Test-Driven Development with Spring and Hibernate

Matt Raible
matt@raibledesigns.com

Agenda ~ Part I

- Introductions
- [Lab] Setup machines: JDK, Ant, AppFuse, Eclipse, MySQL and Tomcat
- Overview of J2EE, Spring and Hibernate
- AppFuse: History, Overview and Demo

Agenda ~ Part II

- [Lab] Creating database tables using Ant and Hibernate
- JUnit and Test-Driven Development
- Review of interface vs. implementation classes
- [Lab] Writing a WeblogDAOTest JUnit test
- Overview of Spring's DAO Support classes for Hibernate 3
- [Lab] Write WeblogDAO interface and WeblogDAOHibernate implementation

Agenda ~ Part III

- Business Facade: What is it and why should I use one?
- [Lab] Writing a WeblogManagerTest with jMock
- [Lab] Writing a WeblogManager interface and implementation
- [Lab] Defining the “weblogManager” bean and per-method transaction attributes
- Alternative I: Generic DAO and Manager
- Alternative II: Code Generation

Agenda ~ Optional

- Pick your favorite Java web framework:
    - JSF, Struts, Spring MVC, Tapestry or WebWork
- Testing Controllers (or page backing objects) for selected framework
- Testing the UI with Canoo WebTest and Ant

Part I ~ Setup

Introductions

- Your experience with webapps?
- You experience with J2EE?
- What do you want to get from this tutorial?
- Open Source experience with Ant, XDoclet, Hibernate, Spring, Eclipse/IDEA?

Who is Matt Raible?

- Developing websites since 1994 (before Netscape 1.0) - Developing in Java since 1999
- Committer on several open source projects: Roller Weblogger, XDoclet, Struts Menu, Display Tag, AppFuse
- J2EE 5.0 and JSF 1.2 Expert Group Member
- Author: Spring Live (SourceBeat) and contributor to Pro JSP (Apress)

Environment Setup

- Download and install:
  - Ant 1.6.2
  - MySQL 4.1.x (or 5.0.x)
  - Tomcat 5.5.7
  - AppFuse 1.8 RC1
  - Eclipse 3.1

J2EE, Spring and Hibernate

- EJB 2.1: difficult to test, overkill for most applications
- JDBC: closing resources, checked exception, cumbersome to populate POJOs
- Hibernate: simplifies persistence and makes working with POJOs easy
- Spring: runtime exceptions, de-couples dependencies, simplifies APIs (JNDI, etc.)

AppFuse - what is it?

- A directory structure, build file and project classes to get your project started quickly
- The hard part is getting started and configuring dependencies
- Uses popular open-source tools: Ant, XDoclet, Spring, Hibernate, Struts (or JSF, Spring MVC, WebWork or Tapestry)
- Top 5 java.net project in hits, accesses and mail traffic

History

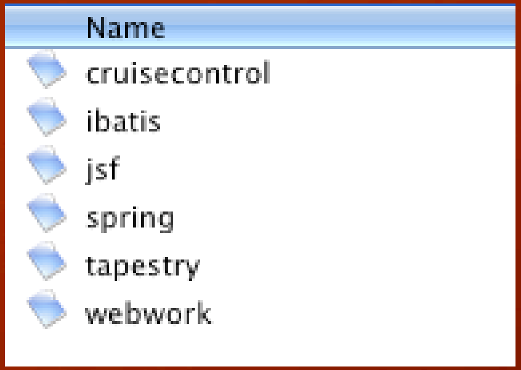
- Started as a sample app for Pro JSP
- Became a toolkit for starting new projects
- Lots of community feedback makes it a "best practices" webapp for J2EE
- Documentation and Tutorials written November 2003
- AppGen - CRUD made easy

