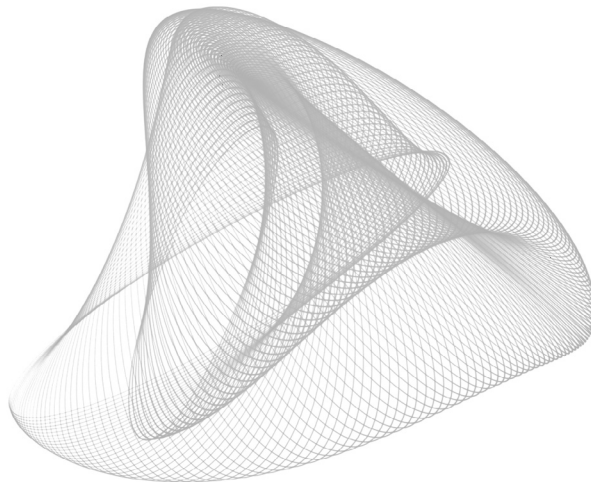

IBM® WebSphere® Application Server Programming



About the Author

Bassem (Max) Jamaledine is a Web Systems Engineer with five years of diversified contractual experience with IBM. He administered and orchestrated the development of several main projects at IBM's T.J.Watson Research Center, from the Java-based network computer to the new generation of IBM's WAS technology.

A strong mathematical background has helped foster his belief that the soul of understanding new computer technology lies in disciplined coding coupled with proper system administration.

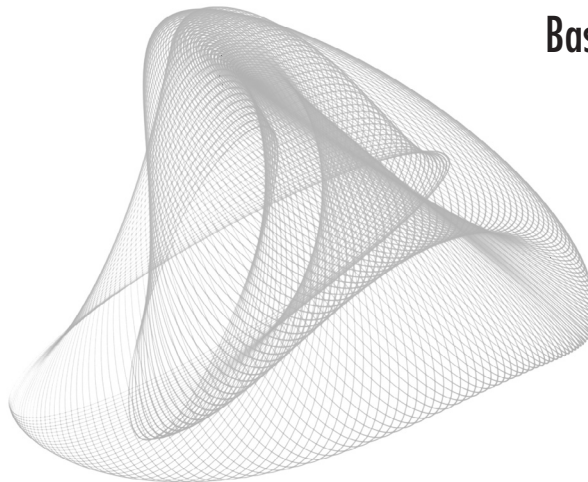
His UNIX knowledge dates from the early stages of PDP-11 system programming to the IBM SuperPower Parallel machines geared by newer AIX technologies. He has lectured at the City University of New York on advanced system programming, and on compiler and assembler construction. During the dot.com explosion, he was a major Object Oriented developer for Wall Street e-businesses, building user registration, authentication, shopping carts, and tracking and debugging routines. His forte is the development of algorithms for pattern recognition and calligraphy, such as the grammar to recognize chromosomes and digital signatures.

He earned an M.S. in Computer Science from the City University of New York, a B.S. in Computer Mathematics from Lebanese American University, and a mathematics diploma from International College in Lebanon.

He started development of Project-9 at Total Computing & Network Design, Inc. He is actively consulting on the East Coast for WebSphere deployment.

IBM® WebSphere® Application Server Programming

Bassem W. Jamaledine



McGraw-Hill/Osborne

New York Chicago San Francisco
Lisbon London Madrid Mexico City Milan
New Delhi San Juan Seoul Singapore Sydney Toronto

The **McGraw-Hill** Companies

McGraw-Hill/Osborne
2600 Tenth Street
Berkeley, California 94710
U.S.A.

To arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact **McGraw-Hill/Osborne** at the above address. For information on translations or book distributors outside the U.S.A., please see the International Contact Information page immediately following the index of this book.

IBM® WebSphere® Application Server Programming

Copyright © 2003 by The McGraw-Hill Companies. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

WASLED/WASMON is Copyright © 2002 by Total Computing & Network Design, Inc. Gramercy Toolkit is Copyright © 2002 by Total Computing & Network Design, Inc. The following terms are trademarks of Total Computing & Network Design, Inc. (TCND): WASLED/WASMON™, WALED™, WASMON™, MrUnicode™, Mister Unicode™, MrThread™, Mister Thread™, MrTop™, Mister Top™, SharkUrl™, WASETTE™.

