Dear Valued Customer,

We realize you’re a busy professional with deadlines to hit. Whether your goal is to learn a new technology or solve a critical problem, we want to be there to lend you a hand. Our primary objective is to provide you with the insight and knowledge you need to stay atop the highly competitive and ever-changing technology industry.

Wiley Publishing, Inc., offers books on a wide variety of technical categories, including security, data warehousing, software development tools, and networking — everything you need to reach your peak. Regardless of your level of expertise, the Wiley family of books has you covered.

- For Dummies – The fun and easy way to learn
- The Weekend Crash Course – The fastest way to learn a new tool or technology
- Visual – For those who prefer to learn a new topic visually
- The Bible – The 100% comprehensive tutorial and reference
- The Wiley Professional list – Practical and reliable resources for IT professionals

The book you hold now, *Mastering Tomcat Development*, is a code-intensive guide to developing J2EE applications using the Tomcat server. This book takes a developer’s perspective on Tomcat, showing you how to use Tomcat with related tools such as JDBC, Struts, Jakarta Taglibs, and Velocity. The result is a comprehensive approach to developing sophisticated and secure Java Web applications. All of the source code used in the real-world examples throughout the book is available for you to download from the book’s companion Web site at www.wiley.com/compbooks/harrison.

Our commitment to you does not end at the last page of this book. We’d want to open a dialog with you to see what other solutions we can provide. Please be sure to visit us at www.wiley.com/compbooks to review our complete title list and explore the other resources we offer. If you have a comment, suggestion, or any other inquiry, please locate the “contact us” link at www.wiley.com.

Thank you for your support and we look forward to hearing from you and serving your needs again in the future.

Sincerely,

Richard K. Swadley
Vice President & Executive Group Publisher
Wiley Technology Publishing

*WILEY advantage*

WILEY
Independent Thinkers

more information on related titles
This book is dedicated to The Apache Software Foundation, to the Tomcat team, and to the spirit of the open source movement, which has created such wonderful tools and made them freely available to the world.

Acknowledgments

Thanks first to Tim Ryan of Ryan Publishing for contacting me in the first place and enticing me to work on this book. Thanks also for all his support and encouragement during the whole writing, rewriting, and production process.

Thanks to Rick Hightower, who contributed two excellent chapters—the one on Ant and the one on Struts. They significantly improve the book, and his writing them ensured that I was able to see the sun a couple of times during the two weeks he was working on them.

Thanks to Liz Welch, my copyeditor, whose suggestions were always insightful and informed, and whose input greatly improved the quality of this book.

Thanks to all my friends, who understood that I didn't have much time to see them during the two-month marathon that the production of this book entailed.

Thanks also to my colleagues at LocalLogic and inkchaser for understanding the deadlines, which often took me away from working on their projects.

Thanks to my downstairs neighbor Said, his restaurant Zaré, and his fine staff, Adel and the others, who let me sit for hours in the afternoon on the patio and work on this book. Such a pleasant change from sitting at my desk in the same room hour after hour!
# Contents

<table>
<thead>
<tr>
<th>Acknowledgments</th>
<th>xv</th>
</tr>
</thead>
<tbody>
<tr>
<td>About the Author</td>
<td>xvii</td>
</tr>
<tr>
<td>Introduction</td>
<td>xix</td>
</tr>
</tbody>
</table>

