



# Visual Web Application Design

with  
NetBeans

Beth Stearns

The NetBeans IDE has many exciting features for visual web application design. This article introduces some of this functionality and shows how it makes it easy to develop for the web.

The NetBeans IDE incorporates many features for visual web application design, a number of which have been available as part of the NetBeans IDE Visual Web Pack 5.5 module. NetBeans 6.0 integrates these visual design features directly into the IDE and adds more features to the mix.

This article gives you a quick overview of the feature highlights of the NetBeans visual web design environment, including the Visual Designer, Page Navigator, Query Editor, and Style Editor. We also show some of the things you can do easily and quickly with these tools when developing a web application.

## Highlights

One of the best aspects of the NetBeans visual development environment is its flexibility. You can develop an application by first designing its individual pages, for which the IDE provides a Visual Designer with a palette of visual and non-visual components. You can add other elements to the palette, such as AJAX-enabled components you develop yourself or obtain from third parties, using the Component Library Manager. The Style Editor provides a graphical interface for perfecting the look of the different components.

You might prefer to first map out the application's logic flow instead of beginning with page design. If that's the case, you can start application design with the Page Navigator. Using the Page Navigator functions, you can create empty pages as stubs or placeholders and link them together in the Navigator window to define the applica-

tion processing, and add the individual page layout and functionality later. You can even go back and forth between the Navigator and Visual Designer modes as you develop your application.

NetBeans also makes it simple for a web application to access a database. You can use the Query Editor functions to form complex SQL queries for applications that need to retrieve data from database tables or update a database. Other visual functions make it an easy matter to handle the display of data retrieved from database tables.

Although much is generated for you, there are still times you have to write your own custom code. When you are ready to write code, you can draw on the palette of generic code clips to help. You also have available all the shortcuts and other helpful features, such as code completion, that the Java source editor provides.

NetBeans 6.0 will introduce more visual design features, including the ability to develop portlets, add Enterprise JavaBeans components to applications, and incorporate web services. These features will rely on and extend the preexisting visual capabilities, so the learning curve for them should be minimal.

## Using Components to Develop Web Application Pages

Page design is of course an important part of developing a web application. NetBeans provides a palette of visual design components that you drag and drop onto a page in the Design window, to set up your page quickly with the desired look and feel. In addition to these visual components, the IDE supports themes, which let you apply a predefined set of styles to visual components throughout a project, thus enabling you to change an application's entire appearance with a single mouse click.

As you design the underlying business logic, you can use the non-visual components – such as the converter, validator, and data provider components – to generate code. When you're ready to write the business logic, you can incorporate generic code snippets into your page bean by simply dropping code clips from the palette onto your bean code in the Java source code editor. All you need to do is add the correct variable names to these code clips.

 You can easily create your own code clips too, just by selecting a snippet of code in the source editor and dragging it to the palette.

NetBeans displays the palette of components in a logical and in-



[netbeans.org/kb/55/wvp-index.html#downloads/index.jsp](http://netbeans.org/kb/55/wvp-index.html#downloads/index.jsp)

NetBeans  
5.5 Visual  
Web Pack  
Documentation  
main page.



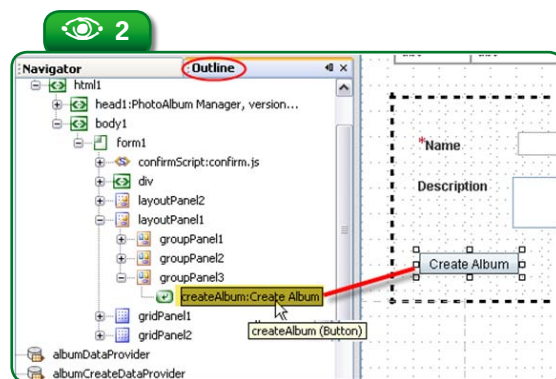
**Figure 2**  
Viewing  
components  
in the Outline  
Pane

tuitive manner. The palette appears only when you're working on a web page for a project. When designing a web page – that is, when you have the page open in the Design window – the palette displays the components used to build the page, like the visual design components (buttons, drop-down lists, checkboxes, tables, etc.), as well as layout components, converters, validators, and data providers. If you used the Component Library Manager to add a set of components, you may have designated that they appear in their own separate category.

To illustrate, we use a sample media management project called PhotoAlbum, which is a web application through which users can manage different types of media (photos and audio data) stored in a local or remote database. You can use this application to organize media data into albums, such as photo albums. For example, if you want to organize a set of images, you can create multiple photo albums, upload these images into the different albums, and view the images as thumbnails or singly in a preview mode. (See the article “Developing a Media Management Application” at [netbeans.org/kb/55/photoalbum.html](http://netbeans.org/kb/55/photoalbum.html) for instructions on downloading this project zip file and installing and running the application.)

We opened the AlbumList page in the Design window and the palette displays the components we can use to build this page (see **Figure 1**). You add components to a page merely by dragging and dropping them on the page – or onto other components, such as in the case of validators and converters. The IDE ensures that no rules are broken, such as dropping a validator on the wrong type of component. Notice, too, that themes are not in the palette but are specific to individual projects and thus appear in the Projects pane.

