Introduction

WebObjects is an environment for developing and deploying World Wide Web applications. For development, it provides a scripting language and objects that you use to create web applications. For deployment, it provides a system of interrelated components that connect your WebObjects applications to the Web.

WebObjects gives you all the benefits of object technology. At the same time, it scales to accommodate the complexity of your programming tasks. You can create simple applications without having to compile anything—just implement the logic in scripts. Scripted applications can even access your corporate database. For more complex tasks, you can easily combine WebObjects with your own compiled objects.

The Ingredients of a WebObjects Application

In its simplest form, a WebObjects application contains these ingredients:

- Dynamic HTML pages (called *components*)
- WebObjects classes that provide an infrastructure for the web application
- An application executable that responds to requests from a web browser

Of these ingredients, you only have to provide the components. Create an application directory under ***<DocumentRoot>*/WebObjects** (***<DocumentRoot>*** is your HTTP server's document root), and place the components in that directory. The application directory must have the extension **.woa**.

A default application executable is provided as part of the WebObjects package. The executable uses instances of the WebObjects classes to receive requests from a web browser and responds to them using the components that you provide.

More complex WebObjects applications may also contain:

- An optional application script that creates and manages application-wide resources
- An optional session script that creates and manages session-wide resources
- Optional compiled code that implements custom data and logic (WebObjects Pro and Enterprise only)

For information on using application or session scripts or including compiled code, see "The Role of Scripts in a WebObjects Application" in "Using WebScript" and "Compiling and Debugging WebObjects Applications."

Components

To write a WebObjects application, you create components and connect them together. A component is a page or portion of a page that has both HTML content and behavior. Each component is located in its own directory named *Component*.**wo** and generally contains these files:

- An *HTML template* (*Component*.**html**) that specifies how the page looks.
- A *script file* (*Component*.**wos**) that defines the component's attributes (variables) and implements its behavior in WebScript.
- A *declarations file* (*Component*.**wod**) that binds the dynamic elements of the template to the script's variables and actions.
- If necessary, any images or other resources the component uses.

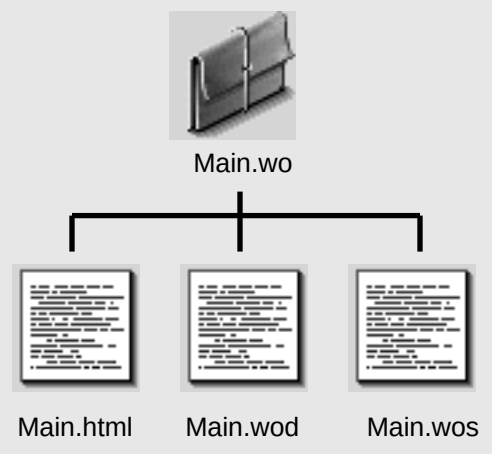

Figure 1. The Contents of a Component Directory

The script file is only included in scripted components. You may also write *compiled components*, which use Objective-C instead of a script file.

The first page of a WebObjects application is usually named **Main.wo**. When users start a session with a WebObjects application, they can specify the name of the first page, but such a practice is uncommon. If no page is specified, WebObjects applications look for the **Main.wo** component.

Note: Not all components represent an entire page. You can nest small, reusable components inside a component representing a whole page. For more information, see the "Creating Reusable Components" chapter.

Application Executables

Applications that don't contain compiled code use **WODefaultApp** (located in **NextLibrary/Executables**). This application uses the resources you provide to respond to user requests.

If you incorporate compiled code into your WebObjects application, you must also provide the application executable. You must write a **main()** function, compile the source code, and link it with the WebObjects library. You place your executable in **NextLibrary/WOApps**. See the "Compiling and Debugging WebObjects Applications" chapter for more information.

Note: If you place the application executable in **NextLibrary/WOApps**, you can also place the **.woa** there as well. For more information, see the description of **WOApps** in "Where Things Go."

