Writing HelloWorld

To create a WebObjects application, you do the following:

- Create a new application directory.
- Create HTML files for each page.
- Write scripts to customize the behavior of the page.
- Map the variables and actions in the script to the dynamic elements of the HTML file.
- Save and run the application.

As an example, let's look at HelloWorld. The source for HelloWorld is located in *NeXT_Root*/**NextDeveloper/Examples/WebObjects/HelloWorld.woa**.

HelloWorld

The HelloWorld application consists of two pages. Figure 1 shows the first page.

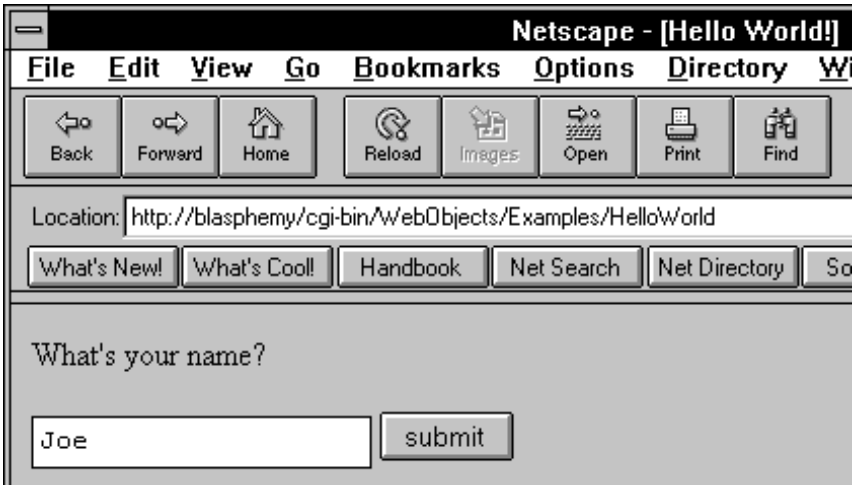

Figure 1. The First Page of HelloWorld

The presentation may vary slightly from browser to browser, but the page elements are the same regardless. The first page contains a single input field in which you type your name. Clicking Submit opens a new page with a personalized greeting. For instance, typing "Joe" and clicking Submit opens a page similar to the one in Figure 2.

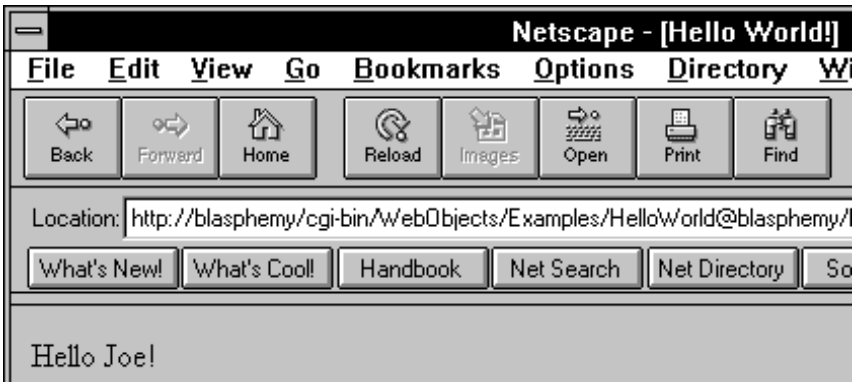

Figure 2. The Second Page of HelloWorld

The HelloWorld application has simple requirements: get the name that's entered in the input field, and when Submit is clicked, dynamically generate the HTML required to display the message in the second page, and then open the second page. More generally:

- Get user input.
- Specify page-to-page transitions.
- Determine dynamic page content and generate corresponding HTML.

Layout of a Web Application

WebObjects applications must be under **<DocumentRoot>/WebObjects** for your HTTP server to find them. For example, HelloWorld is in the **Examples** subdirectory of **<DocumentRoot>/WebObjects**, along with many other WebObjects examples. All WebObjects applications must have the extension **.woa**.

