



Capstone Courseware, LLC

33 Boylston Street
Jamaica Plain, MA 02130

877-227-2477
capstonecourseware.com

Jakarta Struts

Will Provost

Instructor's Guide

Revision 1.13



Revision Notes

Revision 1.1.3 is a maintenance release with minor fixes and enhancements.

Revision 1.1.2 is a maintenance release with minor fixes and enhancements.

Revision 1.1.1 fixes various minor defects including typos and minor code glitches, as well as the following enhancements:

- In Lab 2A, **SendLoveNote.jsp** now provides explicit feedback to the effect that a session attribute "member" could not be found – since this is the primary success factor for **ChooseMember** and students found a blank page confusing in cases where they hadn't gotten the action code right.
- New Lab 3C focuses on migrating from simple **ActionForm** to **DynaActionForm**.
- Lab 6B now automatically adds love notes to all members, so that students don't have to manually send these notes just to test code revisions.
- In main **LevelsBlind** case study, business beans **Characteristics**, **Profile**, **Member**, and **LoveNote** are **Serializable**. This is not necessarily a best-practice example, since business beans ought not to need persistence features, but since they are being used as session attributes in some early versions of the case study making them serializable gets rid of a number of annoying exceptions in Tomcat as it attempts to persist its session state.
- New diagrams in Chapters 1, 3 and 10 highlight Struts' management of the request/response cycle and the central role of form beans.
- Additional discussion in Chapter 5 on the behavior of **<html:form>** in seeking out the upcoming action mapping and form bean.

Revision 1.1 is the initial release.





Module Overview and Philosophy

This is a relatively advanced course, and requires strong Java programming skills along with good knowledge of servlets, JSP, and basic XML. Well-qualified students should need little persuading that Struts is a useful tool, having bumped up against many of the frustrations of building J2EE Web applications that started the Struts project in the first place. The Struts framework ought to seem intuitive once the basics are introduced, and a week full of excited discussions of design patterns and best practices can certainly ensue. Don't be surprised to find well-qualified students jumping ahead or anticipating topics not yet covered; experienced developers will see many of the issues raised in later chapters as soon as they start building Struts applications in early labs.

However, in practical fact many classes will be populated at least partially with students who either don't have the prerequisites in toto or have them on paper but lack experience and can't code and test very productively. This, frankly, is much of the reason for the 5-day timeline, so don't worry too much if less-than-qualified students are present. There should be ample time at least to be certain that everyone completes the labs in the early part of the course.

The course can be partitioned into three sections, each expanding concepts and skills to a larger "ring" of completion. The first part would be the first three chapters, and it is this "kernel" of the Struts framework that is absolutely essential to convey thoroughly, even to the relatively inexperienced student. Chapters 4-8 expand to a more complete practical set of skills including tag libraries and validation, and the remaining three chapters are mostly advanced techniques and best practices, plus an introduction to Tiles.





Timeline

Day 1

Chapter 1	Struts Architecture
Chapter 2	Action Mappings

Day 2

Chapter 3	Form Beans
Chapter 4	Persistent Data
Chapter 5	Struts Tag Libraries

(Spans days)

Day 3

Chapter 6	The JSP Standard Tag Library
Chapter 7	Internationalization and Localization
Chapter 8	Validation

(Spans days)

Day 4

Chapter 9	Under the Hood
Chapter 10	Best Practices

Day 5

Chapter 11	Tiles
------------	-------

Very well-qualified students – with all the formal prerequisites and fairly deep experience with actual, tactical Java development – may cover this course completely in four days.





The Eclipse Overlay

Capstone Courseware provides an optional package of workspace and project files for Eclipse 3.2 for this course. (See the course Setup Guide for download URLs.) Instructors, use this package on your own initiative and at your own risk. You should have experience yourself with Eclipse before using the overlay package in the classroom. The workspace 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. See the documentation inside the workspace itself for general notes on usage: when you first open the workspace you'll see a **ReadMe.txt** that directs you to deeper, HTML-based documentation. Be prepared to walk students through the first few exercises in Eclipse; the notes in this file are for experienced Eclipse users, and will not be clear to many students on their own.

See also the file **Struts Module Notes.html** (available in the IDE as well) for specific matters related to using Eclipse WTP with this course.

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.





Chapter 1

This chapter serves a few purposes. Ostensibly, it gives a high-level tour of the Struts framework, spending as little as a single page on each of several major features that are covered in their own chapters later in the course. It also provides a dead-simple example to give students a sense of what a Struts application looks like, to introduce the configuration and especially to illustrate the emphasis on XML declaration over Java programming in Struts. Finally, it gives the instructor a chance to prove out most of the environment on students' machines, by working through the simple example application build and test.