## Part I  Tomcat Configuration and Management

<table>
<thead>
<tr>
<th>Chapter 1  Introduction to Tomcat</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Apache Software Foundation</td>
<td>1</td>
</tr>
<tr>
<td>Tomcat</td>
<td>2</td>
</tr>
<tr>
<td>Tomcat Versions</td>
<td>3</td>
</tr>
<tr>
<td>Servlets</td>
<td>4</td>
</tr>
<tr>
<td>What's New in the Servlet 2.3 Specification</td>
<td>5</td>
</tr>
<tr>
<td>JavaServer Pages</td>
<td>6</td>
</tr>
<tr>
<td>What's New in the JSP 1.2 Specification</td>
<td>7</td>
</tr>
<tr>
<td>Summary</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2  Installing Tomcat</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installing the Java SDK</td>
<td>9</td>
</tr>
<tr>
<td>Installing the Tomcat Files</td>
<td>10</td>
</tr>
<tr>
<td>Windows Installation</td>
<td>10</td>
</tr>
<tr>
<td>Linux, OS X, and Other Unix Installations</td>
<td>10</td>
</tr>
<tr>
<td>Setting Up the Environment for Tomcat</td>
<td>12</td>
</tr>
<tr>
<td>Windows 95/98/ME</td>
<td>12</td>
</tr>
<tr>
<td>Windows NT/2000</td>
<td>12</td>
</tr>
<tr>
<td>Linux</td>
<td>13</td>
</tr>
<tr>
<td>Mac OS X</td>
<td>14</td>
</tr>
<tr>
<td>Starting Up Tomcat for the First Time</td>
<td>14</td>
</tr>
<tr>
<td>Windows</td>
<td>14</td>
</tr>
<tr>
<td>Linux, OS X, and other Unix</td>
<td>15</td>
</tr>
</tbody>
</table>
Contents

Starting Tomcat on Boot 17
  Windows 17
  Linux 17
  Mac OS X Server 19
  Mac OS X 19
  Other Unix Flavors 23
First Look at Tomcat 24
Summary 24

Chapter 3 Configuring Tomcat 25

Tomcat Configuration Files 28
  The HTTP 1.1 Connector 29
  The WARP Connector 30
  Other Connectors 31
  Webapp Contexts 31
Web Application Configuration 34
Configuring Tomcat with Apache 34
  Changes to Apache 35
Configuring Tomcat with Microsoft IIS 36
  Adding Registry Entries 37
Virtual Host Configuration 38
  Apache Virtual Hosting 38
  Tomcat Virtual Hosting 40
Summary 40

Chapter 4 Hello World Wide Web 41

Handling a Request 44
What’s In a Response 46
Summary 47

Chapter 5 Anatomy and Life Cycle of a Servlet 49

The Servlet Life Cycle 49
Servlet Anatomy 52
Life Cycle Events 57
  javax.servlet.ServletContextListener 57
  javax.servlet.ServletContextAttributeListener 59
  javax.servlet.http.HttpSessionListener 60
  javax.servlet.http.HttpSessionAttributeListener 61
Configuring Life Cycle Listeners 62
Summary 64

Chapter 6 Configuring Web Applications 65

Installing in /webapps 65
The Deployment Descriptor: web.xml 67
WAR Files 69
Configuring Servlets from the server.xml File 70
Summary 71

Chapter 7 The server.xml File in Depth 73

New Features in Tomcat 4.1 74
The Basic Model 74
The <Server/> Node 75
The <Service/> Container 76
## Chapter 10  The Model-View-Controller Architecture 129

### What Is MVC Architecture? 129
- Model Objects
- View Objects
- Communication between Controller and View
- Controller Objects

### Advantages of MVC 131
- Clean Design
- Encapsulation
- Extensibility
- Reuse and Refactoring

### Using MVC on the Web 132

### Summary 134

## Chapter 11  JavaServer Pages 135

### What Does a JSP Page Look Like? 135
- JSPs Are Really Servlets
- Jasper: The Tomcat JSP Compiler
- Expressions
- Scriptlets and Declarations
- Page Directives
- Including Pages
- Working with Beans
  - Scope
- Debugging JSPs

### Summary 154

## Chapter 12  Tag Libraries 155

### Writing a Basic Tag Library 156
- Packaging the Tag
- Using the Tag
  - A Word about Namespaces
- A Practical Tag Example
- Tags with Bodies
- JSTL: The JSP Standard Tag Library
- Other Tag Libraries from the ASF

### Summary 182

## Chapter 13  MVC with Struts 183

### Overview of Struts 184

### A Typical Model 1-Style JSP Application 185
- The Department Model
- The Department Database Schema
- The JSPs for the Model 1 Department Application

