



**Capstone Courseware, LLC**

33 Boylston Street  
Jamaica Plain, MA 02130

877-227-2477  
capstonecourseware.com

# JavaServer Pages

**Robert J. Oberg**  
**Will Provost**

## *Instructor's Guide*

**Revision 2.0.2**



## Course Overview and Philosophy

Web development bridges two disciplines. The first is page design, in which HTML authors create professional quality pages for a web site. The second is programming, which involves implementing business logic, accessing databases, etc. – typically in Java. JSP facilitates a clean separation of these roles, and with the advent of native JSP expressions and JSTL this complete division of labor is a practical option.

This course aims to be largely accessible to both programmers who have a little web background and to page designers who know a little Java. That said, it's primary audience is page authors. Java coding is avoided completely, although there is some review of Java code provided with various exercises, just to clarify the role of Java classes in certain techniques.

The first module covers JSP as a whole, devoting roughly equal time to the various types of JSP content. After an introductory chapter on Web applications and technologies (which may or may not be needed in total by a given student group), the module spends a half chapter on overview of architecture, half on JSP directives, and then a chapter each on scripting, interactive applications, Beans, EL/JSTL, and advanced topics – primarily custom tags.

The second module is a fairly straightforward walk through the JSTL libraries. We start with an overview chapter to introduce (or review) the JSP 2.0 expression language and to introduce the JSTL minimally. This chapter also discusses the evolving JSP 2.0 authoring style, beginning with the page “Going Scriptless.” Much of this section is best-practice discussion, somewhat in advance of learning the specific techniques, but it seemed the best place for it, to put the remaining chapters in some architectural context.

The remaining chapters each introduce one of the four JST libraries. Every action in every library is documented and discussed, using a standard notation at the top of each such page to simplify the learning process and to make the book easier to use as a reference. The chapters are more than just expanded documentation, though: at various points in each chapter, common techniques and best practice relevant to some group of actions is discussed, and these techniques and practices are amplified in the labs.

The course wraps up with a significant workshop exercise, which departs from the other labs in the module in that it does not walk students through the solution step by step. Students are expected at this point to have developed the skills to attack the problem on their own, perhaps with some instructor assistance. If a client prefers it, this workshop is also designed to work well as a “final exam” for the module to show skills development.





## Timeline

### Day 1

Chapter 1	Web Server Fundamentals
Chapter 2	JSP Architecture
Chapter 3	Scripting Elements
Chapter 4	Interactive JSP Applications
This may span Days 1 and 2	

### Day 2

Chapter 5	Using JavaBeans
Chapter 6	The Expression Language and the JSTL
Chapter 7	Advanced JSP Features
This may span Days 2 and 3	

### Day 3

Chapter 1	Effective JSTL
Chapter 2	The Core Actions
Chapter 3	The Formatting and i18n Actions

### Day 4

Chapter 4	The SQL Actions
Chapter 5	The XML Actions





## Eclipse Overlays

Capstone Courseware provides optional packages of workspace and project files for Eclipse WTP 1.5 for this course. (See the course Setup Guide for download URLs.) Instructors, use these packages on your own initiative and at your own risk. You should have experience yourself with Eclipse before using the overlay packages in the classroom. The workspaces and projects have been tested lightly with the course but are not part of the standard product.

That said, this overlay should save a good deal of work for those who wish to use Eclipse instead of the text editor and command-line tools that are standard for the course.

Since the lab software each lab module is configured as a single web application that encompasses all examples, demos, and labs, to get this one big webapp into Eclipse required that we reconfigure the lab image itself. For example, when the installer for the JSP module's Eclipse overlay is run, it:

- Copies the **c:\Capstone\JSP** tree to **c:\Capstone\JSPwithEclipse\Application\JSP**.
- Unzips the Eclipse workspace itself to **c:\Capstone\JSPwithEclipse\Workspace**.
- Updates **c:\Capstone\JSPwithEclipse\Application\JSP\WEB-INF\JSTL.xml** to reflect the new location – just in case you want to run a standalone Tomcat server and install the labs as described in the coursebook, alongside the managed Tomcat server that will run from within Eclipse.

In Eclipse WTP 1.5, open **c:\Capstone\JSPwithEclipse\Workspace**. The workspace is pretty simple: it has a single "Dynamic Web" project for that one big web application, which is pre-deployed to a managed Tomcat 5.0 server. See the read-me file that shows when you first open the workspace for startup and testing instructions.

You should advise students to use **c:\Capstone\JSTLwithEclipse\Application\JSP** as the root whenever they edit files outside of Eclipse. Changes made under **c:\Capstone\JSP** will not be picked up by Eclipse.