Chapter 2

The foundation of Struts, of course, is the action mapping, and this chapter goes slow in bringing out this concept. Several simple examples are provided and students get their first chance to build and configure Struts actions. The first example may seem strange, in that it doesn't use any action classes! At first this may seem to undercut the point of this chapter, but a couple important concepts are proved here: (a) there is value in using action mappings even when connecting one view to the next, and (b) actions are not usually called for where no true "control" is being exerted over the model. This latter idea especially will resurface in later chapters as students confront the design choices typical to Struts: what goes in actions vs. views vs. form beans vs. the model?

Chapter 3

The basic facility that comes with using form beans should be evident right away. However the real power of form beans will probably not be obvious to students just yet, especially because they haven't started working with sufficiently complex applications to see the value of form beans as mediators, and because we have yet to talk about validation. Some hand-waving will be required! Try to sell form beans as the basis for a number of more exciting features, such as the Struts HTML tags and the Struts Validator, and students won't have long to wait to see the real power of form beans in practice.





Chapter 4

This chapter just gives the quickest possible primer on JDBC – though it is not a prerequisite, most students with the actual prerequisites will know some JDBC. At any rate, no direct JDBC coding is called for in this course, but we do want to talk about Struts data sources and see how to use them. This chapter brings more tools into play, and so there will be some time spent proving environment again. Once MySQL is running and Tomcat can find the driver, the student environments will not require much attention for the rest of the course; you may want to extend this promise to them as they're wrestling with class paths and SQL scripts!

Chapter 5

This and the next chapter combine to develop a toolkit of custom tags for developing view components in JSP. As such there is some forward-reference to the JSTL chapter, and the two chapters will feel like a coherent unit in class. The primary focus in this chapter is the HTML tag library, which also continues the case for consistent use of form beans. Seeing how the `<html:form>` and child tags streamline the input and output for an HTML form should solidify students' understanding of the advantages of using form beans in the first place. The exact mechanics of pre-population are tricky, and can be confusing. A page or two and a diagram are provided to clarify this, but don't burn a lot of time on this, even if it doesn't seem to be sinking in. Refer very interested students to the source code for the `<html:form>` tag handler, and also rely on the Periscope demo to show the creation of the global FORM and BEAN objects.

Chapter 6

Although the course title is simply, "Jakarta Struts," the scope really includes effective Web development in general, and this is the first of a few non-Struts topics that we cover. This is of course not an in-depth treatment of the JSTL; we have a separate two-day course that can be recommended as a follow-on to this one. Instead, the chapter has a straightforward survey structure: an overview, including the EL, and one example for each of the four libraries Core, Formatting, SQL, and XML. The Core library is far and away the most useful of these, and should be given the most weight – this is reflected in the lab work as well.





Chapter 7

Internationalization and localization are relevant to many kinds of software – not just Web applications, and not just Struts. Still, Struts integrates i18n so thoroughly that it's really not possible to understand application code or configuration files completely without some grounding in locales, resource bundles, etc. So this chapter is about ½ primer on Java i18n and ½ practical treatment of how Struts uses localized resources in configuration and Java code. This chapter should not take a long time; for many students the Java i18n part will be review, and if time is tight (which it really shouldn't be unless pursuing an abbreviated timeline), Lab 7 is a good candidate for skipping. i18n is quietly threaded into just about every other exercise in the course.

Chapter 8

Validation has become a huge topic within Struts. There are several levels and techniques, plus challenging architectural questions, and so this is one of the longest chapters in the course. (In terms of class time consumed it is the longest.) Give all the sections of this chapter plenty of time, and encourage discussions of tactics and strategy at many points, especially once the Struts Validator has been covered. Lab 8A in particular illustrates many of the issues: it is just complex enough to exceed the declarative capabilities of the Struts Validator for a coarse-grained bean such as MemberBean, and even multi-page validation doesn't quite solve the problem cleanly.

(In the beta offering of this course, in fact, a student exposed a weakness in the answer code that I wasn't aware of: try navigating as far as **Update.jsp**, deleting a required profile field such as age, submitting – which fails validation and returns to the same page – but then changing the password. The validator will continue to complain that age is missing, even though it's not submitted by **ChangePassword.jsp**! Sort of a deep bug, but enough to make the case that multi-page validation is not sufficient to this problem – which is why I've left it in there. If no one notices, you may want to demonstrate this yourself, perhaps during Chapter 10's section on validation best practices.)

Also, make a point of keeping the Struts validation framework well separated from the Struts Validator in discussions. It's all too easy for students to start thinking that the Validator plug-in is the only way to do validation.