### The MVC Struts Version of the Application 200
- List Department Action
- Action Details
- The Struts DeptListing View
- The Struts Custom Tags
Automated Commits 312  
Creating New Objects 316  
Using the Framework 319  
Adding the getObjects() Method 327  
Some Remaining Issues 331  
Summary 331

Chapter 18 Security 333

Configuring Tomcat 334  
SSL, HTTPS, and TLS 334  
Using Tomcat Behind Another Server to Provide HTTPS 335  
Setting Up the HTTPS Connector 335  
Digital Certificates 336  
Configuring the Connector 338  
The Security Manager 339  
Bad Code 339  
Pernicious Code 340  
Weak, Dangerous, Badly Written, or Misconfigured Code 340  
Tomcat and the Security Manager 341  
Using the Security Manager with Tomcat 345  
Debugging Permissions Issues 345  
Permissions 345  
Running as an Unprivileged User 346  
Servlet Development 347  
Using Onetime Passwords 347  
Discovering the Protocol of Your Connection 348  
Using Certificates as User Credentials 350  
Other Techniques for Validating Identity 351  
Summary 351

Chapter 19 The Development Cycle 353

Structuring Your Workspace 354  
Version Control 355  
Build Management 358  
Running the Software You Build 361  
Testing 361  
Developing Unit Tests with JUnit 361  
Integration Tests 366  
Refactoring 369  
Staged Release Environment 369  
Setting Up Your Development and Testing Instances 369  
User Testing 370  
Load Testing and Profiling 370  
Profiling Your Code 371  
Summary 372

Chapter 20 Developing Web Applications with Ant and XDoclet 373

Setting Up Your Environment to Run Ant 374  
What Does an Ant Build File Look Like? 375  
Properties, File Sets, and Paths 375  
Using Properties 377  
Conditional Targets 378  
Using Filters 378
Contents

Creating a Master Build File 380
Using the War Task to Create a WAR File 381
Using XDoclet's webdoclet Task to Create Deployment Descriptors 382
Running Ant for the First Time 392
Using the XDoclet's webdoclet Task to Create Custom Tag TLDs 393
Standard Targets 400
The Hello World Model Project 402
  Overview of Model Classes 402
  Creating a Project Directory Structure for Model 403
  Creating a Build File for a Shared Library 404
  Analysis of the Model Project Build File 404
  Running an Ant Build File 408
The Hello World Application Project 409
  Overview of Application Java Classes 409
  Creating a Project Directory Structure for the Application 409
  Creating a Manifest File for a Stand-alone Application 410
  Creating an Ant Build File for a Stand-Alone Application 410
The Hello World Main Project 413
  Creating a Master Build File 413
  Analysis of the Master Build File 414
The Applet Project 415
  Overview of the Applet Class 415
  Creating a Build File for the Applet 419
  Building the Applet with Ant 420
  Hello World Recap 421
Hello World Model 2 and J2EE 421
  The Web Application Project 422
  Building and Deploying the Web Application 432
  Running the Web Application 434
Summary 437

Appendix A  Server Configuration (server.xml) Reference 439

Appendix B  The Deployment Descriptor (web.xml) Reference 451

Index 479
Ian McFarland is president of Neo Ventures, Ltd., a San Francisco–based consulting firm. His articles on Java and Mac OS X appear monthly in Java Developer’s Journal. He has been developing client-server Java applications since Java version 1.0 alpha 2, and with his team wrote the first client-server application ever written in Java: a demo seating-reservation application used for the Java product announcement at SunWorld in 1995. He first started working on networked hypertext publishing working with Ted Nelson at Autodesk, Inc., in 1989, and was on the launch team of HotWired. He subsequently served as Java Evangelist for Symantec’s Visual Café before returning to consulting. He is the author of a number of articles for various publications, including the Sun Microsystems developer site Java Developer Connection and Developer.com. He is also published in the anthology Drama in the Desert.