Demo of Features

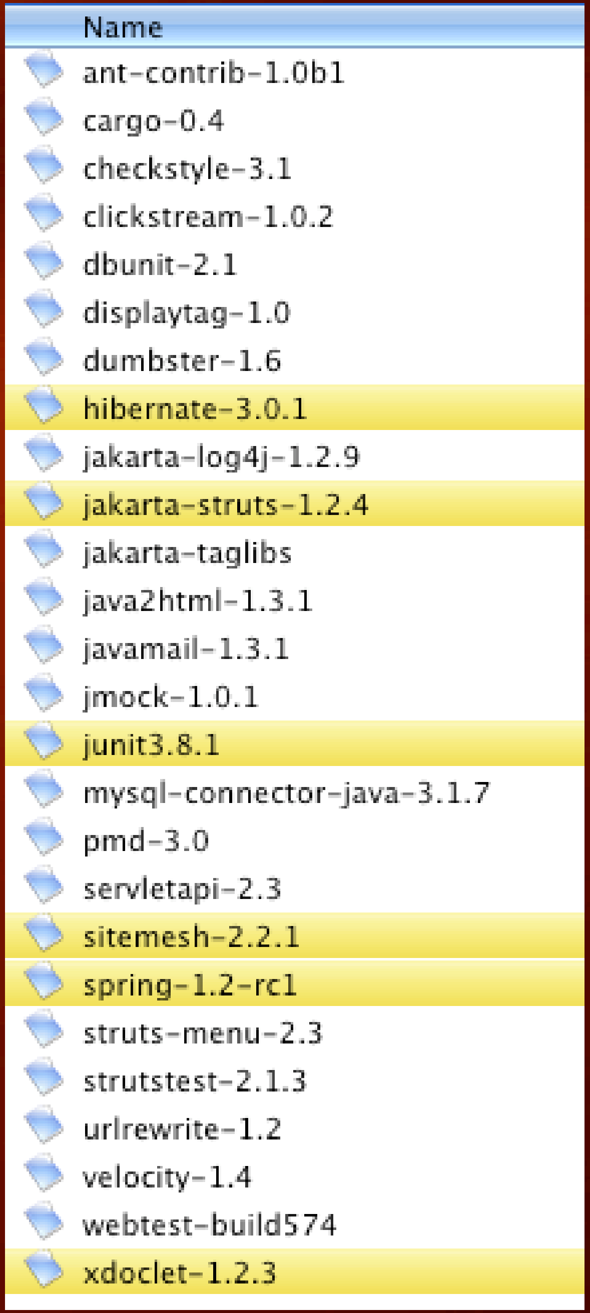
- Acegi Security - makes security more portable between containers
- Remember Me and Self Registration
- GZip Compression Built-in
- Testing environment ready to go, many tutorials, CruiseControl files included
- http://demo.raibledesigns.com/appfuse

Dependencies

Optional Installs

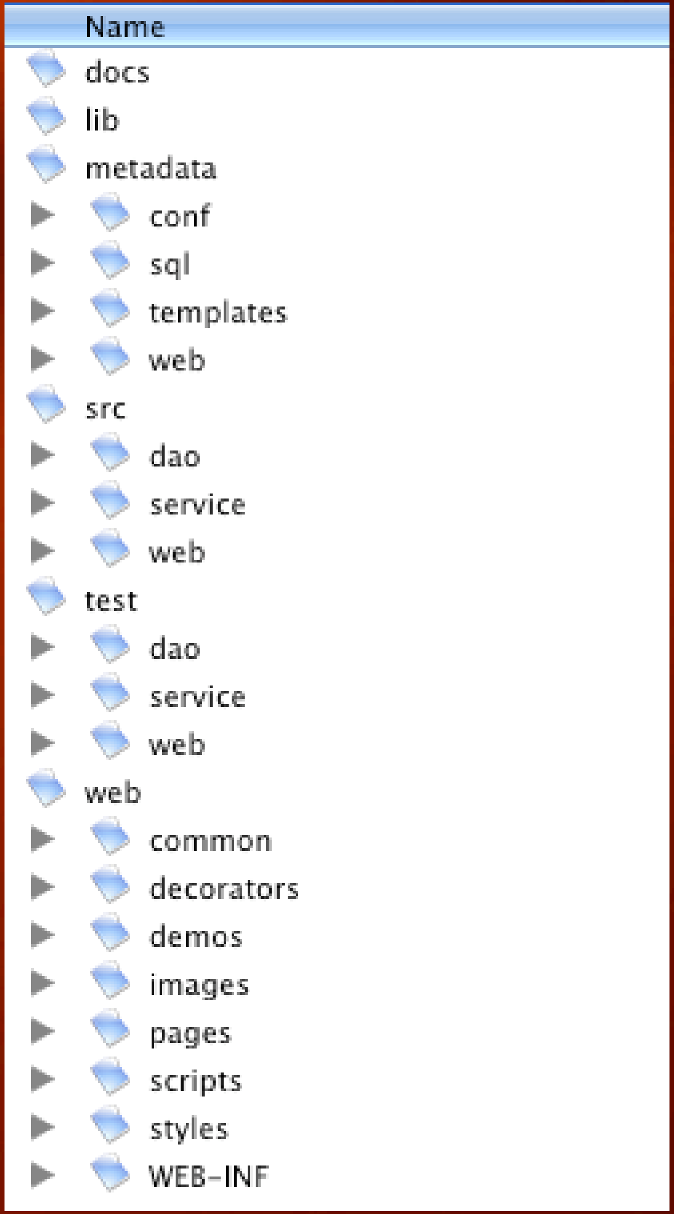


Name
cruisecontrol
ibatis
jsf
spring
tapestry
webwork



Name
ant-contrib-1.0b1
cargo-0.4
checkstyle-3.1
clickstream-1.0.2
dbunit-2.1
displaytag-1.0
dumbster-1.6
hibernate-3.0.1
jakarta-log4j-1.2.9
jakarta-struts-1.2.4
jakarta-taglibs
java2html-1.3.1
javamail-1.3.1
jmock-1.0.1
junit3.8.1
mysql-connector-java-3.1.7
pmd-3.0
servletapi-2.3
sitemesh-2.2.1
spring-1.2-rc1
struts-menu-2.3
strutstest-2.1.3
urlrewrite-1.2
velocity-1.4
webtest-build574
xdoclet-1.2.3

