\")";
}
if(sessionScope.searchState != null & sessionScope.searchState != "") {
 tmpArray[cTerms++] = "(FIELD SearchState = \"" + sessionScope.searchState + "\")";
}
if(sessionScope.searchCountry != null & sessionScope.searchCountry != "") {
 tmpArray[cTerms++] = "(FIELD Country = \"" + sessionScope.searchCountry + "\")";
}
qstring = tmpArray.join(" AND ").trim();
sessionScope.queryString = qstring; // this just displays the query
return qstring // this is what sets the search property
```

   You will not go through the exercise, but this code could be put in the form of a function and placed in a script library. The script library would then be added as a resource to the page and the function accessed by a simple reference. This allows the code to be used in multiple places with central maintenance.

e. Click **OK**.

This code gets the values the user enters on the page by using the session scope variables bound to those edit boxes. The code builds and returns a query string that matches document values in the fields `searchZip`, `searchCity`, `searchState`, and `Country`.

For clarity we're using the same field names as in the documents we're searching, but we could use any names.

16. Adjust the first column to allow the user to open the document in read mode:
    a. On the XPage, click inside the first column of the view.
    b. Under Properties in the bottom pane, click **View Column**.
    c. Check **Show values in this column as links**.
    d. For Document open mode, click **Read-only**.

    Web users looking at search results will not be allowed to edit the documents they find when they follow the link. In a real application you would also use the access control list.
17. Save your changes and preview your new search screen.
18. Preview the page as desired. Try some searches, for example: 02108 or 33432 for the zip code; Boca Raton or Boston for the city; MA for the state; France for the country.
19. Close the SiteFinder XPage by clicking the **X** at the upper right of the center pane.

At this point, your XPage should be similar to SiteFinderFinal which you can open for comparison.

The process shown here is not necessarily the optimal way to organize a search function but it is one way. You might instead assign the search property of the data source from JavaScript code in the Search button. That way it is easier to have multiple sources of query information on the page, perhaps with multiple search buttons for different types of searches.

Other controls are available to better format your results. The Data Table, for example, gives more control over how you lay out the search results, and is often used instead of a view. However, the View control is very simple to use.

Navigation in IBM Lotus Domino Designer

IBM Lotus Domino Designer is installed as an Eclipse perspective arranged into easy-to-access views, palettes, and editors. You can perform tasks using the main Eclipse menu, right-click menus, and icons in the view, palette, and editor title bars.

Since the IBM Lotus Domino Designer perspective is defined by normal Eclipse conventions, you can move, resize, close, and open views. These changes are handled by standard Eclipse code, and can be persisted and saved as an alternative perspective name. You should always be able to retain the predefined Domino Designer perspective by clicking **Window → Reset perspective** in the main Eclipse menu. You can also navigate through views using the default Eclipse keyboard shortcut Ctrl + F7 or by clicking **Window → Navigation** in the main Eclipse menu.

To see any view that is not visible, click **Window → Show View** in the main Eclipse menu, and select the hidden view from the list.

For more information about the IBM Lotus Domino Designer user interface, see the following topics.

Designing with themes

A theme is used to define the look and feel of an application.

Themes are server-side customization of HTML generation that can be used to define the look and feel of an application. A theme can be set globally, to apply to all applications run on that server, or applied to a single application. Different themes can also be applied depending on context.

Themes are different from Style Sheets in that they are not restricted to CSS styles. A theme can assign values to any XML property, including such functional properties as the number of rows displayed by a view. Themes can assign values of JSF expressions, which will be interpreted when a page is created as if that expression was in the source.

Themes can be used to control all XML properties of an XPage. Themes can also control all CSS style properties and the body style attribute for web pages served by Domino. The same theme can be customized to target the XPages and Domino Web environment with different properties.

Themes are one of the design elements available from the Applications Navigator. The Applications Navigator is the left hand tab shown in the user interface. Themes are available under the **Resources** design element.