About the Contributors

Peter Harrison is senior developer and technical team leader for Customer Information Technologies, located in New Zealand. At CIT he leads a team of Java Web application developers who primarily use servlets and JSP. He is responsible for writing and maintaining all standards and procedures; he has implemented the XP methodology of project development; and he manages all testing, including setting up automated unit testing, source-code control, peer-review systems, and planning procedures.

Applications that Peter has developed using servlet/JSP include a digital rights management system, a mailing list server, a security product called Eproxy, and the dynamic content-delivery application that powers www.devcentre.org.

Peter is Webmaster for DevCentre.org, a resource for open source developers that includes articles he has written. He has been a software developer for 11 years, and also teaches a Java class at Glenfield College in Auckland, New Zealand.
Richard Hightower is CTO of Trivera Technologies, a leading Java education company that offers online and classroom training, as well as consulting on enterprise Java application development. Rick is a software engineer by training who specializes in software development tools and processes; and developing enterprise applications using J2EE, XML, UML, JDBC, SQL, and open source technologies. Formerly, he was the Senior Software Engineer for Java Architecture at Intel's Enterprise Architecture Lab. At Intel he led a team of developers, designing and implementing three-tier client-server applications; introduced OO CASE tools; and created several frameworks using a variety of Java, COM, CORBA, and middleware technologies. Rick also created ICBeans and authored the patent application for this technology, which was awarded to Intel.

Rick is the co-author of the popular Java Tools for Extreme Programming, writes frequently for Java Developer's Journal, speaks at Java industry conferences, and teaches a variety of enterprise Java development classes at Trivera.
This is a book about Tomcat 4.1. It covers installation and configuration, and also a wide variety of development topics. It will teach you how to develop Web applications on Tomcat, and how to configure Tomcat to support those applications. It won't teach you Java, so if you're going to be writing code, you should already know Java or have a good basic Java text on hand.

There are two detailed appendices covering the complete syntax of both the server.xml configuration file and the Servlet 2.3 deployment descriptor file. Our experience has shown these to be two key areas of difficulty for people new to Tomcat and servlets. Both topics are discussed extensively in their own chapters as well.

**Who Should Read This Book**

This book is intended to give complete coverage of Tomcat from both a development and an administration perspective. It is primarily oriented toward servlet and JSP developers who inevitably have a need to administer their own development installation of Tomcat. Depending on the size of the organization in which readers find themselves, they may also have responsibility for some server configuration and administration, or may wish to provide guidance to the systems administration staff.

Another important audience for this book is the person whose job it is to administer Tomcat. The first half of the book focuses extensively on configuration and management of the Tomcat server. Not all of the development chapters will be necessary reading for these folks, but a greater understanding of the development process will help them make informed configuration and administration decisions.

The third important audience consists of those responsible for quality assurance. By understanding the whole development process, these readers can better anticipate the
issues, improve testing strategies, and ensure that the development and quality assurance platforms closely match the production environment.

Development, quality assurance, and systems administration are three parts of the same whole. This book gives all sides of the team a common vocabulary and a greater understanding of the development, testing, and deployment process.

Versions

This book covers both Tomcat 4.1 and Tomcat 4.0. There are significant new features in Tomcat 4.1 (see Chapter 1 for more information) that we cover in detail. If you are working with Tomcat 4.0, you may notice some slight differences from the examples and figures in this book, and we are confident that you will find the coverage here quite instructive and useful.

Tomcat 4.x implements version 2.3 of the Servlet Specification and version 1.2 of the JSP Specification. We provide chapters on both of these platforms and their accompanying specifications.

Companion Web Site

This book's companion Web site (www.wiley.com/compbooks/harrison) contains all the code listings from the book. It also contains a bonus chapter (Chapter 21, "Source Code Management with CVS") that describes the range of tools available to help you manage your source code. This chapter focuses on CVS and explains how to set up the tool and use it to manage your software projects.