Connecting to a WebObjects Application

To connect to a WebObjects application from a web browser, you open a URL with the following form:

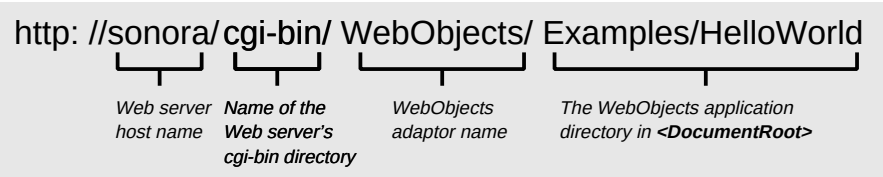

Figure 2. URL to Start a WebObjects Application

Communicating with a WebObjects application involves the following processes:

- *An HTTP server*. Any HTTP server that uses the Common Gateway Interface (CGI), the Netscape Server API (NSAPI), or the Internet Server API (ISAPI).
- *A WebObjects adaptor*. Acts as an intermediary between WebObjects applications and HTTP servers. Adaptors insulate applications from server interfaces by handling all server communication. Simply by switching

adaptors, you use a different HTTP server and a different server interface without modifying application code.

- *A WebObjects application executable.* The application executable receives incoming requests and responds to them, usually by returning a dynamically generated HTML page.

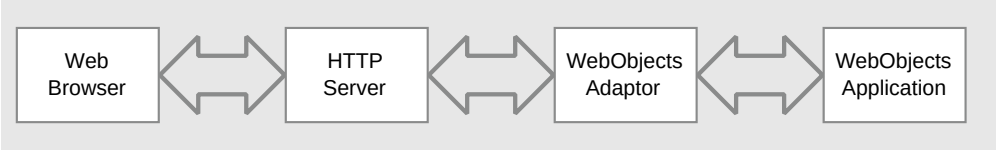

Figure 3. Chain of Communication between Browser and WebObjects

WebObjects Adaptors

The WebObjects adaptor receives requests from the server, repackages the requests in a standard format, and forwards them to an appropriate WebObjects application. All WebObjects adaptors communicate with WebObjects applications in the same way, but they communicate with HTTP servers using whatever interface is provided by a particular server. For example, the WebObjects CGI adaptor uses the Common Gateway Interface, the Netscape Interface adaptor uses the Netscape Server API, and the Internet Server adaptor uses ISAPI. Thus, WebObjects adaptors can take advantage of server-specific interfaces but still provide server-independence.

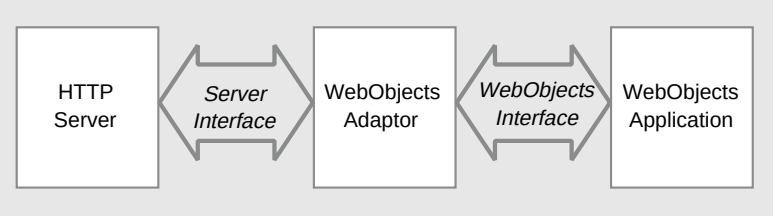

Figure 4. The Role of a WebObjects Adaptor

By default, WebObjects uses the WebObjects CGI adaptor. The Common Gateway Interface is supported by all HTTP servers, so you can use the CGI adaptor with any server—including those that are publicly available. As performance demands increase, use one of the other adaptors with a server that supports the corresponding API (Netscape Server API or Internet Server API). Such servers are capable of dynamically loading the adaptor, eliminating the overhead of starting a new process for each request. As shown in Figure 5, the communication between the adaptor and the HTTP server occurs inside a single process.

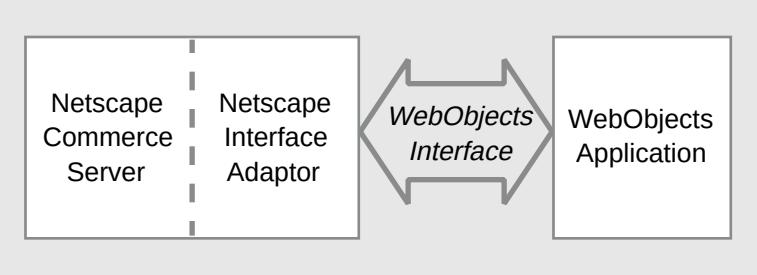

Figure 5. The Netscape Interface Adaptor

"Serving WebObjects" describes how to configure a WebObjects adaptor.