There is a similar process to create files under **c:\Capstone\JSTLwithEclipse**. Coverage in this module is not as broad, meaning that there are more examples in which at least some steps must be taken outside of the IDE for things to work. For the **update** scripts in the LevelsBlind and Replication case study to work, it's essential that **CC\_MODULE** be set to **c:\Capstone\JSTLwithEclipse\Application\JSTL** prior to starting Eclipse. Also, because Eclipse WTP 1.5 deploys to Tomcat in a different way than we do in our non-Eclipse approach, it will be necessary to rebuild the **JSTL** application after running **update**, so that Tomcat notices the new **.tag** files.

There is no decent way to configure data sources for Tomcat 5 contexts from within Eclipse; so the examples that use data sources – **Store** and **Replication** – will have to be deployed to a standalone Tomcat instance outside Eclipse – simply follow the instructions in the coursebook as if Eclipse were not installed.

Finally, note that the server configurations are the same as in the Eclipse workspace for our servlets course. If you're teaching more than one of these modules, and don't want to have to swap workspaces, you can import the JSP and/or JSTL projects into the servlets workspace, and test them from there. Be sure to build the project before trying to publish and test it on the server! This is done in advance in the JSP and JSTL workspaces, but if you forget this step (a) it won't work, and (b) it really will be a huge pain to clean up.

Again, Capstone Courseware can only offer complete technical support on the standard course, and while we hope this overlay is convenient, it is not as thoroughly tested as the core lab image at this time. If a given exercise is giving trouble, please be certain to build and run it from the command line, using the SDK tools as prescribed in the student guide, before contacting Capstone.





## Module 1 – Introduction to JSP

### Chapter 1. Web Applications

This chapter reviews the fundamentals of web technology, including HTML and HTTP. Various approaches to developing web applications are discussed, including CGI. The chapter provides a hands-on introduction to servlets and JSP with some small examples. The important points of this chapter are the Web technology review, introducing servlets and JSP in context, and establishing the authoring environment so that students can do hands-on work throughout the module.

### Chapter 2. JSP Architecture

This chapter provides a conceptual overview of how servlets and JSP work. It also starts a systematic survey of the various kinds of JSP content and the most important implicit objects, such as **request** and **response**. JSP 2.0 has evolved into a broad diversity of possible syntax, and there are many ways to encode the same basic functions: scripting elements, standard actions, JSP expressions and JSTL actions overlap significantly in their capabilities. This chapter is partially meant to establish each of these and to distinguish them from one another ahead of time, so students don't start to get lost as they see multiple ways of solving a problem. Successive chapters will address each of these possible syntaxes.

### Chapter 3. Scripting Elements

This chapter explains the uses of scripting elements: scriptlets, expressions and declarations. These are covered for the sake of completeness, but by the end of the module the use of scripts will have fallen from favor by contrast to the action- and expression-oriented authoring style encouraged by the JSP 2.0 specification. It might be a good idea to soft-peddle these techniques, so students (especially those with Java experience who might find scripts the most natural way to code pages) don't get too attached.

By the end of this chapter the student should already be able to do quite a lot of JSP development. The "odd and even" example gives a simple illustration of processing input data from clients by having the input supplied explicitly after a question mark on the URL. This provides a workable way to exercise some small JSPs and helps students without much Web background understand more about HTTP, without yet the details of HTML forms.





## Chapter 4. Interactive JSP Applications

At this point the student already knows quite a lot about JSP, and the time has come to provide a more interesting example. The “electronic store” is a miniature e-commerce site. HTML forms are introduced for submitting data to the server, and JSP code provides handling of data. The application simulates purchasing of a single item, and provides motivation for session handling to implement a shopping cart. In this chapter the servlet session API is introduced, which is illustrated by a small program. The actual shopping cart example is postponed until the next chapter, where a simpler solution is provided. The chapter also includes a discussion of error handling, including an error page and explicitly throwing exceptions in your own Java code.

In the exercises that use the **errorPage** directive, the coursebook mentions a problem with using IE6. The longer story here is that IE6 shows what it calls “friendly” error messages by default: based only on the HTTP error code, it will show its own standard page, discarding whatever HTML is carried in the actual HTTP response body. This “helpful” feature gets in the way here; to turn it off, choose Tools|Options from the IE6 menu, select the Advanced tab, and under the Browsing category, see “Show friendly HTTP error messages” and uncheck this option. With this change IE6 will work correctly with JSP error pages, as other browsers will out of the box.

