Harness the power of XPages in Lotus Domino Designer

How to Web 2.0 enable your Domino application

Skill Level: Intermediate

Chris Toohey (ctoohey@dominoguru.com)
Chief Solutions Architect
Clearframe

24 Sep 2008

One of the latest additions to the Domino Web developers' toolkit of technologies, XPages, is also arguably its most powerful and revolutionary to date. XPages allows for functionality and capabilities previously thought impossible to achieve in Domino application development. With the inclusion of XPages, you now have the ability to easily create Web 2.0 user experiences in your existing applications. To demonstrate the power of XPages, this tutorial shows you how to Web 2.0-enable the personal address book Domino application template.

Section 1. Before you start

This tutorial is intended for experienced Domino Web application developers who are new to XPages and what this latest enhancement to the Domino Web environment can do for your existing Domino applications. Although this tutorial covers several basic and fundamental XPage development topics, it refers to common Domino Web application development techniques.

About this tutorial

XPage capability is the latest Domino Browser Client design element. XPages allow Domino Web application developers to both extend the functionality and the end-user reach of current IBM® Lotus® Notes and Domino applications, and rapidly create new, rich Web applications -- all while natively adhering to current Web development standards. Delivered through Java™Server Faces (JSF) technology, XPages provide both novice and seasoned Web application developers with nearly limitless data architecture and function previously not available to Lotus Notes and Domino applications.

One of the benefits of XPages is that you may never truly need to understand JSF technologies to benefit from them. JSF, to put it in terms we Domino developers understand, is the equivalent of a custom-class rendering engine. It interprets developer-supplied markup, evaluating the desired end-result based on predefined rendering logic mapped to the JSF custom-class libraries.

To better illustrate the power of XPages, this tutorial makes use of the personal address book template that comes with Domino. Obviously, the personal address book template was not designed with the Web browser client as the primary user. The template, however, is great to use as the example for this tutorial.

Although the personal address book template does more than simply store contact and contact group information, this tutorial focuses on these two aspects of the application template in this tutorial. Specifically, you will use XPages to provide a Web 2.0-based contact and group management and navigation entry point into the personal address book template.

This tutorial includes the following:

- Overview of XPage Domino design elements
- Creation of a functional example View-type XPage
- Creation of a functional example NotesDocument-type XPage
- Creation of a multipurpose, fully functional XPage, displaying both View and Document-type XPage objects in a single XPage

Prerequisites

If you are unfamiliar with the Lotus® Domino Server 8.5 Public Beta, Lotus Notes, and Domino Designer in Eclipse Client installation processes and need additional information before reading on, please use your IBM ID and password (or complete the simple registration) to review the Release Notes specific to your particular environment.

System requirements

XPages capability requires both the Domino HTTP Server and the Domino JSF, available in the Domino Designer in Eclipse 8.5 Client. Download the following, using your IBM ID and password (or complete the simple registration):

- IBM Lotus Domino Server 8.5 (Public Beta 2): XPage capability requires both the Domino HTTP Server and the new Domino JSF, which is included in the Domino Designer in Eclipse 8.5 (Public Beta 2) Client.
- IBM Domino Designer in Eclipse 8.5 (included in the IBM Lotus Notes Client and Domino Server 8.5 - Beta 2 for Windows), which allows you to create, edit, and maintain XPages.

For this tutorial, you can use the personal address book template (pernames.ntf) packaged in the Lotus Notes 8.5 Public Beta or, as mentioned, any Lotus Notes Client Release 6.5 or greater. This template will serve as a Domino application template that we'll infuse with XPages. You will be creating a new instance of the personal address nook on your Domino Application Server, allowing you to follow along at home with the modifications discussed and detailed in this tutorial.

Section 2. Build a view-type XPage

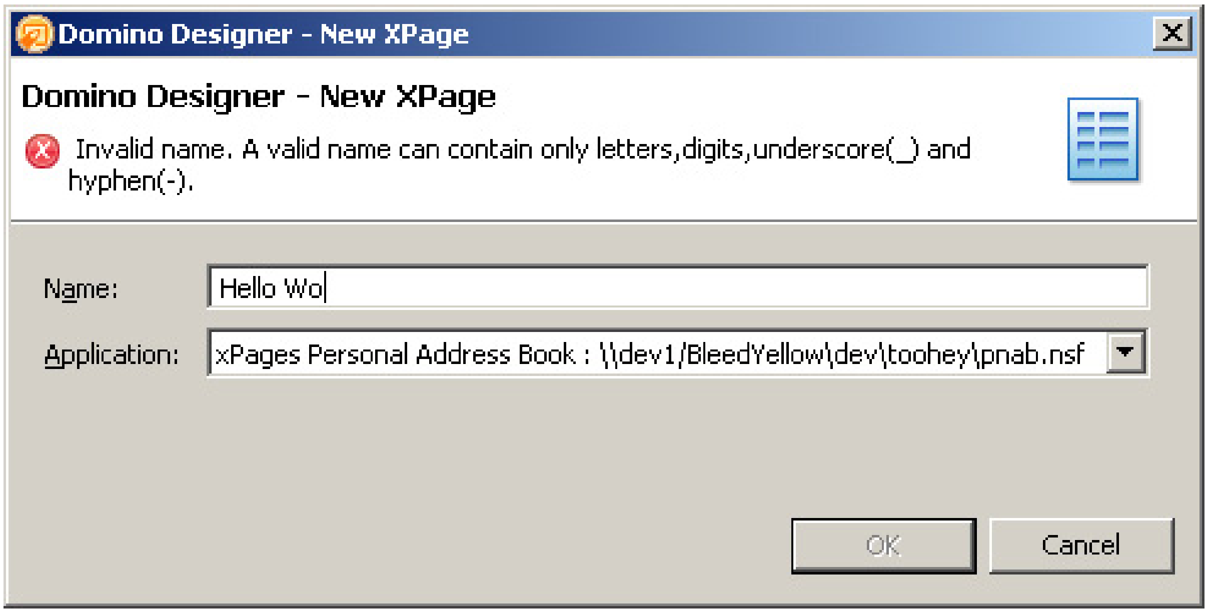
Let's get started by taking the contact from the personal address template. In this section, you create your first XPage, which is used to display contents of the People view in the personal address book application.

Your first XPage

Start the build section of this tutorial by creating a new XPage, which launches the New XPage window shown in Figure 1. This XPage is used to display contacts in the personal address book application, with various Web 2.0-friendly controls and functionality. Typical firsts in application development are titled "Hello World", but your first XPage is named "Contacts" for two reasons:

1. You're not developing for the sake of an example. You have a defined need: the Web 2.0-enabling of an existing, working, and fully-functional Domino application.
2. You'll notice very quickly there are naming scheme restrictions for XPages, including the fact that you can't give an XPage a name that includes a blank space, as shown in Figure 1.

Figure 1. The New XPage window


Domino Designer - New XPage

Domino Designer - New XPage

Invalid name. A valid name can contain only letters,digits,underscore(_) and hyphen(-).

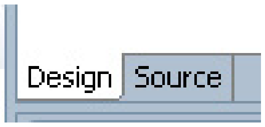
Name: Hello Wo

Application: xPages Personal Address Book : \\dev1/BleedYellow\dev\toohey\pnab.nsf

When you click **OK** from the New XPage window, you are presented with the blank XPage in its Design view.

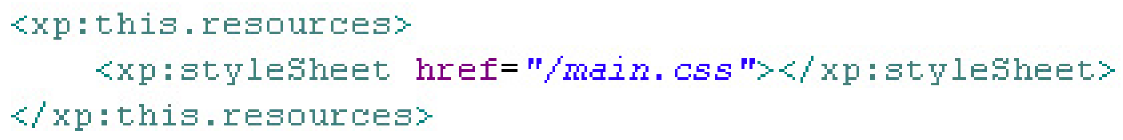
Most WYSIWYG content editors, especially HTML editors, give developers a source view of the WYSIWYG editor content. XPages provides this option, which allows you to toggle between the design and source views at will (see Figure 2).

Figure 2. Design and source panel toggle



For the purposes of this tutorial, you won't be spending much time in the Source view; instead you will be making all design changes via the Rational Application Developer-friendly Design view. However, for those so inclined, you can easily toggle to the Source view and see the markup already generated for your blank XPage (see Figure 3).

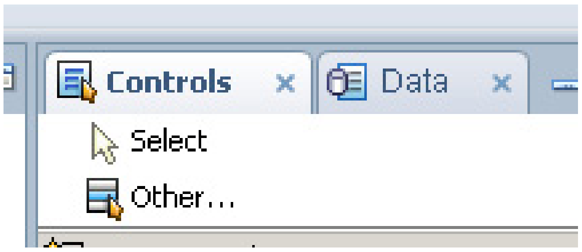
Figure 3. Markup generated for the blank XPage in source view

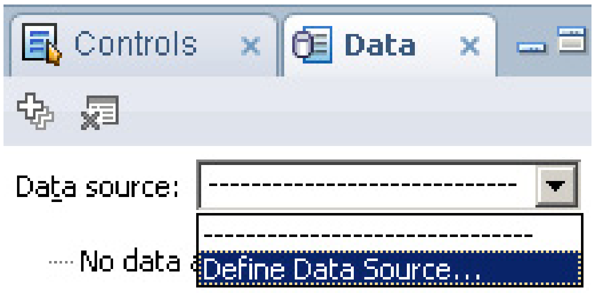

```
<xp:this.resources>
    <xp:styleSheet href="/main.css"></xp:styleSheet>
</xp:this.resources>
```

As you can see, the XPages design tab generates custom schema XML, which the JSF ultimately uses to render your XPages on the Web.

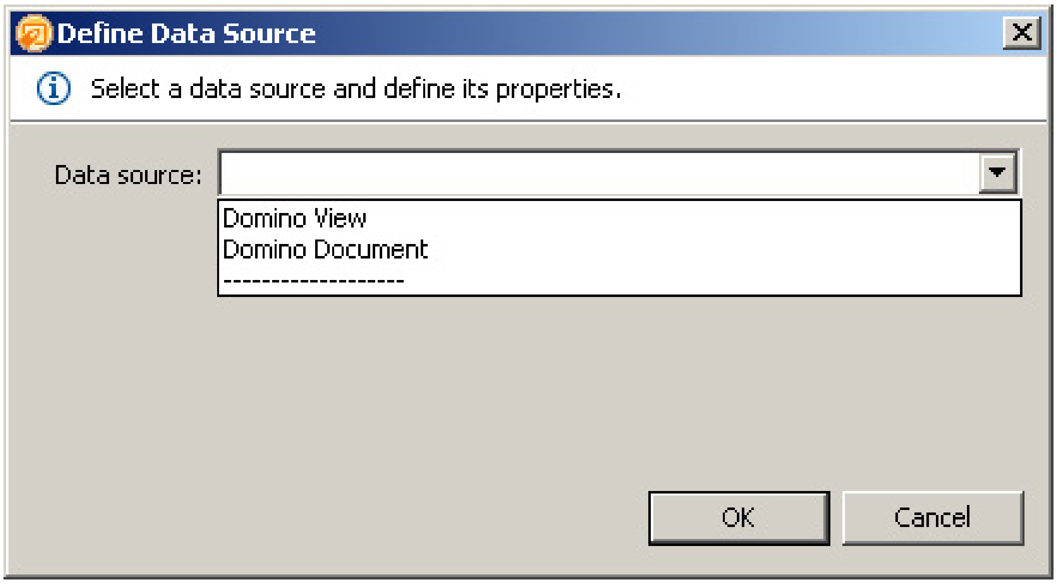
Creating a data source

If your Domino application has a view or a form design element, you are already halfway to having your data source. As luck would have it, the personal address book template has several views and forms to choose from, so follow along to define your data source.

1. In the default perspective (what Domino Designer in Eclipse refers to as its layout), the data source can be defined through the Data tab (Data palette) on the right sidebar in Domino Designer on Eclipse (see Figure 4).
   Figure 4. Defining the data source through the Data tab
   

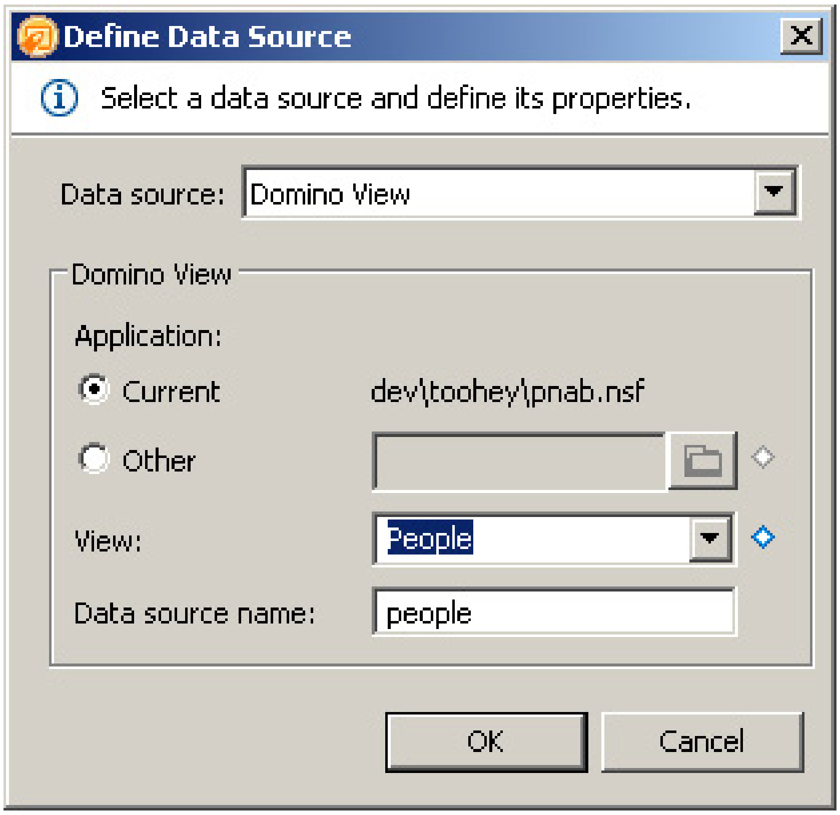
2. Once you are in the Data palette, you can select a data source from the drop-down list provided; however, since this is your first XPage, you'll need to define your first data source. To do this, select **Define Data Source** from the drop-down menu shown in Figure 5.
   Figure 5. Defining data source from the drop-down menu
   

3. This launches the Define Data Source window. You can see two options from the Data source list: Domino View and Domino Document. For this tutorial, you will be defining several different data sources, but for this section, select **Domino View**.
   Figure 6. Launching the define data source window



4. Once you select Domino View from the Define Data Source window, the window changes to the Domino view-specific data source window shown in Figure 7. From here, you can select a view from the selected Domino application (which you will keep as your personal address book application for the purposes of this tutorial).

Figure 7. The Domino view-specific data source window



Data source: Domino View

Domino View

Application:

(●) Current dev\toohey\pnab.nsf

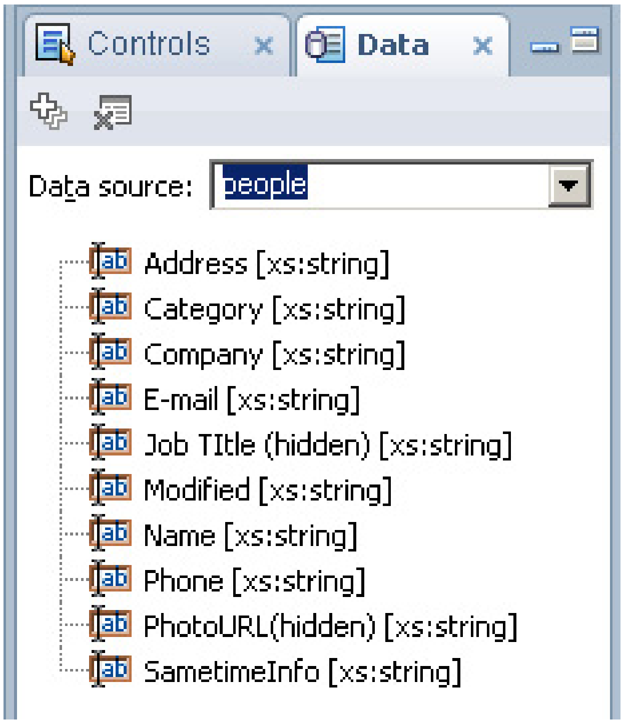
( ) Other

View: People

Data source name: people

5. Once you've defined the Domino view design element, it's of course possible to rename the data source, but for this tutorial, keep the default names. Thus, the data source name remains "people". Select **OK**. The Define Data Source window processes and creates your specified data source in the Data palette, listing each data source object in alphabetical order (see Figure 8).