The WebObjects Application Executable

When a WebObjects application receives a request from a WebObjects adaptor, it processes the request in three phases. As shown in Figure 6, the application uses the page-to-script mappings defined in the declarations files to:

- Extract the values from the request and map them to the script.
- Invoke an action.
- Generate a response page.

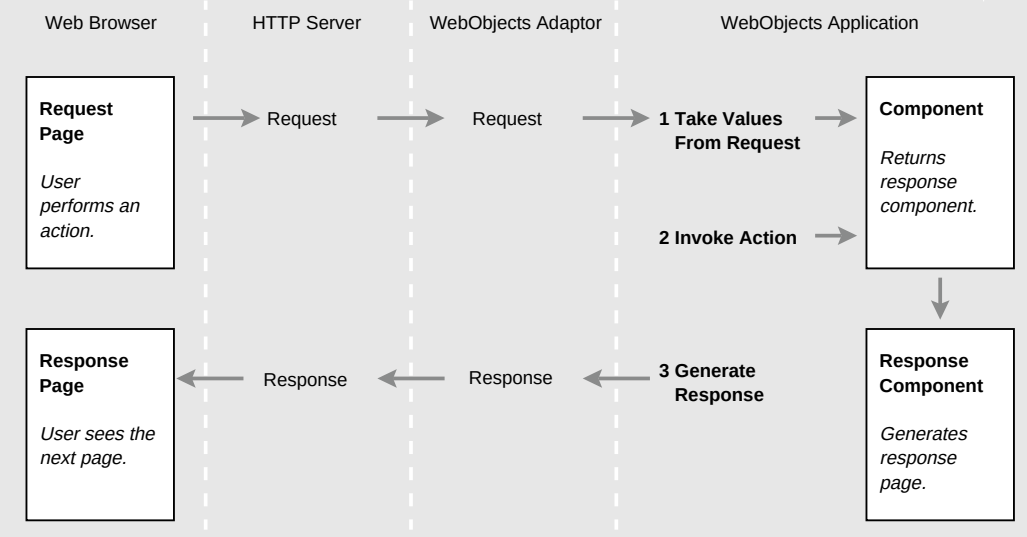

Figure 6. Request-Response Loop

The following sections describes what happens during each phase.

Take Values From Request

The application prepares for the request by updating variables in the request page—the page from which the request was made. That is, if a user has provided any input that maps to a component variable, the application assigns the new value to the variable. For example, when a user clicks Submit in the first page of the HelloWorld example application (in **NextLibrary/Examples/WebObjects/HelloWorld.woa**), the application gets the value from the text field and assigns it to the **visitorName** variable defined in the Main component.

Invoke Action

After preparing for the request, the application determines whether or not the user has triggered an action. If an action has been triggered—for example, if the user clicked a button or a hyperlink—the application invokes the action method that corresponds to what the user did. For example, clicking Submit in HelloWorld has the effect of invoking the **sayHello** action method. An action method returns a component that represents the *response page*—the page that is sent back to the web server. **sayHello** returns a component that represents the

Hello page. If the user does not trigger an action, the components for the request page also represents the response page.

Generate Response

The response page component generates the HTML for the response. Using the HTML template and declarations file, the component generates the HTML that is eventually displayed in the user's web browser. For example, after Submit is clicked in HelloWorld and **sayHello** returns the component for the Hello page, the Hello page component generates the resulting personalized greeting.