Roadmap

This book is divided into two parts, the first concentrating on server configuration issues and the second on servlet development issues. The chapters are arranged in such a way that you can choose to read the book straight through, with later chapters building on topics covered in earlier chapters. However, it is generally quite possible to skip ahead to topics of particular interest to you. If you're not using HTTP Authentication, for instance, there's no particular need for you to read Chapter 8, "Managing Authentication with Realms." This section is intended to help you understand how the book is laid out so that you can quickly find the information you want.

Part I: Tomcat Configuration and Management

Chapter 1, "Introduction to Tomcat," gives you the nickel tour of what Tomcat is, where Tomcat came from, and what's new in Tomcat 4.1.

Chapter 2, "Installing Tomcat," gets you up and running with the latest version of the Tomcat server.

Chapter 3, "Configuring Tomcat," covers the basics of Tomcat configuration. You'll learn how to configure Tomcat for most applications, without spending time on less-used or more esoteric features.
Chapter 4, "Hello World Wide Web," discusses the fundamentals of servlet development, with an emphasis on HTTP servlets. This chapter also explains how Tomcat calls the doGet(), doPost(), and init() methods in response to client requests.

Chapter 5, "Anatomy and Life Cycle of a Servlet," walks you through the structure of a servlet and explains how the container goes about instantiating and managing servlets.

Chapter 6, "Configuring Web Applications," goes into depth on the configuration and file structure of a Web application and examines the structure of the deployment descriptor file. It also talks about how to build Web Archive (WAR) files, which are designed to simplify Web app deployment.

Chapter 7, "The server.xml File in Depth," takes you through the nitty-gritty of the server.xml file, covering the material in Chapter 3 in more detail and also explaining anything we left out.

Chapter 8, "Managing Authentication with Realms," describes how to configure Tomcat to manage HTTP user authentication.

Chapter 9, "The Manager Application," explains the Tomcat manager servlet, which can be used to deploy and manage servlets.

Part II: Tomcat Development

Chapter 10, "The Model-View-Controller Architecture," gives you some background on a key development paradigm used extensively through the technologies in the four chapters that follow.

Chapter 11, "JavaServer Pages," covers the basics of JSP development.

Chapter 12, "Tag Libraries," explains how to create and use tag libraries.

Chapter 13, "MVC with Struts," covers how to use Struts, a Web application development framework based on the Model-View-Controller paradigm.

Chapter 14, "Template Systems," examines template systems generally, and then discusses Velocity, a template system developed by the ASF to further abstract page layout from code development.

Chapter 15, "Session Management," shows you how to use the excellent support for session management that is a part of the servlet platform.

Chapter 16, "Databases, Connection Pools, and JDBC," covers how to integrate databases into your servlets.

In Chapter 17, "Database-Aware Objects," we develop a reusable framework for object persistence that uses a database for the underlying datastore.

In Chapter 18, "Security," we discuss security considerations, both from a software configuration and a development perspective.

Chapter 19, "The Development Cycle," focuses on using the techniques in this book on a day-to-day basis.

Chapter 20, "Developing Web Applications with Ant and XDoclet," is a detailed discussion of using Ant and XDoclet to automate the build and deploy process of your server-side Java Web components for Tomcat.

Appendixes A and B are references for the server.xml file and web.xml file, respectively.
Many of you picking up this book have developed Java Web applications and want to learn the details of Tomcat configuration, management, and development. Some of you may be new to Tomcat and are considering whether to deploy it. Others may be new to servlet programming. In this chapter, we create a common frame of reference for all readers, and put Tomcat in both a technical and an historical context.

**The Apache Software Foundation**

Tomcat is a servlet and JavaServer Pages (JSP) container developed under the auspices of The Apache Software Foundation (ASF). This foundation is an outgrowth of the Apache Project, an initiative that led to the creation of Apache, the most successful Web server platform in the still-short history of the World Wide Web. Apache itself began as a series of patches to the NCSA HTTPd, which we started at HotWired to support the huge (at the time) user load we experienced on the site from the moment we launched. Apache, in fact, derives its name from those patches. (The server was so named because it was a “patchy” version of the NCSA HTTPd.) Brian Behlendorf—who was responsible for some of the first patches and on whose desktop SGI Indy, one of the first two instances of the patched NCSA HTTPds, ran—developed the ASF to promote open source development of key technologies and has become an icon of the whole open source movement. (By the way, the other instance ran on the desktop Indy belonging to one of your authors, Ian McFarland.)

