

Head First Servlets and JSP™

Second Edition

Wouldn't it be dreamy
if there were a Servlets book
that was more stimulating than
deleting spam from your inbox?
It's probably just a fantasy...



Bryan Basham
Kathy Sierra
Bert Bates

O'REILLY®

Beijing • Cambridge • Köln • Paris • Sebastopol • Taipei • Tokyo

Table of Contents (Summary)

	Intro	xix
1	Why use Servlets & JSPs: <i>an introduction</i>	1
2	Web App Architecture: <i>high-level overview</i>	37
3	Mini MVC Tutorial: <i>hands-on MVC</i>	67
4	Being a Servlet: <i>request AND response</i>	93
5	Being a Web App: <i>attributes and listeners</i>	147
6	Conversational state: <i>session management</i>	223
7	Being a JSP: <i>using JSP</i>	281
8	Script-free pages: <i>scriptless JSP</i>	343
9	Custom tags are powerful: <i>using JSTL</i>	439
10	When even JSTL is not enough: <i>custom tag development</i>	499
11	Deploying your web app: <i>web app deployment</i>	601
12	Keep it secret, keep it safe: <i>web app security</i>	649
13	The Power of Filters: <i>wrappers and filters</i>	701
14	Enterprise design patterns: <i>patterns and struts</i>	737
A	Appendix A: <i>Final Mock Exam</i>	791
i	Index	865

Table of Contents (the real thing)

i

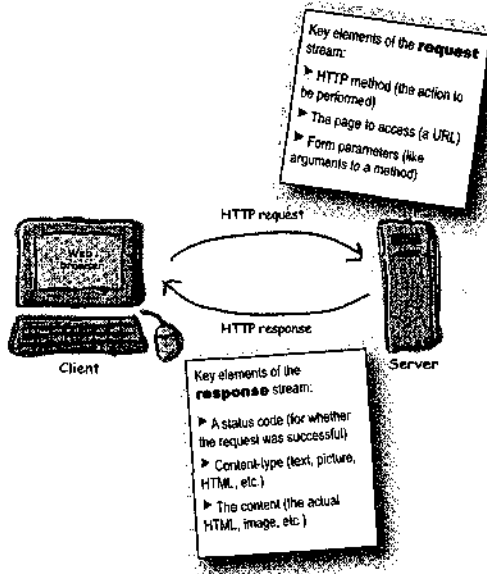
Intro

Your brain on Servlets. Here you are trying to *learn* something, while here your *brain* is doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing Servlets?

Who is this book for?	xx
We know what your brain is thinking	xxi
Metacognition	xxiii
Bend your brain into submission	xv
What you need for this book	xxvi
Passing the certification exam	xxviii
Technical reviewers	xxx
Acknowledgments	xxxix

1 Why use Servlets & JSPs

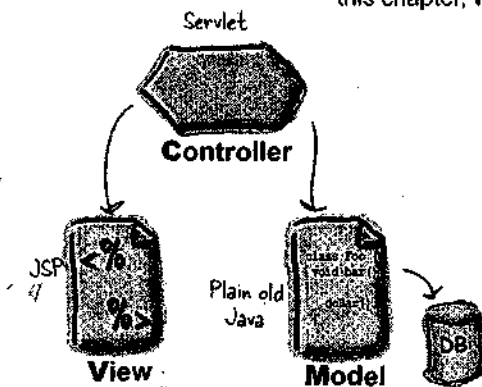
Web applications are hot. How many GUI apps do you know that are used by millions of users worldwide? As a web app developer, you can free yourself from the grip of deployment problems all standalone apps have, and deliver your app to anyone with a browser. But you need *servlets and JSPs*. Because plain old static HTML pages are so, well, 1999. Learn to move from web site to web app.



