CHAPTER

1

Introduction to IBM WAS Programming

IN THIS CHAPTER:

The Simplest WAS View

IBM's Offering for the WebSphere Application Server

What's In the Five Parts of the Book

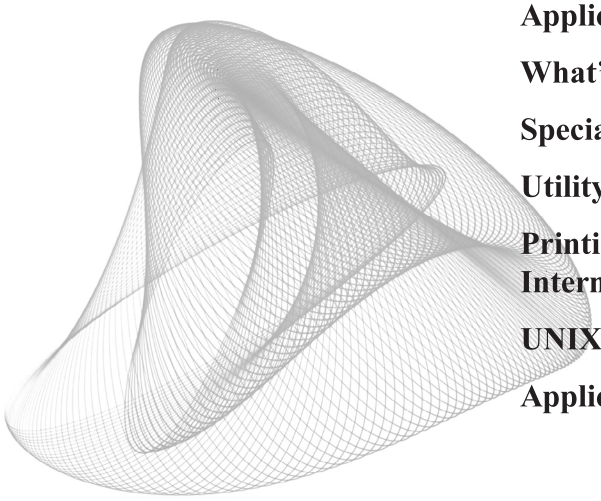
Special Consideration of WAS v5

Utility Scripts

Printing Unicode, Localization, and Internationalization

UNIX Commands, Shell, Perl, and Lynx

Applications Used in This Book



WebSphere Application Server (WAS) is IBM's Java 2 Enterprise Edition application server. WAS is in its fifth year of development, with four official releases and an upcoming fifth release in 2003. WAS is a deployment environment for J2EE enterprise applications implemented on the server side. A J2EE application server defines a platform where HTTP servlets, Java ServerPages, and Enterprise JavaBeans (EJBs) can coexist; it also outlines the framework of how these technologies work together.

After you have installed WAS, you will learn about the scripts that comprise the programmer development environment. These scripts form a group of basic shell programs that are to be included in the user profile. This ensures compatibility between the code developed and the WAS run-time environment on which the code is tested. You will also learn the essential details about the internal structure of WAS and explore the basic system commands to control WAS processes.

Most J2EE application server vendors provide customers with a J2EE Application Assembly Tool and many other (graphical-based) tools that facilitate the process of describing, assembling, (de)archiving, and deploying J2EE applications. The Application Assembly Tool (AAT) that is provided with WAS is graphical and mirrors the internal structure of a J2EE application.

Although the WebSphere AAT is a useful vehicle for generating the details of J2EE applications, it is not convenient for writing and testing web applications in a dynamic environment. Once you understand the essential elements involved in describing a J2EE web application, using alternative text preprocessing with the power of interpreted scripts becomes essential.

Perl is more appropriate for writing simple scripts that automate the process of text preprocessing and replace the lengthy procedure of using a graphical-based tool such as the AAT.

As you study WAS programming, you will learn about the system's capability to handle processes and threads needed by WAS containers. This book explains the underlying features of WebSphere by examining WAS from the outside in—that is, from the system perspective. You will be exposed to many utilities that facilitate your understanding of WAS' systematic consumption to the system resources.

This book discusses WAS generically; therefore, if you are familiar with any version, you will be able to relate to the text because it uses nomenclature common to all versions of WAS to explain the WAS run-time environment. It also describes many UNIX commands and scripts that have been tested on WAS v4, which are backward compatible with WAS v5.

This first chapter introduces the different WAS versions, describes the contents of each chapter and appendix, and discusses the approach taken in the book.

The Simplest WAS View

A client browser uses the HTTP protocol to perform I/O with an Internet server. The Internet server is an HTTP server that is patched with WAS' vendor plug-in: WS-plug-in. Requests from the HTTP server are then forwarded—by means of the plug-in—to the WAS server. The WAS plug-in intercepts these requests intended for WebSphere and forwards them to the application server. The WAS server consists of a WAS engine whose functionality is rendered by an initial

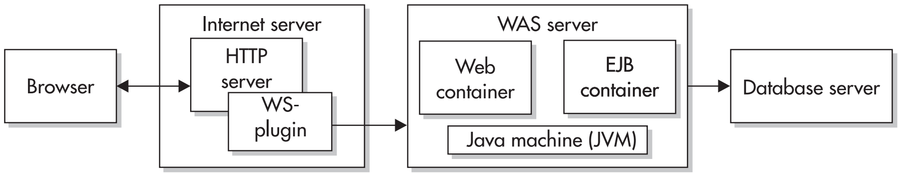


Figure 1-1 *Simple view of WebSphere Application Server*

Java machine; the WAS engine is therefore said to be bootstrapped by such an initial Java machine. The WAS server holds two containers: a web container and an EJB container. WAS communicates with the database server using Java APIs.

IBM's Offering for the WebSphere Application Server

IBM packaged WAS v4 into five deliverable products. The release of WAS v5 is expected to be packaged into three categories. The following two sections list the deliverable products of WAS v4, and (unofficially) of WAS v5.