Figure 8. Data source objects listed in alphabetical order



Data source: people

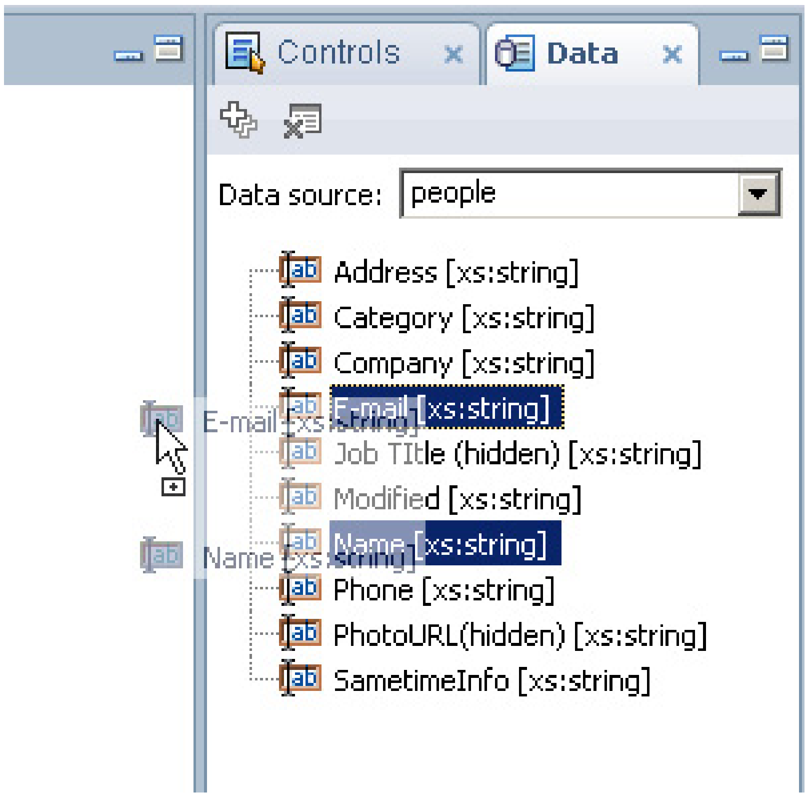
- Address [xs:string]
- Category [xs:string]
- Company [xs:string]
- E-mail [xs:string]
- Job TItle (hidden) [xs:string]
- Modified [xs:string]
- Name [xs:string]
- Phone [xs:string]
- PhotoURL(hidden) [xs:string]
- SametimeInfo [xs:string]

Creating XPage viewPanels with data sources

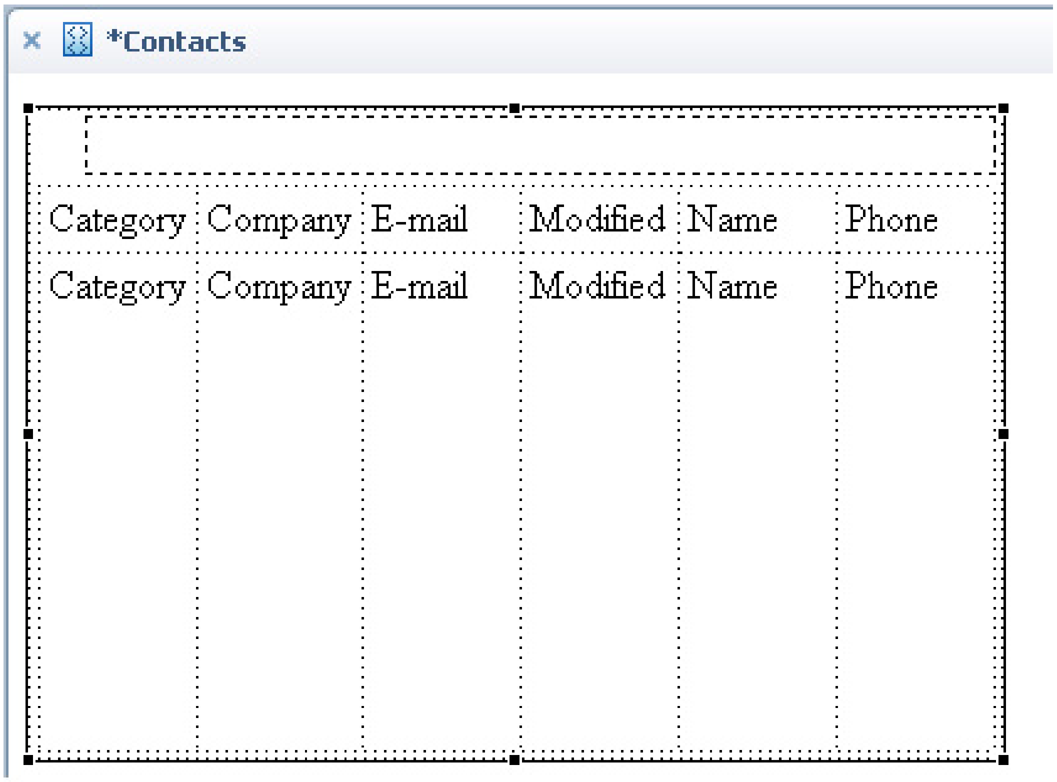
Now that you have your People data source, begin populating your contacts XPage.

1. Select the data source objects (using the Ctrl key on Windows OS for multi-selection) and drag them onto the XPage (see Figure 9).

Figure 9. Dragging and dropping data source objects onto the XPage

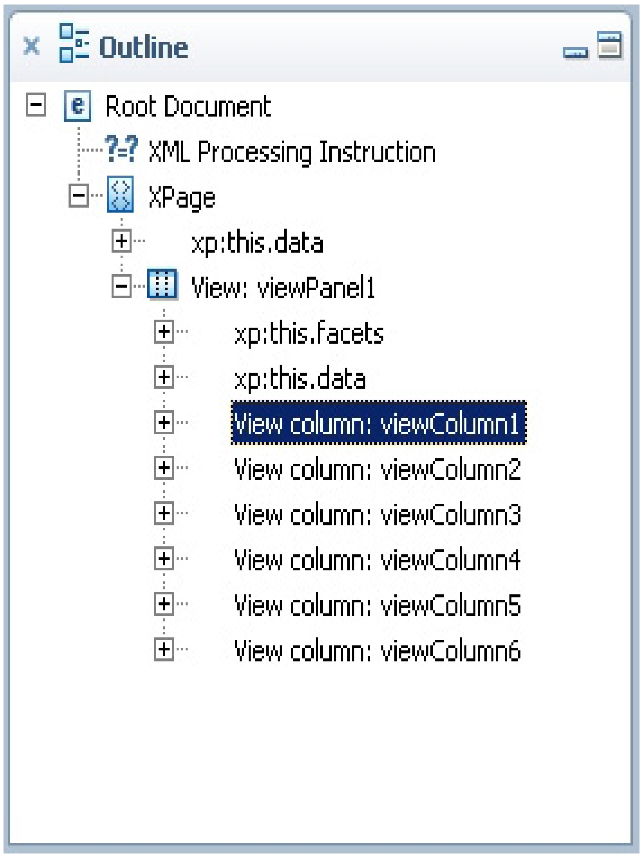


2. Select the following data source objects to mirror the personal address book application's People view:
   - Name
   - Email
   - Phone
   - Company
   - Modified
   - Category
3. Now that you've drag-and-dropped the data source objects onto your contacts XPage, you can see that each data source object has created a column (an XPages viewColumn) in what appears to be a Domino view (an XPages viewPanel). Modify this viewPanel to achieve your desired results (see Figure 10).
   Figure 10. Unsorted data source objects


*Contacts
Category Company E-mail Modified Name Phone
Category Company E-mail Modified Name Phone

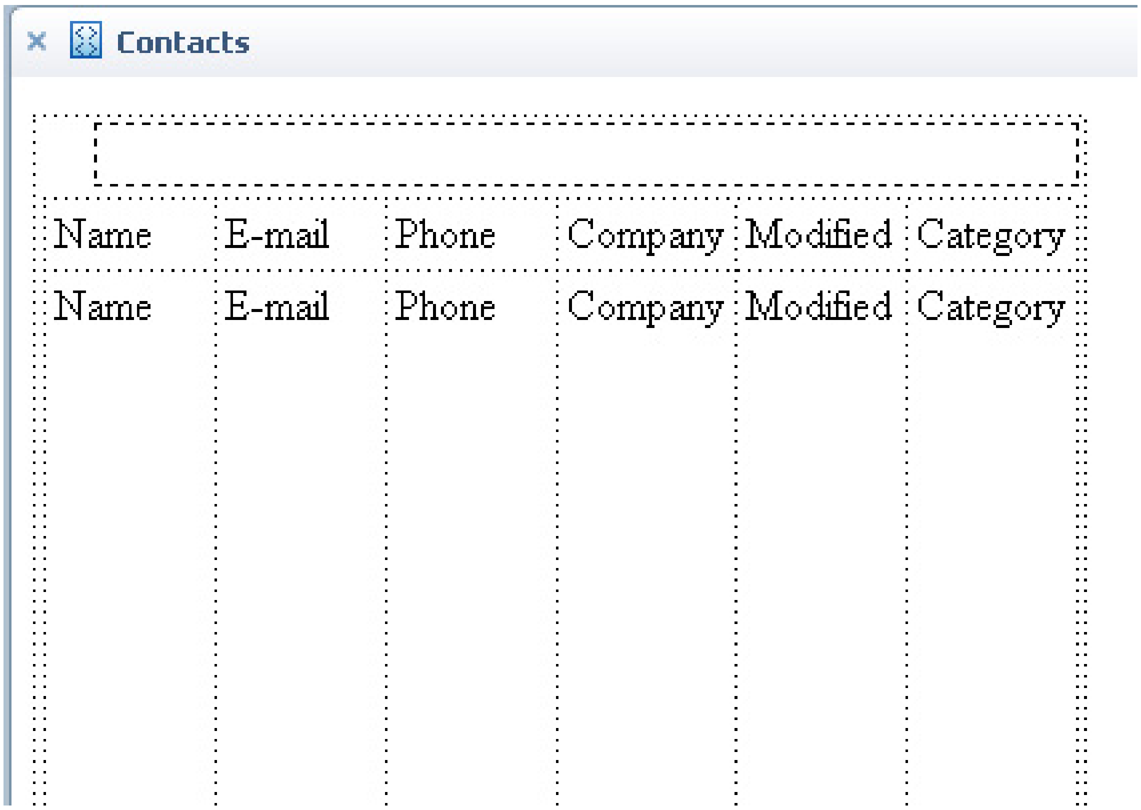
Note that the data source object viewColumns retain the alphabetical order sorting. To mirror the People view of the personal address book application, reorganize these viewColumns so that they match the order of the People view.

4. Toggle to the XPage Source view, because the XPage viewPanel and viewColumns can be easily modified through its XML structure and because you can immediately rename element IDs. Use the Outline palette (located at the bottom-left section in the default Domino Designer in Eclipse perspective) to drag-and-drop and reposition your viewColumns through the tree-like object and element structure (see Figure 11).
Figure 11. Outline panel and object positioning



With these viewColumns repositioned, you can immediately see the similarity between your evolving contacts XPage and the People view-based viewPanel (see Figure 12).

Figure 12. Sorted data source objects



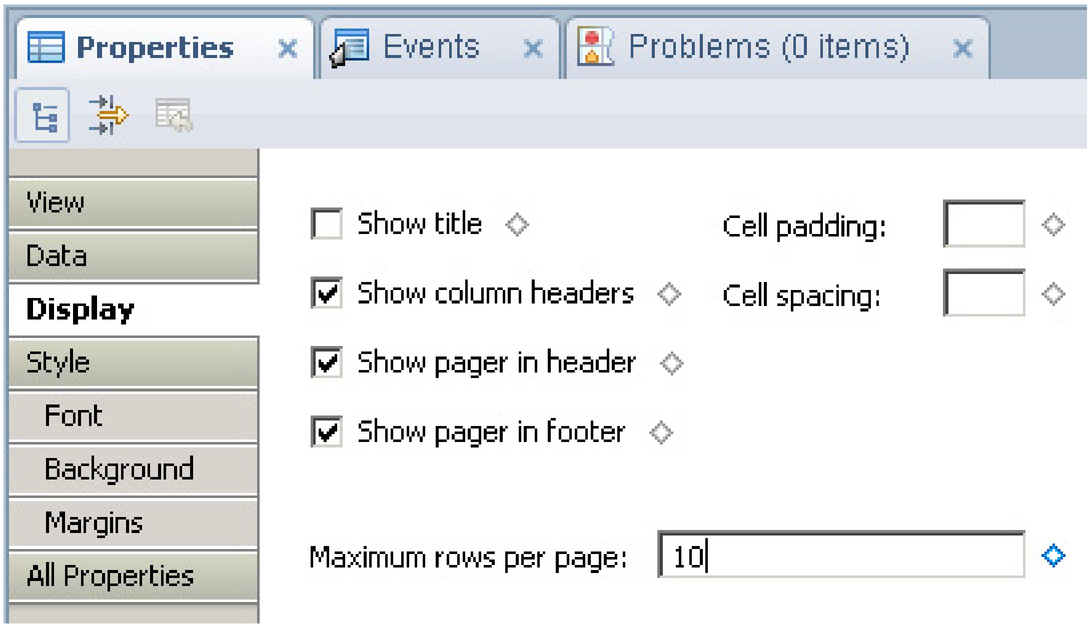
Note: Select both the column header and the column for repositioning, if you wish to maintain the table-like structure.

To duplicate the personal address book application user experience, modify specific XPage viewPanel attributes and properties.

XPage viewPanel properties

The XPage viewPanel properties can be used to set the maximum number of NotesDocuments (or Domino Application data records) to include in the rendered collection. The viewPanel properties also specify the XPage data source, define viewPager placement, and control other viewPanel settings. This tutorial shows you how to modify several viewPanel properties. Figure 13 shows the viewPanel properties with **Maximum rows per page** preference at the bottom.

Figure 13. viewPanel properties -- maximum rows per page



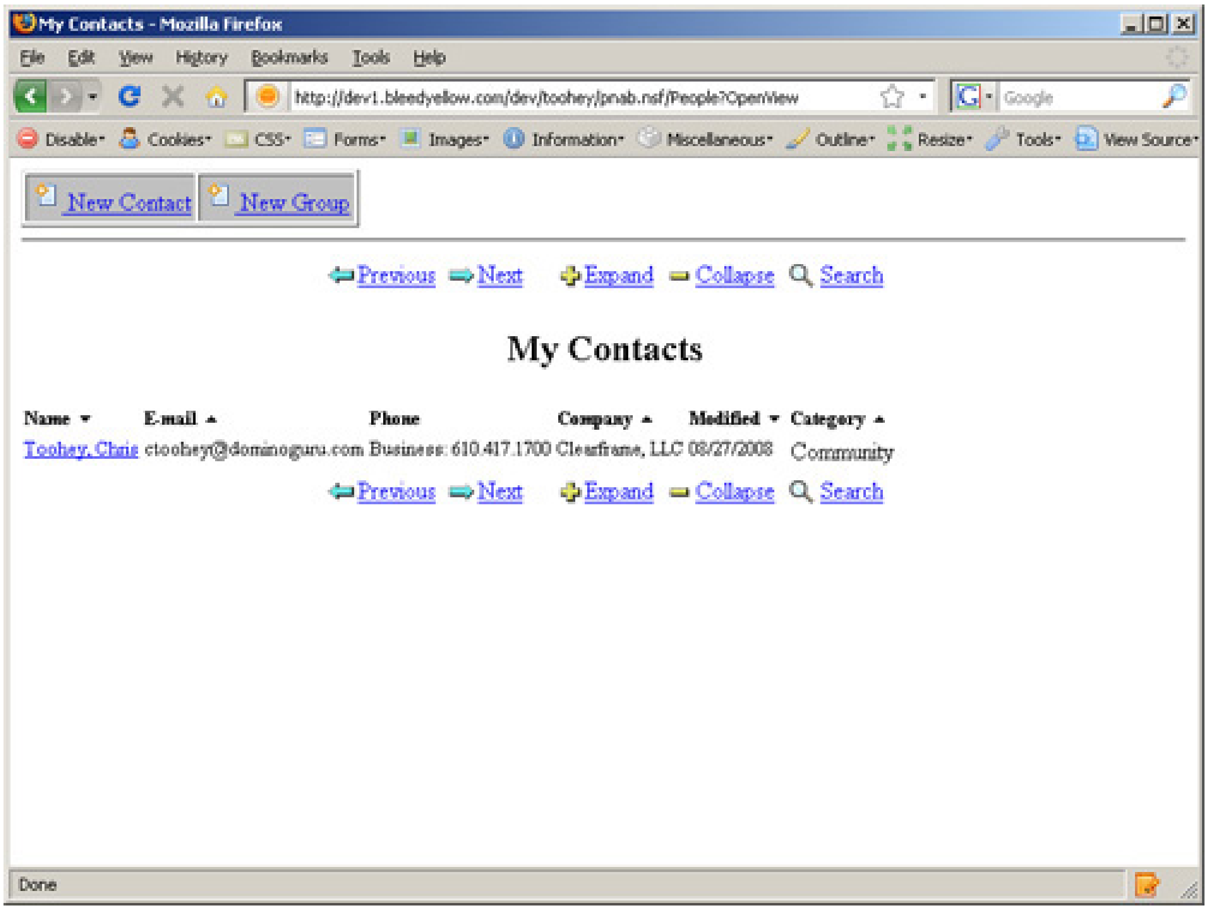
[ ] Show title
[✓] Show column headers
[✓] Show pager in header
[✓] Show pager in footer
Cell padding:
Cell spacing:
Maximum rows per page: 10

The Maximum rows per page value, like many other XPage property values, can be set as static or computed, but for the purposes of this tutorial, set the maximum rows per page to 10.