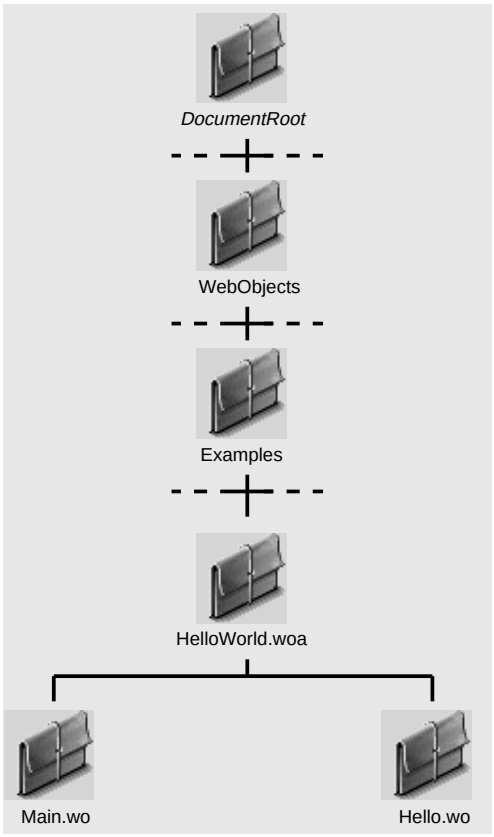

Figure 3. HelloWorld Application Directory

Under HelloWorld are two directories with a **.wo** extension. These directories, and any directory with a **.wo** extension, are called components. Components are the basic building blocks of a WebObjects application.

A component consists of three files:

- An *HTML template* (*Component*.**html**)that specifies how the page looks.
- A *script file* (*Component*.**wos**) that implements the page's behavior.
- A *declarations file* (*Component*.**wod**) that binds the dynamic elements of the template to script's variables and methods.

HelloWorld has two components—**Main** and **Hello**. The following sections describe the files for the Main and Hello components of the HelloWorld application.

Main Component Files

The template for the Main page contains the following HTML elements:

```
<HTML>
<HEAD>
    <TITLE>Hello World!</TITLE>
</HEAD>
<BODY>
    <FORM>
    What's your name?
    <P>
    <WEBOBJECT NAME = "NAME_FIELD"><INPUT TYPE = "TEXT"></WEBOBJECT>
    <WEBOBJECT NAME = "SUBMIT_BUTTON"><INPUT TYPE = "SUBMIT"></WEBOBJECT>
    </P>
    </FORM>
</BODY>
</HTML>
```

The WEBOBJECT elements—a new kind of markup element introduced by WebObjects—are replaced with dynamically generated HTML when the HelloWorld application returns the Main page. The declarations file specifies the kind of objects that perform the HTML substitutions.

The declarations file for the Main page contains the following two declarations:

```
NAME_FIELD: WOTextField {value = visitorName};
SUBMIT_BUTTON: WOSubmitButton {action = sayHello};
```

Each declaration corresponds to a WEBOBJECT element in the template. Each declaration declares an object—a WODynamicElement—to represent its corresponding WEBOBJECT element. The declaration specifies what kind of object to create and how to configure it. Specifically, a declaration associates a WODynamicElement with variables and methods defined in the corresponding script file. Dynamic Elelments are described in more detail in the chapter “How WebObjects Works” in the *WebObjects Developer’s Guide*.

The script for the Main page contains the following lines:

```
id visitorName;
- sayHello
{
```

```
    id nextPage;
    nextPage = [WOApp pageWithName:@"Hello"];
    [nextPage setVisitorName:visitorName];
    return nextPage;
}
```

Together, these three files (template, declarations, and script) establish what action to take when a user clicks Submit, which is to return the second page with a customized greeting. The files do two things:

- Store the name entered by the user.
- Return the Hello page.

Storing the Name

The declaration for the NAME_FIELD WEBOBJECT element:

```
NAME_FIELD: WOTextField {value = visitorName};
```

specifies how to store the name entered by the user. It associates the NAME_FIELD element with the **visitorName** variable declared in **Main.wos**.

The declaration specifies that a WOTextField object generates HTML for the NAME_FIELD element, and that the variable assigned to the **value** attribute of the WOTextField object**—visitorName**— stores user input.

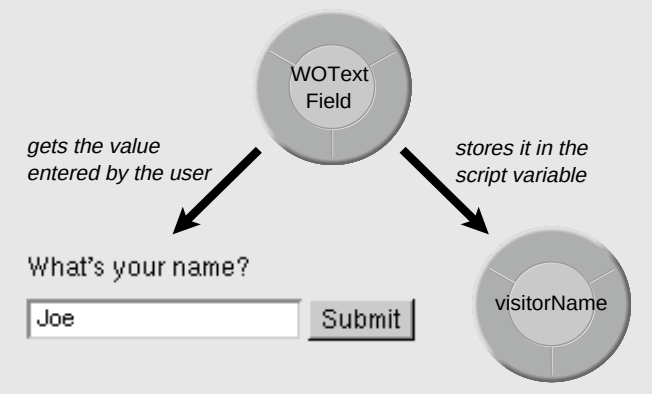

Figure 4. Getting and Storing a Value

For more information on WOTextFields, see the WOTextField description in the Dynamic Elements Reference.