At any time, you can go to the Outline pane to see the components added to a page. If you are unsure of a component's type, hover the mouse over it in the Outline pane to get more information. Or



match its icon with the Palette icons (see **Figure 2**).

While you work on the page design and layout, you also can switch to editing the page's underlying Java code. To see the code in the Java editor, double click on the page background or select the Java view. When you change to the Java view, the Palette displays the available code clips (see **Figure 3**).

To work on a particular method, such as an action handler for a button, double click the button component in the Design window, and NetBeans displays the source code in the Java editor positioned at the button's action handler.

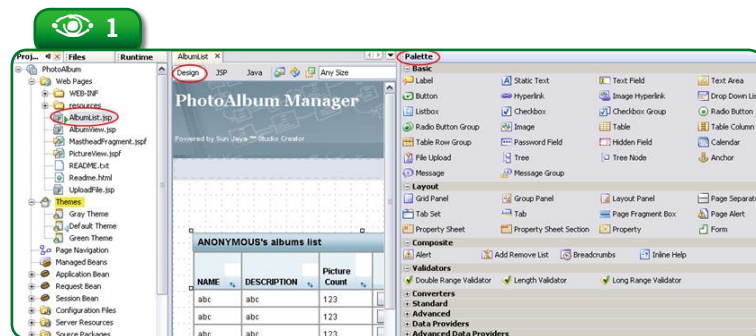
## Designing pages

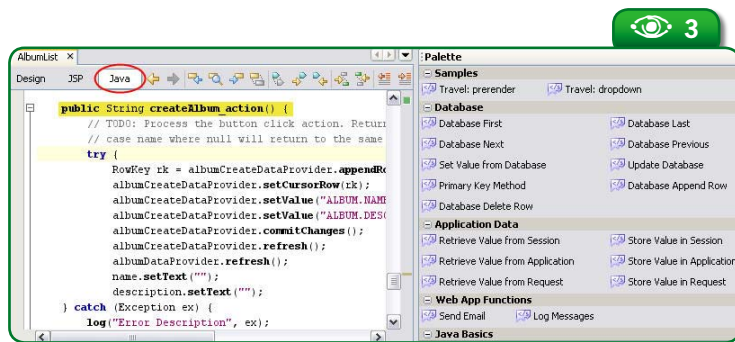
Let's examine how you might design a web application page using the visual components. The components in the Basic section of the Palette are typical GUI components: buttons, drop-down lists, checkboxes, hyperlinks, text areas, hyperlinks to images, and so forth. You use these components to add basic behaviors to your web pages.

These components provide properties to determine their appearance and behavior, and you can customize them using the component's Properties sheet or through dialogs. Although hand-editing a page's JSP code is possible (click



**Figure 1**  
Palette  
of design  
components  
for a Web page





You define this look in a single page fragment, then include that fragment on the application pages. For a banner or masthead, you place the page fragment at the top of each web page. If you later make changes to it, these changes automatically appear on all the pages that include the page fragment.

**Figure 3**  
Palette code clips

the JSP view to see and potentially modify the code), it is far easier and less error-prone to modify component properties in the Properties sheet.

The Layout components help you organize the appearance or layout of your page. Use them to add tab displays, alert boxes, grids, and forms – essentially to control the placement and alignment of other components on the page.

To give you an idea how all this works, let's take a closer look at some page-design issues frequently encountered by web application developers: setting up a consistent look across multiple pages of an application and aligning text and components on a page. Three of the layout components – Grid Panel, Layout Panel, and Page Fragment – handle this nicely.

### Establishing a consistent look across pages

Use page fragments when you want to set up a consistent look to multiple pages of a web application. Page fragments are particularly useful for establishing uniform web page headers, footers, and sidebars. You design the page fragment once, then place it where appropriate on different web pages.

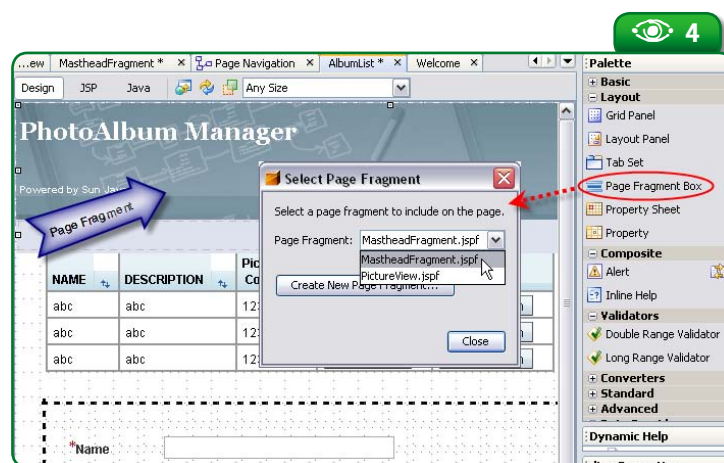
For example, you might want a consistent banner across all application pages.

Suppose you want to create a masthead banner for an application. Create a new page fragment: in the Projects pane, right click the project's Web Pages node and select *New>Page Fragment*. Then design your masthead by placing components in the fragment. Designing and building the page fragment is much the same as building a web page.

