Capstone Courseware, LLC
1 Washburn Place
Brookline, MA 02446

877-227-2477
capstonecourseware.com

Introduction to Spring

Version 3.1

Instructor's Guide

Overview

This course provides an overview of Spring, wandering onto the territory of web applications, persistence, AOP, and other features; but it then settles into a nuts-and-bolts treatment of the Core module, and that mostly means understanding how to declare **<bean>**s in the configuration of a bean factory or application context. This course is offered standalone and also commonly included in other courses that then go on to cover other topics such as web applications or persistence techniques.

We try to separate out several closely related concepts, including inversion of control vs. dependency injection and instantiation vs. configuration vs. assembly – and then follow a progression from the simpler to the more complex of these sets. So we don't really do dependency injection proper until Chapter 4, having first nailed down basic bean declarations. There is some blurring of the lines with instantiation vs. configuration, but only for practical reasons: that is, it's hard to do anything of real interest if we can't even configure a simple-value property such as the dimensions of a shape or a filename for an XML transformer.

However important these conceptual distinctions may be to you, or to your students, they definitely serve to organize the coursework, such that by the end of Chapter 4 students should be able to use Spring to instantiate, configure, and assemble pretty much any graph of JavaBean instances that they like. Chapter 5 covers some intermediate techniques such as configuring collections and using the **util:** schema. Chapter 6 then jumps to a separate topic, though still one that's within the bounds of the Spring Core module, which is validation.

Timeline

The following breakdowns are approximate, and every class will vary.

Day 1

2 hours	Chapter 1
2½ hours	Chapter 2
2 hours	Chapter 3

Day 2

2 hours	Chapter 4
2½ hours	Chapter 5
2 hours	Chapter 6

If you are short on time, one option before cutting content outright is to do either the optional part of Lab 3A or all of Lab 3B, but not both. Otherwise, going lighter on some of the more adventurous configuration techniques in Chapters 3, 4, and 5 will save some time, and while validation is important it is cleanly severable from the rest of the course and so cutting or scaling back Chapter 6 makes sense as well

Teaching Notes

I hope that the coursebook is sufficiently clear and detailed, and so the following notes just capture a few ideas about how to approach a given topic, additional concepts to add to your lectures and discussions, and any surprises you might encounter.

Chapter 2

An interesting side note that you might make during or after the Singleton vs. Singleton exercise: in order to enforce singularity (or prototype behavior) even over the rules set by a Java developer, Spring has to carry out some privileged actions. For this to work, either there must be no security manager in place, or it must have a policy that allows the Spring codebase some special permissions. A rough list is here:

```
permission java.io.FilePermission "", "read";
permission java.io.FilePermission "-", "read,write";
permission java.util.PropertyPermission "*", "read";
permission java.net.SocketPermission
    "www.springframework.org:80", "connect";
permission java.lang.RuntimePermission "accessDeclaredMembers";
permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
```


Chapter 4

This chapter introduces two features that may land with a bit of a thud in some classes: autowiring, and auto-detection of annotated component classes. The previous chapter did introduce component scanning, so that we could consider **@PostConstruct** and **@PreDestroy** hook methods. But now we look at both of these features in detail. Autowiring is controversial: used heavily by many in the Spring community but reviled and labeled outright bad practice at many companies. We try not to advocate one way or the other here, but simply to cover the feature completely and accurately.

Component scanning and annotation processing is a different choice entirely, although this and autowiring are often practiced together. Like autowiring, component scanning seems to tear away some of the declarative benefit of Spring; annotation processing is another matter but where the kingpin annotation seems to be **@Autowired** we do wind up in the same boat. It might be worthwhile to point out that other sorts of annotation post-processors can enable features that even the anti-autowire crowd may find more appealing, such as **@RequestMapping** for the web tier or **@Transactional** for DAOs.

Chapter 5

One note on the Policy demo: to accommodate the different concrete paths at which the code is deployed, code in other courseware modules often rely on the **CC_MODULE** environment variable, and various beans will on their own initiative expand the phrase "${env.CC_MODULE}" found in their property values by looking up this variable. But that seems a bit much for this exercise, and so the exact path of the code source must be entered when configuring the policy. This is simple enough and the path is shown in the demo instructions. But the demo answer will have a path from the **Examples** tree, so any copy-and-paste solution will need some additional editing. It also means that course deployments to non-standard roots (other than **c:\Capstone**) will require editing of the XML file, even for the answer code. Finally, note that this syntax is demanding: the trailing slash matters, and even incorrect case for the drive spec ("C" vs. "c") will result in a failure to match the permission. Read and type carefully!

Revision History

Version 3.1 upgrades this course to Spring 3.1. Major changes include:

- Updated the lab infrastructure to current Capstone standards: especially, there is an **Admin** directory with batch files that simplify environment setup and have generally saved a lot of time on day one as students get up and running more quickly.
- Replaced all uses of the now-deprecated **XmlBeanFactory** with **DefaultListableBeanFactory**.
- There is a new lab track in which the Bank application plays a larger role, with digital signature of application inputs configured in Spring. This serves as an alternative to the XML-Transformer track, which works very well but for some groups has shown to be a bit too much too soon.

Version 2.5 upgrades this course to Spring 2.5, and at this point we also reworked the modular structure of our Spring courseware such that this course has been reduced in scope to cover just the Core module. Major changes include:

- We moved to use of Ant for all projects in this course, where in prior versions of this material there was a DOS **build.bat** script for building. This is a stronger structure, maybe overkill for these simple applications but it then makes for smoother transitions to later material in our other courses.
- What used to be four chapters has been broken into six, with the old chapters 2 and 3 each split into two smaller units. The overall timeline is not much affected by this.
- There are new sections on lifecycle hooks, component scanning, and annotation processing.
- The Eclipse overlay has been updated to Ganymede and WTP 3, and also provides configuration files for use by Spring IDE 2.2.

Version 2.0.1 fixes a handful of typos and omissions in the book. There are no changes to the lab code, but the Eclipse overlay is updated from the old style to the new one that supports WTP 1.5.

Version 2.0 is the initial release of the course.

Troubleshooting

If you run into any trouble with code exercises, the first and best thing to do is to double-check that the classroom machines have been set up precisely according to the course setup guide. Especially, the wrong version of a tool can cause significant problems; don't wander off-book in this way unless absolutely sure you can support the software that you prefer and that we haven't tested. Check environment variable settings carefully, too; these are the cause of a great many classroom glitches.

Errata

At this time there are no errata for this version of the course.

Feedback

We very much appreciate whatever feedback we can get on our courseware – especially from the instructor's perspective. Naturally, the more specific, the better, and we strongly encourage you to make notes on issues you may encounter in the classroom, whether they're typos, missing files, or suggestions for clearer language to explain a concept. We can't guarantee that we'll act on every suggestion, but we're aggressive about stamping out problems and try to be highly responsive. Hopefully this means that when you give us good feedback, you get a better course the next time you need to teach it.

Please direct all courseware feedback to

Will Provost
Capstone Courseware
mailto:provost@capcourse.com
877-227-2477

For anyone who's interested, we have a very informal defect-tracking system, based in Excel spreadsheets with columns to capture defect location, nature, status, and author feedback. Ultimately, feedback goes into these sheets, so if you want a template, we'll be happy to provide one, to facilitate the reporting process.