1234567890 CUS CUS 0198765432

ISBN 0-07-222459-2

Publisher	Brandon A. Nordin
Vice President & Associate Publisher	Scott Rogers
Editorial Director	Wendy Rinaldi
Project Editor	Elizabeth Seymour
Technical Editor	Chris Moss
Copy Editor	Chrisa Hotchkiss
Proofreader	Paul Medoff
Indexer	Valerie Perry
Computer Designers	Carie Abrew, Lucie Ericksen, Apollo Publishing Services
Illustrators	Michael Mueller, Lyssa Wald
Series Designer	Roberta Steele
Cover Series Designer	Greg Scott
Cover Illustration	Akira Inoue/Photonica

This book was composed with Corel VENTURA™ Publisher.

Information has been obtained by **McGraw-Hill/Osborne** from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, **McGraw-Hill/Osborne**, or others, **McGraw-Hill/Osborne** does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from the use of such information.

This book is dedicated in loving memory to my sister,
Dima Jamaledine

Contents at a Glance

Part I	Getting Started	
Chapter 1	Introduction to IBM WAS Programming	3
Chapter 2	Installing the WAS Repository	19
Chapter 3	Setting Development Environment Prerequisites	37
Chapter 4	Installing WAS on Linux, Windows NT, and AIX	77
Chapter 5	Defining a WebSphere Domain	113
Chapter 6	Testing Your Installation: WAS Tools and Examples	135
Part II	The Essential Administrative Guide for WAS Developers	
Chapter 7	WAS Report Extrapolation with Perl/WSCP	167
Chapter 8	A Quick and Essential Guide to Administering WAS	187
Part III	Programming for WAS	
Chapter 9	Preparing the Database	223
Chapter 10	Accessing the Database in Java: DataAccessComponent	243
Chapter 11	Developing a J2EE Web Application in WAS	261
Chapter 12	HTTP Servlet Programming	283
Chapter 13	Java Server Pages (JSP)	319
Chapter 14	The J2EE Web Application in WAS: A Detailed View	339
Chapter 15	Classes in WAS: Loading Order versus Visibility Order	357

viii IBM WebSphere Application Server Programming

Chapter 16	Session Identification and the HTTP Protocol	383
Chapter 17	Session Scope and IBM Session Persistence	423
Chapter 18	Enterprise JavaBeans Programming	455
Chapter 19	Apache SOAP Programming in WAS	499
Chapter 20	Fundamental Security Programming: Applying JAAS	523
Chapter 21	Enterprise Application Development	563
Part IV	Stress-Testing, Tracing, and Debugging	
Chapter 22	Stress-Testing	593
Chapter 23	Writing an Exception Handler, Logging, and Debugging	619
Part V	Monitoring, Tuning, and Risk Management	
Chapter 24	WAS Monitoring with WASLED™ and WASMON™	645
Chapter 25	Monitoring and Tuning the System Resources	665
Chapter 26	Risk Management with WASMON	693
Part VI	Appendixes	
Appendix A	Retrieving Information and Code Distribution	725
Appendix B	Backing Up and Restoring	731
Appendix C	Demystifying Java 2's Internationalization with MrUnicode	737
Appendix D	Gramercy Toolkit Scripts and the WASDG Environment	747
Appendix E	WASLED/WASMON Quick Reference	753
Appendix F	Support for WAS v5	763
	Index	769

Contents

	<i>Acknowledgments</i>	xxiii
	<i>Foreword</i>	xxv
Part I	Getting Started	
Chapter 1	Introduction to IBM WAS Programming	3
	The Simplest WAS View	4
	IBM's Offering for the WebSphere Application Server	5
	WebSphere Application Server v4	5
	WebSphere Application Server v5	6
	What's In the Five Parts of the Book	6
	Part I	7
	Part II	7
	Part III	8
	Part IV	9
	Part V	10
	Appendixes	10
	Special Consideration of WAS v5	11
	Utility Scripts	12
	Stress Testing	12
	Printing Unicode, Localization, and Internationalization	13
	UNIX Commands, Shell, Perl, and Lynx	13
	WAS Programming on the UNIX Platform	14
	Using Lynx	14
	Perl Scripts in Enterprise WAS Programming	15
	Applications Used in This Book	16
Chapter 2	Installing the WAS Repository	19
	Rationales to Install a Database	20