To add the masthead to other pages, drop a Page Fragment Box on a page, select the specific page fragment from the dialog, and position it where you want it to appear – at the top of the page in this case (see **Figure 4**). Fragments can also be copied and pasted among projects.

### Controlling page layout

Grid Panel and Layout Panel components are useful for arranging text and other components. When you drop a Grid Panel on a page, it creates a table to which you then add other components. The added components display starting from left to right and from top to bottom. By default, a Grid Panel has one column and as many rows as needed to accommodate components dropped on it. You can change the number of columns and the display direction in the Grid Panel's Properties sheet.



**Figure 4**  
Adding a page fragment to a Web page

A listing of reference articles and tips for Java Studio Creator IDE. Most of the information applies to the NetBeans 5.5 Visual Web Pack.

developers.sun.com/prodtech/javatools/javacreator/reference/index.jsp

Grid Panel components can also be nested within other Grid Panel components, giving you even finer control to position components on a page. To nest Grid Panels, drop a Grid Panel on top of a Grid Panel already placed on a page.

 When dropping one component on top of another, be sure that the component already on the page is highlighted with a blue outline.

You can resize a Grid Panel directly on the page. For greater control, use the Style Editor, which you open by clicking the Grid Panel component's **style** property. You can use this property to set background color, margins, size, position, and so forth (see **Figure 5**). The Style Editor allows you to fine-tune the appearance of all the visual components.

The example MastheadFragment shown in **Figure 6** uses several Grid Panel components to control the display of its two hyperlinks (Home and Help) in the bottom panel. We used Grid Panels to place these links on the right, bottom side of the page. First, we dropped a Grid Panel (which we called "bottomPanel") on the page. Then we resized its width to match the top banner and its height to a size that would accommodate the images and text. Also, we set its **columns** property to two columns and set a background color.

Next, we dropped two Grid Panels on top of **bottomPanel**. Since

**bottomPanel** displays two columns and the display direction is left-to-right, these two Grid Panel components display within **bottomPanel** from left to right. Also, the **bottomPanel** background color carries through as the background color for the components dropped onto it.

Of the two additional Grid Panels, the one in the left column (highlighted in yellow here) is a placeholder, and its width is set to cover the left half of the page. The Grid Panel on the right side holds the two links (see **Figure 6**). Since we want the links to display side-by-side, we changed the **columns** property to two for this Grid Panel.

After getting everything in position, we dropped two Image Hyperlink components onto the right panel. For each, we went to its Properties sheet and set its **text** property to the label – "Home" or "Help" – that we wanted to appear on the page. We also set each component's **imageURL** property to

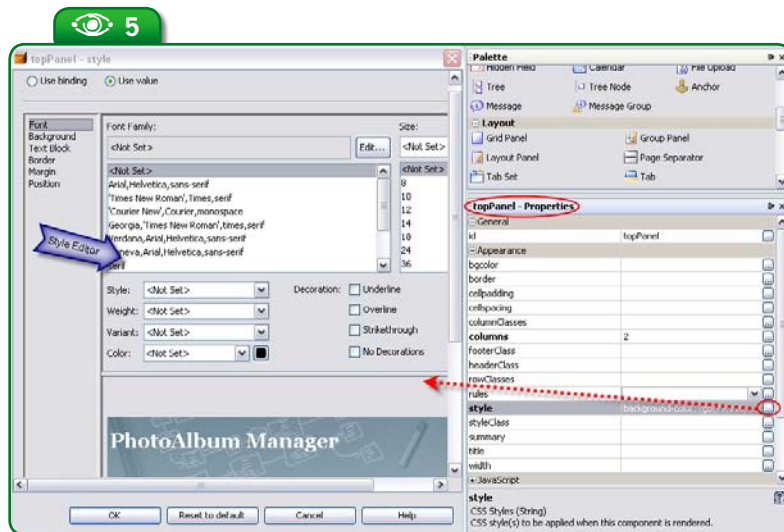
suitable image files for the display icons. In our application, these files are in the project's **/resources** folder, but the **imageURL** property can point to wherever the files are located, of course.

We use the **textPosition** property to ensure that text is positioned to the right of the image associated with the hyperlink. (The **text**, **textPosition**, and **imageURL** properties are in the Properties sheet Appearance section.) We set the hyperlink's **url** property (found in the Behavior section) to the appropriate web page, so that a user is taken to that page when the link is clicked.