The ASF has a number of larger subdivisions, called projects, each focused around a different technology area. Perhaps the three key projects of interest to the Java
developer are the HTTP Server Project, which maintains the Apache Web server we just talked about; the Jakarta Project, which has a Java focus; and the Apache XML Project, which, as the name implies, focuses on XML tools.

The Jakarta Project’s mission statement contains a concise summary of the Project’s goals: “Jakarta is a Project of The Apache Software Foundation, charged with the creation and maintenance of commercial-quality, open source, server-side solutions for the Java Platform, based on software licensed to the Foundation, for distribution at no charge to the public.”

The Jakarta Project is responsible for many significant and award-winning Java tools, and Tomcat is one of them. Another very significant project, one that has gained substantial momentum, is Ant, a portable build management system written in Java but applicable to development in any language (see Chapter 20 for in-depth coverage of Ant). Nearly half of the remaining subprojects are related in some way to servlets. A number are template systems or application toolkits. We'll look at two of them, Struts and Velocity, in this book. Other subprojects include the following:

- Turbine is a framework for developing secure Web applications.
- JMeter is a performance testing tool you can use to load test your servlets.
- Log4J is a very highly regarded logging package, originally developed elsewhere, substantially improved by IBM, and then transferred to the Jakarta Project.
- ORO and Regexp are two regular expression packages, also donated to the foundation.
- Jetspeed is a portal platform built on servlets.

Although the Apache XML Project focuses on XML, not Java, there are a large number of Java-based projects there as well, many of particular interest to developers of Web applications. The Xerces parser library has long been a staple of XML development. Xalan is a fine Extensible Stylesheet Language Transformations (XSLT) engine. Axis, and its predecessor, Apache SOAP, are implementations of the Simple Object Access Protocol, which is at the root of the current Web services craze. Cocoon is an XML- and XSLT-based publishing framework. Batik is a Java library for working with Scalable Vector Graphics (SVG) files. And so on.

All of this software has the dual advantages of being open source and, at the same time, commercial-application friendly. If you are developing a commercial application, you can safely use the libraries provided, and the Apache License allows you to embed and redistribute those libraries. In a few cases, third-party libraries with redistribution restrictions are required, notably the security libraries that are part of the Java platform. But for the most part, if you build software using the Apache tools, you can redistribute it as you see fit.

**Tomcat**

Tomcat is the principal servlet engine developed under the auspices of the ASF. Tomcat is not, however, the first servlet engine developed by the foundation. The JServ Project predated it by a substantial interval. The key reason for Tomcat’s success was that it
Tomcat Versions

Tomcat is currently in its fourth major release, Tomcat 4.1. The core of Tomcat 4.x is the Catalina servlet container, which is based on a new architecture and completely rewritten in order to make it more component-oriented and flexible. The other key component is Jasper, the JSP compiler, which is what initially gave Tomcat its edge.

Tomcat 4.x is the reference implementation for both the Servlet Specification, version 2.3, and the JSP Specification, version 1.2. As a result, Tomcat requires a Java 2 runtime, with Java Development Kit (JDK) 1.3.1 or later recommended. It also has excellent support for JDK 1.4.

Tomcat 4.1 improves upon Tomcat 4.0.x in several other areas, the most noticeable being the enhanced management tools. Tomcat 4.1 includes a new Struts- and JSP-based administration tool, and a much improved management servlet, also with a Struts- and JSP-based user interface. There is a JMX interface for administering the server, and the manager Web application also has improved tool support, including features enabling better integration with development tools, and a set of Ant tasks that leverage those tools. There is a new and improved connector class, which implements HTTP 1.0 and
1.1, HTTPS, and AJP 1.3. The Jasper JSP engine has also been rewritten, and there are a number of memory usage and performance improvements throughout.