Contacts preview

Now that you've repositioned the viewColumns in your viewPanel to mirror the People view from the personal address book application and updated the viewPanel properties, look at the People view rendered on a Web browser shown in Figure 14.

Figure 14. Standard People view preview

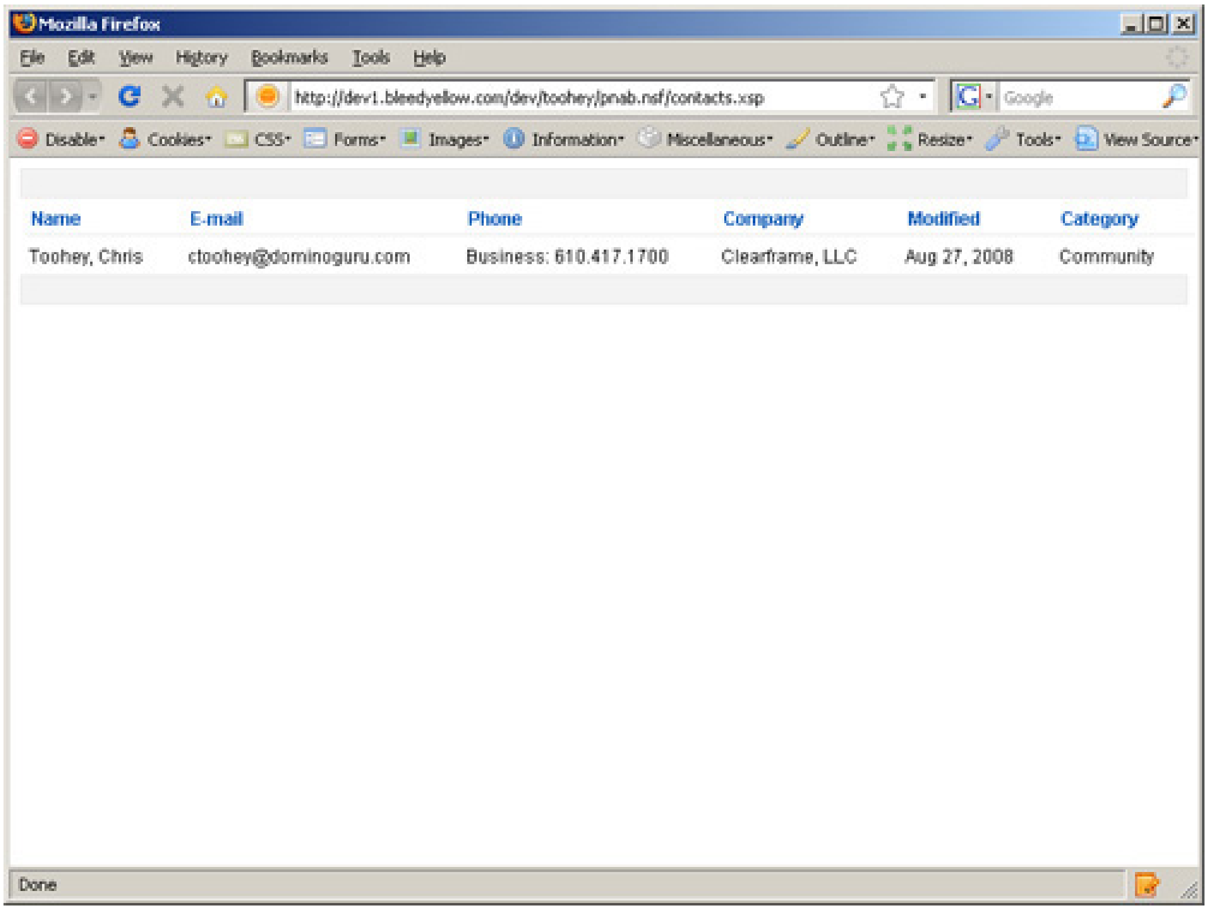

My Contacts - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/People?OpenView
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source
New Contact
New Group
Previous Next Expand Collapse Search
Done

My Contacts

Name	E-mail	Phone	Company	Modified	Category
Toohey, Chris	ctoohey@dominoguru.com	Business: 610.417.1700	Clearframe, LLC	08/27/2008	Community

The view is certainly functional, but the default Domino Design Element Web browser client rendering does not compete with the functional and stylized experiences most users now expect. Figure 15 shows how the contacts XPage looks so far.

Figure 15. Contacts XPage preview


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/contacts.xsp
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source
Done

Name	E-mail	Phone	Company	Modified	Category
Toohey, Chris	ctoohey@dominoguru.com	Business: 610.417.1700	Clearframe, LLC	Aug 27, 2008	Community

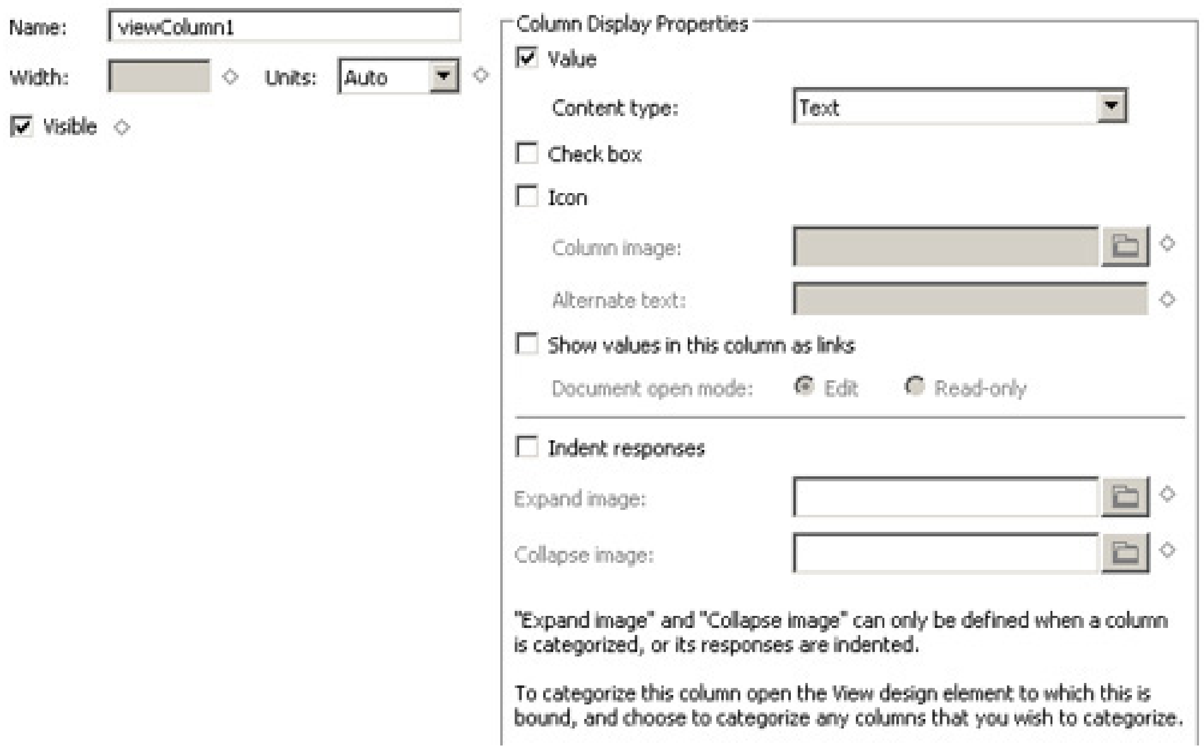
This view looks good considering you really did nothing more than drag-and-drop a few data source objects onto an XPage, right? The problem, however, is that although you see the contents of the People view, you can't do anything with them. You need to create a link in the first viewColumn contents that opens the given NotesDocument, a task covered in the next section.

Defining viewColumn actions and advanced properties

As mentioned, the viewColumn for your contacts does not allow you to open a NotesDocument. Consider changing the contents of the Name viewColumn to a clickable link that launches the given NotesDocument in read-only mode. Select several configuration options from the view column properties of the Name viewColumn.

The viewColumn properties are located at the bottom properties area on the default Domino Designer on Eclipse perspective, and become available once a viewColumn is highlighted.

Figure 16. viewColumn properties -- column display properties


Name: viewColumn1
Width: Units: Auto
[✓] Visible

Column Display Properties
[✓] Value
Content type: Text
[ ] Check box
[ ] Icon
Column image:
Alternate text:
[ ] Show values in this column as links
Document open mode: (●) Edit ( ) Read-only
[ ] Indent responses
Expand image:
Collapse image:

"Expand image" and "Collapse image" can only be defined when a column is categorized, or its responses are indented.

To categorize this column open the View design element to which this is bound, and choose to categorize any columns that you wish to categorize.

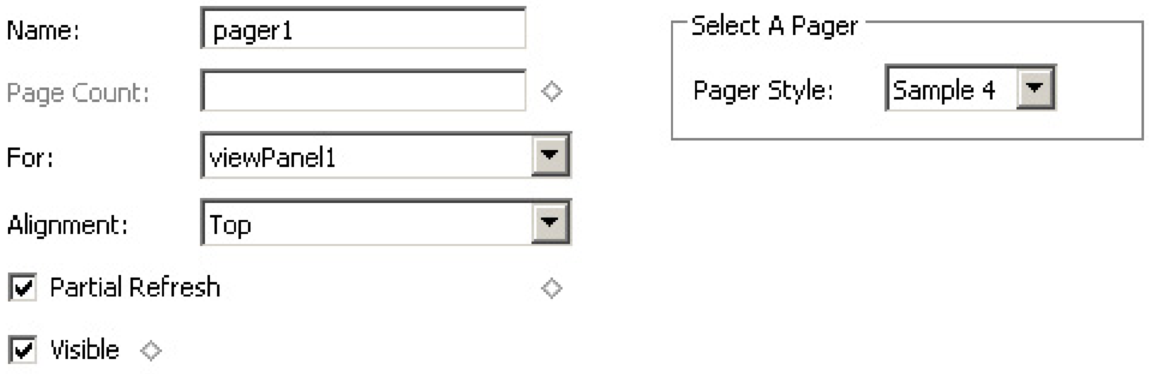
To control the viewColumn properties that allow you to open the NotesDocument for the particular entry in the collection, check **Show values in this column as links**. Once checked, you can specify the edit-state of the NotesDocument, electing whether the NotesDocuments be in edit or read modes when opened via the link generated in this viewColumn. The view column properties additionally allow you to specify static and computed viewColumn styling information, content-type declarations, and other such viewColumn properties.

XPage pagers

A pager in XPages is a control for navigation of data source collections. These pager controls allow for specific viewPanel navigation, Ajax-based collection navigation, and custom-defined pager types. For your contacts XPage, you're going to configure top and bottom pagers, both using a preset theme that lends itself to collection paging, and enabling the Ajax-based collection navigation.

Selecting the top pager in your viewPanel gives you the pager preferences, which are located at the bottom of the Domino Designer on Eclipse client in the default perspective. Select Sample 4 for your pager style, check **Partial Refresh**, which gives you the AJAX-based collection navigation, set the alignment, and specify that this pager will be refreshing the content of your viewPanel (viewPanel1).

Figure 17. XPage pager properties



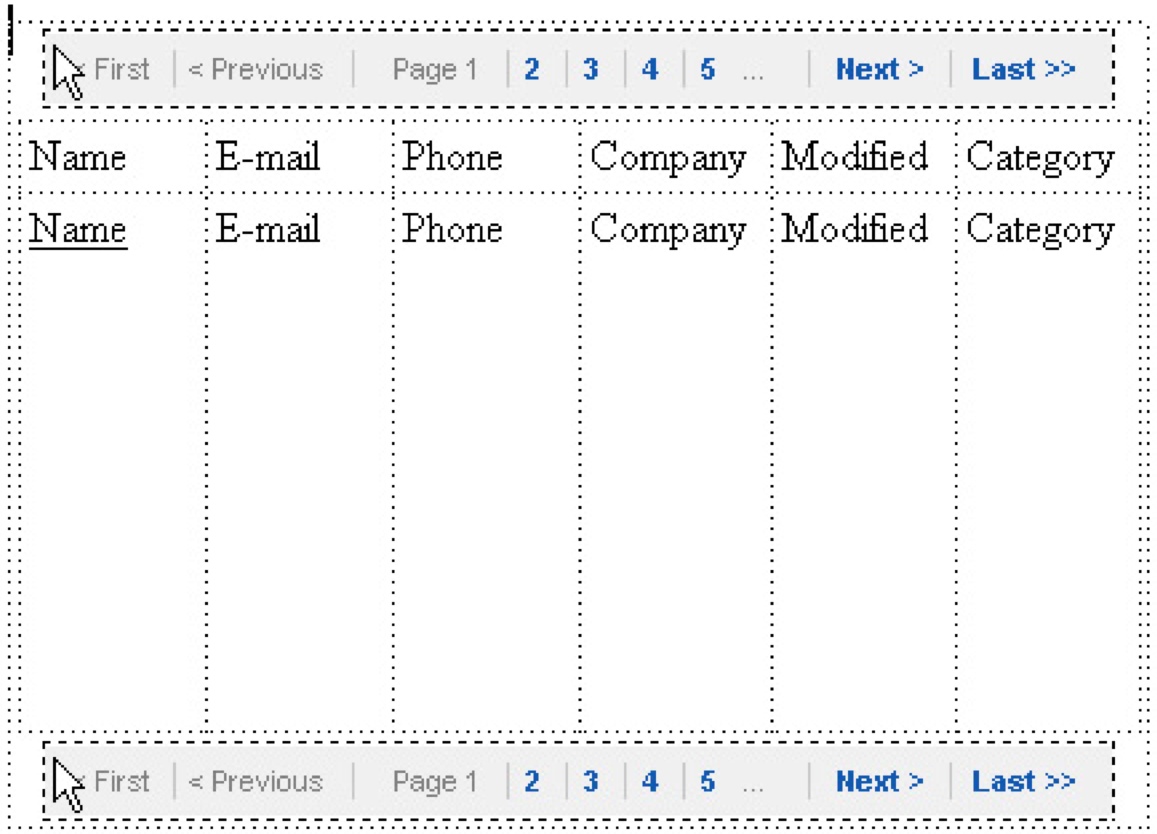
Name:	pager1
Page Count:	
For:	viewPanel1
Alignment:	Top
☑ Partial Refresh	
☑ Visible	

Select A Pager

Pager Style: Sample 4

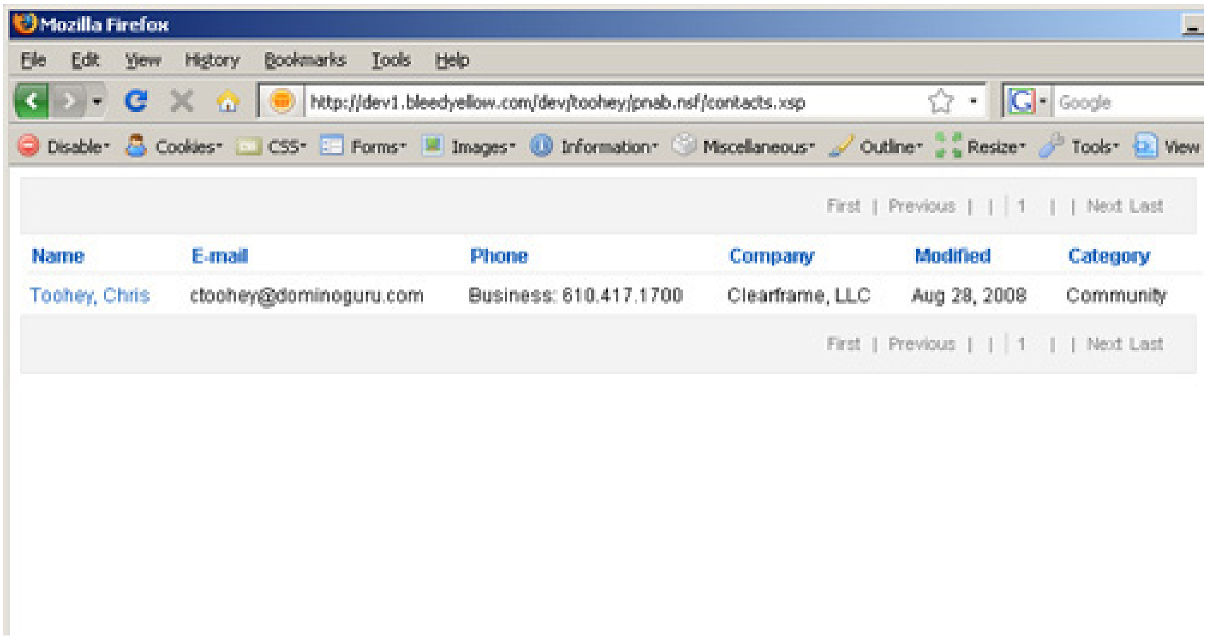
Now that you've set the pager preference, the Design viewPanel displays an example of your pager (see Figure 18).