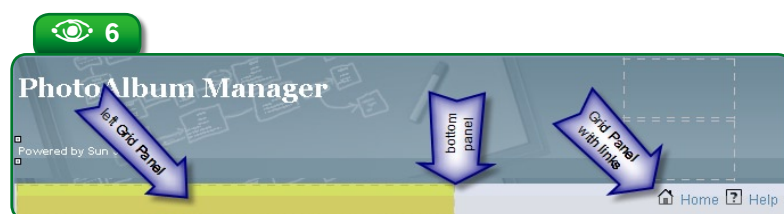
A Layout Panel component is much like a Grid Panel – you place a Lay-

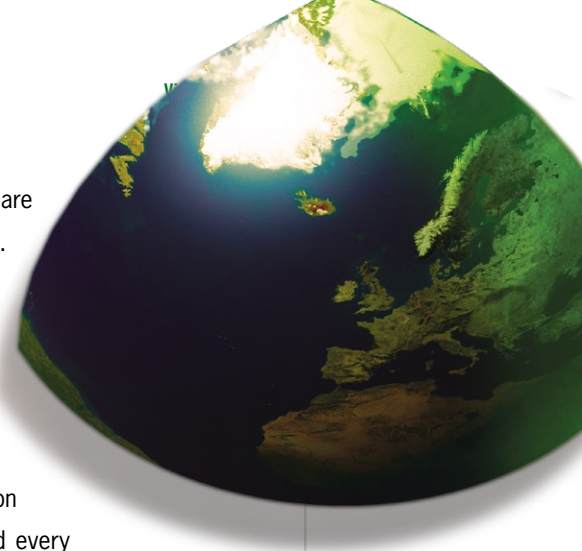


**Figure 5**  
Style Editor with Grid Panel



**Figure 6**  
Controlling layout with multiple Grid Panels





out Panel on a page and then drop components onto the Layout Panel. But a Layout Panel gives you more flexibility in arranging components. Whereas a Group Panel can be placed on the same line as other components, a Layout Panel always appears on its own line, separated from components above and below it, when the page is rendered at runtime.

A Layout Panel lets you arrange added components in a flow or grid layout. When flow layout is used, the IDE places components dropped on a Layout Panel starting in the top left corner, adding components to the panel from left to right, across the top row of the panel until that row is filled. Subsequent components are added from left to right in the next row down, and so forth. You can drop a new component to the left of another component by hovering over the previously added component until a vertical mark appears to the left of that component.

A Layout Panel works in grid layout mode only if the Snap to Grid option is set. You set this option from the *Tools>Options>Visual Designer* settings; at the same time, you can also customize the grid pattern size. When set to grid layout, added components appear in the panel aligned to the grid location at which they were added. Click the Align pop-up menu option for a Layout Panel (or for a component within a Layout Panel) to position components in the panel relative to the nearest grid corner.

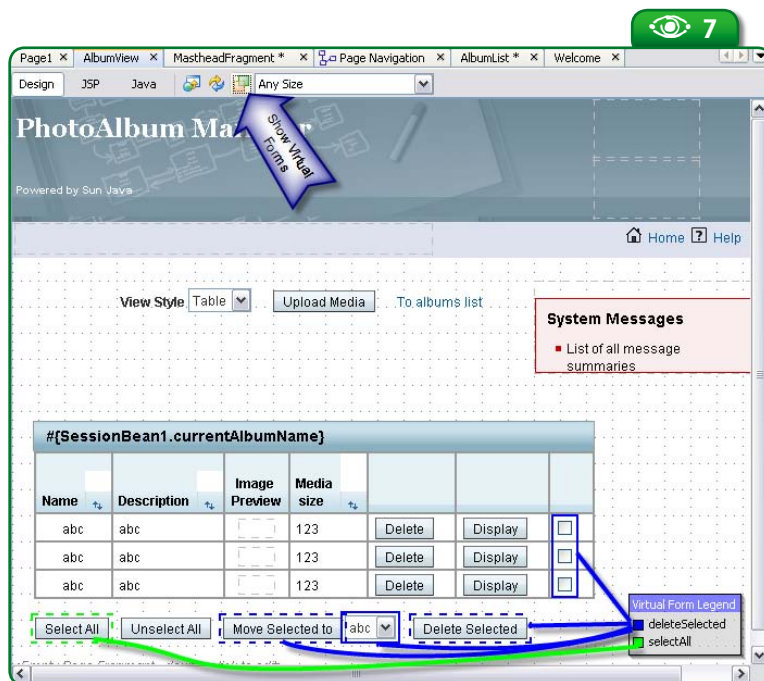
## Controlling user input

The Visual Designer includes a feature, called *virtual forms*, which lets


you limit the portions of user input that are processed when a page is submitted. This is useful because you may have linked a number of input fields to validators (which means that the user-entered data in these fields is validated against some criteria when the page is submitted); but, given the application logic, you don't want all fields validated every time the page is submitted.

Let's look at how you might use virtual forms for a page. Our example page (see **Figure 7**) defines two virtual forms – **deleteSelected** and **selectAll**. Several components are included in these forms. Notice that the buttons, drop-down lists, and checkboxes for selecting, moving, and deleting files from an album are outlined in green or blue. Components outlined in green are part of the **selectAll** virtual form, while those outlined in blue are part of the **deleteSelected** virtual form.