## Chapter 5. Using JavaBeans

This is a key chapter, because it is with JavaBeans that some serious inroads can be made in removing business logic from the page itself. Actually, you can put business logic in ordinary Java classes, which can be called from simple scriptlet code or using standard actions.

## Chapter 6. The Expression Language and the JSTL

The newest features of JSP are covered here: the native expression language and the use of JSTL. The students are well on their way now to seeing the whole 2.0 authoring style, in which business logic lives completely off-page, and even presentation logic can be packed into reusable fragments. Scripts are becoming a thing of the past.

While the EL is laid out in detail, only the most basic JSTL is covered; we’re just trying to understand the role of JSTL in page authoring and to learn to recognize its use.





## Chapter 7. Advanced JSP Features

The course concludes with a chapter that briefly introduces several more advanced topics. The topic of custom tags is both very large and very important, and really warrants a course in its own right, especially as custom tag libraries are becoming more widely available. The chapter discusses both the more traditional approach of developing custom tags with Java handler classes and the newer, more facile technique of building tag files with snippets of callable JSP code.

The final topic that is discussed is threading. A simple example illustrates concretely how a race condition can arise on a JSP, and various approaches are discussed that will eliminate such a race condition. The example makes clear a fundamental difference between a JSP tag that separately declares a variable and scriptlet code that declares a local variable. You could mention this issue in the first module when both declarations and scriptlets are introduced, but it is probably better not to spend much time on the issue then. The Student Guide provides a simple forward reference to this last chapter.





## Module 2 – The JSP Standard Tag Library

### Chapter 1. Effective JSTL

Since students have already understood the EL and a bit about the role of JSTL, this chapter is meant to re-introduce the JSTL, and to delve more deeply into its role and practical use in JSP. Detailed study of particular actions is deferred for the remaining chapters. Instead, focus on the architectural level and the concepts of decomposition in page authoring and organization.

### Chapter 2. The Core Actions

Now students get down to brass tacks, learning the core library in a somewhat lengthy chapter that gives plenty of exercise in procedural coding. Do make a point of hammering on the difference between static and dynamic attributes: the basic distinction is obvious enough, but, as the book points out over a couple specific cases, actual usage can be far from intuitive. Explain exactly why the **target** attribute of `<c:set>` can't be primed with a literal string, so students have an idea of where EL evaluation actually occurs and how it can and can't be leveraged.

Note that it is not until partway through this chapter that we do the usual environment setup pages. These are placed here because they are not strictly needed earlier: none of the code examples in Chapter 1 calls for testing. If you think students would be interested in seeing example code running right off the bat, you may want to jump ahead to these pages and to get students' environments set up early.

### Chapter 3. The Formatting and i18n Actions

This chapter breaks into two sections: one on formatting dates, times, and numbers, and one on multiple language support. Students may or may not have experience with i18n problems, so it's possible that some additional discussion of locales, time zones and resources will be appropriate. Having the J2SE SDK documentation on hand in a browser may be useful for reference throughout the chapter.

Note that the book calls for browsers to be configured to establish preferences for one language over another to support the labs. Rather than include explicit instructions for a given browser in the student guide, we ask the instructor to be familiar with this process for the browser of choice in a given classroom, and to walk students through it.





## Chapter 4. The SQL Actions

This chapter addresses use of relational data using SQL and JDBC data sources. This chapter is the shortest in the module, partly because (as the book points out) this practice is generally discouraged in JSPs for distributed systems, and partly because there just aren't that many actions to discuss. The SQL library is a very simple wrapper around SQL connection and syntax; between container management of connections and resource references and the literal encoding of SQL text in query and update actions, there's not much to discuss that doesn't wander too far down the road into teaching SQL or JDBC itself. If students do have a great interest in this library, you might want to discuss the use of result collections a bit further, and possibly to plan on an expansion of Lab 5C, which combines SQL and XML skills in a replication exercise. (Perhaps encourage students to write the replication code in both directions.)

## Chapter 5. The XML Actions

This final chapter covers XML actions, which brings both XPath and XSLT into play. Students may or may not know XPath or XSLT. The use of each technology is intentionally limited so as not to assume too much about student background in these areas. Even so, there is likely to be a need for expanded discussion of XML, XPath and XSLT to audiences without good experience in them.

This chapter is the longest of the bunch, not in page count but in suggested time. XML is of course an increasingly important language and technology area for JSP authors, and it makes sense to dwell on these actions and techniques, more so than for the SQL actions, for instance. Also, the switch to XPath from EL in the various attribute values will take some getting used to, as will working with XSLT.