x IBM WebSphere Application Server Programming

	A Database Used as an Administrative Repository	20
	A Database Used as a Data Source	21
	WAS Compatibility with UDB	21
	Binary Distribution and Basic Installation of UDB	22
	Installing the Components of the Universal Database (UDB)	24
	UDB Installation on AIX	29
	Classifying the Universal Database JDBC Driver Types	33
	JDBC Type 2: The app Driver	33
	JDBC Type 3: The net Driver	34
	Wrapping Up	34
Chapter 3	Setting Development Environment Prerequisites	37
	The HTTP Server	38
	Installing the HTTP Server	38
	Starting/Stopping the HTTP Server	42
	HTTP Server Log Files	42
	The Java Machine	44
	Installing the Java Machine	45
	Setting Up the Java Computing System Environment	46
	The Windows NT User Environment	62
	A Refresher on Windows NT's Batch Commands	62
	Setting the User Environment on Windows NT	63
	The Shell Interpreter on Windows NT	64
	Working with the Java Machine	65
	Hello World	66
	Working with Java Packages: A Simple View	66
	HelloURL Example	69
	The CGI Version of Hello World: HelloWorld.cgi	72
	Testing Remote Database Connection with the JDBC Driver	73
	Wrapping Up	75
Chapter 4	Installing WAS on Linux, Windows NT, and AIX	77
	Conventions Used in This Book	78
	Home Directory Conventions	78
	Conventions of WAS Processes	80
	Conventions of the WebSphere Domain	80
	Products Dependency	80

Requirements	81
Hardware Requirements	81
Operating System–Level Requirements	82
GUI Interface Dependency	82
Database Requirements	84
Preinstallation Verification	85
Successful Database Connectivity	86
Availability of an HTTP Web Server to be Plugged Into the WAS Installation	86
Java Machine Availability	87
Installing WAS with a UDB Configuration Repository	88
Suppressing the prereq_checker	89
Installing WAS v4 Advanced Edition Single Server	90
WAS Startup and First-Time Configuration	92
Configuring the Common Resources	93
Setting the JDBC Driver	94
Virtual Hosting	96
Virtual Hosts Are Not Node Specific	96
Reasons for Setting the Virtual Hosts	97
Virtual Hosts Name Resolvability	102
Serving the Test Servlet: snoop	102
File Locations	104
Setting the WAS Development Environment	105
Setting the Commands	105
Setting the Desktop	108
An Undocumented Shortcut to Test WAS Installation	109
Uninstalling the Product	111
Replicating an Installation on Several Machines	112
Wrapping Up	112
Chapter 5	
Defining a WebSphere Domain	113
Understanding the WebSphere Region	114
Understanding the WebSphere Domain	116
Background About the WebSphere Domain	116
Benefits of a WebSphere Domain	118
Defining the Name for the WebSphere Domain	121
Administering a WebSphere Domain	123

xii IBM WebSphere Application Server Programming

	The Structure of a WebSphere Domain	126
	Ports Used by WAS	130
	Comparing WAS v3.5 and v4	131
	Suppressing the Servlet Redirector	131
	Suppressing Augmented datasources.xml Configuration	132
	JSP Levels 1.0 and 1.1 Are the JSPs Supported in WAS v4	132
	EJB Level 1.1 Is the Only Level Supported in v4	132
	Models and Clones Are Only for Application Servers in WAS v4	133
	Wrapping Up	134
Chapter 6	Testing Your Installation: WAS Tools and Examples	135
	WAS Systemwide Tools	136
	The Application Server Start/Stop Commands	136
	The WebSphere Administrative Console	137
	The WebSphere Control Program: WSCP	137
	DrAdmin: Generating Thread Dumps	138
	Enterprise Application Archive (EAR)	139
	WAS Testing Tools	144
	Sectioning an EAR Sample File: sampleApp.ear	146
	The Petstore Sample	150
	Installing the Petstore Sample	150
	Making the Changes Manually	158
	Miscellaneous WAS Tools	160
	The Log Analyzer	160
	The Resource Analyzer	161
	Wrapping Up	164
Part II	The Essential Administrative Guide for WAS Developers	
Chapter 7	WAS Report Extrapolation with Perl/WSCP	167
	Introduction to WSCP	168
	The wscp Command	169
	Starting WSCP	169
	The WSCP Property File	171
	A Conversation with WSCP	172
	WSCP and Tcl	173
	Sole Installation of WSCP on a Workstation	175