Tomcat 3.x is the reference implementation for both the Servlet Specification, version 2.2, and the JSP Specification, version 1.1. Tomcat 3 will also run on JDK 1.1, so if you need to run a servlet engine on a platform that still doesn’t support Java 2, this is the place to turn.

**Servlets**

We’ve already talked about servlets indirectly, but it’s worth taking a moment to give an overview of servlet technology and talk about some of its advantages. Servlets are simply Java classes that are used to handle incoming requests to a server, providing a response. Using the intrinsic ability of Java to bind objects at runtime, the servlet container can dynamically load instances of these classes to handle requests. Since Java is platform-neutral, the same compiled servlets, usually packaged in the form of a Web application, can be deployed on any machine with a compliant servlet container, like Tomcat.

Servlets have some key advantages over other technologies when it comes to delivering dynamic content. Since servlets are loaded into the core server to execute, they run in a smaller footprint and use fewer resources than do systems, such as traditional Perl CGI, that create new processes to handle requests. This makes servlets very well suited to the job.

Beyond this, because they’re Java objects, they give the developer access to a huge library of functionality to handle XML parsing, server-to-server communication, database access, encryption, image manipulation, email—almost anything you can think of. Java has a very rich core runtime library, and a vast amount of code has been written to the platform over the seven years since it was introduced.

One of the main differences, however, is that the servlet specification requires all servers to manage session state. This is harder than it sounds, given that HTTP is intrinsically a stateless protocol. If you’re writing servlets or JSPs, you never have to worry about managing sessions. Sessions are automatically created and maintained for you by the runtime system. If you want to store some information specific to a given session, you can store a full-fledged Java object right in the session object itself, and go and retrieve it at any point while the session is still valid. This has been a notable source of brittleness for other platforms.

Servlets are also highly reliable, and scale well to very large applications. As a result, they have been very widely adopted for large, mission-critical applications, often when other technologies have fallen short and have had to be abandoned. Java has also done very well in enterprise integration settings, providing excellent access to legacy data. Further, because servlets are part of the larger Java 2 Enterprise Edition (J2EE) platform, there is an upward migration path to Enterprise JavaBeans (EJB), Java Message Service (JMS), and Java Transaction API (JTA) for applications that are highly transactional, require robust messaging, or need to be run on distributed systems.

Servlets are used so often to handle HTTP requests to produce HTML pages that people often think of them only in that context. They can in fact be used to handle other
request-response-oriented protocols, and serve other datatypes beyond HTML as well. In practice, so many communication protocols are using HTTP now that the ability to handle other protocols seldom comes into play; however, the ability of servlets to produce other datatypes is used quite often. Images can be generated on the fly; documents can be converted from one type—say, XML—to one or more other types—say, HTML and Website Meta Language (WML). Files can be compressed on the fly, or encrypted, or otherwise manipulated. Servlets are often used to fulfill these functions. Further, new protocols like SOAP are leveraging HTTP as their low-level communication protocol, and communicating by sending XML messages back and forth. Coupled with the dynamism of Java as a language and platform, this makes servlets a compelling platform for developing custom protocols, and for serving this new wave of backend-to-backend communication.

We cover servlet technology extensively throughout this book, starting with Chapter 4, “Hello World Wide Web.”

What’s New in the Servlet 2.3 Specification
The Servlet 2.3 Specification adds a few new features to an already extensive and full-featured standard. Since many of you are already experienced servlet engineers, let’s take a moment to highlight the new features introduced in 2.3.

Life Cycle Events
The new specification adds support for event notification for a number of events relating to two major subcategories: context events and session events. Context events provide notification when contexts are created or destroyed; and when attributes on those contexts are added, removed, or modified. Similarly, session events provide notifications when sessions are created or destroyed, and when attributes are added, removed, or modified on those sessions. We talk more about life cycle events at the end of Chapter 5, “Anatomy and Life Cycle of a Servlet.”