The final lab is optional, and can serve as a safety valve of sorts if the class is running ahead or behind schedule: it can be skipped if you're out of time, and it can also be allowed to run long, perhaps even to 2 hours, if there is extra time and students are interested in delving further into techniques. Possible expansions of the lab (one of which is mentioned under Chapter 4 notes) are to add transaction control to the replication process at various levels and to implement bi-directional replication.





## Revision Notes

**Revision 2.0.2** updates the course to use JSTL 1.1 and to adopt more recent courseware standards. Major changes include:

- Tomcat is now bundled with the lab software, which simplifies classroom setup.
- Java classes required by various exercises are now pre-compiled to the **WEB-INF/classes** directory. The **compile** step that was part of many exercises is no longer required. Now almost all examples can be tested right out of the box. There are a few at the end of the course that still require manual compilation: the custom-tags example has alternative versions of the tag handlers, so we still explicitly compile those; and we manually run **javac** as part of the deployment exercise at the end of the course.
- With the change to JSTL 1.1, we can use the current and final taglib URIs, ending with `/jsp/jstl/???`. No more dancing around with “normal” and “-rt” versions that were common with JSTL 1.0, and we assume a JSP 2 container for everything.
- Database setup is much simpler now thanks to a bundled MySQL 5.0 installation.
- There is now full support for Eclipse WTP 1.5, including integrated deployment and testing of the web applications.





**Revision 2.0.1** updates the course to the final JSP 2.0 specification, and includes the following changes:

- Labs have been migrated to the stable release of Tomcat 5 and tested against Netscape 7.1 and IE 6.0.
- Introductory material on WWW history has been cut down considerably, allowing JSP proper to be introduced much earlier in the first chapter.
- The <Context> deployed to Tomcat for each module is now reloadable, which should simplify a lot of the lab work
- Tag libraries and handlers have been migrated to the JSP 2.0 schema from the earlier DTDs.
- A new “example” called **Schema** has been added with copies of the servlets, JSP and tag-library schema for students’ reference.
- A new example **ElectronicDJ\ELFunctions** illustrates the use of function calls from within JSP native expressions.
- The logistics of using custom tag files in the JSTL Love is Blind example track have been simplified quite a bit. Instead of compiling, running a separate script to update the tag files (which didn’t always work), and then restarting Tomcat, in most cases it is enough to run a single update script and to let the server pick up changes and refresh.
- A bug in the internationalization of Love is Blind has been fixed: the Results.jsp was not publishing the resource bundle to request scope, and so the members.tag wasn’t loading resources correctly.
- The Calculator JSP has been rebuilt with more correct HTML code for its layout.
- A new example JSTL\ElectronicDJ\Scriptlet has been added, to remove a dependency on a code path that was actually in the JSP module. This example now serves the description in JSTL Chapter 1 about sharing objects between JSP, JSTL and other code.
- Fixed the screenshot for the DateTime example in Chapter 3, which was mistakenly caught while browsing with the language set to French.





**Revision 2.0** involves a significant restructuring of our JSP course materials as well as an update to the JSP 2.0 specification. The course has doubled in length, preserving most of the content from its prior two-day version and adding two days on JSTL 1.0.

- This course is now a combination of two modules: Module 112, "Introduction to JSP," and Module 113, "The JSP Standard Tag Library." 112 is essentially the evolution of the prior Course 109, and 113 is new for this revision.
- The previous Chapter 2 has been split into two chapters, one on JSP architecture and one on JSP development using scripting elements.
- The chapter on interactive applications now holds the discussion of cookies, which has been removed here from the final advanced-topics chapter.
- The previous Chapter 5 on relational database access using JDBC has been removed.
- A new chapter 6 covers the basics of the JSP expression language and the role of the JSTL.
- The final chapter of the first module now focuses on custom tags – Java- and JSP-implemented – and on threading and use of XML in JSP. A new example shows how JSPs can be used to implement a primitive REST-style Web service.
- Support for the J2EE reference implementation server has been removed. Tomcat is the most natural choice for this course, but if another Web server, including the J2EE RI, is desired, it should be fairly straightforward to install this and to test out the module software on that server, but we no longer include instructions or support for anything other than Tomcat.
- Similarly, we've removed explicit support for RDBMS other than MySQL. Since data sources are used only in the JSTL SQL actions, reconfiguring to use another database product should be a snap, but we don't provide instructions or support for this.