Figure 18. XPage pager Design panel preview



Now that you've added the pagers to the header and footer of the viewPanel, look at the Contacts XPage, shown in Figure 19.

Figure 19. XPage pager browser preview

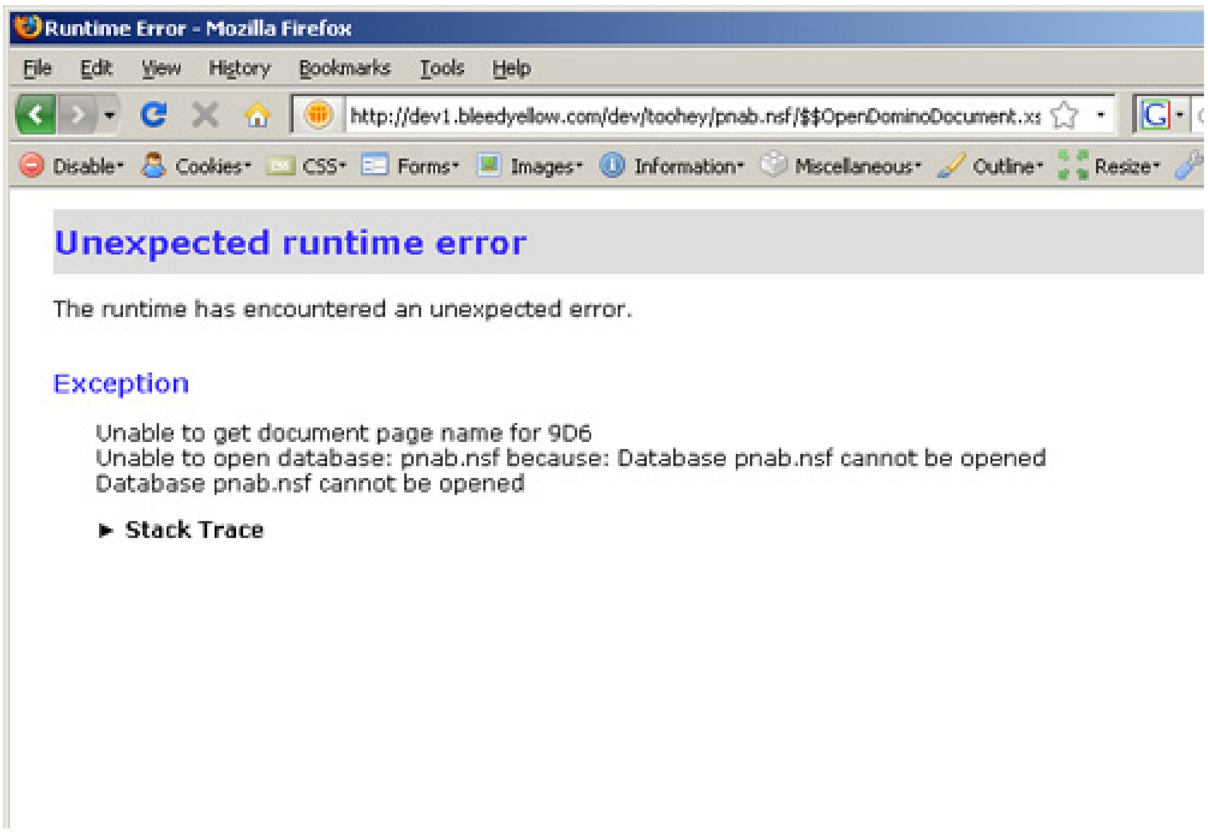

Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/contacts.xsp
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View
First | Previous | | 1 | | Next Last

Name	E-mail	Phone	Company	Modified	Category
Toohey, Chris	ctoohey@dominoguru.com	Business: 610.417.1700	Clearframe, LLC	Aug 28, 2008	Community

Your first XPage error

If you click on the Name viewColumn now that you've specified as your link to open the NotesDocuments, you get an error message as shown in Figure 20.

Figure 20. XPage error example


Runtime Error - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/$$OpenDominoDocument.xs
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize

Unexpected runtime error

The runtime has encountered an unexpected error.

Exception

```
Unable to get document page name for 9D6
Unable to open database: pnab.nsf because: Database pnab.nsf cannot be opened
Database pnab.nsf cannot be opened
```

► Stack Trace

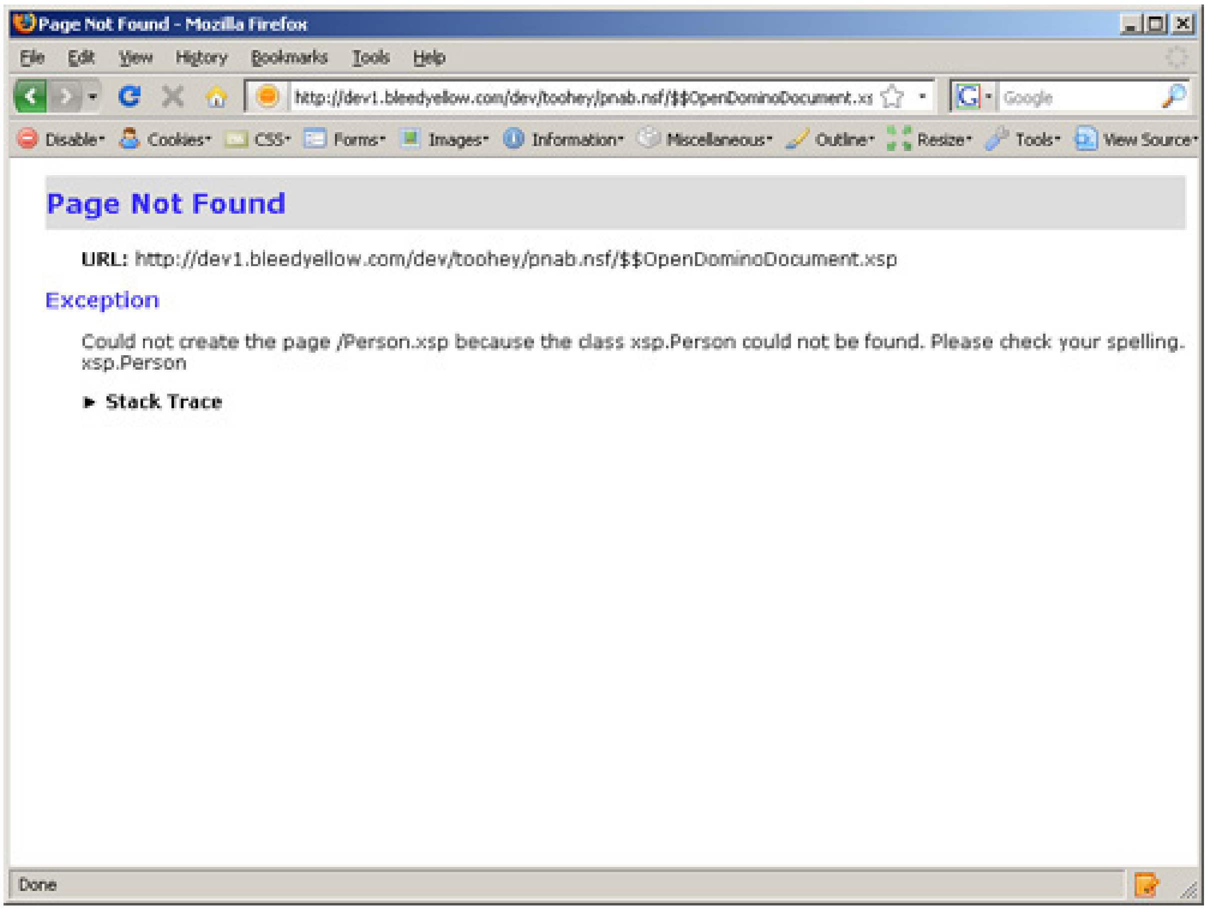
The URL syntax of the viewColumn link is as follows:
`http://server/pnab.nsf/$$OpenDominoDocument.xsp?databaseName=pnab.nsf&doc`

Although this error message might have you thinking that there is an issue with the pnab.nsf (which is the OS-level name of the example personal address book application), that's not the case. Focus on the first line of the error message. To illustrate the actual issue generating your error and resulting in your inability to open the requested NotesDocument, modify the URL to the following:
`http://server/pnab.nsf/$$OpenDominoDocument.xsp?documentId=9D6&action=edi`

In the above URL, the databaseName attribute and value from the query string have been removed.

The error message generated from this modified request URL informs you that the JSF (the XPage rendering engine) is unable to locate an XPage to render the NotesDocument (see Figure 21).

Figure 21. XPage Error Example -- modified URL command


Page Not Found - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/$$OpenDominoDocument.xs
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize Tools View Source
Done

Page Not Found

URL: http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/$$OpenDominoDocument.xsp

Exception

Could not create the page /Person.xsp because the class xsp.Person could not be found. Please check your spelling.
xsp.Person

▶ Stack Trace

To address this issue, simply create an XPage that will be used to render the NotesDocument. This makes sense if you take into consideration that you're now in the XPages environment and not in the standard Domino Design Element environment. Since you've only defined a View XPage, there is no NotesDocument XPage to display the requested NotesDocument. You'll create that in the next section.

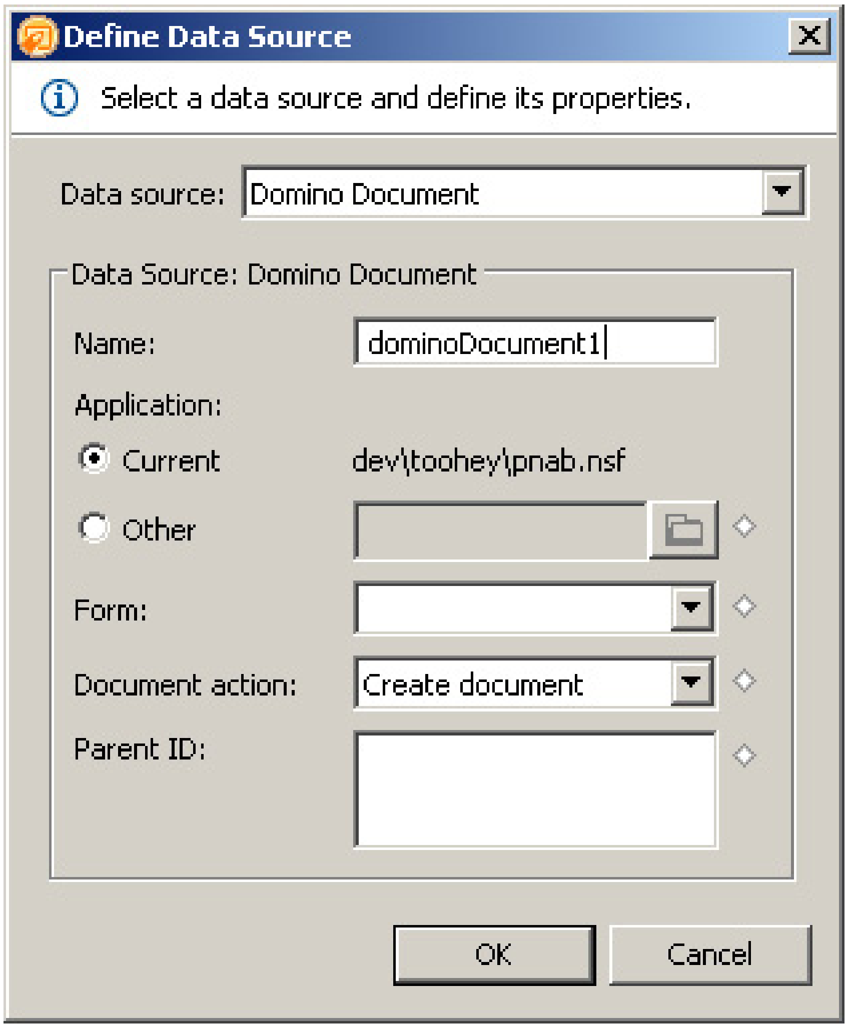
Section 3. Build a NotesDocument-type XPage

In the previous section, you created a simple Domino View XPage, which allows you to read and page through Domino Data in your personal address book application. In this section, you create a simple Domino Document XPage, which you use to render individual NotesDocument content and initiate functional interaction between your XPage-based user interface and your Domino data, creating a link between your Domino View XPage and your Domino Document XPage.

Domino Document data sources

Start the initial build of the NotesDocument XPage that you'll use to render NotesDocuments from the Contacts XPage by creating a new XPage called Person. Immediately define a Document Data Source from the Data panel by following the same Define Data Source window prompts you used to create the viewPanel on the Contacts XPage.

Figure 22. Define data source window -- Domino document



Select a data source and define its properties.

Data source: Domino Document

Data Source: Domino Document

Name: dominoDocument1

Application:

[✓ Current] dev\toohey\pnab.nsf

Other

Form:

Document action: Create document

Parent ID:

When selecting **Domino Document**, notice that a different set of options are provided specific to NotesDocument creation, modification, and read-only rendering. Change the data source name to **Person**, select the **Person** Domino form design element, and change the option from **Create document** to **Open document**.

Once you've changed these options and selected **OK** from the data source window, you see a population of all data source objects in the data panel that match the

NotesItems of the person form. This option lets you include any NotesItem value from your NotesDocument in your people XPage.

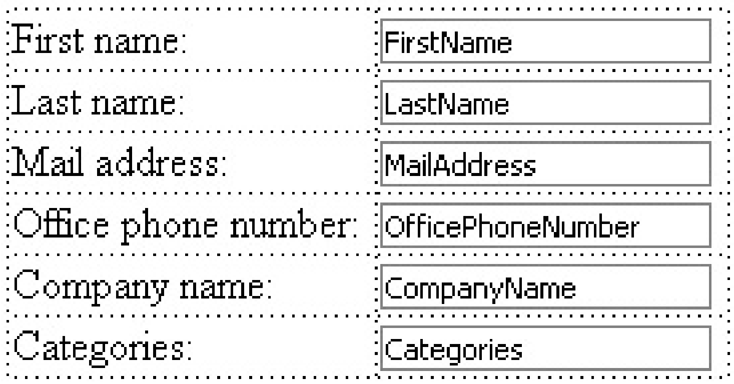
XPage data object tables

For this example, keep your people XPage relatively simple and initially only include the following data source objects:

- FirstName
- LastName
- MailAddress
- OfficePhoneNumber
- CompanyName
- Categories

Select the above data source objects and use them to populate your Person XPage via drag-and-drop to create an XPage data object table. An XPage data object table is to a NotesDocument what an XPage viewPanel is to a NotesView, and renders with a label-and-edit box control combination for each Data Source Object added (see Figure 23).

