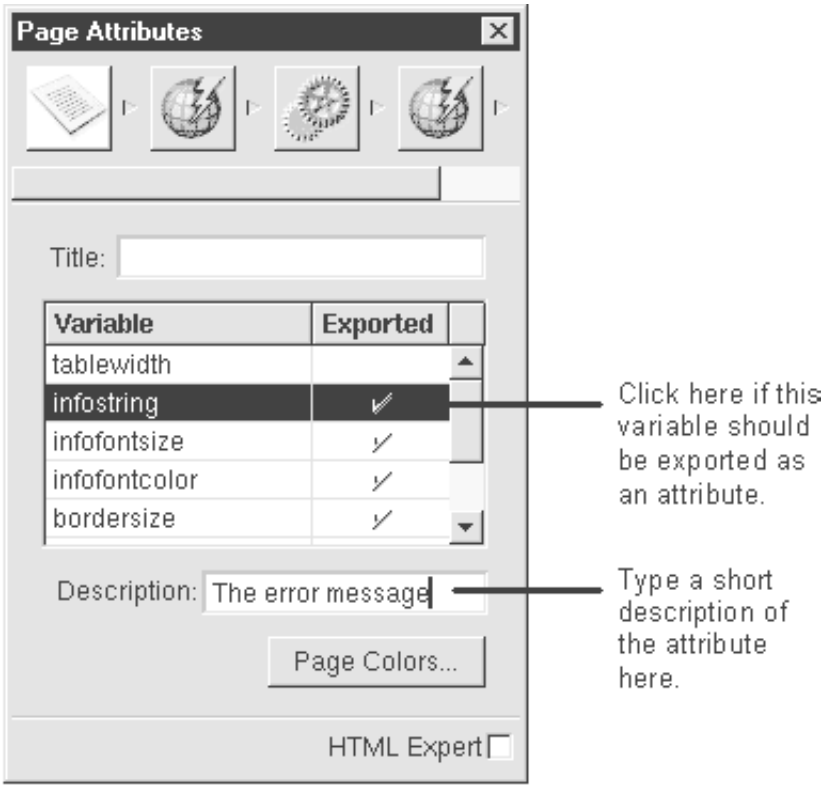
Advanced WebObjects Builder Topics

This chapter describes things that make creating applications in WebObjects Builder easier.

Creating Reusable Components

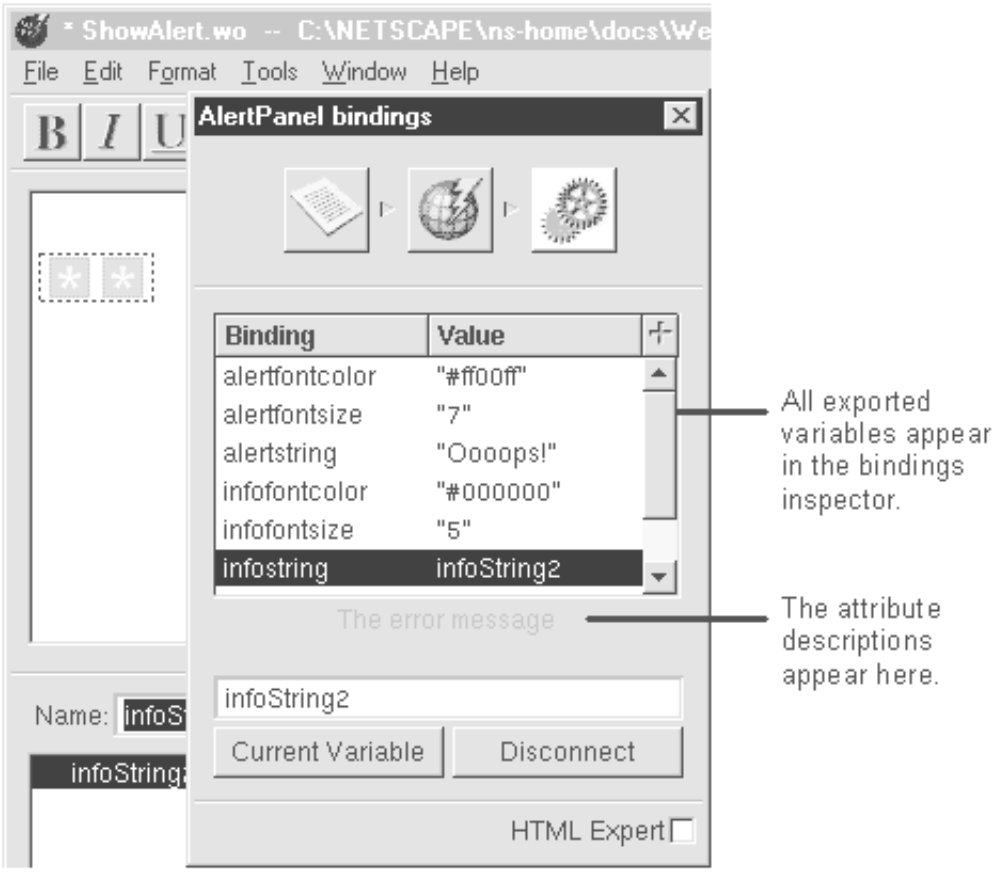
1. Create the component either inside an application or outside of an application by choosing File->New.
2. Write the component's HTML and script the same way you normally do.
3. Open the component's page inspector. (Click the inspector button to bring up the inspector window.)
4. Click the "Exported" field for each variable that the parent component should bind to.
5. Type brief documentation for each exported variable in the Description field.