Where Things Go

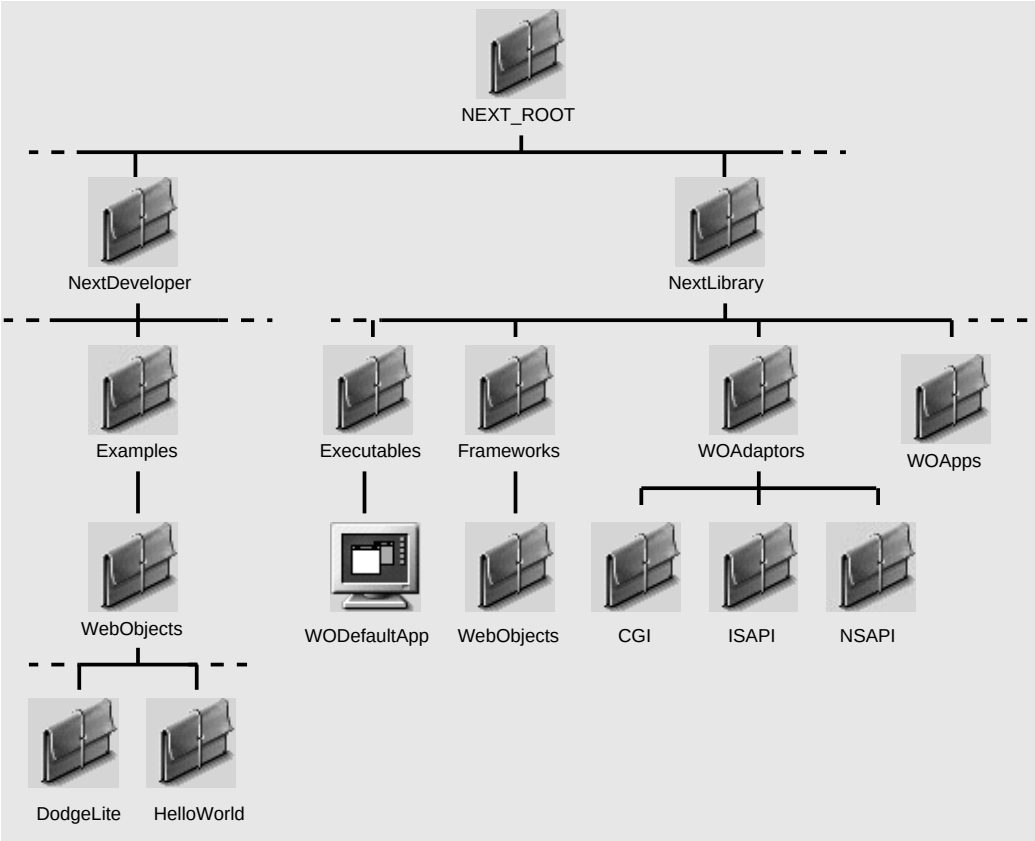

Figure 7. WebObjects Directories

NeXT_ROOT, the installation directory, depends on the platform you are using:

Platform	Installation Directory
Mach	**/**
Solaris	**Defined at installation time.**
Windows NT	Defined at installation time.

After you install WebObjects, you'll find the following items:

- **NextDeveloper/Apps/WebObjectsBuilder.app** contains the WebObjects Builder application (WebObjects Pro or Enterprise only).
- **NextDeveloper/Examples/WebObjects** contains several sample WebObjects applications.
- **NextLibrary/WOAdaptors** contains the WebObjects adaptors. If you have WebObjects Pro or Enterprise, this directory also contains adaptor source code, which you can compile for additional platforms.

  Your HTTP server does not access the adaptor in this directory. Instead, it accesses a link or copy of it in the server's **<cgi-bin>** directory.
- **NextLibrary/WOApps** is an empty directory when first installed. If you create applications that contain compiled code, you place the application's executable in this directory.

  If the application's executable is in **WOApps**, you can also place the application's **.woa** directory in **WOApps**. This ensures the **.woa** directory's privacy; if you place the **.woa** under the document root and outside users have read access on **.wos** and **.wod** files, they have access to the application's source. However, if the application imports any images or sounds, you must leave a "sparse" copy of the application in the document root so that the client's browser can find these resources. In this case "sparse" means that the application's directory structure is reproduced in the document root, but the only files it contains are the static resources that the server must dispense to a client's browser.

  Note: Applications in **WOApps** must include the executable file. If you have a WebObjects application that relies on the generic application executable (**NextLibrary/Executables/WODefaultApp**) and you want to place it in **WOApps**, you can still do so. Copy **WODefaulApp** into that application's directory and name the copy after your application.

- **NextLibrary/Executables** contains **WODefaultApp**, the generic application executable.
- **NextLibrary/Frameworks** contains the WebObjects framework, the library of WebObjects classes. This is also where you find header files.