A solid outline indicates an input component that participates in the virtual form, and a broken or dotted outline indicates a submission component, which is a component that, when clicked, submits the virtual form for processing. (To see the virtual forms legend for a page, toggle the virtual forms display icon at the top of the Design window, as shown in **Figure 7**)



**Figure 7**  
Virtual forms displayed on a page

 **Figure 9**  
Generic Table  
component and  
displaying a  
database table

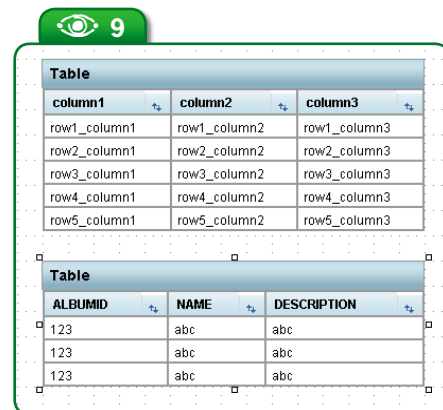
You add components to a virtual form via the dialog that opens when you click a component's pop-up menu *Configure Virtual Forms* option. The *Configure Virtual Forms* dialog shows that the Drop Down List component `albumList` participates in the `deleteSelected` virtual form. You can change whether a component participates or submits for any virtual form on the page by double clicking in the appropriate row and column. The Participate and Submit column entries display a Yes/No pull-down list if the selected component is of the right type, since only certain types of components can submit a page for processing. The New button in the dialog lets you create a new virtual form for the page (see **Figure 8**).

When the web page user interacts with a virtual form's submission component, such as by clicking the *Move Selected to* button, processing affects only the virtual form's input or participant components and ignores other input components on the page. In this manner, you can confine certain application operations to selected files or database table rows. For example, a user might check the boxes of several media files from the displayed list, then click the *Move Selected to* button with a target album designated from the drop-down list. The virtual forms feature ensures that subsequent processing moves only the selected files from the current album to the designated album.

## Retrieving and displaying database data

The Visual Designer simplifies displaying tabular data on a page, and especially makes it easy to retrieve and display database data. To display tabular data, first drop a Table component from the Palette onto your page. The Visual Designer creates a generic three-column, multi-row table display (see **Figure 9**).


To have this component display data from a database table, you

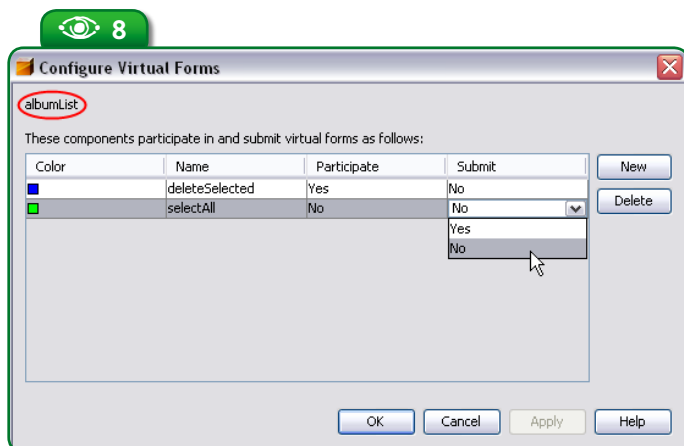


only need to drop the database table onto the generic table component on the page. (Database tables appear in the Runtime pane, beneath the Databases node. You must connect to the particular database or subschema before you can see its individual tables.)

After you drop a database table on top of a generic table component, the Visual Designer binds the database table to the component and creates a default SQL query to retrieve the table data. The table display on the page changes to reflect the columns in the database table, while the rows indicate the type of data (again, see **Figure 9**).

When you bind a database table to a Table component, the IDE adds a **RowSet** to the project's Session Bean. (Every project has, in addition to its individual pages, a Session Bean and an Application Bean. The Session Bean maintains session scope variables, while the Application Bean is used for project-wide variables.) The **RowSet** includes a default SQL statement selecting all the columns in the database table. NetBeans also places the query's SQL statement in the Session Bean's constructor using the method `rowset.setCommand(String sqlCommand)`. Here, `rowset` is the **RowSet** for the table dropped on the page and `sqlCommand` is the query SELECT statement. Thus, you

 **Figure 8**  
Configuring  
virtual forms




can edit the query in the Session Bean Java source code in the Java editor as well as through the Query Editor.

### Customizing table data display

When you drop a database table on a Table component, the component display changes to show all columns from the database table in the same order they are returned from the database, and the database column names appear for the column headings. You can change the display using the Table Layout dialog, which you open from the Table's pop-up menu.

The Table Layout dialog consists of two tabs. These together give you options to remove columns, modify column headings, change column order, and even add new columns beyond what's in the database. From the Columns tab, use the Up/Down buttons to change the column display order, and use the left arrow to remove a selected column from the display (the double left arrow removes all columns). Use the right arrow to add available columns and New to create a new column. You can also modify column header text, column width, alignment, and sort capabilities (see **Figure 10**).

 Setting the **sortable** option for a column enables users to sort the table data in real time by clicking the arrows in the column heading.