Exam objectives	2
What web servers and clients do, and how they talk?	4
Two-minute guide to HTML	7
What is the HTTP protocol?	10
Anatomy of HTTP GET and POST requests and HTTP responses	16
Locating web pages using URLs	20
Web servers, static web pages, and CGI	24
Servlets Demystified: write, deploy, and run a servlet	30
JSP is what happened when somebody introduced Java to HTML	34

2 Web app architecture

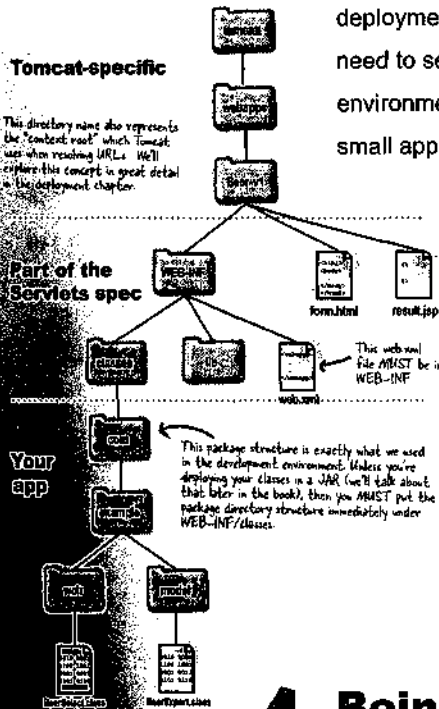
Servlets need help. When a request comes in, somebody has to instantiate the servlet or at least allocate a thread to handle the request. Somebody has to call the servlet's `doPost()` or `doGet()` method. Somebody has to get the request and the response to the servlet. Somebody has to manage the life, death, and resources of the servlet. In this chapter, we'll look at the Container, and we'll take a first look at the MVC pattern.



Exam Objectives	38
What is a Container and what does it give you?	39
How it looks in code (and what makes a servlet)	44
Naming servlets and mapping them to URLs using the DD	46
Story: Bob Builds a Matchmaking Site (and MVC intro)	50
A Model-View-Controller (MVC) overview and example	54
A "working" Deployment Descriptor (DD)	64
How J2EE fits into all this	65

3 Mini MVC tutorial

Create and deploy an MVC web app. It's time to get your hands dirty writing an HTML form, a servlet controller, a model (plain old Java class), an XML deployment descriptor, and a JSP view. Time to build it, deploy it, and test it. But first, you need to set up your *development* environment. Next, you need to set up your *deployment* environment following the servlet and JSP specs and Tomcat requirements. True, this is a small app... but there's almost NO app that's too small to use MVC.

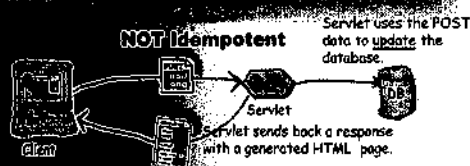


Exam Objectives	68
Let's build an MVC application; the first design	69
Create the development and deployment environments	72
Create and test the HTML for the initial form page	75
Create the Deployment Descriptor (DD)	77
Create, compile, deploy, and test the controller servlet	80
Design, build, and test the model component	82
Enhance the controller to call the model	83
Create and deploy the view component (it's a JSP)	87
Enhance the controller servlet to call the JSP	88

4 Being a Servlet

Servlets need help. When a request A servlet's job is to take a client's **request** and send back a **response**. The request might be simple: "get me the Welcome page." Or it might be complex: "Complete my shopping cart check-out." The **request** carries crucial data, and your servlet code has to know how to find it and how to use it. And your servlet code has to know how to **send a response**. Or not...

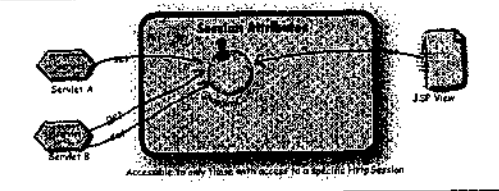
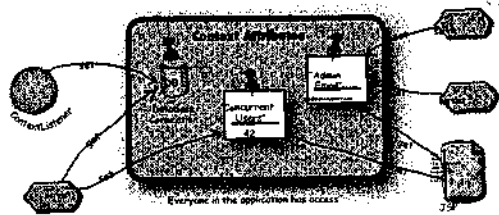
Idempotency is nothing to be ashamed of...



Exam Objectives	94
A servlet's life in the Container	95
Servlet initialization and threads	101
A Servlet's REAL job is to handle GET and POST requests.	105
The story of the non-idempotent request	112
What determines whether you get a GET or POST request?	117
Sending and using parameter(s)	119
So that's the Request... now let's see the Response	126
You can set response headers, you can add response headers	133
Servlet redirect vs. request dispatcher	136
Review: HttpServletResponse	140

5 Being a web app

No servlet stands alone. In today's modern web app, many components work together to accomplish a goal. You have models, controllers, and views. You have parameters and attributes. You have helper classes. But how do you tie the pieces together? How do you let components share information? How do you *hide* information? How do you make information thread-safe? Your job may depend on the answers.