Directory Structure



```
Name
docs
lib
metadata
  ▶ conf
  ▶ sql
  ▶ templates
  ▶ web
src
  ▶ dao
  ▶ service
  ▶ web
test
  ▶ dao
  ▶ service
  ▶ web
web
  ▶ common
  ▶ decorators
  ▶ demos
  ▶ images
  ▶ pages
  ▶ scripts
  ▶ styles
  ▶ WEB-INF
```

Part II ~ Testing DAOs

Create database & table

- Create new project and create database with Ant
- Create Weblog.java POJO and generate Hibernate mapping file with XDoclet
- Configure Spring to be aware of Weblog object
- Create "weblog" table from POJO using Ant

JUnit and Interfaces

- TDD makes you think before you code
- JUnit is easiest to test with because of plethora of tools and examples
- Writing interfaces allows de-coupling of layers and implementation
- Write code to test interface (integration test) or implementation (unit test)

Let's write some code!

- What is a DAO?
- Create WeblogDAOTest - Test First!
- Create WeblogDAO Interface
- Create WeblogDAOHibernate implementation
- Create "weblogDAO" bean definition in Spring context file
- Run JUnit Test

Spring simplifies testing

- Spring's org.springframework.test package has a number of base classes to simplify testing
    - AbstractDependencyInjectionSpringContextTests - can do both setter or field-based dependency injection, cached context files.
    - AbstractTransactionalDataSourceSpringContextTests - exposes a JdbcTemplate for easy querying and rolls back any data entered into the database.

Part III ~ Testing with Mocks

Business Facades

- Encapsulates business logic for multiple clients
- Similar to Business Delegate pattern for EJBs
- Provide client-friendly, transactional access to data layer
- Facilitates calling multiple DAOs within the same transaction
- Can be exposed as web services

Yee haw ~ more code!

- Create WeblogManagerTest ~ a true unit test that uses jMock for mocking dependencies
- Create WeblogManager Interface
- Create WeblogManagerImpl implementation
- Run JUnit Test
- Create "weblogManager" bean definition

So this is better, huh?

- Created 7 classes, modified 2 files, generated 1
- Faster than traditional EJB/JDBC, but still painful
- Faster options:
    - BaseDAO/Manager classes in AppFuse for Generic CRUD on any POJO
    - Generate/modify files with AppGen or AppFuse Generator

AppGen

- To create CRUD for a table, it required you create 11 files and modify 5
- AppGen requires you create 1 and modify 1
- Uses BaseHibernateDAO and BaseManager for generic CRUD methods
- Still requires you to “pretty up” the UI

Part IV ~ Web Development

Creating the UI

- Pick your favorite Java web framework:
  - JSF, Struts, Spring MVC, Tapestry or WebWork
- [Lab] Generate UI classes and pages using AppGen
- Review controller (or page backing object) tests for selected framework
- Testing the UI with Canoo WebTest and Ant

End of Coding

- Any code tweaks you'd like to see?
- Deploying to production
  - Setup MySQL with create-tables.sql
  - Setup Tomcat with JDBC Drivers
  - Deploy using Tomcat's Manager application
  - Hosting: kgbinternet.com, contegix.com

AppFuse Roadmap

- Continue to try and make IDE integration easier
- Support/Documentation for more app servers
- iBATIS Tutorials, refactor build/test process
- Other things you'd like to see?

Equinox

- AppFuse Light - designed for quick apps with few requirements (i.e. prototypes)
- No build-time dependencies (i.e. XDoclet), no out-of-the-box security
- Includes 5 MVC implementations: JSF, Spring MVC, Struts, Tapestry and WebWork
- Includes 5 Persistence frameworks: Hibernate, iBATIS, JDO, OJB, Spring JDBC
- Located at http://equinox.dev.java.net

Questions?

- AppFuse Official Homepage:
  - http://raibledesigns.com/appfuse
- AppFuse Demo:
  - http://demo.raibledesigns.com/appfuse
- This Presentation:
  - http://appfuse.dev.java.net/TDDWithAppFuse.pdf