	The WscpCommand Interface	176
	Practical Reports Extraction with Perl/WSCP	177
	Running WSCP from Perl	177
	Parsing the WSCP Output	178
	WSCP Message Codes	185
	Wrapping Up	186
Chapter 8	A Quick and Essential Guide to Administering WAS	187
	Starting, Stopping, and Testing WAS	188
	Starting WAS	188
	Stopping WAS	189
	Stopping/Starting/Testing the HTTP Server	190
	Testing WAS	190
	Checking for WAS Version and Release Level/Date	194
	Keeping Track of the Version of Your WAS Java Machine	196
	Log Files and Startup Messages	196
	If WAS Fails to Start	196
	Formatting of the Log File	197
	The activity.log File	199
	Be Cautious When Editing Properties Files	200
	Working with WAS Processes	203
	Demystifying WAS Processes	203
	Acting on a Group of Processes	205
	The /etc/services Ports	205
	The Default Ports Used by WAS	206
	Debugging Default Ports	207
	Monitoring the Ports	207
	The Administrative Configuration Files	210
	The WAS Startup Java Machine	211
	Java Machine Parametric Tune-Up	212
	Disabling the Just-In-Time Compiler	213
	Select a Heap Size That Fits Within the Physical Memory Constraint	214
	Managing the WAS Repository	214
	Common Administrative Practices	216
	A Quick WAS Diagnostic: Invoking showCfg	216
	Checking on the WAS Database Repository	217

xiv IBM WebSphere Application Server Programming

Monitoring the Application Archives	217
Archiving the WAS Directory	218
Archiving the WAS directory on Windows NT	219
Wrapping Up	220

Part III Programming for WAS

Chapter 9	Preparing the Database	223
	Introduction	224
	UDB Authority on UNIX	224
	Creating the Database	225
	Populating the Database	228
	Java and JDBC	228
	Using the DBD::DB2 Module	233
	Wrapping Up	240
Chapter 10	Accessing the Database in Java: DataAccessComponent	243
	Defining the DataAccessComponent	244
	Examining Characteristics of the DataAccessComponent	249
	Text Substitution in the DataAccessComponent	250
	Data Retrieval Using the DataSet	251
	Retrieving Data for the EmpAccnt and CliAccnt	252
	Understanding the Build Process	258
	Creating the Java Package Tree	259
	Mediocre Compilation of the Java Package	259
	Compiling .java Programs Collectively: The jall Script	260
	Wrapping Up	260
Chapter 11	Developing a J2EE Web Application in WAS	261
	Loading a J2EE Web Application: A Simplified View	262
	Creating the First J2EE Tree	262
	Building the Web Application	265
	Creating a More Complete J2EE Tree	274
	Adding a New Build to an Already Loaded Web Application	275
	Understanding the j2tree Script	277
	Wrapping Up	281

Chapter 12	HTTP Servlet Programming	283
	From CGI Programming to Java Servlets	284
	Using the DumpEnv Servlet to Dump CGI Environment Variables	286
	Using the DumpEnv Servlet: The service() Method	288
	Developer's Tactics While Programming Servlets	288
	Understanding the HTTPD Log Record	289
	WAS Redirection to Standard Input/Output	289
	Using Lynx to Check on HTTP Requests	290
	Recompiling Servlets Using the svlbuild Script	290
	Turning HTML Content into a Java String	291
	Understanding the Need for Servlet Programming	292
	Using a Servlet to Print HTML Content	292
	Using a Servlet to Access the UDB	293
	Using a Servlet to Print Contents of a File	299
	Using Servlets for the Teller/Client Example	305
	The Teller Login Servlet: LoginScreen	305
	Teller Authentication: TellerLogged	308
	Teller to Credit/Debit a Client Account: CreditDebit	311
	Batch Processing with Lynx	317
	Wrapping Up	318
Chapter 13	Java Server Pages (JSP)	319
	JSP Programming: A First Example	321
	A Second Example: Using the Request Dispatcher	324
	Servlet Communication	328
	Intrасervlet Communication	329
	Interservlet Communication	329
	Custom Tags: JSP Tag Libraries	329
	Automating the Process of JSP Tags Library Programming	332
	Registering a Tags Library in the Web Application Descriptor	335
	Summarizing the Steps to Program with a JSP Tags Library	335
	Customizing a Tags Library to Print Typical Messages	336
	Wrapping Up	338
Chapter 14	The J2EE Web Application in WAS: A Detailed View	339
	IBM-Specific Deployment Descriptors: A Detailed View	340
	IBM Extensions	340