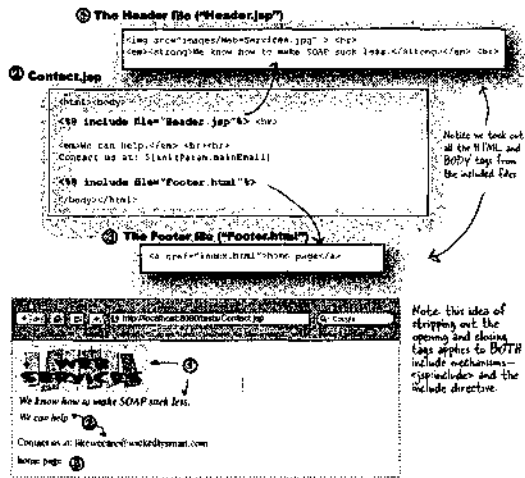


Exam Objectives	148
Init Parameters and ServletConfig to the rescue	149
How can a JSP get servlet init parameters?	155
Context init parameters to the rescue	157
Comparing ServletConfig with ServletContext	159
She wants a ServletContextListener	166
Tutorial: a simple ServletContextListener	168
Compile, deploy, and test your listener	176
The full story, a ServletContextListener review	178
Eight Listeners: they're not just for context events...	180
What, exactly, is an attribute?	185
The Attribute API and the dark side of attributes	189
Context scope isn't thread-safe!	192
The problem in slow motion...	193
Trying out Synchronization	195
Are Session attributes thread-safe?	198
The SingleThreadModel	201
Only Request attributes and local variables are thread-safe!	204
Request attributes and Request dispatching	205

8 Script-free pages

Lose the scripting. Do your web page designers really have to know Java? Do they expect server-side Java programmers to be, say, graphic designers? And even if it's just *you* on the team, do you really want a pile of bits and pieces of Java code in your JSPs? Can you say, "maintenance nightmare"? Writing scriptless pages is not just possible, it's become much *easier* and more flexible with the new JSP 2.0 spec, thanks to the new Expression Language (EL). Patterned after JavaScript and XPATH, web designers feel right at home with EL, and you'll like it too (once you get used to it). But there are some traps... EL *looks* like Java, but isn't. Sometimes EL behaves differently than if you used the same syntax in Java, so pay attention!

Don't expect ME to strip out your redundant opening and closing tags.



Exam Objectives	344
When attributes are <i>beans</i>	345
Standard actions: useBean, getProperty, setProperty	349
Can you make polymorphic bean references?	354
The <i>param</i> attribute to the rescue	360
Converting properties	363
Expression Language (EL) saves the day!	368
Using the dot (.) operator to access properties and map values	370
The [] gives you more options (Lists, arrays...)	372
More dot and [] operator details	376
The EL implicit objects	385
EL functions, and handling "null"	392
Reusable template pieces—two kinds of "include"	402
The <code><jsp:forward /></code> standard action	416
She doesn't know about JSTL tags (a preview)	417
Reviewing standard actions and include	417

9 Custom tags are powerful

Sometimes you need more than EL or standard actions. What if you want to loop through the data in an array, and display one item per row in an HTML table? You *know* you could write that in two seconds using a for loop in a scriptlet. But you're trying to get away from scripting. No problem. When EL and standard actions aren't enough, you can use *custom* tags. They're as easy to use in a JSP as standard actions. Even better, someone's already written a pile of the ones you're most likely to need, and bundled them into the JSP Standard Tag Library (JSTL). In *this* chapter we'll learn to use custom tags, and in the next chapter we'll learn to create our own.

```

<table border="1">
  <tr>
    <td>
      <c:forEach var="listElement" items="${movies}" >
        <c:forEach var="movie" items="${listElement}" >
          <tr>
            <td>${movie}</td>
          </tr>
        </c:forEach>
      </c:forEach>
    </td>
  </tr>
</table>