WebSphere Application Server v4

IBM packages the latest official releases of WAS v4 into five commercial products:

- **WebSphere Application Server Advanced Edition (WAS AE)** Provides the application server that is scalable with distributed security and command-line-based administration via the WebSphere Control Program (WSCP). This version is used to group a cluster of machines (called *nodes* in WAS v4 and *cells* in WAS v5) in a logical group, hence providing scalability, failover, and high performance. This book refers to WAS AE v3.5 and WAS AE v4 simply as WAS AE. Because WAS AE is used on production computer systems to run the final web application code, it is not suited for your programming needs. WAS AE is used in a production environment after the programmers' code has been tested on the smaller version, WAS AEs.
- **WebSphere Application Server Advanced Edition Single Server (WAS AEs)** Provides an application server that is easily manageable, with all the features you need to get started programming for WAS. It is easy to install, administer, and test code with. You can test code quickly using this version because it stops and restarts WAS much faster than any other version. If you are using Linux, you can get started by downloading a free, fully functional trial version of WAS. The code developed in this book runs on both versions of WAS: WAS AE and WAS AEs. (Part III uses WAS AEs.)
- **WebSphere Application Server Advanced Edition Developer (WAS AEd)** Provides the application server with many other IBM applications and tools to write applications that run under WAS. WAS AEd is an expensive solution, and its dependency on other IBM tools hides many development aspects that a beginner needs

to be aware of. Use WAS AEs instead, so that you will have a productive environment that is cheaper and more solid than the one WAS AEd offers.

- **WebSphere Application Server Enterprise Edition (WAS EE)** Provides application servers that run on many platforms. The functionality of WAS EE is similar to WAS AE, with some extras at an extra cost. This product is intended for enterprises with a variety of platforms.
- **WebSphere Application Server for IBM z/OS and IBM OS/390** Provides a scalable, high-performance, highly available architecture with workload management for IBM zSeries and IBM S/390. This application server is also used for production environments, and if you test code using WAS AEs, you should be able to port the code to this application server.

WebSphere Application Server v5

IBM has recently announced the upcoming release of WebSphere Application Server version 5.0. As of this writing, the packaging of the new product is targeted as follows:

- **IBM WebSphere Application Server** Equivalent to WAS AEs, this product is to be used as a single-server development environment that runs on a programmer desktop. It supports standards-based programming of web, EJB, and other components, including web services.
- **WebSphere Application Server Network Deployment** Equivalent to WAS AE, this product fits into departmental computing scenarios. It supports clustering, caching, and a centralized administration for multiple application servers.
- **WebSphere Application Server Enterprise** Equivalent to WAS EE, this product is similar to the one described in the previous bullet, but it also provides additional administrative accessories—content distribution and dynamic workload management—which makes it fit in a large production environment.

What's In the Five Parts of the Book

This book consists of five parts and six appendixes. In Part I, you will learn about the WASDG environment and the many systems involved in a WebSphere region. Part II introduces a quick and essential administrative guide that you can use to understand and control the processes involved in WAS. In Part III, you will use the WASDG environment to program the WASDG application. In the process, you will learn about WAS session management, its startup classloader, its containers, and its support to Apache SOAP (Simple Object Access Protocol) programming and to JAAS (Java Authentication and Authorization Service) programming. In Part IV, you will stress-test the WASDG application and add the WASDG exception handler to the application. And in Part V, you will use WASLED/WASMON to monitor WAS and use many utilities to collect statistics that assist you in tuning WAS performance.

Part I

In Part I, you will learn how to install WAS, how to write effective scripts for a cost-effective enterprise development environment, and how to test the functionality of WAS.

The development environment used in Part I is WASDG. This environment is sufficient for enterprise development at colleges and small or large businesses that use primarily UNIX and Windows NT systems. Although the cost of the WASDG environment is minimal, it is solid, portable, and manageable because of the use the Korn shell, Perl, Makefile utilities, and other UNIX commands. If you decide to learn programming for WAS and to start directly with Part III, you do *not* need to learn about the complexity of setting the WASDG environment. A system administrator can setup the environment as described in Part I by altering the user login profile.

Of course, WAS is not a programming language; the phrase *WAS programming* refers to the many system commands, scripts, and programs and the configuration and programming of many other applications that are used in a WAS region. Only a subset of Java programming is required, and employing complex Java programs in WAS programming can drastically inhibit the performance of WAS. Chapter 5 introduces the WebSphere domain and defines its constituent elements. You will understand the benefits of a WebSphere domain and the WebSphere nodes that can participate in it.

Chapter 5 also takes a broader look at the systems and services involved in rendering a WebSphere domain usable; as a result, the WebSphere region is also defined. This chapter concludes with a comparison between WAS v3.5 and WAS v4.

Chapter 6 discusses WAS tools and sample Web applications that you will learn to deploy to test the functionality of WAS.

