



Introduction

This book describes the Web-based shell, a new kind of user interface (UI), that until now has not been available to Web developers. Like all computing environments, the UI determines what a user can do with a system, how a user perceives its capabilities, and how effectively a user can perform tasks.

The nature of a UI drives the way a computer is used. For example, a bank's ATM terminal guides its users through performing simple account management tasks. These tasks are made accessible through a UI that presents users with simple buttons and dialog boxes to encourage ease of use and quick transactions. On the other hand, the same banking system provides very different UIs to its employees. A bank teller, for example, must be given access to multiple accounts with the ability to perform advanced tasks. Still, a bank system administrator would be given a more complex UI. Conceivably, all these UIs would run on the same system and would have access to the same underlying capabilities. But they are tailored to provide only the capability necessary to make the user more effective, depending on the user's role.

Since the advent of the Web and Web-based applications, only a few new tools have been tailored for Web development. Therefore, Web developers typically do their work in either a UNIX shell, or a client-side GUI such as Visual Studio. Both of these development environments are quite good for their designed purposes, but they weren't designed for the Web. Shells are good for developing command-line applications, and GUI tools are good for creating GUI applications. But each fails to provide a true integrated development environment (IDE), for the Web developer.

If GUI applications are best developed with GUI IDEs, and command-line applications are best and most often developed in shell environments, why not use Web-based applications to develop other Web-based applications? Web developers can benefit from working with

Web-based IDEs in practical ways that make their common tasks simpler. This book discusses the advantages of developing Web applications with Web IDEs in general, and with the Web shell in particular.

Why the Web Shell?

The most common UI today is the windowing environment (sometimes called a GUI or graphical user interface). Most computers sold to the general public have a commercial implementation of this kind of UI. GUIs are mouse-based graphical environments that use windows to encapsulate programs and interact with the user through graphical cues and dialog boxes. This kind of UI is good for many tasks, but it doesn't work well across networks (if you've ever logged into your Windows computer remotely, you know how true that is), and they don't provide convenient access to the professional user. Although client GUI applications are popular, their Web-based counterparts have shown themselves to be spectacularly more effective when you are performing tasks that require network content.

Before the wide adoption of GUI environments, the most common interface was the shell. A shell is a text-based interface in which users issue commands to the computer by typing them on a command line. A shell user, instead of double-clicking a folder icon to see its contents, would type a command, such as `ls` to list the file contents of a directory to the screen. This simple example doesn't demonstrate an advantage with either the shell or the windowing environment. But what if the task were to require getting the directory contents and determining whether the output appeared in the contents of a file? Such a task would be simple in a shell environment, but might require several steps in a GUI if it even supported such unusual functionality. Shells also work well across networks. It is usually just as convenient to run a shell on a remote computer as on the local host.

All major operating systems still supply shell interfaces alongside their polished graphical environments, because a command-line environment provides quick, networked access to a library of powerful commands, many of which can be combined to perform a complex set of related instructions. To a skilled user, a shell can be more powerful than a GUI. If the user knows which commands to use, it is just a matter of will. The user need not even lift his hands from the keyboard. The primary disadvantage of a shell is that its users need to learn the command set before it becomes useful. Until then, to the inexperienced user, most shells behave rudely.

The wide adoption of the graphical user interface has made computing accessible to the inexperienced users and has helped make personal computing a mass-market phenomenon, because GUI designers have worked to make computers as simple to use as the breadbasket at the dinner table. To move a file, grab it, move its pictorial representation (its icon) to where you want it, and let go. Similarly, ATMs have had great success partly because of their simplicity. Whereas banks previously wouldn't have allowed access to their computer systems from a publicly available terminal on a sidewalk, the ATM protects the integrity of the bank's computing environment and makes the user experience simple and effective.

You can bet, though, that the people who maintain banking systems use shell interfaces for the bulk of their tasks. As in so many corporate and industrial arenas, the workhorse UI for professional users — even though windowing environments have become very mature and sophisticated — is the ugly-looking but powerful shell interface.

The Web shell uses both of these concepts to its advantage. It provides a command-line interface to a Web-based system, but it also limits a user's access to a system. That means that a Web host can safely provide Web shell access to a system, just as a bank can place an ATM on a sidewalk, without risking the file areas that should be off limits to a remote user, and the user has the power of a Web-enabled shell environment.