xvi IBM WebSphere Application Server Programming

	Getting the IBM Extensions	343
	Using the EARExpander	350
	Summary of IBM Extensions Attributes	350
	WAS Parsing for the web.xml	351
	Wrapping Up	355
Chapter 15	Classes in WAS: Loading Order versus Visibility Order	357
	Classpaths and WAS' Classloaders	358
	The Classpaths Tree Hierarchy	359
	The WAS Startup Classpath	362
	Adding Jar Archives to the WAS Startup Classloader	364
	Getting the Application Classpath from within a Servlet	364
	Initializing and Reloading a Servlet	365
	Module Visibility and Class Reloading	367
	Module Visibility	367
	Module Visibility and Reloading Simulation: Hey—Yo	370
	Hot Deployment and Dynamic Reloading	378
	Wrapping Up	382
Chapter 16	Session Identification and the HTTP Protocol	383
	Identifying User Logins	385
	Getting the mime_header	385
	Cookies	386
	Cookies with CGI	387
	Using Cookies Through HTTP Java Servlets	391
	Limitations and Naming Conventions of Cookies	399
	Session Management	399
	Identifying a Session with the JSESSIONID Cookie	400
	WAS Session Identifier Visibility to CGI Programs	408
	SessionFairy	409
	Gathering Information from the Session Identifier	410
	Generating the SessionFairy.jsp	412
	URL Rewriting and Hidden Parameters	416
	URL Rewriting	417
	Form Hidden Fields	419
	Wrapping Up	421

Chapter 17	Session Scope and IBM Session Persistence	423
	Examining Session Scope and Affinity	424
	Understanding Session Scope	424
	Understanding Session Affinity	426
	Examining Session Persistence: Setup, Configuration, and Testing	427
	Setting Up and Configuring a Session	427
	Using WAS Components in Persisting a Session	433
	Testing Session Persistence	433
	Programming Considerations	436
	Serializing Persistent HTTP Session Objects	437
	Invalidating the Session on Logoff	437
	Committing Changes in a Persistent Session	438
	Understanding Session Impact When Reloading a Servlet or a .jsp	439
	Writing to the Persistent Database: sync()	439
	Using a Teller Login Scenario to Test Session Persistence	439
	Examining a Teller Login Scenario Using the Browser	440
	Examining a Teller Login Scenario Using Lynx	444
	Using Multiple Sessions to One Web Container	447
	Tuning the Session Manager	448
	Considering Page Sizes Larger Than 4K	450
	Understanding the Reasoning Behind Persistence	451
	Wrapping Up	452
Chapter 18	Enterprise JavaBeans Programming	455
	Enterprise JavaBeans for the WASDG Application	457
	Turning the Programs into Session Beans	466
	The Data Access Component as a Session Bean: DataAccessComponentBean	470
	Extending the BeanBase to Communicate the Returned Objects	473
	Optimizing the EJB Container with a Larger Bean: MapRequest.setOperation()	473
	Regulating the Beans Nomination	479
	Deploying the EJB Module WasdgBeans.jar	481
	Generating the EJB Module: WasdgBeans.jar	481
	Verifying the EJB Module	482
	Generating and Registering the Enterprise Application: WasdgBeans.ear	483
	Using ejbdeploy.sh to Generate the Code for Deployment	488