Figure 23. XPage data object table example



First name:	FirstName
Last name:	LastName
Mail address:	MailAddress
Office phone number:	OfficePhoneNumber
Company name:	CompanyName
Categories:	Categories

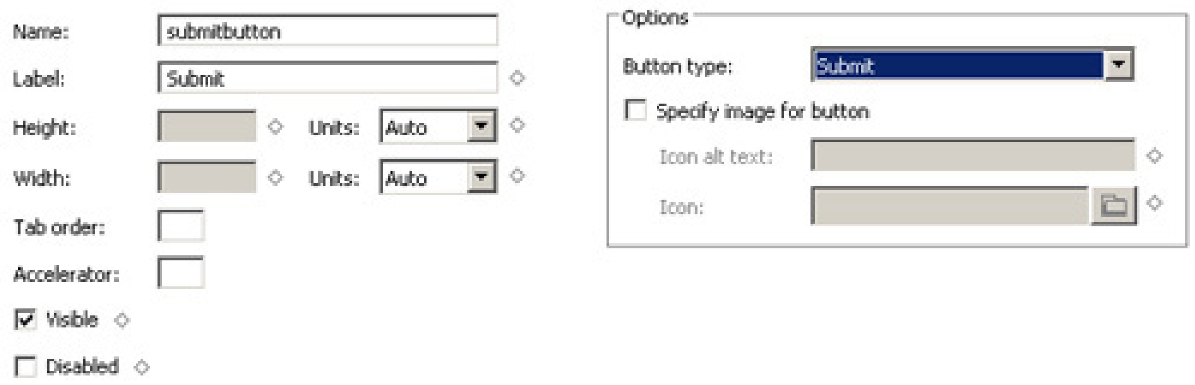
As with the XPage viewPanel, the table rows are initially listed in alphabetical order, and will require you to reposition to a more standard, logical order. Again, as in the viewPanel, you can easily do this in the Source view or through the Outline panel.

Once you have repositioned the Table rows, add an Edit button control so you can toggle the read-only and editable state of the XPage-rendered NotesDocument. To use the same XPage for both the read-only and editable states, modify both the Button control action and its label.

Adding button controls to an XPage data object table

Up to this point, you have really concentrated on the Data panel when creating XPage Data Objects for your viewPanels and Tables via drag-and-drop functionality. However, the Control panel (located on the right sidebar panel next to the Data panel in the Domino Designer on Eclipse default Eclipse Perspective), contains various Controls used for the content management capabilities available in XPages. Add a simple Button Control to your Form either from the Control panel, or from the Create\Other Menu option. Once added to the XPage, you can edit the Button Control label, as well as its functional parameters by using the Button Control Properties (see Figure 24).

Figure 24. Button control object properties



Name: submitbutton
Label: Submit
Height: Units: Auto
Width: Units: Auto
Tab order:
Accelerator:
[✓] Visible
[ ] Disabled

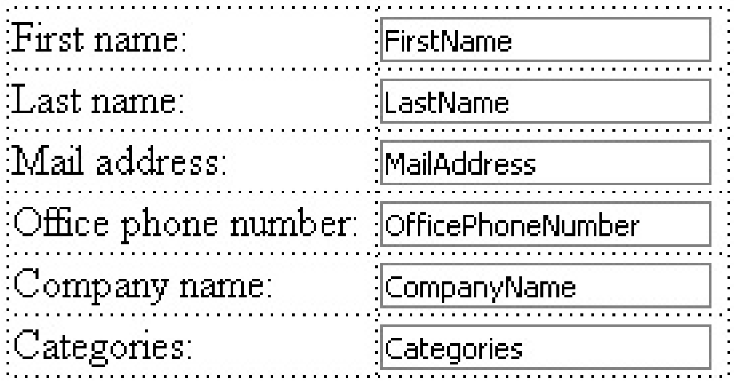
Options
Button type: Submit
[ ] Specify image for button
Icon alt text:
Icon:

From here, as shown in Figure 24, you can change the button control instance name, label, and the button type. For the purposes of this tutorial, change the button control to a submit button, changing both its name, label, and button type to fit your specific needs.

Using the Person XPage

Make one more preference modification to your contacts XPage to allow it to render NotesDocument requests with your person XPage by setting the `view` properties in the viewPanel (see Figure 25).

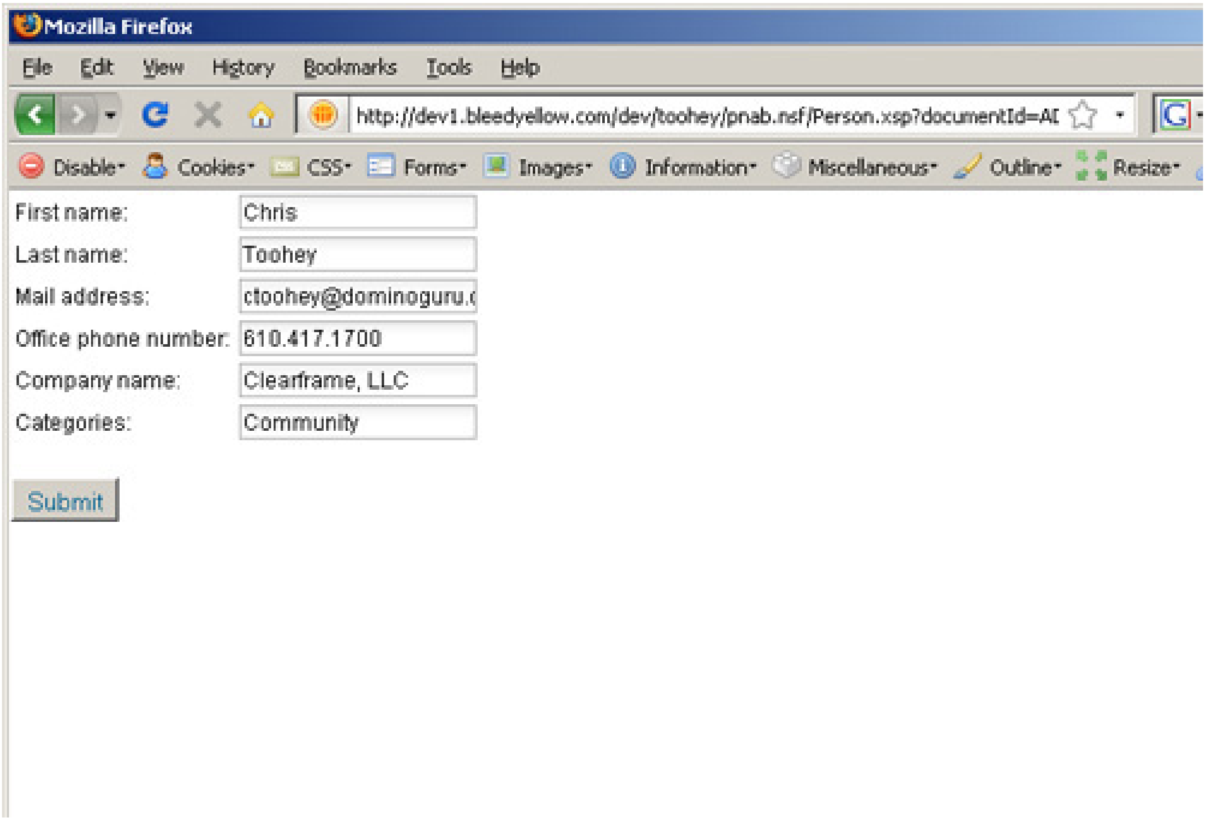
Figure 25. viewPanel view properties



First name:	FirstName
Last name:	LastName
Mail address:	MailAddress
Office phone number:	OfficePhoneNumber
Company name:	CompanyName
Categories:	Categories

By setting the pageName option (labeled **At runtime, open selected document using**) to **Person**, you are now finally ready to see the person XPage in action. To do this, simply open your contacts XPage, and click on the value in the name viewColumn specific to the NotesDocument you want to open (see Figure 26).

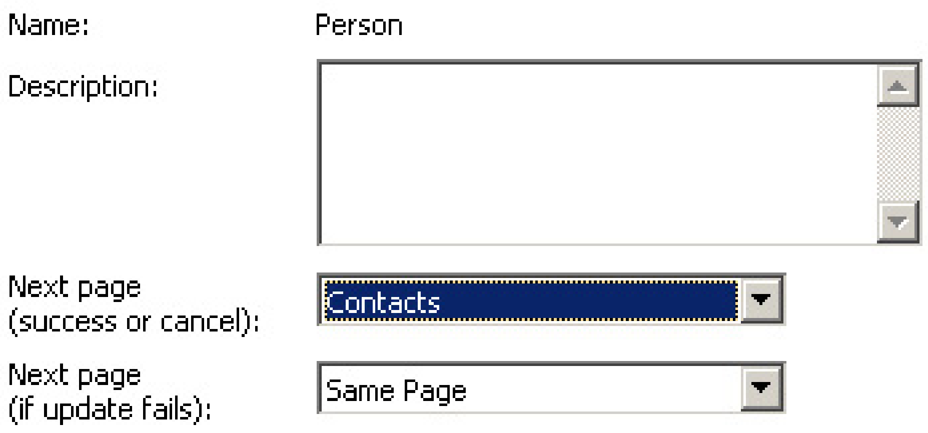
Figure 26. Person XPage for NotesDocument edit mode rendering


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/Person.xsp?documentId=AI
Disable Cookies CSS Forms Images Information Miscellaneous Outline Resize

First name:	Chris
Last name:	Toohey
Mail address:	ctoohey@dominoguru.c
Office phone number:	610.417.1700
Company name:	Clearframe, LLC
Categories:	Community

Now that you can open NotesDocuments in XPages in edit mode, configure the post-submission redirect for your person XPage. In non-XPages Domino Web development, this is typically handled via the `$$Return` reserved FIELD. This functionality, however, is directly configurable from the person XPage preferences as shown in Figure 27.

Figure 27. XPage submission redirection options



Name: Person

Description:

Next page (success or cancel): Contacts

Next page (if update fails): Same Page

By setting the navigationRules preferences to redirect to the contacts XPage on successful submission or cancellation, you now have a completely functional XPages representation of contact navigation and person NotesDocument management. However, looking at this example of XPage functionality really begs the question: "I can do that today with regular Domino Web development, so why would I choose to use XPages?" The next section will answer this question.

Section 4. Add an XPage front-end

So far you have reviewed very basic XPage examples of the rendering of data objects for both Domino View and Domino Document-based data sources. In this section, you use those techniques in combination with new control objects to create a fully-functional XPage front-end to your Domino Web-based personal Web application, allowing users of the new Web 2.0 application to input their own contact information.

Defining your application user interface

Now that you've moved beyond the simple duplication of existing functionality within the person address book application, you can create a simple yet useful Web-based front end using the more advanced features and functionality provided by XPages.

Start by defining your expected end-result XPages-based user interface:

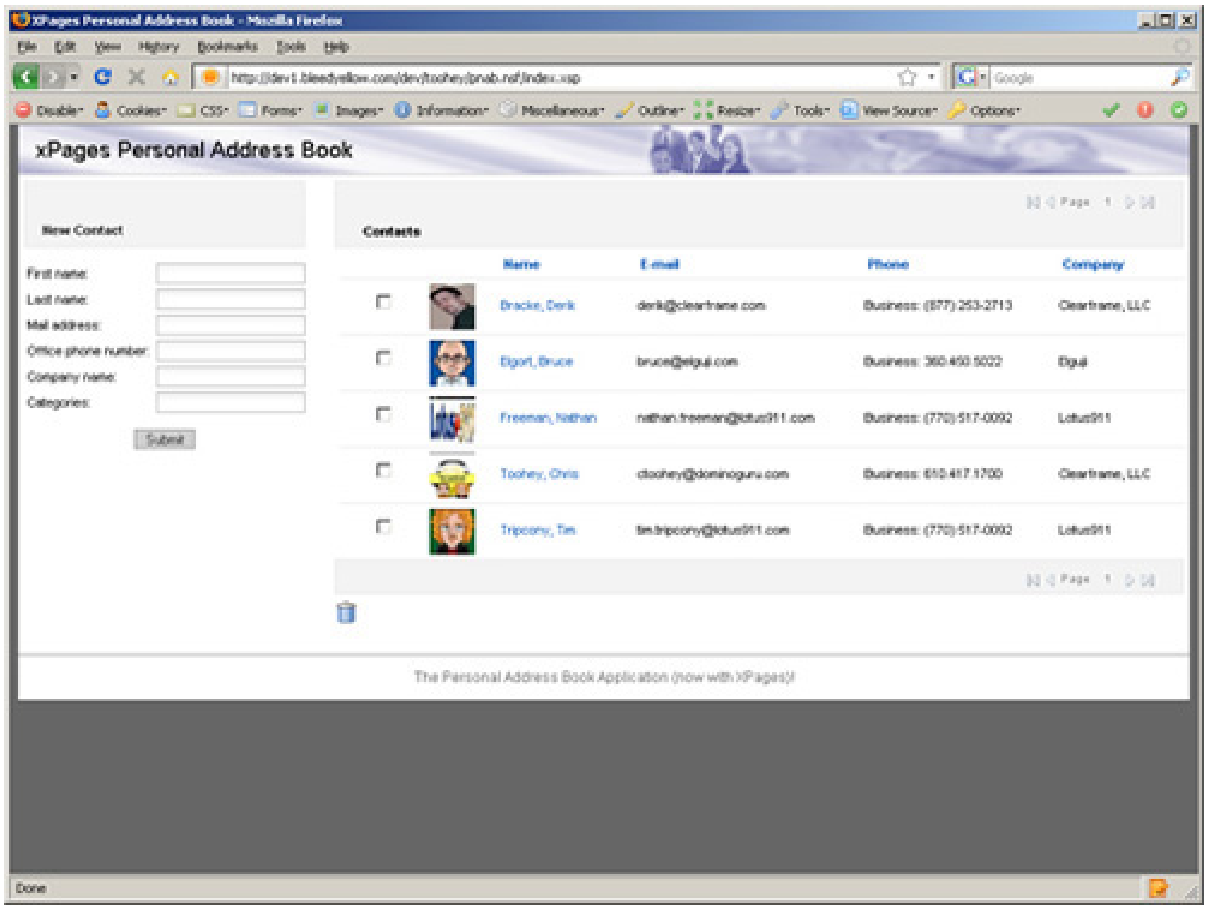
- A simple main page that displays contact information in a listing that mirrors the people view
- In-view inclusion of the ContactPhoto per contact
- Multi-document deletions with confirmation prompt
- A CSS-based layout that avoids tables
- Simple contact form for quick contact entry

The next section covers how to use XPages so that the user experience feels like an application.

Main page preview

This might be considered cheating, but I'm going to show the end-result before you begin to build any of the main page, but being able to see what you're building lets you better visualize the architecture process (see Figure 28).

Figure 28. Index XPage main page preview



XPages Personal Address Book - Mozilla Firefox

http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/index.xsp

xPages Personal Address Book

New Contact

First name:	
Last name:	
Mail address:	
Office phone number:	
Company name:	
Categories:	

Submit

Contacts

Name	E-mail	Phone	Company
Dracke, Derk	derik@cleartrame.com	Business: (877) 253-2713	Cleartrame, LLC
Elgort, Bruce	bruce@elgui.com	Business: 360 450 5022	Elguji
Freeman, Nathan	nathan.freeman@lotus911.com	Business: (770) 517-0092	Lotus911
Toohey, Chris	ctoohey@dominoguru.com	Business: 610 417 1700	Cleartrame, LLC
Tripcony, Tim	tim.tripcony@lotus911.com	Business: (770) 517-0092	Lotus911

The Personal Address Book Application (now with XPages)!

Done

The main page layout is based on a "fixed-layout" designed for 1024x768 resolution display.