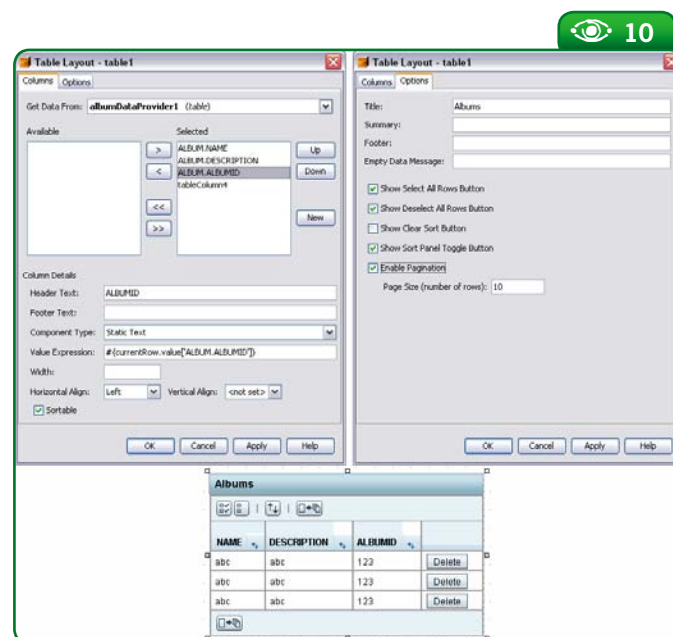
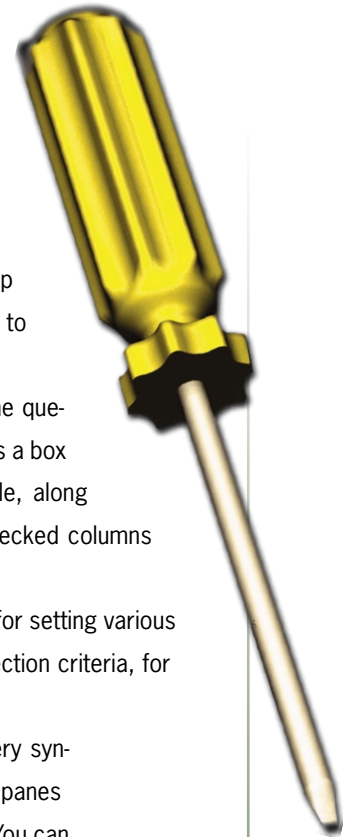
From the Table Layout Options tab, you can do such things as enable pagination, establish the page size (number of rows per page), and configure navigation buttons (select all rows, deselect all rows, clear sort, and so forth). These are automatically provided for you; no manual coding is required for the paging buttons.


### Customizing SQL queries

The Query Editor provides a graphical interface through which you can edit and customize the SQL SELECT statement contained in a table's **RowSet** component.

The Query Editor has four panes in which you can right click to bring up appropriate pop-up menus (see **Figure 11**). These are, from top to bottom:

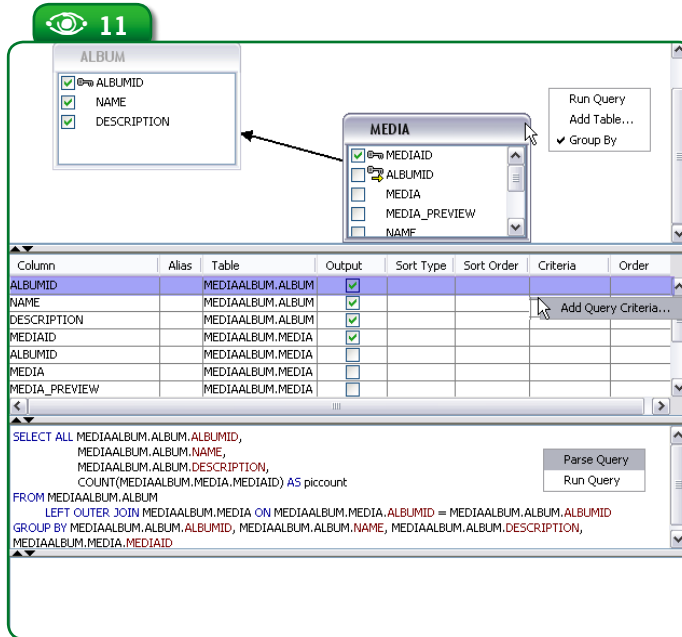
- The **Diagram Pane** graphically represents the query. Each table dropped on the page appears as a box that indicates all the columns within the table, along with the table's primary and foreign keys. Checked columns are retrieved when you run the query.
- The **Grid Pane** displays a multicolumn table for setting various query conditions, such as sort order and selection criteria, for each table dropped on the page.
- The **SQL Pane** displays the actual SQL query syntax. Changes made in either of the top two panes are automatically reflected in all three panes. You can change the SQL query directly. The pane has a pop-up menu *Parse Query* option to update the other two panes. Use the *Run Query* option to test the query.
- The **Results Pane** at the bottom displays the results of a query you tested via *Run Query*.



 **Figure 10**  
Table Layout dialog



**Figure 11**  
Query Editor  
with sample  
SQL query



What are some of the query customizations you can do using the Query Editor? The Output column (in the grid pane) lets you select the columns to retrieve from the database and display. The Alias column lets you insert an AS clause into the SELECT clause. You can also control the sort order of the retrieved data.