xviii IBM WebSphere Application Server Programming

	Testing the EJB Module WasdgBeans.jar's Beans	489
	Installing the WASDG Application	490
	Using Explicit Copy to Merge the EJBs to the WASDG Development Tree	490
	Using a Symbolic Link to Link the EJBs to the WASDG Development Tree	492
	Modifying and Verifying the EJB Module Code	493
	Test Scenario	494
	Assembling the EJB Module WasdgBeans.jar: Using the AAT	496
	Generating the WasdgBeans.jar Using the AAT	497
	Modifying the Code of an Assembled EJB Module	497
	Wrapping Up	498
Chapter 19	Apache SOAP Programming in WAS	499
	Deploying the SOAP Application: wasdgsoap.ear	500
	The soap.war Web Application	502
	Testing the wasdgsoap.ear and Demystifying the SOAP Message	504
	Uninstalling and Installing the wasdgsoap.ear	505
	Apache SOAP Deployment Descriptors Used by WAS	506
	Generating an Apache SOAP Deployment Descriptor Using gensoap-ejb	508
	Passing Parameters	511
	Qualifying Names on the SOAP Client: QName	512
	Specifying Type Mappings in a SOAP Deployment Descriptor	513
	Programming the DataSetSerializer	515
	Matching a Client's QName() to Its Server Mappings	518
	Accessing Beans Using a SOAP Client	520
	Setting the SOAP Client Environment	520
	Testing with SOAP Clients	521
	Debugging SOAP Client in a Nutshell	521
	Wrapping Up	522
Chapter 20	Fundamental Security Programming: Applying JAAS	523
	Installing the Java Authentication and Authorization Service (JAAS)	525
	Programming JAAS	526
	A JAAS Example: FetchFile and PrivilegedFetchFile	527
	Another JAAS Example: ZenFile Servlet and PrivilegedFetchFile	536
	Securing the Teller Login/Logout	543
	JAAS-Enabling the Logout of the WASDG Application	544
	Segregation of the Developer's Roles	544

	The Future of JAAS	553
	Cipher	554
	Wrapping Up	561
Chapter 21	Enterprise Application Development	563
	Reconsidering the Build Process Using the make Utility	565
	Compiling Code Using the make Utility	566
	Automating the Process	568
	An Initial Makefile for the BASE_DEV	571
	Building and Deploying a Two-in-One Script: AppBuild	576
	Concurrent Development: Source Code Control, Compiling, and Testing	578
	A User-Specific Development Tree: Holding Tree	579
	Installing Multiple Instances of WebSphere Domains on the Same Server	581
	Considering the Environment for Concurrent Programmers	586
	Documentation	589
	Wrapping Up	590
Part IV	Stress-Testing, Tracing, and Debugging	
Chapter 22	Stress-Testing	593
	Timing (Basic Timing) and Placing HTTP Requests	595
	Programming a Stress-Tester	597
	Special Considerations for the HTTP Methods	597
	Forking Concurrent Processes for Concurrent Users	598
	Iterating to Place Multiple Requests Within Each Process	599
	The ZappUrl Script	599
	Stress-Testing the WASDG Application with SharkUrl	606
	Files Preprocessed by the SharkUrl Script	606
	Running SharkUrl	608
	Generating and Interpreting a Report	610
	Sniffing the Network to Measure WAS Performance	614
	A Final Note about the Preprocessed Includes	616
	Wrapping Up	617
Chapter 23	Writing an Exception Handler, Logging, and Debugging	619
	Logging, Tracing, and Debugging with Exception Handling	620
	Logging Information: Log.log()	623

xx IBM WebSphere Application Server Programming

Bundling Information with the BundleManager	627
Considering the log.properties File	631
Considering the Exception-Handling Properties Files	631
Exception Handling	633
Testing the Programs	639
IBM Object Level Trace (OLT) and Object Level Debugger (OLD)	641
Wrapping Up	641

Part V Monitoring, Tuning, and Risk Management

Chapter 24	WAS Monitoring with WASLED™ and WASMON™	645
	Monitoring Objective	646
	Component Message Numbers as LED	646
	Installing and Starting Up WASMON™	649
	The First Pane: The User Input Pane	651
	The Second Pane: The WASLED Activities Pane	652
	The Third Pane: The WASMON Activities Pane	652
	Activating and Connecting to the WASMON Server	653
	Monitoring WAS Containment	654
	Filtering WAS Events and Firing Action with WASMON	654
	E-Mailing an Alert	659
	E-Mail: The Subject Dialog Box and Send E-mailButton	659
	Filtering Events and E-Mailing Users	660
	Triggering Scripts	660
	Conditional Monitoring with the Logical Alert Directives	662
	Wrapping Up	664
Chapter 25	Monitoring and Tuning the System Resources	665
	Preparing Your Systems for the Performance Commands	666
	Performance Commands on AIX and Linux	666
	Installing the System Agent on AIX	667
	Requirement for Windows NT	668
	Essential Monitoring Commands: ps , sar , and vmstat	668
	Monitoring Processes with the ps Command	668
	Monitoring Memory and CPU with sar and vmstat	671
	Considering Windows NT Memory Usage	673
	Considering Windows NT Network Usage	675