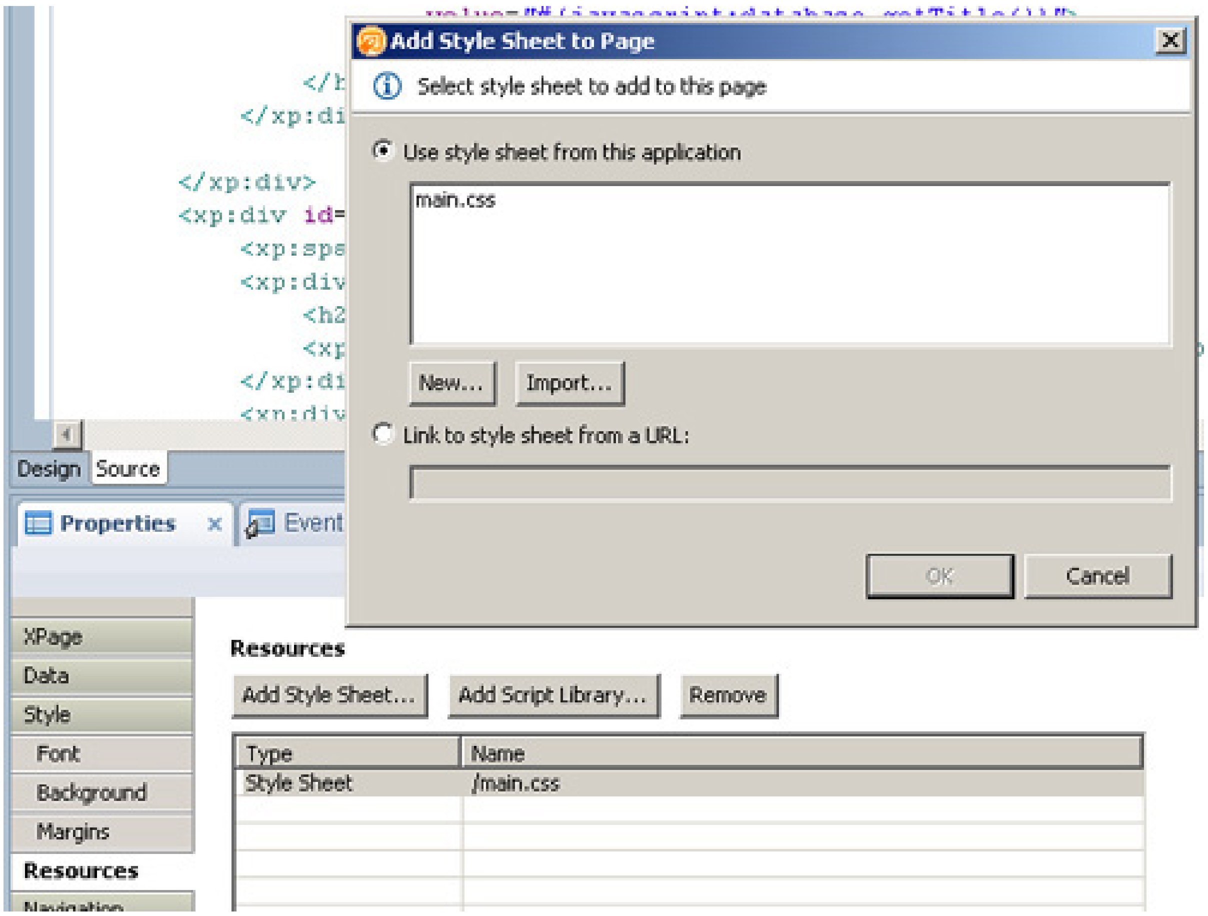
Styling and layout

To construct the main page, create a new XPage named index.

Create a new style sheet resource named main.css, which controls the CSS-based layout as well as the overall design and presentation of your index XPage. Keep both of these design elements, the index XPage and the main.css style sheet, open in your Domino Designer on Eclipse client, since you will be working with each for the remainder of this tutorial.

To add your style sheet to the index XPage, use the **Add Style Sheet to Page**, window, which is located in the resources section of the XPage properties panel (see Figure 29).

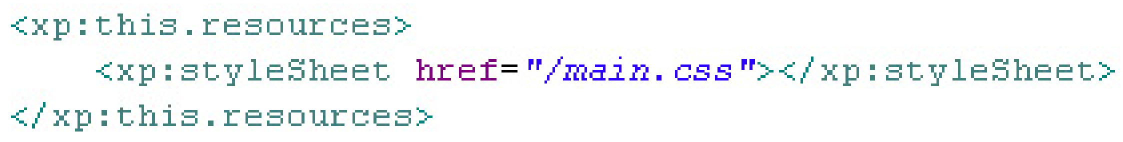
Figure 29. XPage Add Style Sheet window



Type	Name
Style Sheet	/main.css

Alternately, you can add your style sheet directly through the source panel as shown in Figure 30.

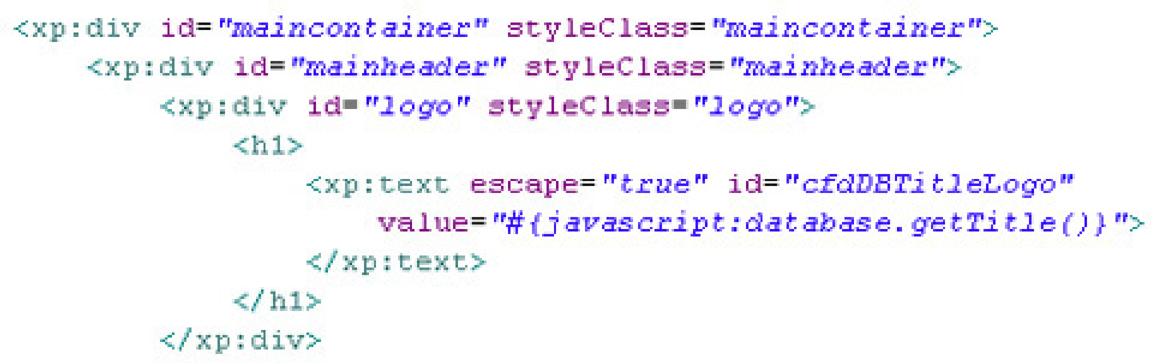
Figure 30. XPage style sheet inclusion markup



```
<xp:this.resources>
    <xp:styleSheet href="/main.css"></xp:styleSheet>
</xp:this.resources>
```

From here, create DIV control objects. These DIV control objects, or DIVs, render as HTML DIV elements at runtime, and used for the CSS-based layout construction. Once you become more familiar with the source panel -- and aided by the XPages source panel type-ahead -- adding DIV Control Objects is as easy as writing markup in your preferred text editor (see Figure 31).

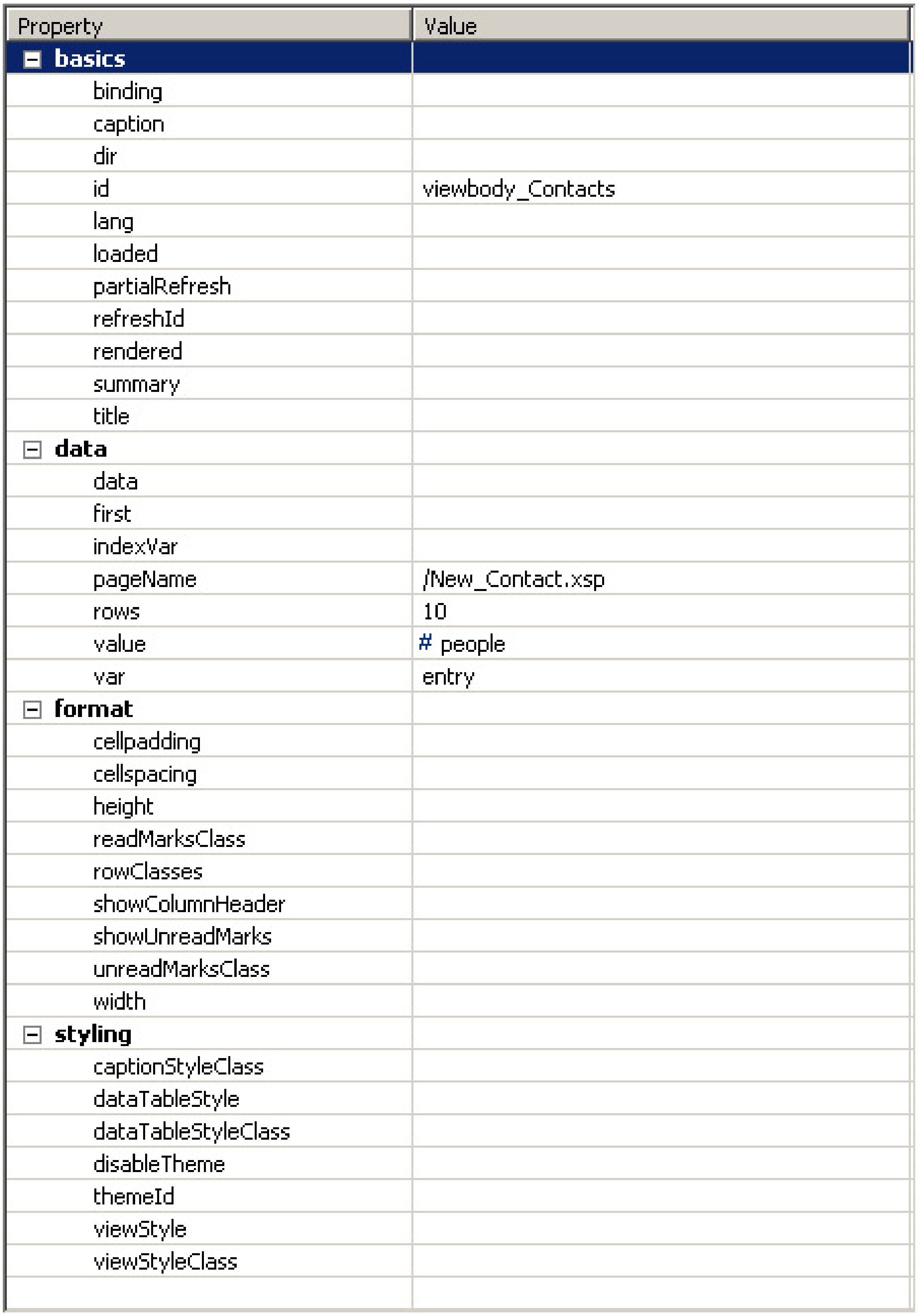
Figure 31. DIV control object markup example



```
<xp:div id="maincontainer" styleClass="maincontainer">
    <xp:div id="mainheader" styleClass="mainheader">
        <xp:div id="logo" styleClass="logo">
            <h1>
                <xp:text escape="true" id="cfdDBTitleLogo"
                    value="#{javascript:database.getTitle()}">
                </xp:text>
            </h1>
        </xp:div>
```

Alternately, these same control object attributes can be modified from the object's properties panel in the Domino Designer on Eclipse client, as seen in Figure 32 in the All Properties section.

Figure 32. ID and styleClass assignment



Property	Value
⊟ **basics**	
binding	
caption	
dir	
id	viewbody_Contacts
lang	
loaded	
partialRefresh	
refreshId	
rendered	
summary	
title	
⊟ **data**	
data	
first	
indexVar	
pageName	/New_Contact.xsp
rows	10
value	# people
var	entry
⊟ **format**	
cellpadding	
cellspacing	
height	
readMarksClass	
rowClasses	
showColumnHeader	
showUnreadMarks	
unreadMarksClass	
width	
⊟ **styling**	
captionStyleClass	
dataTableStyle	
dataTableStyleClass	
disableTheme	
themeId	
viewStyle	
viewStyleClass	

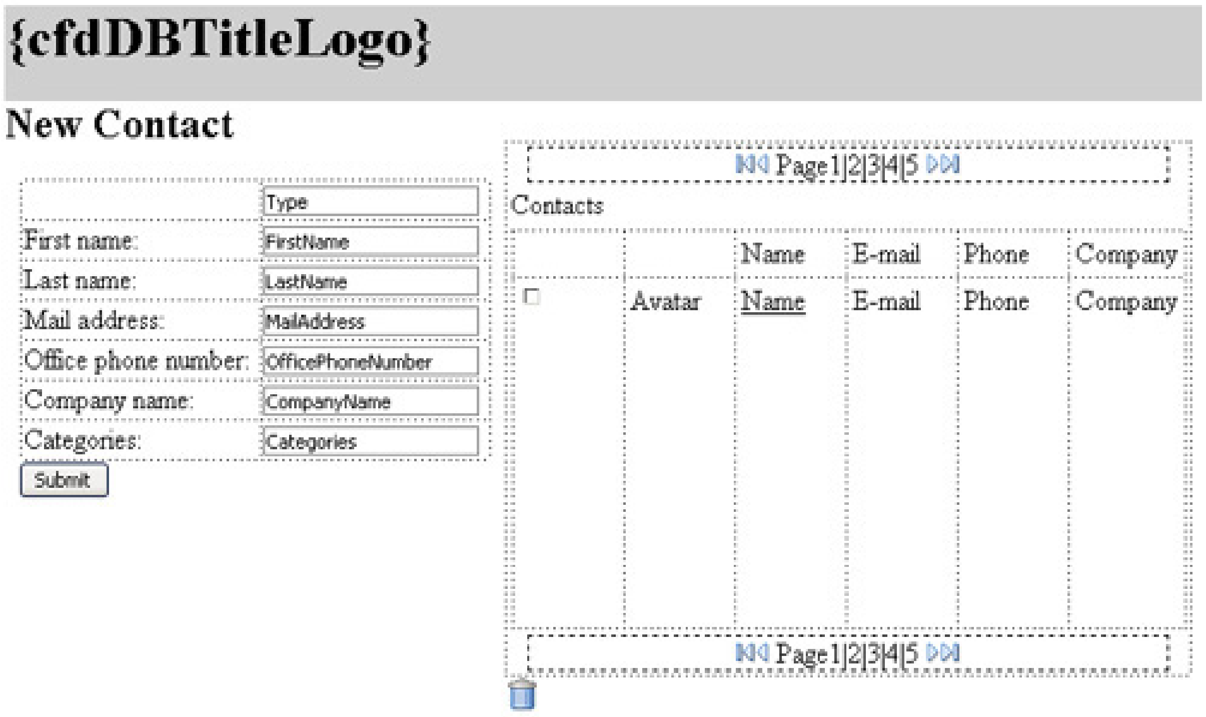
Once you've added your DIV control Objects to your index XPage, apply some basic layout style in your main.css style Sheet. Start with some very standard CSS shown in Listing 1.

Listing 1. Standard CSS

```
body {
background-color:
#666;
margin: 0px;
padding: 0px;
text-align:
center;
}
div.maincontainer
{
width: 1000px;
margin: 0px auto;
background-color:
#fff;
}
div.maincontainer
div.mainbody
div.navigator {
float: left;
clear: left;
width: 250px;
}
div.maincontainer
div.mainbody
div.content {
margin-left:
270px;
width: 725px;
clear: right;
}
```

One of the most useful features of XPages in Domino Designer on Eclipse is the real-time rendering of CSS in the Design panel. This feature allows you to tweak and modify the layout of your DIV Container Objects to meet the needs of your specific application. Forget the constant Save-Toggle-Preview tango that Domino developers currently using CSS for styling and layout design often have to perform!

Figure 33. Index XPage main page -- Design panel preview



{cfdDBTitleLogo}

New Contact

	Type
First name:	FirstName
Last name:	LastName
Mail address:	MailAddress
Office phone number:	OfficePhoneNumber
Company name:	CompanyName
Categories:	Categories

		Name	E-mail	Phone	Company
☐	Avatar	Name	E-mail	Phone	Company

This capture of the design panel from the index XPage shows your CSS-based layout applied in real-time. The main.css style sheet uses a combination of standard CSS-layout techniques to give a "column-ed" appearance for your user interface.

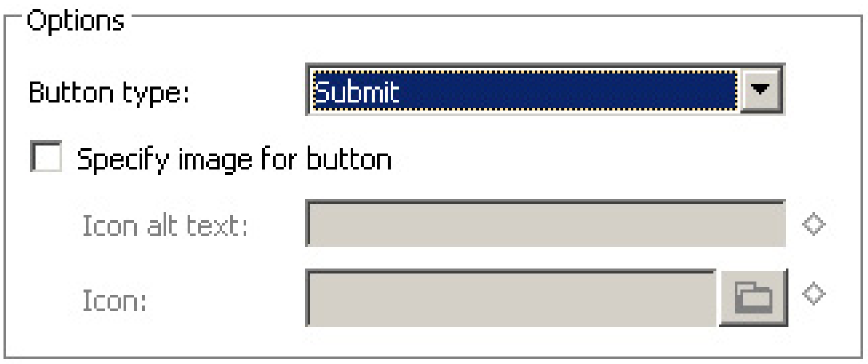
New_Contact XPage

You need to create a copy of the person XPage as you will be modifying the design of that XPage to meet your particular main page requirements. Although you can use the person XPage here, you want to maintain the examples for reference purposes. Create a new XPage named "New_Contact", and copy the following data source elements:

- Type
- FirstName
- LastName
- MailAddress
- OfficePhoneNumber
- CompanyName
- Categories

Once added, create a simple button control object called `submitbutton`. In the Button Properties panel, set the options and set the **Button type** attribute (value) to **Submit** (see Figure 34).

Figure 34. Button control object properties



Options

Button type: Submit

☐ Specify image for button

Icon alt text:

Icon:

Modify the type edit box control object, setting a default value of **Person**, as the view selection formula for the people view requires that the Type NotesItem equal "Person". Modify the default value attribute (defaultValue) in the data section of the Edit Box Properties panel.

