

Getting started with Struts

Part I - Building a (free) development environment



Introduction

- ◆ Advisory Developer - delta.com
- ◆ 12 years experience
- ◆ Sun Certified Programmer and Web Component Developer for the Java™ 2 platform
- ◆ Open source developer – CacheCow, Edgar, Commando, WebDoctor
- ◆ Interests include Java, Linux, web development
- ◆ Maintains blog at jason.blog-city.com



Contents

- ◆ The fundamentals
- ◆ Designing our development environment
- ◆ Building our development environment
- ◆ Using our development environment – basics
- ◆ Demo
- ◆ Using our development environment – Struts app development
- ◆ Demo
- ◆ Alternatives
- ◆ Demo

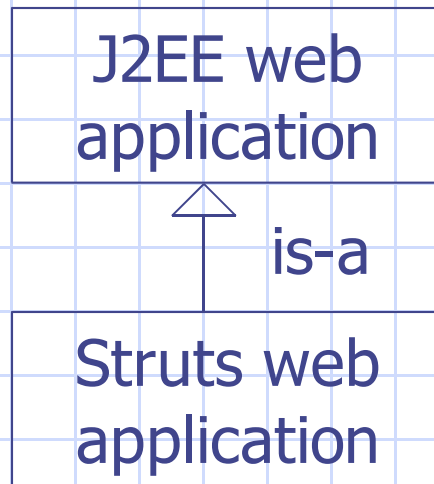




The fundamentals



What is a Struts application?



What is a Struts application?

◆ What is a J2EE web application?

- web.xml, Java classes, servlets, images, content, JSPs, JARs, custom tag libraries

◆ What is a Struts application?

- J2EE web application +
- struts-config.xml, Actions, Resource bundles, Struts tags, libraries

◆ Understanding a J2EE web application is half the battle

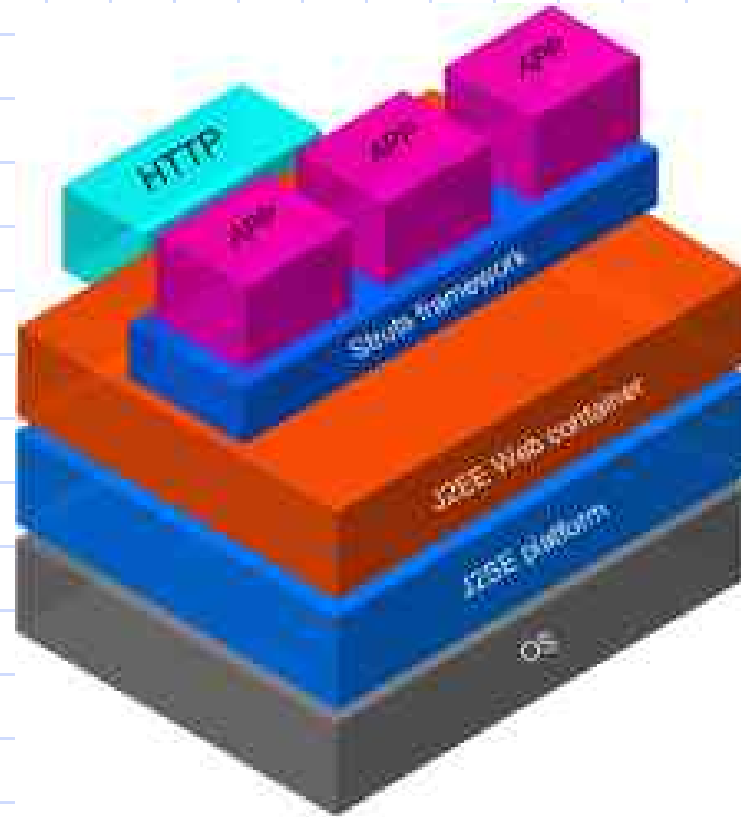
◆ Can leverage web application development environment for Struts application development

◆ If new to J2EE web applications, check out Hunter's Java Servlet Programming book from O'Reilly

<http://www.oreilly.com/catalog/jservlet2/>



What is a Struts application?



What is a development environment?