```

The ArrayList request attribute

One of the String arrays that was assigned to the outer loop's "var" attribute

From the first String array

From the second String array

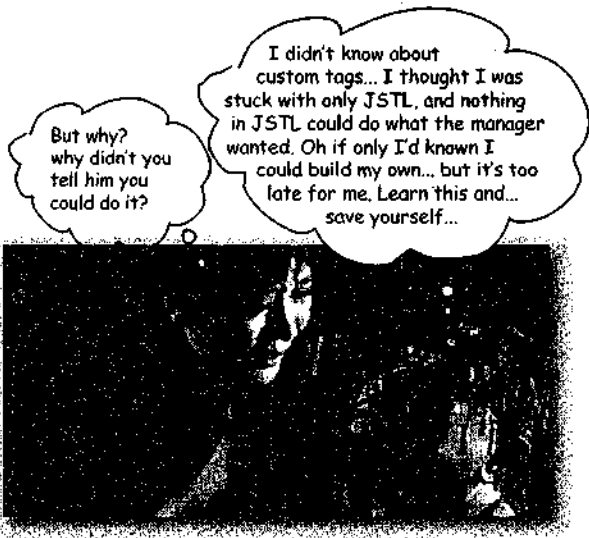
Matrix Revolutions
Kill Bill
Boondock Sains
Amelie
Return of the King
Mean Girls

Exam Objectives	440
Looping without scripting <c:forEach>	446
Conditional control with <c:if> and <c:choose>	451
Using the <c:set> and <c:remove> tags	455
With <c:import>, there are now <i>three</i> ways to include content	460
Customizing the thing you include	462
Doing the same thing with <c:param>	463
<c:url> for all your hyperlink needs	465
Make your own error pages	468
The <c:catch> tag. Like try/catch... <i>sort of</i>	472
What if you need a tag that's NOT in JSTL?	475
Pay attention to <rtexprvalue>	480
What can be in a tag body	482
The tag handler, the TLD, and the JSP	483
The taglib <uri> is just a name, not a location	484
When a JSP uses more than one tag library	487

10

When even JSTL isn't enough...

Sometimes JSTL and standard actions aren't enough. When you need something custom, and you don't want to go back to scripting, you can write your own tag handlers. That way, your page designers can use your tag in their pages, while all the *hard* work is done behind the scenes in your tag handler class. But there are three different ways to build your own tag handlers, so there's a lot to learn. Of the three, two were introduced with JSP 2.0 to make your life easier (Simple Tags and Tag Files).



Exam Objectives	500
Tag Files: like include, only better	502
Where the Container looks for Tag Files	509
Simple tag handlers	513
A Simple tag with a body	514
What if the tag body uses an expression?	519
You still have to know about Classic tag handlers	529
A very small Classic tag handler	531
The Classic lifecycle depends on return values	536
IterationTag lets you repeat the body	537
Default return values from TagSupport	539
The DynamicAttributes interface	556
With BodyTag, you get two new methods	563
What if you have tags that work together?	567
Using the PageContext API for tag handlers	577

11

Deploying your web app

Finally, your web app is ready for prime time. Your pages are polished, your code is tested and tuned, and your deadline was two weeks ago. But where does everything go? So many directories, so many rules. What do you name your directories? What does the client think they're named? What does the client actually request, and how does the Container know where to look?

Reference to a local bean

```
<jsp:local-ref>
  <jsp:ref name="/WEB-INF/classes/LocalBean"
    class="com.example.jsp.LocalBean"
    scope="page" type="page"
    </jsp:ref>
</jsp:local-ref>
```

The JNDI name you'll use in code

These need to fully-qualified names of the bean's exposed interfaces

A LOCAL bean means the client (in this case, a Servlet) and the bean must be sitting in the same JVM.

Reference to a remote bean

```
<jsp-ref>
  <jsp:ref name="/RemoteBean"
    class="com.example.jsp.RemoteBean"
    scope="page" type="page"
    </jsp:ref>
</jsp-ref>
```

Exam Objectives	602
Key deployment task, what goes where?	603
WAR files	612
How servlet mapping REALLY works	616
Configuring welcome files in the DD	622
Configuring error pages in the DD	626
Configuring servlet initialization in the DD	628
Making an XML-compliant JSP: a JSP Document	629

12 Keep it secret, keep it safe

Your web app is in *danger*. Trouble lurks in every corner of the network. You don't want the Bad Guys listening in to your online store transactions, picking off credit card numbers. You don't want the Bad Guys convincing your server that they're actually the Special Customers Who Get Big Discounts. And you don't want *anyone* (good OR bad) looking at sensitive employee data. Does Jim in marketing really need to know that Lisa in engineering makes three times as much as he does?