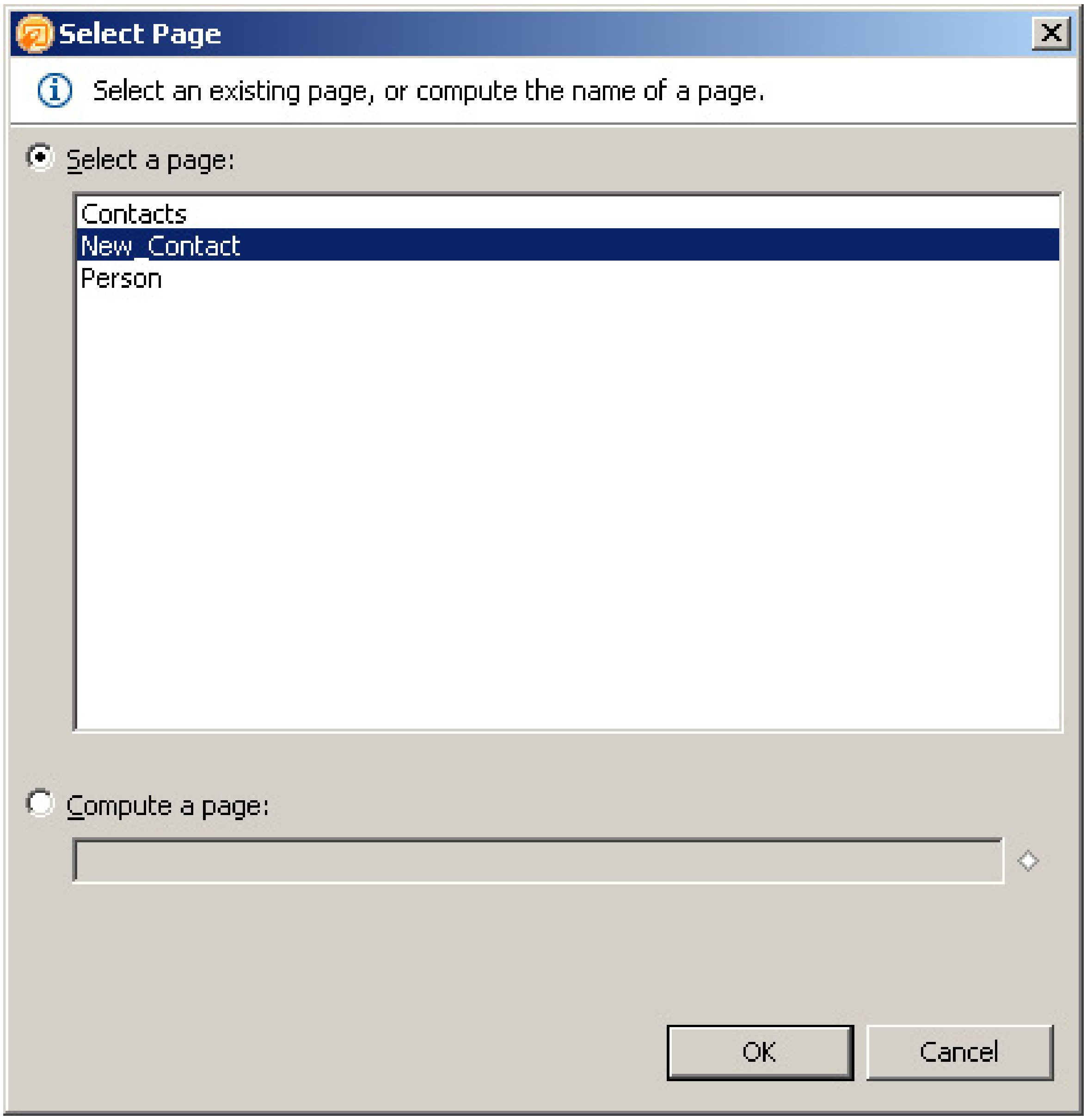
Once you have your New_Contact XPage, you're ready for it's inclusion in the main page.

Nesting XPages with Include Page

XPages can serve as both form and subform design elements in that an XPage can be inserted into an XPage, a feature that allows for the creation of reusable components in XPage Development. Use this XPage feature to create a simplified embedded "New Contact" Form on your main page.

Position your cursor in your navigator DIV Control Object, and select **Create\Container Controls\Include Page** from the Domino Designer on Eclipse application menu and selecting your New_Contact XPage from the Select Page window (see Figure 35).

Figure 35. XPage Include Page window



This action embeds your New_Contact XPage in your navigator DIV control object.

Alternately, use the controls panel to drag-and-drop an Include Page Control Object onto your XPages, or toggle to the Source panel and add the following markup:
`<xp:include pageName="/New_Contact.xsp" id="include1"></xp:include>`.

People view revisited

Earlier in this tutorial, you reviewed the simple creation of a viewPanel which, having defined the Domino View-based data source as the People view, allowed you to display viewColumn information in your Contacts XPage. The index XPage requirements, however, mandate that you create two additional viewColumns containing information and functionality that do not currently exist in your People view.

The two additional viewColumns that you need to create are:

1. A check box selection for multiple document processing, named (simple) viewColumn1 in the example.
2. A column displaying the ContactPhoto image for each contact, named viewColumn_Avatar in the example.

Because you are relatively familiar with the creation of viewPanels and inserting and positioning viewColumns, here's are the main steps:

Create a new viewPanel on your index XPage named "viewbody_Contacts" and set the following viewPanel properties shown in Table 1.

Table 1. viewPanel properties

ID:	viewbody_Contacts
pageName:	/Person.xsp
rows:	10
value (Data Source):	people
var (variable):	entry

Section 5. Add multiple document deletions

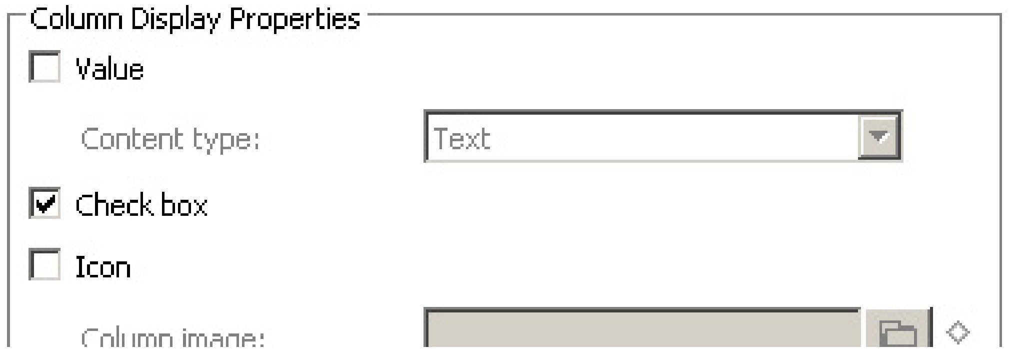
Based upon the previously-defined main page requirements, you want to be able to select single or multiple documents for deletion. To do so, use a combination of check boxes in the first viewColumn of your viewbody_Contacts viewPanel, and a simple onclick-event action on an embedded Image Resource. This is an excellent showcase of exactly just how powerful XPages are and how much time they'll save the Domino Web application developer. Simple, yet XPages makes this combination extremely powerful!

Enabling check boxes in viewColumns

Build this **Multiple Document Deletions Engine** with the viewColumn1.

To enable check boxes in viewColumns, modify the Column Display properties (see Figure 36) in the View Column section of the viewColumn Properties panel to show only Check Box (`showCheckBox = true`).

Figure 36. Check box viewColumn assignment via column display properties



Column Display Properties

- [ ] Value

Content type: Text

- [✓] Check box
- [ ] Icon

Column image:

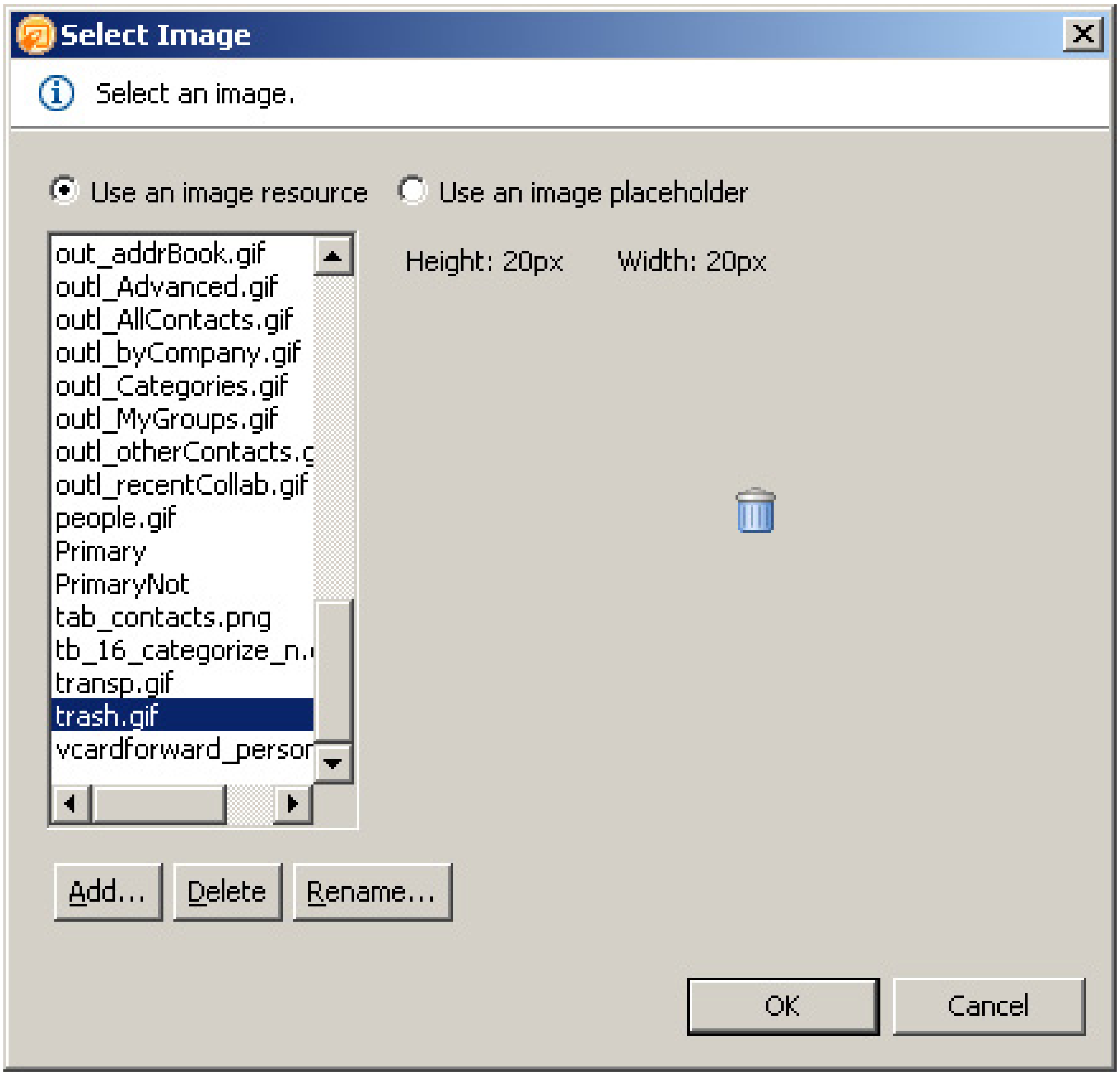
This option can be set in both the All Properties section and the source panel.

Adding actions to image resources

Now that you have check boxes per viewColumn entry, assign an action to an element or object on your XPage. Add a simple Image Resource and define an action for the Image Resource.

Either drag-and drop from the Controls panel, select from the Domino Designer on Eclipse Application Menu, or use markup in the Source panel to include an Image Resource on your XPage. The Controls panel or the Domino Designer on Eclipse application menu provides you with a Select Image window, which you'll use to select the trash.gif Image Resource (see Figure 37).

Figure 37. Insert Image Resource window



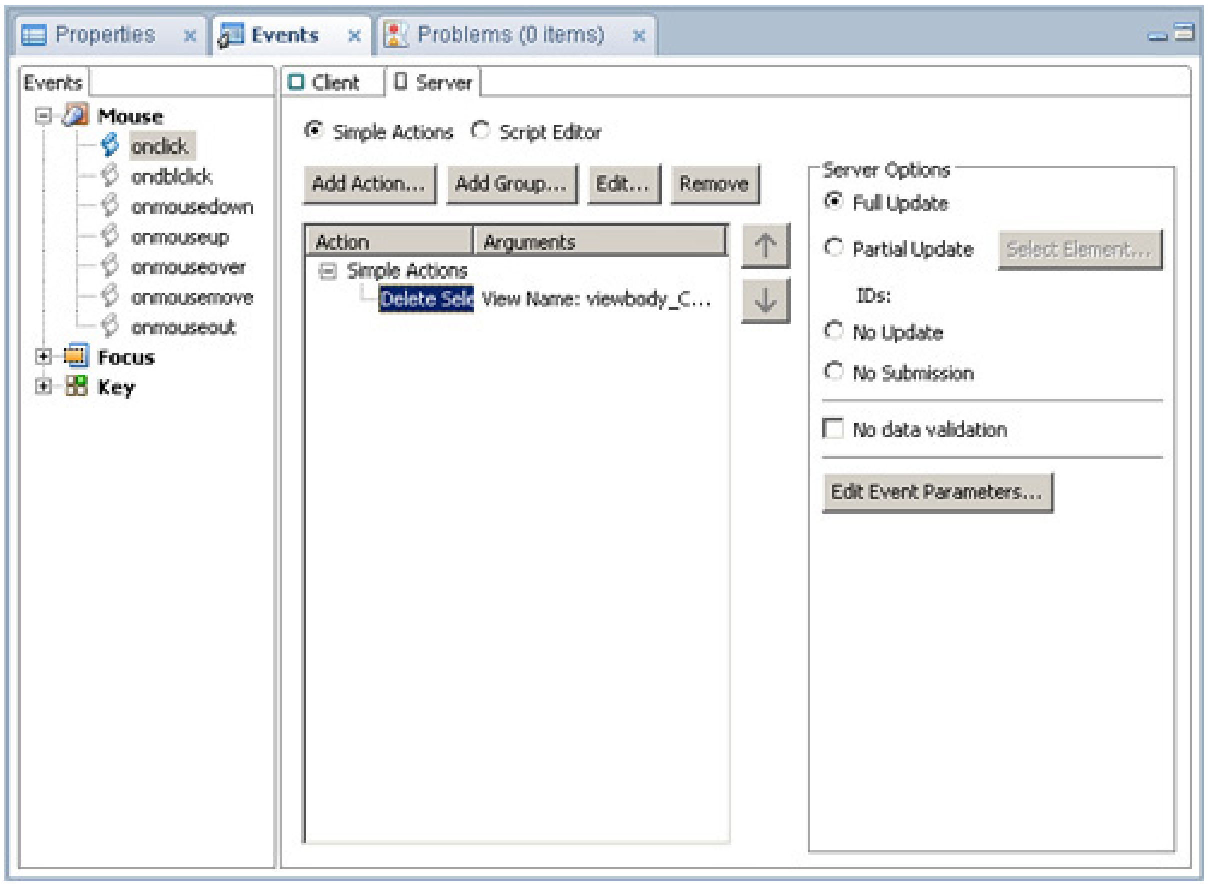
[✓ Use an image resource] Use an image placeholder

out_addrBook.gif
outl_Advanced.gif
outl_AllContacts.gif
outl_byCompany.gif
outl_Categories.gif
outl_MyGroups.gif
outl_otherContacts.g
outl_recentCollab.gif
people.gif
Primary
PrimaryNot
tab_contacts.png
tb_16_categorize_n.
transp.gif
trash.gif
vcardforward_persor

Height: 20px Width: 20px

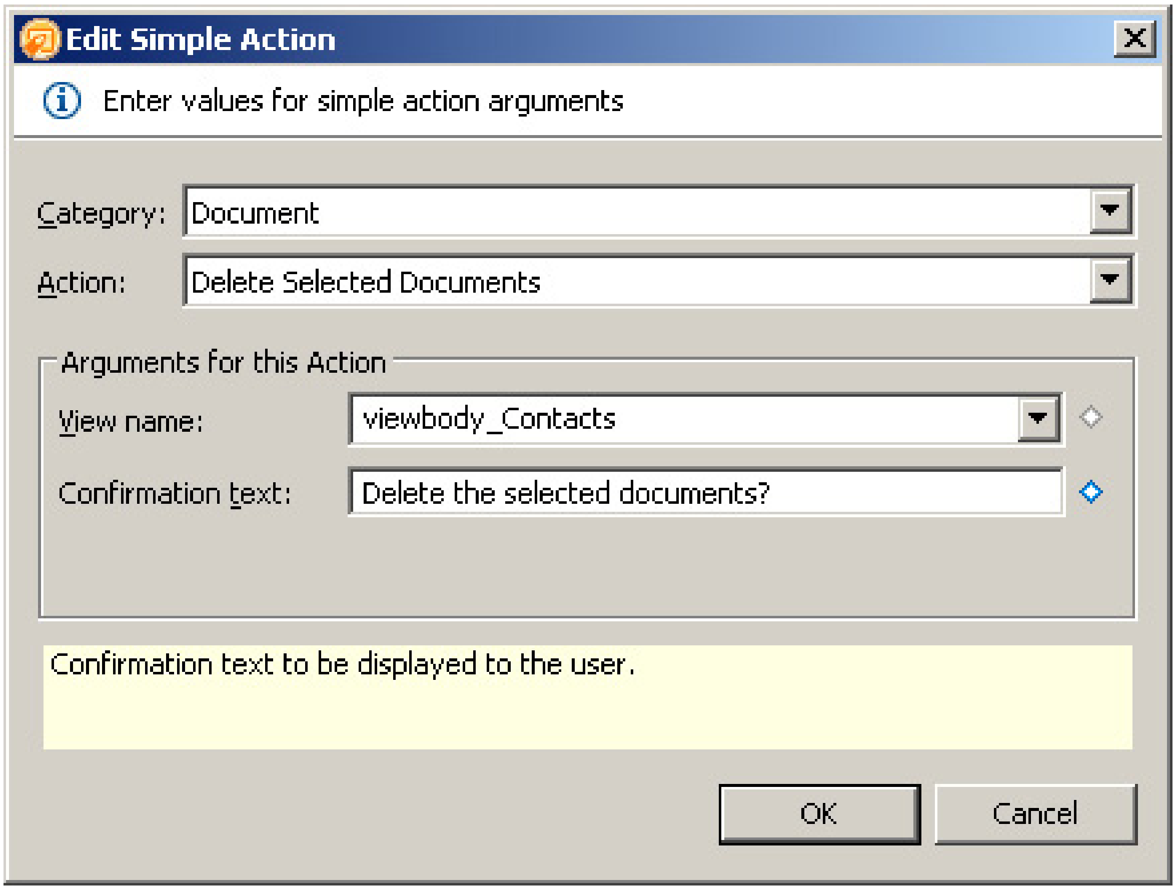
Once your image resource is on your XPage, you can add an Event Action via the Events panel, which is located at the bottom of your Domino Designer on Eclipse default perspective (see Figure 38).