The Diagram Pane's Add Table pop-up menu option lets you include additional tables in the query, essentially adding a JOIN clause to the SQL. The Group By option lets you group results by data type, which adds a GROUP BY clause to the query.

You can also limit a query to return only selected rows that meet some specified criteria. Use the *Add Query Criteria* pop-up menu option to open a dialog through which you add conditions for the query selection. You can select rows using comparison operators (such as equal to, greater than, less than, not equal to, and so forth), in addition to LIKE and IN. You can apply selection criteria to multiple table columns and specify the order to evaluate these criteria. Also, when you test a query for which you have specified selection criteria, a dialog prompts you to enter data for the selection parameters.



## Defining application flow

Use Page Navigation to define the process flow of an application. Page Navigation, accessed from the Projects pane, lets you visually design the application flow by drawing links be-

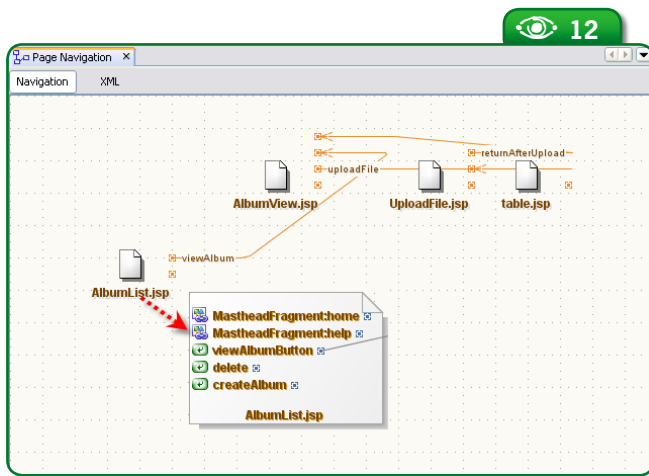
tween the application pages and thus model the interactions of the application.

You can use the Page Navigation feature at any point in the application development process, and you can revisit Page Navigation to fine-tune interactions among pages, particularly when you want a specific action component on one page to link to another page.

Typically, you start to define the application flow by working at the page level, linking one page to another. Then, as you implement application details on the various pages – that is, place buttons, links, and so forth – you can return to the Navigation window and create navigational links at the component level within individual pages. You can even add buttons and other links to pages directly in the Navigation window.

The Navigator indicates the linkage between the pages of an application. (We illustrate the linkage between pages using the PhotoAlbum application, which includes several pages: *AlbumView.jsp*, *AlbumList.jsp*, *UploadFile.jsp*, among others. As with any application that has multiple pages, there needs to be a defined navigation between the pages.)

Click an individual page, such as we did for *AlbumList.jsp*, and the Navigator expands the page icon to show the individual linkable components on the page and where they link to, if anywhere (see **Figure 12**). Click and drag from one page to another to create a link between the two pages. Similarly, drag from a page's individual components, such as buttons, to another page to create links from the specific component on the first page to the second page. The Navigator adds the backing code to the pages and component action handler methods.



**Figure 12**  
Page Navigation

cate the event in the JavaScript section of its property sheet; then enter the JavaScript code directly, or click the “...” button to open an editor window.

To illustrate, suppose you want the user to confirm an action connected to a button click. First select the button component; then, in the JavaScript section of the component's Properties sheet, open the editor window of the target event, which in this case is **onClick**. You might enter the following JavaScript code:

## Working with JavaScript code

Many web application developers want to work closely with JavaScript code. The Visual Designer makes it easy to invoke JavaScript from within an application. As you know, JavaScript code can be embedded within the JavaServer Pages code, where it is identified by its own set of tags.

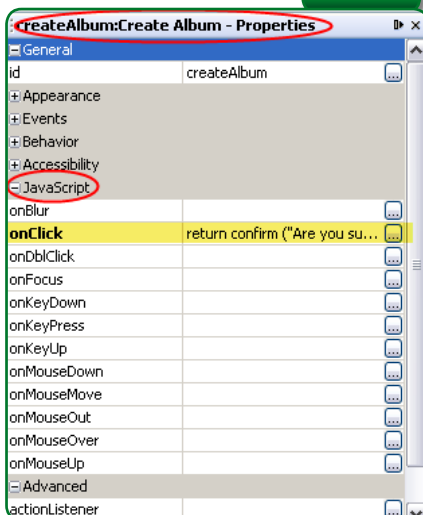
When you work with JavaScript code, you bind the code to a component event so that the code is invoked when the event occurs. You can do this binding through the JavaScript section of a component's properties. Select the component and lo-

```
return confirm ("Are you sure?")
```

When you're finished, the component's Properties would look like **Figure 13**.

One good use of JavaScript code is to obtain confirmation from a user for an operation. For example, an application might bind JavaScript code to a delete button, to compel the user to respond to a confirmation message before the deletion occurs. JavaScript can also be used for actions that change component state, for validation checks, and for setting file names when retrieving data. The JavaScript code can be fairly complex, such as code for a button's validation checks (see **Figure 14**). Such complex code is best added using the Properties editor window.