To run the WebObjects applications you write, your web server must be able to access the adaptor and the application's code or script. Therefore, after installing WebObjects, make sure your web server's directories contain the following links or copies:

- *<cgi-bin>/WebObjects* is a copy of or link to the CGI adaptor **NextLibrary/WOAdaptors/CGI/WebObjects** (or **WebObjects.exe**).
- *<DocumentRoot>/WebObjects/Examples* is a copy of or link to **NextLibrary/Examples/WebObjects**.

Figure 8 shows the resources that should be present in your web server after the WebObjects installation process is completed. If any of these links or copies are missing, check the instructions "About WebObjects."

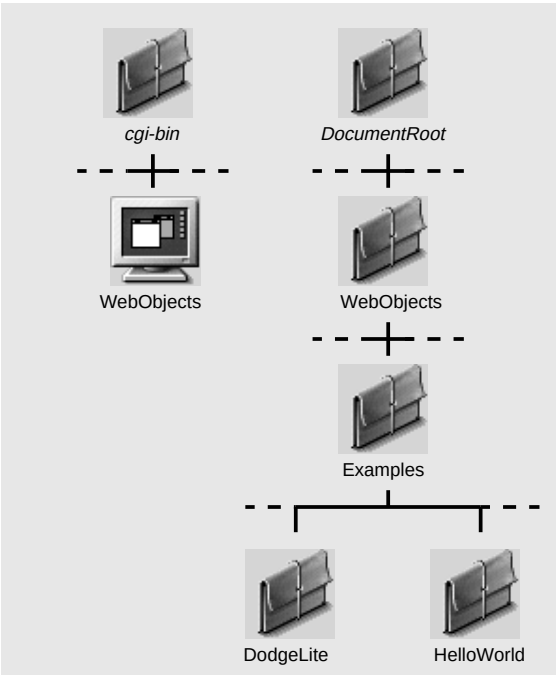

Figure 8. The Contents of Your Web Server's Directories After Installing WebObjects

Summary

What's in a WebObjects Application?

A typical WebObjects application contains the following ingredients:

- Components that specify the content, presentation, and behavior of the application's pages
- An optional application script that creates and manages application-wide resources
- An optional session script that creates and manages session-wide resources
- Optional compiled code that implements custom data and logic
- WebObjects classes that provide an infrastructure for the web application

What Parts Do I Write?

You write the following parts of a WebObjects application:

- Components consisting of HTML templates, script files, and declarations files
- An optional application script
- An optional session script
- Optional compiled code

How Do I Run a WebObjects Application?

To run a WebObjects application, you open an URL with the following form:

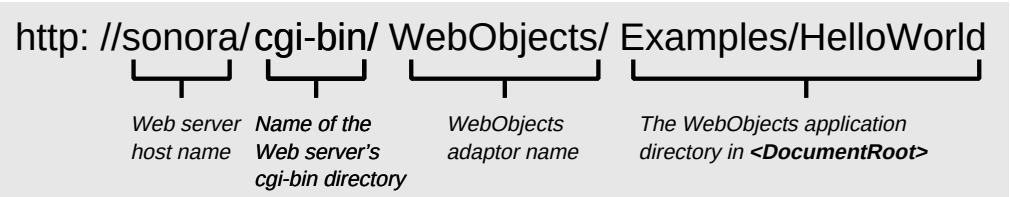

Figure 9. URL to Start a WebObjects Application

How Do I Connect a WebObjects Application to the Web?

To connect a WebObjects application to the Web, you need the following:

- *An HTTP server*. You can use any HTTP server that uses the Common Gateway Interface (CGI), the Netscape Server API (NSAPI), or the Internet Server API (ISAPI).
- *A WebObjects adaptor*. A WebObjects adaptor connects WebObjects applications to the Web by acting as an intermediary between web applications and HTTP servers.
- *A WebObjects application executable.* An application executable receives incoming requests and responds to them, usually by returning a dynamically generated HTML page.

What Happens Behind the Scenes?

Behind the scenes of a running WebObjects application, the application enters a request-response loop each time it receives a request. In the request-response loop, a WebObjects application uses the page-to-script file mappings defined in declarations files to:

- Take values from the request.
- Invoke an action.
- Generate a response page.