**Revision 1.3** includes a few major upgrades:

- The course is now available for both Windows and Linux systems.
- We've retested everything on Tomcat 4.0.3.
- The exercises are all built to run on MySQL, while including previous support for Cloudscape and MS Access. The student guide only mentions MySQL, but Cloudscape and Access versions have been preserved in the code distribution and are still functional. (We moved off of Cloudscape because Informix no longer provides it for free. MySQL should continue to be a free downloadable since it's open-source.) See the course setup guide for more.

... and some other fixes:

- Added a discussion and demonstration of Web-application deployment using WAR files. For convenience, most of the course exercises are visited in-place by the Tomcat container thanks to a context entry made at the beginning of the class, but it is useful for students to see the "real-world" way of doing things.
- Added error handling to the Store example step 1 for the intro module. It had been accepting incorrect inputs without returning errors.
- Fixed screenshot on page 3-10 of the intro module, which had been incorrectly drawn from the step 2 Store example and so included an item count.





**Revision 1.2** effects the following updates:

- Splits the course into two modules:
  1. "Introduction to Java Server Pages" – lab module **JSPIntro**
  2. "Java Server Pages in Multi-Tier Development" – lab module **JSPNTier**

This provides a natural break that allows for additional tools to be supported in one module or the other, and also allows the intro module to be re-used in other Capstone courses. The installation of lab software is now a two-step process, where it had been a single step. The table of contents includes directions to install lab software: it is a simple matter of running the install script for each module.

- Adds support for the J2EE reference implementation server for the first module's software. Tomcat is still the preferred server for this course (and the only one tested for both modules); J2EE is available for other courses, especially for custom courses in which EJB modules are also in use; thus both the JSPIntro and EJB materials can be run on one server instead of having to switch midstream. Pages or points in the student guide specific to either Tomcat or J2EE are flagged with appropriate icons.
- Adds support for Microsoft Access for the second module's JSP/JDBC chapter. Access 97 and Access 2000 databases are provided in parallel with the pre-existing Cloudscape databases, and Java code to use the JDBC-ODBC bridge driver to connect to the Access databases is provided. Pages or points in the student guide specific to either Cloudscape or Access are flagged with appropriate icons.
- Replaces the tag-library example with a more substantial one. The old one was a "Hello" example, which was not very compelling and had been developed for 1.0 when the server in use didn't even support tag extensions. The new example uses a single JSP to present personal information for six people (a hypothetical query result) using custom tags. Two different sets of tag handlers present the information two different ways: one as a straightforward table and one as a dictionary list with a more verbose description based on the simple fields.





**Revision 1.1** effects the following updates:

- Replaces the outdated JWSDK with the **Tomcat** server, now the reference implementation server for JSPs.
- **Cloudscape** is now the primary RDBMS for chapter 5 on JSP and JDBC. Access is still part of the plan, and Access 97 and 2000 versions of the databases are provided, but at this point, only Cloudscape source code is provided.
- Icons are used in the student guide to flag material that is specific to one or both of these tools.
- The directory structure of the software and installer has been revised to fit with other Capstone courseware.

**Revision 1.0** is the initial release.





## Errata

At this time there are no errata for the course.

## Troubleshooting and Tool Tips

First, be sure to be familiar with the lab and environment setup instructions in the Table of Contents. Assure that the instructions in the module Setup Guide have been followed, and that you know the locations of the installed tools. If, having followed those steps, a classroom machine is failing to run demos or lab answers correctly, here is a list of items to consider and to doublecheck:

- Generally, Tomcat is good at reloading and recompiling JSPs after they're updated, and as of version 2.0.1 of this course the process of updating tag files and Java classes is much more fluid and should not require any server restarts. If a student runs into trouble with JSP or Java class development and everything seems like it should be working but isn't, try shutting down and restarting Tomcat.
- The **update.bat** script for the Love is Blind and Replication tracks simplifies build and test cycles, and works reliably in all cases but one: when moving from LovelsBlind/Step5 to Step6, or back the other way. This has to do with the addition of a new **pictureURL** attribute to the custom tags, and Tomcat's willingness to let go of loaded tag handlers. Even in this case, we've found the update script will persuade Tomcat to reload in most cases, resulting in successful testing of updated code. When it doesn't, simply stop and restart Tomcat, and run the update script again.





## Feedback

We truly do welcome feedback, both of a specific nature (pointing out mistakes) and general suggestions. For the former sending email with a numbered list of corrections would be most helpful.

Please send feedback to:

Will Provost  
Capstone Courseware  
<mailto:provost@capstonecourseware.com>  
[www.capstonecourseware.com](http://www.capstonecourseware.com)