Part II

In Part II, you will learn the fundamentals about the processes and the many systemic and network resources used by WAS; you will learn the effective commands to administer WAS; and you will learn to extrapolate WAS configuration data with automation tools. Chapter 7 introduces the basic programs and scripts that automatically generate administrative reports about the WAS domain. The technique is called WAS *report extrapolation.* Such a technique is generic, and you can use it with WAS v3.5 and v4, in which the WSCP is distributed. You can also adapt it to WAS v5 by writing similar scripts to wrap the `wsadmin` command, which replaced the `wscp` command of prior versions.

Chapter 8 is a quick but efficient chapter filled with commands to administer and look closely at the processes of WAS. A distinction is drawn between AIX threads and Linux threads (which are really forked processes). The chapter also introduces the essential administrative commands to quickly troubleshoot WAS in a multiplatform environment.

Part III

Part III discusses programming J2EE web applications specifically targeted to be deployed and tested on WAS. The web application developed is called the WASDG application: a simple bank application in which an employee of a bank, known as a teller, acts on the bankers' or clients' accounts. Based on the WASDG application, you will learn how to quickly develop a J2EE application.

In Chapter 9, you will create the database for the WASDG application. The chapter also addresses Java Database Connectivity (JDBC) programming to manipulate the database and to get the database's metadata. Using the Perl Database independent (DBI) module shows the efficiency of simple scripts to populate the database and fetch data from it. You will use such scripts during the course of Part III to quickly rebuild the database and to monitor the subsequent inserts, updates, and deletions of its records. For instance, in Chapter 17 you will use qry_session, a Perl script that uses DBI, to get a quick snapshot of the SESSION database.

Chapter 10 progressively shows how to build a data access component (DAC) to transparently manipulate the records in a database. This data access component is called the DataAccessComponent class. You can set the JDBC driver type of this component through a properties file so that you will be able to practice with the JDBC drivers of type 2 and type 3, which are discussed in Chapter 2. The data access component has been carefully designed so that it can be turned into an Enterprise JavaBean in Chapter 18. This EJB forms the essential interface to the database and will be used in implementing the business logic by many other EJBs.

Chapter 11 discusses a development approach for web applications based on J2EE that apply specifically to WAS. The approach taken here is based on Perl and shell scripts to create and deploy a web application with a single command: `svlbuild`. As an example, you will use this method to build the WASDG application that is used throughout this text.

Chapter 12 uses the DataAccessComponent (of Chapter 11) within the J2EE web application (of Chapter 11) to write servlet programs that access the database. As servlets are loaded in the WAS web container, the servlets directly use the data access component module to fetch and store data in the database. Merging the data access component with the J2EE application is a simple operation: just add the com/tcnd/wasdg tree to the development tree, and the package is available to the WASDG application.

Chapter 14 gives a more detailed view about the elements describing a J2EE application, and the additional elements introduced by IBM as an extension to these descriptors. You will learn how to obtain these descriptors and how to (de)archive enterprise and web applications through the use of simple customized scripts. Such scripts allow you to effectively and quickly control J2EE application during the development process.

Once you have established the method to quickly write servlets and test them, you will be comfortable writing specific code to explore with WAS: JSP programming and JSP tag libraries programming are the subjects of Chapter 13, and WAS bootstrap classloader is discussed in Chapter 15.

In Chapter 15, the classes' loading order versus their visibility order is justified using a simulation process.

Chapters 16 and 17 provide a solid foundation in programming session management and configuring IBM session persistence; this is validated by using very short programs and testing techniques by tapping into the communication between WAS and the client browser. This is made possible by using Lynx and the scripts developed in Chapter 9.

In Chapter 18, the data access component of Chapter 10 is turned into an Enterprise JavaBean: DataAccessComponentBean. This EJB forms the essential interface to the database, and it will be used in implementing the business logic (separately from the servlets) through many other EJBs. Using the web application development tree from Chapter 16, you will merge to it the business logic development tree either by using an explicit copy of the EJB development tree or by establishing a symbolic link to the EJB development tree.

In Chapter 19 you will learn how to use Apache SOAP with WAS to turn the EJBs in Chapter 18 into Web services. The discussion is specific to SOAP programming for WAS. The chapter discusses the technique of parameter passing between the SOAP client program and the SOAP server. You will learn how to edit the SOAP deployment descriptor and how to programmatically use qualified parameters names (QNames), the SOAP mapping registry (SOAPMappingRegistry) and mapTypes() method, and the SOAP org.apache.soap.util.xml. Serializer and org.apache.soap.util.xml.Deserializer.

Java Authentication and Authorization Service (JAAS) programming is not addressed by WAS v4, but because it is part of Java 2 v1.3 and will be supported by WAS v5, JAAS is covered in Chapter 20. Applying JAAS programming to the servlets in the context of WAS v4 has proven that WAS technology is totally coupled to the Java Virtual Machine API. Chapter 20 shows how to add JAAS authentication to the same data access component developed in Chapter 10. The programming shows how only privileged servlets can access the database to modify the timestamp of a teller logout. Chapter 20 concludes by showing how to use Cipher in the context of a servlet writer to protect signed HTML pages such as a registration page.

Finally, Chapter 21 shows how to run multiple WAS instances on a single server and how to apply source control commands to the development tree that is shared by many developers. The method uses the make utility and is adequate for developing web applications in enterprises and colleges.