The Web Shell

The Web shell is a Web-based implementation of the shell concept. Crudely speaking, it can be considered a series of Web pages that look and behave like a traditional shell and act upon the server hosting the application. The Web shell is written in H₂O, a secure, cross-platform scripting language. It parses the commands the user submits and responds to the commands accordingly. When a user submits a command, it is interpreted and executed through this scripting language. The result is a platform-independent product that doesn't rely on any shell functionality native to the host.

The Web shell is unique in that it enables users to operate a remote computer by way of a Web browser that acts like a command-line interface: A user can log onto a remote computer using a Web browser on any type of system, whether it's a desktop computer or a Web-enabled cell phone, and perform tasks on the remote system. No command-line environment is required on either the host or the client. And because the Web shell can be hosted by all the major platforms, the user could perform such functions on any type of system that is running Web-shell software. This means that a copy of the Web shell running

on Windows will behave in exactly the same way as a copy running on, for example, BSD. A developer writing in a cross-platform language using the Web shell therefore generally doesn't have to account for operating system issues.

Web Computing

From a Web perspective, the scope of a file system extends to those files that are directly accessible from a browser. This file area is known as the sandbox. From the perspective of the Web shell, the sandbox is the entire file system. One advantage to this kind of file system is that because the Web is mostly platform independent, the file system can be platform independent from the perspective of the user. Additionally, a Web sandbox contributes to system security because sandboxed users see only files relevant to Web functionality and not those files used to manage the server.

For the majority of configurations, the sandbox area comprises all files that are accessible from a browser. On a typical UNIX system, this corresponds to everything in the `htdocs` and `cgi-bin` directories. Because, from the perspective of the Web shell, the sandbox is the same across all supported types of Web servers, Web shell users by default are forced to write cross-platform Web applications. This will be true as long as the script interpreter and/or database functionality is supported on the other platform as well. Perl, PHP, H₂O, and HTML/OS work well across major platforms. HTML/OS conveniently provides an integrated database, too. A Web shell user writing a Web application in H₂O, HTML/OS, PHP, or PERL, for example, won't stray from the sandbox area where file systems can vary across different operating systems. This book explains how to use the Web shell commands `pack` and `unpack` to package and deploy cross-platform applications. It also covers how to encrypt your source-code when you deliver an H₂O or HTML/OS-based Web application to a client.

On the browser side, Web files can be rendered in many ways from within the Web browser. For example, whereas a traditional shell can provide a text-based list of the image files (`gif`, `jpg`, `bmp`, and so on) in any given directory, the Web shell, as any Web-based application, could also actually display the images, referenced by HTML image tags, from within its Web-based interface. The Web shell was written with these distinctions in mind. The Web content should be manageable as it is from within a traditional shell environment, but it should also be viewable as the application user sees it. The Web shell maintains in its various capacities the perspective of the browser-based Web. Therefore, although you use a

typical command-line interface when working within the Web shell, you can also view files as though you were working within a browser.

It is because of its Web-based perspective on the server side, as well as on the client (browser) side that the Web shell can be thought of as a Web IDE.

Practical Uses

The Web shell was designed for use as a file-management system as well as a Web development tool. As a file-management system, the Web shell has file manipulation capabilities similar to those of a traditional shell. These basic shell operations, include adding, deleting, moving, and renaming files and directories within the Web sandbox, and more advanced capabilities, such as text file manipulation, searching, and file processing. This book covers the commands available in the Web shell as well as how to use these commands to create Web applications and to manage Web systems.

Because most Web development consists of editing text files and testing them in a Web browser, the Web shell also ships with a Web-based file-editing tool supplied as an alternative to traditional command-line text editing tools, such as `vi`. The shell's editor is a browser-based interface that naturally allows for development in any Web scripting language. Some of the more powerful of the languages supported are `H2O`, `HTML/OS`, `PHP`, `C#.NET`, `Perl`, and `Java Server Pages (JSP)`. The only caveat is that files need to be in the Web sandbox to be visible to the Web shell. A file outside of the sandbox (outside of the `cgi-bin` or `htdocs` folder) will not be accessible from the Web shell. This seeming restriction ends up being an advantage to developers and hosting providers, as you'll see below.

Obviously, there are several ways of developing a Web application. A developer using a traditional shell logs into a system via a remote connection, such as `Telnet` or `SSH` (if such access is provided by the host) and then edits a file using a text-based program, such as `vi`. This process requires knowledge of `vi`'s arcane command set as well as how to use an uncorrelated browser interface to view the resulting application.