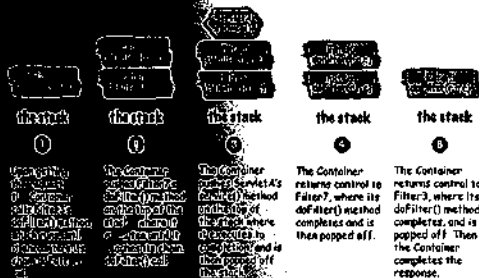
Top Ten Reasons to do your security declaratively

- 10 Who doesn't need more XML practices?
- 9 Often maps naturally to the existing job roles in a company's IT department...
- 8 Looks great on your resume.
- 7 Allows you to use servlets you've already written in more flexible ways.
- 6 It's on the exam.
- 5 Allows application developers to reuse services without access to the source code
- 4 It's just cool.
- 3 Reduces ongoing maintenance when your application grows.
- 2 Finally, a way to justify the cost of that Container...
- 1 Supports the idea of component-based development.

Exam Objectives	650
The Big 4 in servlet security	653
How to Authenticate in HTTP World	656
Top Ten Reasons to do your security declaratively	659
Who implements security in a web app?	660
Authorization roles and constraints	662
Authentication: four flavors	677
The FOUR authentication types	677
Securing data in transit: HTTPS to the rescue	682
Data confidentiality and integrity sparingly and declaratively	684

13 The power of filters

Filters let you intercept the request. And if you can intercept the *request*, you can also control the *response*. And best of all, **the servlet remains clueless**. It never knows that someone stepped in between the client request and the Container's invocation of the servlet's `service()` method. What does that mean to you? More vacations. Because the time you would have spent rewriting just *one* of your servlets can be spent instead writing and configuring a filter that has the ability to affect *all* of your servlets. Want to add user request tracking to *every* servlet in your app? No problem. Manipulate the output from *every* servlet in your app? No problem. And you don't even have to touch the servlet.



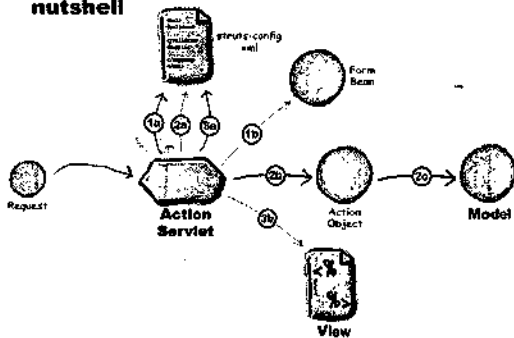
Exam Objectives	702
Building the request tracking filter	707
A filter's life cycle	708
Declaring and ordering filters	710
Compressing output with a response-side filter	713
Wrappers rock	719
The real compression filter code	722
Compression wrapper code	724

14

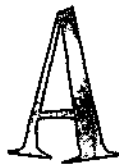
Enterprise design patterns

Someone has done this already. If you're just starting to develop web applications in Java, you're lucky. You get to exploit the collective wisdom of the tens of thousands of developers who've been down that road and got the t-shirt. Using both J2EE-specific and *other* design patterns, you can simplify your code *and* your life. And the most significant design pattern for web apps, MVC, even has a wildly popular framework, Struts, that'll help you craft a flexible, maintainable servlet Front Controller. You owe it to yourself to take advantage of everyone *else's* work so that you can spend more time on the more important things in life...

Struts in a nutshell



Exam Objectives	738
Hardware and software forces behind patterns	739
Review of software design principles...	744
Patterns to support remote model components	745
Overview of JNDI and RMI	747
The Business Delegate is a "go-between"	753
Time for a Transfer Object?	759
Business tier patterns: quick review	761
Our very first pattern revisited... MVC	762
Yes! It's Struts (and FrontController) in a nutshell	767
Refactoring the Beer app for Struts	770
Review of patterns	778



**COFFEE
CRAM**

The final Coffee Cram Mock Exam. This is it. 69 questions. The tone, topics, and difficulty level are all virtually identical to the *real* exam. *We know.*

Final mock exam	791
Answers	828



Index

865