Whereas Chapter 5 defines the WebSphere *domain,* Chapter 21 proves the looseness of the nomination of the WebSphere domain. WAS v5 does not use the word “domain” in the context of WebSphere clustering. However, the same information is communicated to you when drawing the similarity between a WebSphere domain that spans many nodes (WAS v4) and a WebSphere cluster that spans many cells (WAS v5).

Part IV

Part IV consists of two chapters: Chapter 22 presents a stress tester for web applications, and Chapter 23 presents the programming of a logger and an exception handler.

Chapter 22 focuses on how to fork multiple processes, each of which generates a set of web hits. Such understanding is used in programming a stress tester, an application that is used to measure web application performance. In addition, this chapter shows the significance of using a network analyzer (similar to Sniffer) to analyze the performance of the web application.

Chapter 23 shows the programming of a log writer, a bundle manager, and an exception handler that can be added to any web application. This exception handler—called the WASDG exception handler: WasdgException—writes messages similar to the messages that WAS writes to its standard output.

Part V

In Part V, Chapters 24 through 26 form an essential basis from which to monitor WAS and its containers, web applications and the exceptions thrown by these applications, and the systems in a WebSphere region. Performance tuning is also briefly considered but supported with efficacious scripts to monitor network resources, system resources, and threads.

Chapter 24 introduces a new lightweight application to monitor WAS, called WASLED/WASMON. The application allows the WASMON console to monitor WAS' web container, EJB container, and run-time components. WASLED shows WAS run-time severity errors in green, orange, and red.

Chapter 25 offers a few scripts to monitor the system and network resources during WAS run time. In particular, because WAS is a thread-intensive application, you will learn how to gather threading information on AIX and Linux systems. Two exclusive scripts called MrThread and MrTop are explained and used. MrThread runs on UNIX systems where the `pstat` command is available and generates an output similar to the `pstat` command found on AIX. When MrThread runs on AIX, it shows the creation of new threads and how they are allocated to a specific CPU on Symmetric Multiprocessing (SMP) systems. Any change in a thread is also detected between different instances. MrTop monitors threads on a Linux system. (The distinction between AIX and Linux threads is outlined in Chapter 8.) Chapter 25 also discusses EJB caching and offers a method for parametric tuning of the database used in session persistence.

Chapter 26 demonstrates how to use WASMON to switch into supervisor mode so that a roaming operator can supervise and remotely control (via e-mail) the systems in a WebSphere region. WASMON can also supervise web applications that use the WASDG exception handler implemented in Chapter 23.

Appendixes

There are six appendixes that complement the material in the chapters. These appendixes elaborate on material and the usage of tools and commands that might have been a disruption to the reader if included in the chapters.

Appendix A lets you know where to find and download the distribution code and documentation for the applications and programs in this book. Assuming the reader has an internet connection, he will be able to get up-to-date distribution code and last-minute information on errata whenever it is made available on the Web.

Appendix B shows you how to back up and restore WAS. Back up and restore rely on the use of UNIX archiving commands and WAS' XMLConfig.

In Appendix C, you will get your hands on a powerful tool, MrUnicode, and you will learn how to generate servlets that maps the Unicode as provided by the Unicode Consortium. This appendix replaces the discussion of localization and internationalization in a separate chapter and provides a clear understanding of the differences between Sun Microsystems and IBM in their supports to character sets.

Appendix D complements Chapters 3 and 4 to show you how to print the environment variables as set by WASDG environment. In addition, it presents the syntax for a few of the tools that have been used in the chapters.

Appendix E complements Chapter 24 and 26 and provides a guideline on WAS monitoring using WASLED/WASMON.

Appendix F details the differences between WAS v4 and WAS v5 in their support to the Java APIs and shows you how to change the scripts explained in this book so that they run on WAS v5.

Special Consideration of WAS v5

The programming approach used in this text is also applicable to WAS v5. Appendix F outlines the differences between WAS v4 and WAS v5 in their level of support for the Java 2 API, JSP, and EJB.

Although this book specifically uses WAS AEs v4 in the programming chapters, WAS v5 has been given special consideration:

- As programming is explained, the text takes WAS v5 into account so that deprecated methods and classes are not used, and deployment specific to a particular J2EE component version or level is contrasted.
- To eliminate confusion, this book does not discuss communication between servlets that live in different web containers because this subject is obsolete.
- This book does not discuss subjects that might be irrelevant in a future release of WAS. For example, IBM supports servlet chaining in WAS v3.5 and v4 (many servlets being dispatched to generate a final output to be shipped to a browser—known since JDK v1); however, this topic is not discussed because Java 2 v1.3 supports a more efficient approach using <filter>. Although WAS v4 uses IBM Java 2 v1.3, <filter> is not supported. Servlet filtering is now possible with WAS v5, which uses IBM Java 2 v1.3.1.
- JAAS programming is not addressed by WAS v4 but because JAAS is part of Java 2 v1.3, it is possible to add JAAS APIs to the WAS v4 startup classloader (the subject of Chapter 15). Although JAAS is used in light of Java 2 v1.3, the text mentions JAAS support with Java 2 v1.4.[1]
- Before exploring WAS v5, you must understand the definition of a WebSphere domain. Although the word "domain" is no longer used in WAS v5, the explanation provided specifically in Chapters 2, 5, 8, and 21 is the main information you need to understand.