Filters
Filters are used to modify the behavior of the servlets they are applied to. They can intercept and modify the request or the response, which allows for a wide range of possible behavior. Functionally, filters are similar to the Tomcat Valves construct, which we discuss in Chapter 7, “The server.xml File in Depth.” Filters can also be chained together, which makes it easy to factor their functionality into handy reusable classes. Some things you can do with filters include:

- Managing authentication and access control
- Logging and auditing
- Converting documents to other formats, for example, via XSLT
- Converting image formats
- Compressing or encrypting data on the fly
- Using filters to trigger notifications when resources are accessed
You could even use filters to attach headers and footers, or use them to proxy requests off to an entirely different server, or translate the requested page into Swedish Chef or Pig Latin, if you were so inclined. Filters have the freedom to modify both the incoming request and the outgoing response, so they can really do almost anything you want them to do.

**Platform Requirement Changes**

The Servlet 2.3 Specification now leverages features of Java 2 Standard Edition (J2SE), particularly in the area of security, and as such requires a J2SE-compliant platform on which to run. If you need a servlet engine that can run on JDK 1.1, you can use Tomcat 3, but of course you won’t have access to the new features mentioned earlier.

The Servlet 2.3 Specification also leverages some of the optional J2SE packages, notably the Java Naming and Directory Interface (JNDI), and various Java security extensions. You will notice that Tomcat 4.1 includes jndi.jar, among others, and you’ll also notice a number of references to JNDI as you read through this book.

This specification is also a subset of J2EE version 1.3, so any 1.3-compliant J2EE platform is required to support the Servlet 2.3 Specification.

Finally, 2.3 is a superset of 2.2, so Servlet 2.2-compliant Web applications should run fine in a 2.3-compliant container like Tomcat 4.1.

**JavaServer Pages**

JSP is a technology for embedding executable Java code into HTML pages or other text-based documents, such as XML documents. It is quite easy to use JSP to create interactive forms. Coupled with the session management features of the underlying platform and the JavaBeans model for constructing self-describing data components, JSP provides an effective way of encapsulating functionality into reusable objects, allowing Web pages to be created by Web designers rather than by engineers. This typically results both in decreased cost and more attractive Web pages. Of particular advantage to the engineers involved, this type of encapsulation also means that Web designers are not prone to break the functional parts of the code.

There are even design tools, most notably Macromedia Dreamweaver, that will allow the components to execute at design time, so that designers can look at real data in their design view and avoid nasty surprises at integration time.

Functionality can be further abstracted out into tag libraries, also known as taglibs. This mechanism allows the engineer to create new custom tags that designers can use just like the standard built-in tags that are part of HTML. These tags are interpreted on the server and can do pretty much anything that Java can do. What gets rendered out in place of the tag is up to the tag developer. The tag could encapsulate anything from a login box, to a system status message, to a mechanism for managing ad banners, to a system for managing user preferences, to smart form components. The server interprets the tag as defined in the tag library, and renders out whatever HTML is appropriate. The client just sees regular HTML, so there is no client compatibility issue.

We provide a more detailed exploration of JSP in Chapter 11, “JavaServer Pages,” and Chapter 12, “Tag Libraries.”
What’s New in the JSP 1.2 Specification

The biggest change in JSP 1.2 is simply that it sits on top of the Servlet 2.3 platform. Beyond this, it has enhanced support for internationalization; better support for tool-based generation; and a number of refinements providing for better syntax validation, a more stringent contract for tag handling, and a newly completed XML Document Type Definition (DTD) that allows JSP pages to conform to XML validation rules.

Summary

In this chapter, we put Tomcat into both a technical and an historical context. You should now have a better idea of what Tomcat is, how it came to be, and its relationship to both the Servlet and JSP standards. We covered the major changes in Tomcat 4.1, and in the current versions of the Servlet and JSP specifications, versions 2.3 and 1.2, respectively. We also laid out a roadmap to the chapters ahead, so you should have a good sense of what this book covers, and where to go from here.

We wish you a good read, and hope the information you find here will make you more productive and help you get the most out of a very powerful platform.