Chapter 9

This and the next chapter form a pair, too. This one, as the title suggests, gives a tour of the less-well-known pieces of the API and framework. For most of the topics in this chapter, the intent is just to point out that something exists and to give an idea when it might be used; we're not working in depth the way we have been through the first eight chapters. Plug-ins are the one exception – these are simple and useful ways to extend Struts for an application, and as such the second of the chapter's two labs is the more important. Still, assuming there's enough time, students should do both 9A and 9B.

The section on logging is another Struts-independent one, and as with JSTL it seems sufficiently a part of Web-development best practice that it's worth discussing in this context. For some students this too will be review, but exercises in combining logging with Web applications and Struts plug-ins should be rewarding for everyone.

Chapter 10

Here we rise up out of the tactical muck and discuss design and architecture in more depth. There are three discrete sections. The first one is meant to de-mystify the various possible relationships in Struts configuration. That is, until now students have seen the simpler structures of one view, one mapping to one action forwarding to the next view on "success." Going into this chapter there will likely be a lot of questions remaining for them about what one can and can't do, and this section extends that by challenging each possible multiple relationship and discussing not just can/can't but should/shouldn't. Hopefully a much clearer picture of effective application design strategies should emerge from this, along with some tactical dos and don'ts.

The section on BeanUtils is a bit of a carryover from the previous chapter; it is explored here because it has such an impact on tier design and implementation patterns. The solution is not the most elegant, but it's a major step forward and should suggest to students the sorts of things they could do in building their own base or delegate form-bean classes as part of a larger development effort. The last section revisits validation and essentially leaves students with the assertion that there is no one perfect architecture here; the Struts Validator is a clever but limited tool and application-specific design choices will almost always be necessary.





Chapter 11

The Tiles tag library has been held out of the Struts-tags chapter and deferred until now, and in a way this chapter is of more of a piece with Struts-tags and JSTL than with anything that precedes it more directly. (It is quite possible to cover this chapter right after JSTL, too, if you prefer. The lab starter code includes developments from Chapters 8-10, but these should be transparent for the moment and shouldn't interfere with lab work. But it has been held for last in the book because, something like the JSTL chapter, it provides an introduction/overview where a full multi-day course would be possible. It also introduces a fairly broad new framework of its own – on a par in complexity with the Struts Validator – and is not integral to Struts development as shown in Chapters 7-10. It seems easiest to take a deep breath after Chapter 10 and then dive into this new world of view management.

In my view, the real advantages in using Tiles are (a) basic reuse of content, à la `<jsp:include>` but with the clever inversion of who's doing the inserting and controlling reuse, and (b) the ability to progressively classify view types using definitions. This chapter focuses on those "wins" but stops short of another that has its proponents: using definitions as forward targets in action mappings. I personally don't see this as an attractive choice in most cases – it seems overly centralized and awkward. There are different schools of thought on this, and if you do like this technique, by all means consider extending the chapter or exercises in some way to show that. We'd be interested in hearing any feedback in this area, or in seeing examples you may cook up.





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, those steps having been followed, a classroom machine is failing to run demos or lab answers correctly, here is a list of items to consider and to double-check:

- If the Ant build for Love is Blind fails to find the driver and data-source classes, most likely `MYSQL_DRIVER` is not set correctly to the ConnectorJ home. If the application builds but can't find the data source at runtime, most likely the Tomcat classpath does not correctly include the ConnectorJ home.
- The technique for application deployment in this course is a little different from that used in either our servlets or JSP/JSTL courses. Servlets builds and copies a WAR file; JSP and JSTL define a single context each for the entire installed module, to simplify rapid re-testing of JSPs edited in-place. This course builds any given project to a **Build** directory and then deploys a context XML file to Tomcat pointing to that directory. This is reliable and is explained in the student guide, but it's a good idea to get familiar with the basic build structure in **build.xml** and **StrutsTargets.xml**. Note especially that **ant all** or **ant deploy** is necessary to redirect Tomcat to the build directory for a specific project space the first time that project is used. For instance, there are 26 versions of Love is Blind in the module (!), but they all deploy to the same context, so that they can be tested with the same URLs. If the generated context XML file is not copied to Tomcat's config directory, students will be coding in one place and testing in another! If a student is getting strange results, the first thing to check is: is the context file in place and pointing to the right project workspace? Look in Tomcat's **conf\Catalina\localhost** directory for **LevelsBlind.xml**, **Ellipsoid.xml**, etc., and check the **docBase** attribute there.
- On slower machines, Tomcat startup can take a minute or so, especially once multiple applications have been built and deployed. The technique mentioned above should make it unnecessary to restart Tomcat very often, but you might want to let students know that between runs of the server, context files for applications not in use can be safely deleted from **conf\Catalina\localhost**, thus saving several seconds of startup time each.
- Be sure that the **root** user's password has been set to **root** on MySQL, or many JDBC exercises will fail. This is detailed in the setup guide.





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