[1] WAS v4 uses IBM Java 2 v1.3, which is limited and does not include the JAAS API. WAS v5 uses IBM Java 2 v1.3.1, which includes the JAAS API. However, none of the WAS versions yet support Java 2 v1.4.

The method discussed in Chapter 7 to extrapolate WAS reports using Perl and WSCP can also be modified and applied in light of Perl and wsadmin. In WAS v5, wsadmin replaced WSCP, but the JACL API is still the means of programming for both wsadmin and WSCP.

Utility Scripts

This book includes sufficient tools and utilities that help you absorb the material efficiently. You can download the Gramercy Toolkit and WASLED/WASMON[2] as explained in Appendix A, so that the following utilities are available on your system:

- Utilities to stress-test web applications
- Utilities to monitor WAS
- Utilities to monitor WAS threads consumption
- Miscellaneous utilities to complement WAS programming environment

Stress Testing

The time it takes for a number of clients to access the web application(s) concurrently—with each client placing a number of hits—represents a realistic measurement for the performance of WAS. Chapter 22 shows how to use the SharkUrl tool to simulate multiple client browsers by forking multiple processes and to place a sequence of hits per each client.

Monitoring WAS

This book considers WAS monitoring by using WASLED/WASMON, a monitoring application developed at Total Computing & Network Design, Inc. WASMON provides WAS monitoring for the standard output and standard error of the application server. It also provides a smart supervisor to monitor a WebSphere region and engage in a secure e-mail conversation with a roaming operator.

The application allows you to monitor the WAS components and its containers' activities. WASMON is configurable to filter specific events from WAS' log file; consequently, it triggers corrective administrative scripts, sends e-mail as alerts, and enters into a remote conversation with an operator. Chapters 24 and 26 explain how to use WASMON for risk management of WAS.

Monitoring WAS Threads

Because WAS is a thread-intensive application, MrThread is presented as the unique tool to monitor the kernel thread table and the variation in the activity of a specific thread during WAS run time.

[2] Refer to Appendix A on downloading the Grammercy Toolkit and WASLED/WASMON. You must read the license agreement before using the toolkit or the monitoring application.

Miscellaneous Utilities

In addition, the text introduces many utility scripts that facilitate the generation of J2EE applications and their deployment.

Printing Unicode, Localization, and Internationalization

The text does not include a separate chapter on localization and internationalization, but it focuses on the servlet writer and on its printing for the Unicode, as provided by the Unicode Consortium. Appendix C shows how to use MrUnicode to generate servlets and search converters and charsets.[3] for specific encoding that is supported in Java 2. MrUnicode clarifies the differences in nomenclature between IBM and Sun Microsystems and their support for converters in their many releases of Java 2 versions.

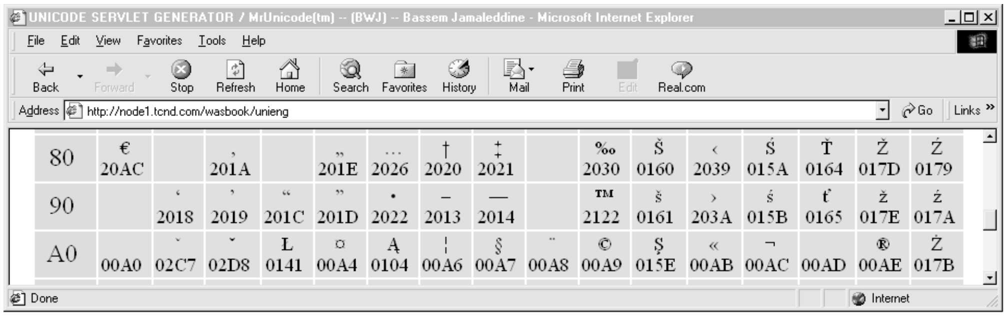
For example, to reveal these character sets for which a Unicode mapping for the Euro sign has been defined, use this command:

```
# MrUnicode -info -desc euro
```

or to view its representation as it is printed in a browser, use MrUnicode to generate the servlet whose output can be viewed in a browser:

```
# MrUnicode -t cp1250.txt -s Uni_Eng.java -c windows-1250
```

By requesting the Uni_Eng servlet, you can view the Unicode mapping for the Euro sign of the Unicode mapping found in the cp1250 converter.