The JavaScript you enter through the editor window is inserted into the JSP code for the component. For example, the JavaScript code for the button's **onClick** property shown in **Figure 14** appears as follows in the JSP:



```
<ui.button action="#{AlbumView.btnMoveSelected_action}"
  binding="#{AlbumView.btnMoveSelected}"
  id="btnMoveSelected"
  onClick="var i;.&#xa;for (i=0;i<document.forms[0].elements.length;i++)
  {.&#xa; if (document.forms[0].elements[i].checked){.&#xa;
  return true; &#xa; }.&#xa;}&#xa;alert(&quot;
  No items selected&quot;);.&#xa;return false;"
  text="Move Selected"/>
```

When you use the Properties editor window to enter JavaScript code, the Visual Designer handles inserting the JavaScript into the JSP page with the proper tags and formatting. Of course, you can always edit the JSP yourself and manually add JavaScript code.

There are other ways to use JavaScript in an application. You can put



PDF compilation of the Java Studio Creator reference articles and tips.

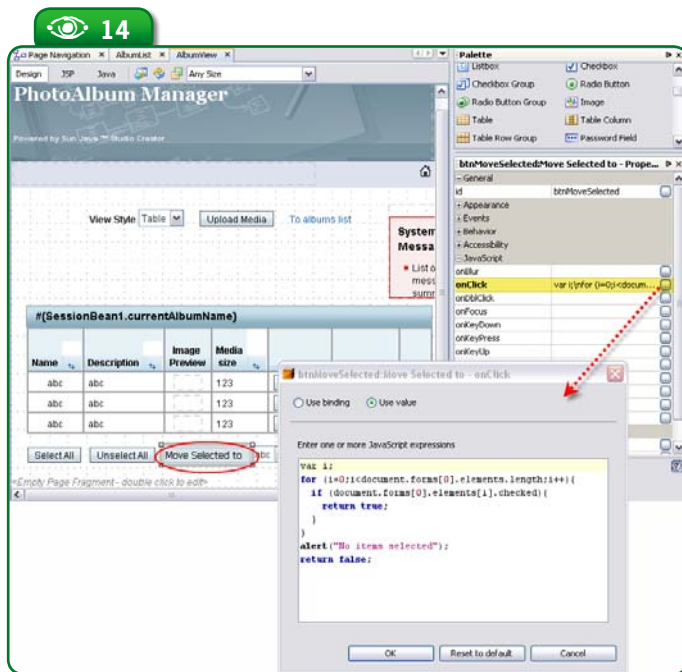
[developers.sun.com/prodtech/javatools/creator/reference/pdf/creator\\_prog\\_guide.pdf](http://developers.sun.com/prodtech/javatools/creator/reference/pdf/creator_prog_guide.pdf)



**Figure 13**  
Editing JavaScript properties



**Figure 14**  
JavaScript  
for button  
validation  
check



### Beth Stearns

(beth.stearns@sun.com) has been writing developer articles and books concerning the Java language since 1995. She has written extensively on Java Enterprise, JavaBeans, and Java Native Interface technologies as well as the Java Studio Creator and NetBeans development tools. Beth has co-authored numerous books in the Java Series, most recently, *Designing Web Services with the J2EE 1.4 Platform* and *Applying Enterprise JavaBeans, Second Edition*.

the code in a separate file and then use the Script component to bind to that file. First, enter the code to a file, giving it a .js extension, and then add the file to the project. (Place the file in a subdirectory of the project directory.) Next, drop a Script component, found in the Advanced section of the Palette, onto a page, and bind the Script component's url property to the .js file.

You can also drop the Script component onto the page and add your JavaScript code directly in the JSP editor. When added to a page, a Script component does not display but you should see the following line added to the JSP page:

```
<ui:script binding="#{AlbumView.script1}" id="script1"/>
```

You can then insert your JavaScript code within that tag, being sure to include the ending `</ui:script>` tag. For the simple confirmation shown before, for example, you might have:

```
<ui:script binding="#{AlbumView.script1}" id="script1"
return confirm("Are you sure?")
</ui:script>
```

Although it is more complicated to use the Script component, it allows you



to add generic JavaScript code that is not bound to an event.

## Conclusions

This introductory article should have given you a good idea of how easy it is to develop visual web applications with NetBeans and its Visual Designer tools. The Visual Designer provides a palette of customizable components that you use to design web pages, along with a Style Editor to help you fine-tune their appearance. In addition, non-visual components provide user-input validation and conversion code automatically, plus a virtual forms feature that simplifies user input processing. There is also a palette of sample code clips that you can use when you have to write backing code. And if you are comfortable with JavaScript code, the Visual Designer facilitates writing JavaScript for different components.

You also have available to you a set of components for specifically working with database tables and tabular data, and these components make it a straightforward matter to retrieve and display database data. The Query Editor provides a graphical interface to help you develop more elaborate SQL queries.

Page Navigation is another feature that helps you design the flow your application. You use this interface to link pages together, either at the page level or at the component level.

All in all, these tools take care of many of the page layout and coding chores required to develop a web application. You can concentrate on getting the right look for your application's web pages and its business logic, and leave the rest to NetBeans. ☺



**NetBeans**



**NetBeans**  
magazine