	Threads and Processes	678
	Threads Statistics on AIX	678
	Monitoring the Threads with <i>MrThread</i> (Mister Thread)	680
	Monitoring Linux Processes with <i>MrTop</i> (Mister Top)	684
	Considering the Swap Paging Space	686
	Considering EJB Caching	687
	Considering Performance for the SESSION Database	688
	Wrapping Up	692
Chapter 26	Risk Management with WASMON	693
	Risk Management with WASMON: A Simple View	695
	Putting WASMON in Supervisor Mode	695
	Putting the Supervisor in an Active State	698
	Entering in Parole with the Supervisor	699
	Checking on the WAS Web Container	705
	WASMON's Four Types of Internal Variables	706
	Static Variables: s-vars	707
	Boolean Variables: b-vars	707
	Differential Variables: d-vars	707
	Glob Variables: g-vars	710
	Preparing the Internal Variables	710
	WASMON Delegation with the Helper Program: wasmonhelper	711
	Using the Internal Variables in wasmon.conf	713
	Using b-vars and d-vars in Logical Expressions	714
	Considering the WASMON Helper Program: wasmonhelper	715
	Understanding Differential Variables: d-var	716
	Generic Monitoring with WASMON	717
	Monitoring Web Applications with WASLED/WASMON	717
	Special Considerations When Running WASMON	719
	Wrapping Up	720
Part VI	Appendixes	
Appendix A	Retrieving Information and Code Distribution	725
	Essential Code Distribution Web Site	726
	Installing the Distribution Code	727
	Installing the Gramercy Toolkit	728

xxii IBM WebSphere Application Server Programming

	Setting the GRAMERCY_DIR Environment Variable	728
	Getting the Version Number of the Tools	729
	Installing WASLED/WASMON	729
	Using This Book in Academic Disciplines	729
	Generalizing J2EE Application Deployment Under WAS AEs	730
Appendix B	Backing Up and Restoring	731
	Backing Up Your WAS Configuration Data	732
	Backing Up and Restoring WAS AEs	732
	Backing Up WAS AE	733
	Restoring WAS AE	733
	Migration Considerations	734
Appendix C	Demystifying Java 2's Internationalization with MrUnicode	737
	Using MrUnicode for Internationalization and Unicode	738
	Printing Information about Converters and Character Sets	741
	Sun Microsystems vs. IBM Nomenclature in Java 2 Encoding	744
	Printing Language Code with MrUnicode	745
Appendix D	Gramercy Toolkit Scripts and the WASDG Environment	747
	Checking the WASDG Environment	748
	Gramercy Toolkit Scripts	750
	BwjSort	750
	cpfl	751
	modjar	751
Appendix E	WASLED/WASMON Quick Reference	753
	Starting and Configuring WASMON	754
	Using Logical Expressions	756
	WASMON Internal Variables	758
	A Quick Monitoring Scenario	761
Appendix F	Support for WAS v5	763
	Determining WAS' Support Through the DTDs	764
	WAS v5 Supports J2EE API 2.3	766
	Reconsidering Scripts with WAS v5	766
	Index	769

Acknowledgments

My thanks are extended to all those who stood by me during the writing and editing of this work; to those who exercised patience at my obstinacy in re-working the text for greater clarity, and to those who, sometimes, lost patience with me. I am particularly grateful to Chris Moss who, all the way from Down Under, carefully checked my ramblings for clarity and correctness, and even found WASMON a ripper. I am deeply indebted to Elizabeth Seymour and Chrisa Hotchkiss, as well, for having glanced fair enough at each and every letter in this manuscript.

Foreword

This text explains the fundamentals of the functionality and programming of WebSphere Application Server (WAS) that are essential for readers whose aim is programming or administering the product. The text is intended to give the reader an understanding of the basic functions of WAS from the computer system perspective. The explanation follows a practical approach to understanding WAS independently of a specific version. Therefore, the text shows a (programming) methodology that is applicable to WAS v3.5, v4, and to the upcoming release of WAS v5.