UNICODE SERVLET GENERATOR / MrUnicode(tm) -- (BWJ) -- Bassem Jamaleddine - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Real.com
Address http://node1.tcnd.com/wasbook/unieng
Go Links
Done
Internet

80	€ 20AC		‚ 201A		„ 201E	… 2026	† 2020	‡ 2021		‰ 2030	Š 0160	‹ 2039	Ś 015A	Ť 0164	Ž 017D	Ź 0179
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014		™ 2122	š 0161	› 203A	ś 015B	ť 0165	ž 017E	ź 017A
A0	00A0	ˇ 02C7	˘ 02D8	Ł 0141	¤ 00A4	Ą 0104	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	Ş 015E	« 00AB	¬ 00AC	00AD	® 00AE	Ż 017B

UNIX Commands, Shell, Perl, and Lynx

This book uses UNIX commands and shell scripts, Perl scripts, and the Lynx textual browser. Whereas the Perl interpreter and Lynx are available for many UNIX operating systems and Windows NT, the UNIX commands and the UNIX shell interpreter are available on all UNIX systems and only on Windows NT where the MKS Toolkit has been installed.

[3] Appendix C defines converters, charsets, and canonical names (used by Sun Microsystems).

The MKS Toolkit is available for DOS and Windows with tools that give you the ability to execute UNIX commands and shell scripts from the command prompt. Yet, it is always less expensive and more resourceful to use Linux than to use Windows NT and the MKS Toolkit.

By convention we will use the hash (#) to represent the command prompt, and a computer command will be printed following the hash. We will also use a few verbs that are derived from UNIX commands: to *head* a file and to *tail* a file refer to the execution of the `head` command and the `tail` command against a file, respectively. In addition, we will often use environment variables (as set by the WASDG environment) to refer to specific files; for instance, the $WASCFG refers to WAS server configuration file, and `$STOPWAS`; `$STARTWAS`, when typed on the command prompt, will restart WAS.

WAS Programming on the UNIX Platform

Due to its native scripting ability, UNIX offers the only development platform for enterprise Java applications. Many of the UNIX commands can simplify the build process of a web application. For instance, Chapter 18 shows how one programmer can develop, compile, and test the business logic as a set of Enterprise JavaBeans in a separate directory (such as /BOOK/18/DevSess), and another programmer can use these EJBs in his or her own web application development tree (such as /BOOK/18/Code) simply by establishing a symbolic link to the development tree of the first programmer.

If you are using Linux or AIX, you can bring together all the JVM processes started by WAS with a simple command:

```
# ps  -A -o "%p  %a" | grep java | awk '/AppServer/ {system("kill -9 “$1)}'
```

To compile all source .java files in the development tree of a Java package, consider the following three lines of script to be executed in the parent directory of the Java package:

```
#!/bin/sh
find . -follow -name "*.java" > javafiles.list
cat javafiles.list | xargs javac -deprecation
```

You will learn more about such handy commands and scripts throughout this book.

Using Lynx

This book often uses Lynx as the browsing agent for two reasons: first, so that you can clearly see what is being sent and received in the HTML headers; second, to break the dependency on a graphical interface and allow the programmer to issue instantaneous bulk requests to the web application. Using textual commands is a valid strategy that enhances knowledge and quick programming and testing, which we will follow in this book.

For instance, to provide a clear explanation of the creation and deletion of session persistence data, Chapter 17 uses Lynx followed by a subsequent query to the SESSIONS table to prove that the record is in accordance with what is returned in the HTTP header of a Lynx command. As a practical example, Listing 1-1 shows the response of the servlet TellerLogged when logging in teller1. The header shows a session id that is similar to the one written to the SESSIONS table when IBM session persistence is enabled.

Listing 1-1 *Issuing a Lynx command to log in teller1*

```
# echo "userid=teller1&password=secret1&---" | lynx
      -accept_all_cookies -post_data -mime_header
      http://node2.tcnd.com/wasbook/tellerlogged | grep Set-Cookie
Set-Cookie: JSESSIONID=0001BL13IMQRF5R2LBOOPTJ1M1Q:-1;Path=/
```

Listing 1-2 is the output of the Perl/DBI script qry_session that reveals the content of the SESSIONS table.

Listing 1-2 *Output result when issuing qry_session to query the SESSIONS table*

```
Connected.
Row 0 ------
...
Row 1 ------
0 ID -- BL13IMQRF5R2LBOOPTJ1M1Q
1 PROPID -- BL13IMQRF5R2LBOOPTJ1M1Q
2 APPNAME -- default_host/wasbook
3 LISTENERCNT -- 0
4 LASTACCESS -- 1019550936764
5 CREATIONTIME -- 1019550936764
6 MAXINACTIVETIME -- 1800
7 USERNAME -- anonymous
8 SMALL -- (object as stream of bytes)
9 MEDIUM --
10 LARGE --
```

Perl Scripts in Enterprise WAS Programming

Perl is an essential interpreter that is included as a standard interpreter on UNIX systems like AIX and Linux. Because a Perl interpreter exists for almost every platform, the interpretation of Perl scripts is platform-independent. As a result, Perl scripts written on UNIX can also be interpreted by Perl on Windows NT or IBM OS/390.