- ◆ An integrated suite of tools to aid the development of software
- ◆ Without it, we can't do our job
- ◆ Worth spending the time to design build and manage your development environment
- ◆ A good development environment will make you more productive
- ◆ A bad development environment combined with poor process planning can cause significant project delays in the long run
- ◆ Takes time, patience and effort to design & build – but is well worth the effort!
- ◆ Review regularly – every six months



Devenv design considerations

- ◆ “But it works in dev!”
- ◆ Compatibility with test and production environments
- ◆ In-expensive
- ◆ Flexibility – customizable by developer
- ◆ Company standards – sometimes you don’t have a choice



It's just Java

- ◆ The good thing about Java is you have a choice
- ◆ The bad thing about Java is you have a choice
- ◆ The selections I make for my development environment, may not work for you or your organization





Designing our development environment



Shopping list – bread and water

- ◆ An OS which is capable of supporting Java
- ◆ A Java Development Kit
- ◆ Web container
- ◆ A machine with as much capacity as you can afford



Shopping list – cheesecake

- ◆ An IDE – highly recommended
- ◆ An XML editor – highly recommended (sometimes included with the IDE)
- ◆ Build tool – highly recommended
- ◆ Unit testing tool – highly recommended
- ◆ Version control – highly recommended even if you are the only developer
- ◆ Modeling tools – recommended
- ◆ Load testing/QA tools – essential.. but more for the test environment and not for the development environment



Which OS for development?

- ◆ Windows (latest 1.4.2_01)
 - ◆ Linux (latest 1.4.2_01)
 - ◆ Solaris x86 (latest 1.4.2_01)
 - ◆ Mac OS X (latest 1.4.1)
 - ◆ FreeBSD (latest 1.3.1)
-
- ◆ Recommendation – most develop on Windows with an increasing number developing on Linux and OS X
 - ◆ Tip – give Linux a try if you haven't already. You are going to see a lot more Java on Linux deployments over the coming years. Check out <http://java.sun.com/linux/>



Which JVM/compiler/JDK?

- ◆ Too numerous to mention
 - ◆ Windows – Sun, IBM, BEA
 - ◆ Solaris - Sun
 - ◆ OS X – Apple
 - ◆ Linux – Sun, Blackdown, BEA, IBM
 - ◆ FreeBSD – FreeBSDFoundation.com
- ◆ Recommendation – use the Sun JDK unless it's not available on your platform. It's good enough for most needs. Make sure the development JDK matches what you plan to use for test & production.



Which web container?

- ◆ Too numerous to mention
 - ◆ Tomcat, Orion, WebLogic, WebSphere, Jetty, JRun, ServletExec...
 - ◆ Some free, some commercial, some free for development
- ◆ Recommendation – Tomcat is good enough for most developer needs. Has a large user base + available resources. Make sure it implements the same spec levels (JSP/Servlet) as your container in production.



Which IDE?

- ◆ Too numerous to mention
 - ◆ Eclipse, NetBeans, IntelliJ IDEA, JBuilder...
 - ◆ Notepad + DOS prompt + javac ... p-l-e-a-s-e!
 - ◆ Some free, some commercial
- ◆ Recommendation – Eclipse because it's free, has large user base, plug-in availability, SWT based native GUI, nice refactoring support, integrates well with Ant, JUnit, CVS - and because I like it!



Which build tool?

- ◆ Several to choose from:
 - ◆ Make, Ant, Maven
- ◆ Recommendation – Ant should meet most of your needs. Ant also has good integration with most IDEs. Keep an eye on Maven. Forget about Make - not optimal for Java
- ◆ Tip – integrate early, integrate often – hourly builds vs. nightly builds. Tools such as Cruise Control can help out with continuous integration.



Which unit testing tool?

- ◆ Several to choose from:
 - ◆ JUnit, JTest, StrutsTestCase
- ◆ Recommendation – JUnit is fine. Nice integration with IDEs and with Ant. As you get more into Struts, consider looking at StrutsTestCase.
- ◆ Tip – Consider Test Driven Development approach. Automate unit testing by weaving test execution into the build process.



Which version control tool?