As necessary background, the reader should possess programming ability in some high-level language. One full course in Java programming and some rudimentary knowledge of Perl and shell script programming constitute the level of competence required for all parts of this text. Programming for WAS is not confined solely to Java. Because J2EE application servers and J2EE application programming demand the editing of many metafiles and descriptor files, we have chosen to use Perl, shell, the UNIX make utility, and other UNIX tools, to do the text preprocessing, compilation, and deployment of code. The choice of UNIX as a development environment is justified since it is the only system that promotes application development in enterprises.

Basic application server programming is taught in the context of distributed systems that vary in platforms; therefore, various operating systems were considered. Although the Java machine is common to all platforms, threads management is operating system dependent and greatly impacts the performance of the application server. Here we have chosen AIX and Linux as platforms to highlight this difference.

Explanation of the structure of WebSphere Application Server has been made generic for the most part, without targeting any specific version of the product. The text addresses the fundamentals of WAS programming and integration, and offers many utility scripts that can be used regardless of the specific version of the product. WAS AE stands for WAS Advanced Edition. The version implied by the term WAS AE refers to either v3.5 or v4. WAS AEs stands for Advanced Edition Single Server v4. The code and utility scripts represented herein are backward compatible with WAS v5.

The reader will use WAS AEs in most of the programming (Part III), because this version is the most adequate (and is free of charge) for writing J2EE applications based on the IBM J2EE application server.

J2EE is not treated as a separate topic; rather, it is introduced in the context of web application programming and deployment under WAS. This is done very specifically, based on the assumption that most readers will probably receive a complete treatment of J2EE in another course or on their own by consulting online documentation.

xxvi IBM WebSphere Application Server Programming

The reader who would like to write web application programs and test them with WAS may begin with Part III of this book. The only requirement is a Linux sandbox with a connection to the Internet to download the WAS AEs trial version, the IBM UDB v7.2 trial version, and the code distribution of this text. No other third party software is required.

Part III can be used on its own, provided a system administrator has prepared the development environment to be used by the reader, as explained in chapters 3 and 4, or alternatively, as it is explained in Appendix A.

Material has been sequenced scrupulously to avoid forward references and to present sufficient steps that allow the reader to program with WAS AEs' free trial (but fully functional) version on a home computer. The approach taken here is to present the reader with some mechanisms to build web applications using IBM's J2EE application server—WebSphere Application Server—without the interference of any other WebSphere product. In this way, the reader will gain a clear understanding of WAS, making the IBM application server an openly available product that can, for example, be taught in a college course.

The text also breaks the dependency on using a graphical interface for programming, debugging, deployment, and administration. This is achieved by applying Perl scripts, and by using applications such as Lynx and the WebSphere Control Language (known as WSCP, and replaced by wsadmin in WAS v5). In addition, many other standard UNIX commands are used, which should be at the fingertips of any competent programmer. Using Lynx is appropriate because it provides an instantaneous view to the HTML header. Perl is an easy language that permits the automation of text processing, and therefore fits to generate the deployment descriptor of J2EE applications automatically.

The material has been classified into six parts so the text can be used in a university for two full courses as an introduction to WAS. Part I targets the preparation of the development environment; Part II supplies the requisite knowledge to administer WAS; Part III teaches programming for WAS; Part IV teaches stress-testing and debugging; Part V examines WAS monitoring, tuning, and risk management; finally, Part VI forms the appendices.

The manuscript was written at Total Computing & Network Design, Inc. (TCND), in New York City, and was carefully reviewed by several IT professionals at IBM. TCND provided the computing facilities on which programming and testing took place. It also provided the tools and applications that helped generate the code listings.

The Gramercy Toolkit and the WASLED/WASMON application are the result of Project-9, likewise conducted at TCND. The toolkit and the application are freely distributed with the purchase of this text, and licensing is waived solely for educational use (non-commercial use): that is, specifically to accredited Computer Science departments and/or readers who will use the product on their home computers.

The most recent update of Gramercy Toolkit and WASLED/WASMON may be obtained from the website www.tcnd.com. The author welcomes readers' comments and critiques about the project, and asks that they be sent to wasbook@tcnd.com.

Bassem W. Jamaledine
Total Computing & Network Design, Inc.
New York, New York
November 2002