Perl is an extremely simple and powerful interpreter. Its use in this book is justified for the following reasons:

- To offer a limited introduction on using Perl/CGI scripts that is essential to the understanding of session management. Perl/CGI is briefly used in Chapters 12 and 16.
- To quickly format and print database records, like the example shown in the previous section.
- To extrapolate WAS reports. Chapter 7 uses the Perl interpreter in conjunction with WSCP to carry out such an extrapolation.

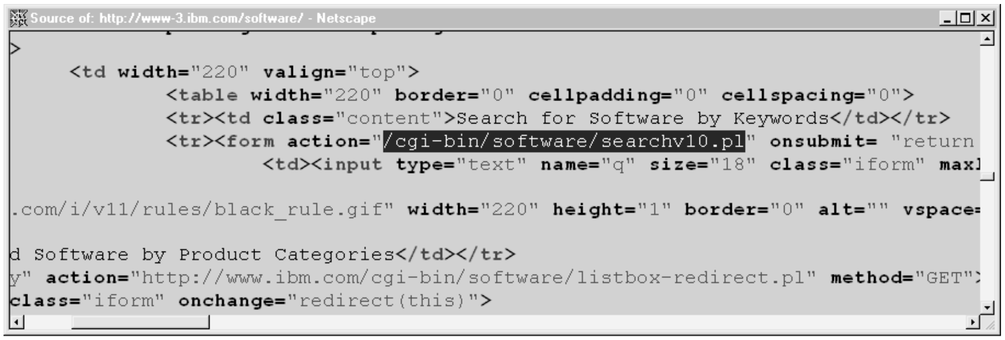
- To automate text processing and to facilitate the build process of an enterprise application. Specifying web application properties and descriptive values using graphical applications is tedious and time consuming.
- To clarify the meaning of a language processor such as JSP. In this context, the Perl interpreter is used for pedagogical reasons to show you the connection between text preprocessing and the language processors used in conjunction with application server programming such as JSP. Chapters 16 and 17 present the SessionFairy servlet that automatically introspects sessions and generates a JSP file. Chapter 22 shows how to preprocess anonymous Perl subroutines to stress-test a web application.

Finally, besides the traditional thrust of WebSphere marketing concerned with eliminating bad CGI to replace it with good Java-based servlets and JSPs, Perl/CGI is apparently still in existence. Some major web sites still favor the power of Perl to parse regular expressions.

Even on the WebSphere web site, http://www-4.ibm.com/software, you'll find a few important references to Perl scripts. On a Linux system, a quick dump exercised on the page seen in Figure 1-2 can be realized using this simple command:

```
# GET -e 'www-4.ibm.com/software' | grep "\.pl"
```

The content of the page can also be revealed using the browser's view source capability, as shown in the following illustration.



```
>
    <td width="220" valign="top">
           <table width="220" border="0" cellpadding="0" cellspacing="0">
           <tr><td class="content">Search for Software by Keywords</td></tr>
           <tr><form action="/cgi-bin/software/searchv10.pl" onsubmit= "return
                  <td><input type="text" name="q" size="18" class="iform" maxl

.com/i/v11/rules/black_rule.gif" width="220" height="1" border="0" alt="" vspace=

d Software by Product Categories</td></tr>
y" action="http://www.ibm.com/cgi-bin/software/listbox-redirect.pl" method="GET">
class="iform" onchange="redirect(this)">
```

Applications Used in This Book

Besides the WebSphere Application Server software product, this book also uses the following software:

- IBM Universal Database (UDB).
- Standard UNIX interpreters and commands, such as `Perl`, `awk`, `sed`, and `ksh`.
- Perl, Perl/CGI, and Perl modules.
- MKS ToolKit from Mortice Kern Systems, Inc. When this toolkit is installed on Windows NT workstations, UNIX commands and shell scripts can be executed;

consequently, such a workstation can then be merged to a heterogeneous distributed environment. We will use Samba to mount the NFS-exported directory.

- Exceed for Windows NT. Install Exceed if you would like to export your X Window display to a Windows NT workstation.
- Internet browsers: Lynx, Netscape, and Microsoft Internet Explorer.
- IBM Dataglance, the network analyzer that is similar to Sniffer and that runs on IBM OS/2. Used in Chapter 22.
- The AIX performance-monitoring agent package. Used in Chapter 25.
- WASLED and WASMON from Total Computing & Network Design, Inc. A monitoring and supervising application that can monitor WAS v4 and v5. Used in Chapters 24 and 26.