In contrast, a Web shell user logs into the Web shell through a browser and edits a text file using the Web-based text editor provided. The Web shell user edits the file in a text area in his browser. No text editing application is required by the host, and the user needs no special knowledge or application. And because the development environment is the same

as the deployment environment, the developer has the ability to run the script from within the Web shell platform. No longer does the developer have to coordinate the development process with application testing. The editing environment of the Web shell provides testing functionality that is an integrated part of the development environment.

Hosting Providers

The Web shell's file system sandbox provides added usefulness to hosting providers. By supplying Web shell access, hosting providers achieve a balance between providing access to clients and keeping inexperienced or malicious users from interfering with the system and other users. The Web shell keeps individual Web shell users in their file system sandboxes: Users can manage the files needed to run a Web site but don't see files that should only be of interest to a system administrator. In this way, a hosting provider of shared accounts on a single machine can provide full Web-enabled shell access to power users, and keep distinct users from hindering each other.

About This Book

This book will show you how to use the Web shell to manage server files and to develop Web applications. As you proceed, you will be able to log in to your own copy of the Web shell and test the concepts and examples that we will cover. The only resources you will need are a Web shell account (which we provide for you), a Web browser, and an Internet connection. You won't need any technical knowledge either, although an understanding of HTML, structured programming, and basic Web concepts would help.

Initially, this book covers basic concepts relating to how to get started with a copy of the Web shell. Subsequent chapters show you how to use more advanced features relating to application development and deployment. As you progress, you build various small Web applications to learn how Web development is accomplished using the Web shell's editor. By the end of the book you should have a good understanding of how to use the Web shell to manage a Web site, create a Web application, deploy a project, protect your source code, and even add your own customized commands to the Web shell's command set.

Chapter 1, "Getting Started," explains how to gain access to a copy of the Web shell that you can use as you progress through this book. It covers how to access the copy of the Web shell provided with the purchase of this book. It also shows you how to install and use your

own copy of the Web shell. You also get a tour of the Web shell's functionality and the environment in which it runs, H₂O or HTML/OS.

Chapter 2, "File Management," shows you how to use the Web shell as you would use a traditional shell. It covers basic commands and methodologies you can use to administer a Web file system with the Web shell. It also covers more advanced topics relating to the HTML/OS virtual file system including sandboxing and mirroring.

Chapter 3, "The Shell Environment," shows you how to develop Web applications with the Web shell. It explains how to use the Web shell as an IDE to create and deploy Web applications and introduces you to the primary language used throughout this book, HTML/OS. Finally, it walks you through developing and testing a simple Web application using the Web shell's editor.

Chapter 4, "Redirection," covers how to combine shell commands to achieve more powerful functionality and how to write the output of a command to a file. It also shows you how to use some of the Web-specific commands you can use to transfer Web-based files from a remote server to your host, as well as between your host and your local machine.

Chapter 5, "Useful Web Shell Commands," provides examples of how to use some of the Web shell's more commonly used commands to perform your daily development and administrative tasks, how to use the shell's Web-based nature to make Web management easy, and how to package, deploy, and install a Web application using the shell's deployment commands.

Chapter 6, "Creating Custom Commands," shows you how to write your own commands for the Web shell. This chapter covers the H₂O language in more detail and shows you how to use H₂O to program the Web shell's command API.

About Web Shell Culture

The first iteration of the Web shell was developed in California at the offices of the Web language company, Aestiva. It is written in Aestiva's own language called H₂O. H₂O is a cross-platform Web development environment. HTML/OS is an extension of H₂O that includes an advanced database. Both H₂O and HTML/OS include the Web shell. Although the Web shell is most commonly used to develop H₂O and HTML/OS applications, it is not

restricted. This book does not assume the reader will necessarily be developing in H₂O or HTML/OS.

The Web shell can be used when developing ASP, PHP, PeRL, and JSP-based applications.

This book does cover H₂O basics but is not a comprehensive resource on the subject. For detailed information on using H₂O visit h2o.aestiva.com or look into the book, *Advanced Web Sites Made Easy*, by D.M. Silverberg.

The Web shell continues to be updated and maintained by its original authors as well as by new recruits, so you can expect exciting new features on a regular basis. In addition to adding more commands, the Web shell is evolving to support interface enhancements that reduce both typing and mouse usage.

As the first book on the subject, *Learning the Web Shell* provides an important resource for developers of browser-based applications. Most important, since the Web shell is free and available for all the major hardware platforms, its popularity is likely to explode, making this book a must-read for everyone interested in gaining an appreciation for this exciting new technology.