Figure 38. Object Events properties panel



As seen in Figure 28, you can add an action to a given event -- in this case, onclick of your Image Resource. When you select **Add Action** from the Events panel, you are presented with the Edit Simple Action (see Figure 39).

Figure 39. Simple Action window -- Delete Selected Documents



Enter values for simple action arguments

Category: Document

Action: Delete Selected Documents

Arguments for this Action

View name: viewbody_Contacts

Confirmation text: Delete the selected documents?

Confirmation text to be displayed to the user.

From here, it's a simple matter of selecting your Action category, Action (where you find that **Delete Selected Documents** is already defined!), bind this action to a viewPanel, and even define text for a confirmation prompt.

Once you finish this modification and build your index XPage, you can immediately start selecting multiple documents and, with a click of your trash icon, delete your selection.

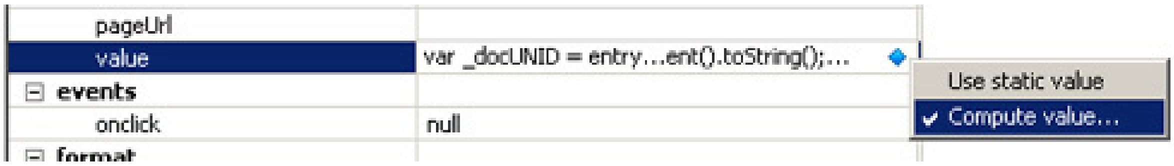
Programming for viewColumn content

The second requirement for your viewbody_Contacts viewPanel is to display the ContactPhoto for the Contacts listed in your personal address book application. Modify viewColumn_Avatar to both display the ContactPhoto as well as add an Event action to open the given entry in Edit Mode (similar to the defined viewColumn_Name Link Action).

You initially defined a variable for your viewbody_Contacts viewPanel, specifically defining *entry* as the variable name for each NotesDocument entry in your viewPanel. Now use that variable declaration to get information on your entry, while combining it with markup to render a simple context-sensitive IMG element per entry.

Once you select the viewColumn_Avatar, you can define its Column Display properties to render its value as HTML (`contentType = HTML`). You can then, through the All Properties section, select it's Data value (value) to use a Computed value (see Figure 40).

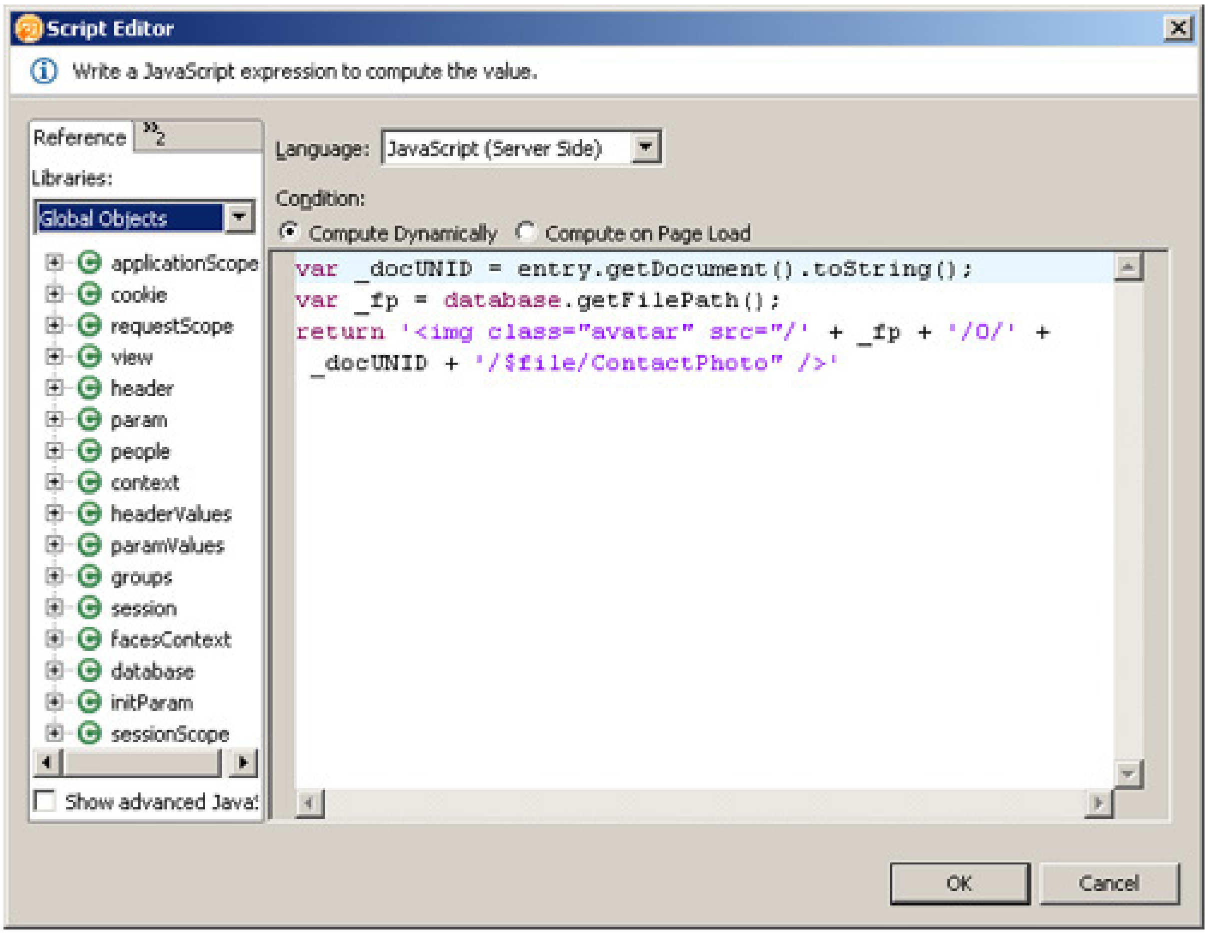
Figure 40. viewColumn computed value properties



pageUrl	
value	var _docUNID = entry...ent().toString();...
⊟ events	
onclick	null
⊟ format	

This action produces the script editor window, which allows you to use your *entry* variable to get a handle on the current NotesDocument DocumentUniqueID, the Database FilePath, and use those values to create your per-entry IMG element (see Figure 41).

Figure 41. XPage script editor window



Script Editor

Write a JavaScript expression to compute the value.

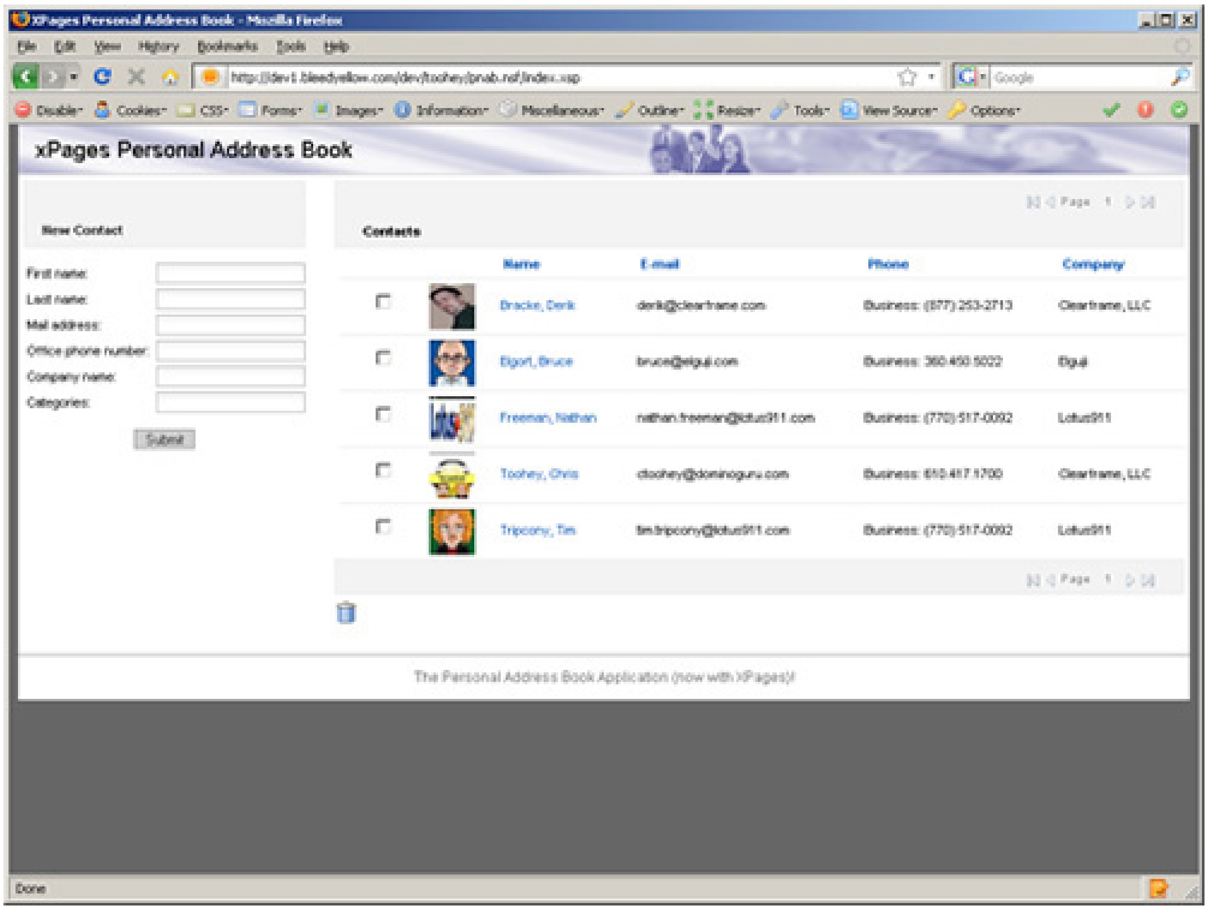
Language: JavaScript (Server Side)

Condition: [✓ Compute Dynamically] Compute on Page Load

```
var _docUNID = entry.getDocument().toString();
var _fp = database.getFilePath();
return '<img class="avatar" src="/' + _fp + '/0/' +
 _docUNID + '/$file/ContactPhoto" />'
```

This is a very simple example of just how powerful the technologies and capabilities in XPages are. A handle on a NotesDocument is the entry point to any NotesItems, attributes or any information about that NotesDocument. Note how that information can be used at runtime to deliver some truly advanced functionality. See again the finished Web 2.0 application shown in Figure 42.

Figure 42. Completed index XPage main page example



XPages Personal Address Book - Mozilla Firefox

http://dev1.bleedyellow.com/dev/toohey/pnab.nsf/index.xsp

xPages Personal Address Book

New Contact

First name:	
Last name:	
Mail address:	
Office phone number:	
Company name:	
Categories:	

Submit

Contacts

		Name	E-mail	Phone	Company
☐		Dracke, Derik	derik@clearframe.com	Business: (877) 253-2713	Clearframe, LLC
☐		Elgort, Bruce	bruce@elguji.com	Business: 360 450 5022	Elguji
☐		Freeman, Nathan	nathan.freeman@lotus911.com	Business: (770) 517-0092	Lotus911
☐		Toohey, Chris	ctoohey@dominoguru.com	Business: 610 417 1700	Clearframe, LLC
☐		Tripcony, Tim	tim.tripcony@lotus911.com	Business: (770) 517-0092	Lotus911

The Personal Address Book Application (now with XPages)!

Done

Building and viewing the main page now renders the preview results: a simple New Contact form next to a View of Contacts, allowing you multiple document deletion capabilities and simple document editing, albeit from a simplified Person XPage.

These examples can be expanded, of course, to allow additional data maintenance capabilities; however, the main page provides a fully functioning example implementation of an XPage-based Domino Web application user interface.

One last thing to mention. Although it wasn't covered in this tutorial, before using an example like the one created here, modify the Application Control List (ACL) to:

- Disable anonymous access
- Add individual or group entries with author, editor, and deletion rights

- Assign appropriate access control roles per user functional requirements.

Section 6. Summary

If you haven't already, be sure to download personaladdress.zip, which contains the finished personal address book that has been modified into a Web 2.0 application with the help of XPages.

Looking through the example application, notice the other Control Object types included: computed fields, paragraphs, and direct HTML Markup through the Source panel. So take it apart and play around with this off-production personal address book application. XPages is a new technology; the best way to get started is to dive in head-first!

In the coming months, be prepared to see such applications shipping with both the Gold release of IBM Lotus Notes/Domino 8.5, through the IBM Business Partners, and through the Lotus Development Community.

Downloads

Description	Name	Size	Download method
personal address book source code	personaladdress.zip	1.1MB	HTTP

Information about download methods

Resources

Learn

- Meet Domino Superior (or A Tutorial on Why Xpages Are Better Than Cake and Ice Cream): Read Nathan T Freeman's Blog entry on XPages.
- XPages in Domino Designer 8.5: Read Rob McDonagh's blog entry on XPages.
- XPages in Domino Designer: Read Andrew Pollack's article covering Xpages.
- YellowCast - Episode 4 - A New Hope: Chris Toohey and Tim Tripcony, with Nathan T Freeman give an audio presentation on XPages.
- Taking Notes Podcast - Episode 83 - XPages and Yellow Water: Learn more about XPages from Bruce Elgort, Julian Robichaux, Rob McDonagh, and John Head.
- Themes in XPages - More insane power: Read Andrew Pollack's additional coverage of some of the advanced capabilities in XPages.
- XPages Round Up: See Alan Lepofsky's blog covering XPages.
- Lotus Domino Web Application Development Wiki: Visit the wiki for Lotus Domino Web Application Development Best Practices.
- PlanetLotus.org XPages Search: You can find blogs for all Domino topics here including XPages.
- Best Practices for Building Domino 8 Web Applications: Read the IBM Redbook.
- IBM Lotus Designer 8.5 - XPages Hello World sample application: This is part of a getting started tutorial for XPages in the 'Lotus Domino Designer Basic User Guide' under the section 'Getting started using XPages - Basic'.
- Lotus Domino Designer documentation: Get access to white papers, Redbooks and more.
- Looking ahead to version 8.5 : Some basic information on the Domino Designer 8.5.

Get products and technologies

- Domino Release 8.5 Public BETA Server and Domino Designer in Eclipse Client Get the 8.5 Beta release. You will need an IBM ID and Password.

Discuss

- Participate in the discussion forum for this content.
- Get involved in the developerWorks community by participating in

developerWorks blogs.

About the author

Chris Toohey
A published developer and Webmaster of DominoGuru.com, a Lotus Notes/Domino-themed "Tips & Tricks" Web site and Weblog, Chris Toohey is the Chief Solutions Architect for Clearframe, and specializes in integrating IBM Lotus Notes/Domino with other enterprise-level solutions. Since entering the IT industry in 1998, Chris's unconventional methodologies, forward thinking, and his ability to uniquely analyze and attack a given problem with award winning solutions has afforded him recognition as an expert in his field, as well as yielding many happy customers.