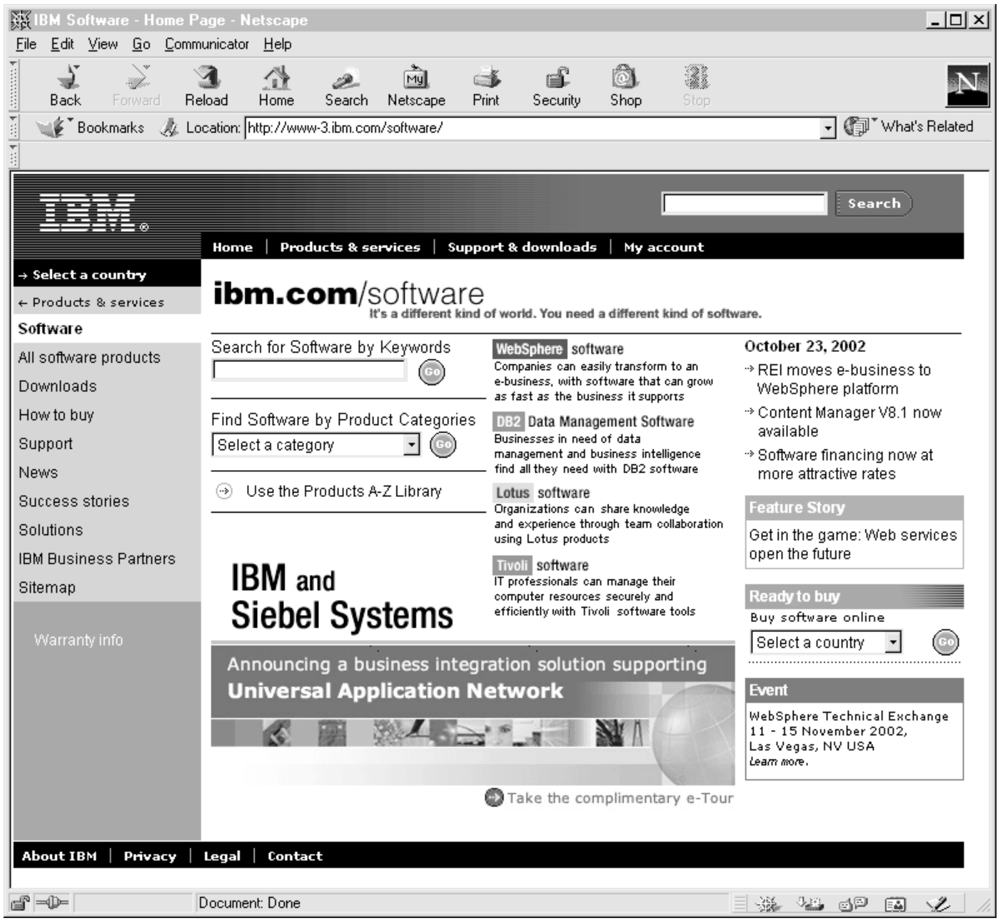

IBM Software - Home Page - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://www-3.ibm.com/software/ What's Related
IBM
Search
Home | Products & services | Support & downloads | My account
→ Select a country
← Products & services
Software
All software products
Downloads
How to buy
Support
News
Success stories
Solutions
IBM Business Partners
Sitemap
Warranty info
ibm.com/software
It's a different kind of world. You need a different kind of software.
Search for Software by Keywords
Go
Find Software by Product Categories
Select a category
Go
Use the Products A-Z Library
IBM and Siebel Systems
Announcing a business integration solution supporting
Universal Application Network
Take the complimentary e-Tour
WebSphere software
Companies can easily transform to an e-business, with software that can grow as fast as the business it supports
DB2 Data Management Software
Businesses in need of data management and business intelligence find all they need with DB2 software
Lotus software
Organizations can share knowledge and experience through team collaboration using Lotus products
Tivoli software
IT professionals can manage their computer resources securely and efficiently with Tivoli software tools
October 23, 2002
→ REI moves e-business to WebSphere platform
→ Content Manager V8.1 now available
→ Software financing now at more attractive rates
Feature Story
Get in the game: Web services open the future
Ready to buy
Buy software online
Select a country
Go
Event
WebSphere Technical Exchange
11 - 15 November 2002,
Las Vegas, NV USA
Learn more.
About IBM | Privacy | Legal | Contact
Document: Done

Figure 1-2 *IBM corporate web site*

- The Gramercy Toolkit from Total Computing & Network Design, Inc. This toolkit provides many of the utility scripts and necessary libraries to run system management commands used in this book.
- A text editor such as emacs or vi, which are freely distributed with any version of Linux.

In addition to these products, some add-on components to WAS v3.5 (that are now included with v4.0.3 and v5) are used:

- WebSphere Resource Analyzer
- WebSphere Log Analyzer

This text does not use WebSphere Studio Application Developer (WSAD) or any of the Visual Age products that are available from IBM to provide a front end for developing code for WebSphere Application Server. Our enterprise development environment will be realized with a system startup profile that is derived from and parallel to WAS configuration. It is a development environment, where, upon user login, the environment is set and you are ready to start programming.

Using Visual Age products does not provide a cost-effective development environment administratively, financially, or educationally. In fact, using such products diverts you from the specific applications of WebSphere Application Server and makes the material seem to be more complicated than it should be.

The focus is on the application server processes, and anyone can get started by supplementing the material in this book with a $600 PC to run Linux and an Internet connection to download the code.

We will tackle the installation, usage, programming, and tuning of WAS like any other UNIX-distributed application, by using simple commands and an integration strategy that is nothing but a classical approach in any enterprise computing environment. It is classical in the sense that it s not specific to a particular version of WAS or any of its components, but it can be adapted to future versions of the product.