Returning the Hello Page

The declaration for the SUBMIT_BUTTON WEBOBJECT element:

```
SUBMIT_BUTTON: WOSubmitButton {action = sayHello};
```

specifies how to return the Hello page. It associates the SUBMIT_BUTTON element with the **sayHello** method defined in **Main.wos**.

The declaration specifies that a WOSubmitButton generates HTML for the SUBMIT_BUTTON element, and that the action assigned to the WOSubmitButton object, **sayHello**, is invoked when a user submits the form.

The **sayHello** method:

```
id visitorName;
- sayHello
{
    id nextPage;
    nextPage = [WOApp pageWithName:@"Hello"];
    [nextPage setVisitorName:visitorName];
    return nextPage;
}
```

finds or creates a component object to represent the Hello page by sending a **pageWithName:** message to WOApp—the global variable representing HelloWorld's application object. If an object representing the Hello page doesn't exist, **pageWithName:** finds the **Hello.wo** component directory and creates a component object from it.

Next, **sayHello** sets the Hello component's **visitorName** variable by sending a message to the Hello component. To access the variables declared in another script file, you use *accessor* methods. There are two kinds of accessor methods: *set* methods that set the value of a variable and *get* methods that return the value of a variable.

Set methods have the form **setVariableName:**, where **variableName** is the name of the script variable. For example, the Hello page declares the variable **visitorName**, so the method **setvisitorName:** is automatically available to set its value. Notice that the "n" in the variable name is lowercase, but in the set method name, it's uppercase. The method name for a set method capitalizes the first letter of the variable name if it's not an uppercase letter, and then prepends the word "set" to it.

Get methods have the form **variableName**, where **variableName** is the name of the variable. For example, to get the value of the Hello component's **visitorName**

variable, you invoke a method of the same name. In WebScript, both set and get accessor methods are automatically available for all script variables.

Note: It is customary to start all variable names with lowercase letters.

After setting the Hello component's **visitorName** variable, **sayHello** returns the Hello component.

Hello Component Files

The files for the Hello component establish how to generate the personalized greeting. The template for the Hello page contains the following HTML elements:

```
<HTML>
<HEAD>
    <TITLE>Hello World!</TITLE>
</HEAD>
<BODY>
    Hello <WEBOBJECT NAME = "NAME_STRING"></WEBOBJECT>!
</BODY>
</HTML>
```

The declarations file contains the following declaration of a WOString object to substitute the user's name for the NAME_STRING WEBOBJECT element:

```
NAME_STRING: WOString {value = visitorName};
```

Like WOTextField objects, WOString objects have a **value** attribute. The WOString is responsible for getting the value in **visitorName** and putting it in the corresponding page.

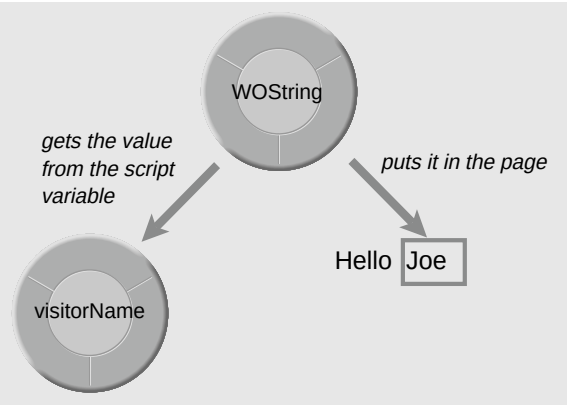

Figure 5. Getting a Value and Displaying it in a Page

The script for the second page contains only one line:

```
id visitorName;
```

The **visitorName** variable must be declared so it can be associated with the NAME_STRING element in the template file. Recall that **visitorName** is set from the Main component in the **sayHello** method.

Running the Application

To run HelloWorld, start up a web browser and enter this URL:

```
http://localhost/cgi-bin/WebObjects/Examples/HelloWorld
```

or, more generally:

`http://`*web_server_host*`/`*cgi-bin_directory*`/`*adaptor*`/`*application_directory*

For more information about the format of this URL and to learn what happens when you run a WebObjects application, see "Connecting to a WebObjects Application" in the *WebObjects Developer's Guide*.