In WebObjects Builder, a reusable component is simply a component that exports it variables. Other than that, there's no difference between creating a component to be reused and creating a component that's used only once. See "Reusing Components" to learn how to use a component that has been created for reuse.

When you use one component (the *child component*) inside of another component (the *parent component*), the child component is treated as a dynamic element. Like most dynamic elements, the child component may have attributes that the parent component should bind to variables or methods from the parent's script. To make such a binding possible in WebObjects Builder, you must be able to see the child component's attributes in the bindings inspector when you inspect it from the parent component.

You set up the attributes in the child component's page inspector. The page inspector lists all of the component's variables. Click "Exported" for each variable that should be considered an "attribute," that is, that the parent component might want to bind a variable or method to. When you do this, you'll see the following when you inspect the child component.



If you don't export a variable, the child component has total control over that variable's value. It's for internal use only. Even if a component variable is exported, it's a good idea to initialize it to some default value in the child's `init` method so that the application doesn't fail if the parent doesn't provide a value.

The chapter "Creating Reusable Components" in the *WebObjects Developer's Guide* tells you how to design a component so that it can be reused easily. In addition, read "Reusable Component Limitations" (in this guide) to learn about some important limitations when you use WebObjects Builder to create a reusable component.

Reusable Component Limitations

When you create a reusable component in WebObjects Builder, there are some limitations:

- Reusable components stored on a custom palette must use only lower-case letters in the names of exported variables.

  When you store a component on the palette, all of its exported variables are stored on the palette along with it. The palette ignores case. When you drag the component off of the palette, the exported variable names appear in the inspector in all lower-case letters and are written out in the parent component's .wod file that way. (The .wod file stores bindings.)

  For example, the child component may have an exported variable alertString. After the child component is dragged from the palette to the parent component, the bindings inspector displays that variable as alertstring. The parent component then binds to an alertstring attribute. At run time, WebObjects tries to bind the parent component's script to the alertstring variable in the child component fails because alertstring does not exist.

- Form elements are surrounded by <FORM> tags.

  When you drag an element from the Form Elements palette to a component, WebObjects Builder inserts a <FORM> tag before that component and a </FORM> tag immediately after it. If you create a reusable component in this manner and intend to use it inside a parent component's form, it won't work because of the extra <FORM> tags.

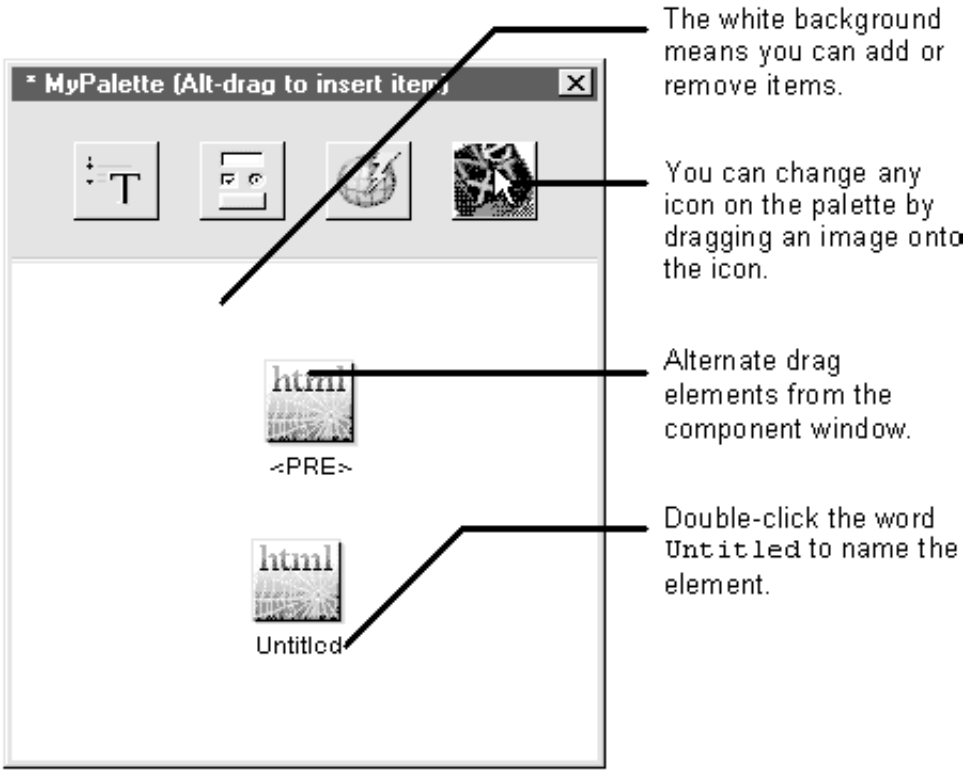
  As a workaround, you can do the following:

  1. Create the component in WebObjects Builder, and then save it.
  2. Open the component's HTML file (*MyComponent*.wo/*MyComponent*.html) in a text editor.
  3. Delete the <FORM> start and end tags, then save the file.

  If you're storing the component on a custom palette, do this before you add the component to the palette.

Creating a Palette

1. Choose Tools->Palettes->New.
2. Choose a name and a location and click Open to create a new empty palette.
3. Add items to the palette.
   - Add components to the palette by dragging them from the application window or from the file system onto the palette.
   - Add other elements from the component window by selecting them and then holding down the Alternate key and dragging them from the component window onto the palette.
4. When you're finished adding items, choose Tools->Palettes->Toggle Editing to turn off edit mode.
5. Save your changes by choosing Tools->Palettes->Save. Be sure to choose Save from the Palettes menu, not the File menu.



By creating a custom palette, you can store portions of an HTML page for later reuse. You can store any combination of elements from a page — an individual element, several elements including text, or an entire page — as a single palette item.

For example, you can use custom palettes to store:

- Boilerplate text, such as a header, that you add to every page
- References to all your icons and background images
- Custom tags that you create
- Tags that require extra set up that you use frequently (for example, if you frequently use a preformatted element, you might want to store it on the palette).
- Reusable components

When storing a reusable component on a palette, you can drag it from one of three places: from the file system, from the application window, or from the parent component's window. If you drag from the parent component's window, any bindings that were made before dragging the component are preserved.

Note: If you're storing a reusable component, make sure all of its exported variables are in lower-case letters. See "Reusable Component Limitations"

The custom palette appears in the palette window along with the standard palettes and you use them the same way. When you first create a custom palette, it is displayed in editing mode. In this mode, you can drag items on to and off of the palette. That is, dragging an item off of a palette deletes it. When your palette is ready to use, perform the command Tools->Palettes->Toggle Editing. Editing mode is turned off, and the background becomes the same as for

the other palettes. Once editing mode is turned off, dragging an element off of the palette does not remove it—it adds the element to the destination page just as you would expect. (As the command name suggests, you can use Toggle Editing to turn editing mode on again later if you wish to make changes.)

Important: Images and components placed on the palette are copied into the palette. If you add an item to the palette and then wish to change the item:

1. Change the item to work the way you want.
2. Choose Tools->Palettes->Toggle Editing to turn on the palette's editing mode.
3. Drag the original version of item off of the palette.
4. Drag the new version onto the palette.

Loading and Unloading Palettes

- Choose Tools->Palettes->Open to load a palette.
- Choose Tools->Palettes->Close to unload a palette.

The palette window always displays the standard palettes (Static Elements, Form Elements, and Abstract Elements). It can also display your custom palettes or palettes that others have created. For example, the palette NeXT_Root/NextDeveloper/Palettes/ClientSideComponents.wbpalette contains *client-side components*, components that are written in Java and execute on the client. Even though this palette is distributed with the WebObjects product, it's not loaded until you specifically request it.

If you want the palette window to display a nonstandard palette, choose Tools->Palettes->Open and select the .wbpalette file to open. Once you have opened a palette, WebObjects Builder always displays it unless you specifically tell it not to. That is, if you quit WebObjects Builder and restart it, the nonstandard palette is opened at launch time.

If you don't want WebObjects Builder to display a palette any more, select the palette in the palette window and choose Tools->Palettes->Close. After you close a palette file, WebObjects Builder assumes you never want it opened. If you quit WebObjects Builder and restart it, it no longer opens that palette.