- ◆ Several to choose from:
 - ◆ None, CVS, Subversion, ClearCase, BitKeeper, SpectrumSCM...
- ◆ Recommendation – CVS will probably meet your needs for basic version control. Keep an eye on Subversion. SpectrumSCM looks interesting – plus they are a local company! Linus apparently uses BitKeeper for Linux kernel development. Use something unless you are truly writing disposable throw-away code and you are the only developer. CVS runs best on *NIX although there is an NT version available.



Final ingredients

- ◆ Windows XP*¹
 - ◆ JDK: JDK 1.4.2_01 – java.sun.com
 - ◆ Web container: Tomcat v5 – jakarta.apache.org
 - ◆ Build tool: Ant 1.5.4 – ant.apache.org
 - ◆ Unit testing tool: JUnit 3.8.1 – junit.org
 - ◆ IDE: Eclipse 3.0M3 – eclipse.org
 - ◆ IDE plug-ins: Sysdeo Eclipse Tomcat Launcher plug-in 2.1.1 – sysdeo.com
 - ◆ Modeling tool: none
 - ◆ Version control: none
 - ◆ Cost*²: \$0
- ◆ And the icing on the cake: Struts 1.1 - <http://jakarta.apache.org/struts/>

*1 - I use SuSE 8.1 at home for development

*2 - does not include hardware costs and/or OS license costs



Side dishes to consider

- ◆ Struts Console - <http://www.jamesholmes.com/struts/console/>
- ◆ Easy Struts - <http://easyststruts.sourceforge.net/>
- ◆ Too numerous to mention...
- ◆ <http://jakarta.apache.org/struts/resources/tools.html>

*1 - I use SuSE 8.1 at home for development

*2 - does not include hardware costs and/or OS license costs





Building our development environment



Installation tips

- ◆ Install from the ground-up starting with the JDK
- ◆ Centralize all component installations where possible e.g. packages folder/directory
- ◆ Test every step of the way
- ◆ Consider putting component under version control
- ◆ Configuration files requiring changes e.g. server.xml should be placed under version control



Installation order

- ◆ JDK
- ◆ Ant
- ◆ JUnit
- ◆ Tomcat
- ◆ Eclipse
- ◆ Eclipse plug-ins
- ◆ Struts



Special steps for Tomcat

- ◆ Test installation – <http://localhost:8080>
- ◆ Need to setup a manager account to support hot deployments – disabled by default for security reasons
- ◆ Add something like the following to `$CATALINA_HOME/conf/tomcat-users.xml`

```
<user username="manager" password="manager" roles="manager"/>
```
- ◆ Restart Tomcat then test the manager - <http://localhost:8080/manager/html> and logging in using the above credentials
- ◆ More info – check out <http://localhost:8080/tomcat-docs/manager-howto.html>



Special steps for Eclipse

- ◆ Install Eclipse first by unzipping the download (Tip: on Linux, use unzip command – don't use jar xvf)
- ◆ Test the installation
- ◆ Shutdown Eclipse
- ◆ Extract the Tomcat plug-in from Sysdeo into the plug-ins directory of the Eclipse installation
- ◆ Restart Eclipse
- ◆ Verify the plug-in was correctly installed: Help->About Eclipse Platform->Plug-in details. Look for SYSDEO
- ◆ If you are new to Eclipse, check out this quick tutorial <http://www.onjava.com/pub/a/onjava/2002/12/11/eclipse.html>

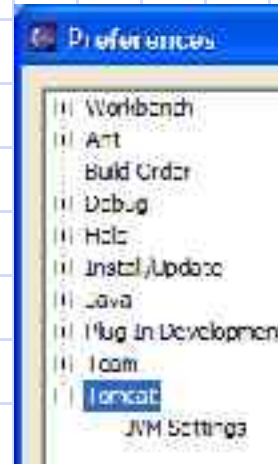


Configure the Sysdeo Tomcat plug-in

- ◆ Window->Preferences
- ◆ Click on Tomcat
- ◆ Fill out the form to point the plug-in to your Tomcat installation
- ◆ Switch to the Java perspective
- ◆ You should see the Tomcat icons and a Tomcat menu



- ◆ Click to  start Tomcat



Ant/Eclipse integration

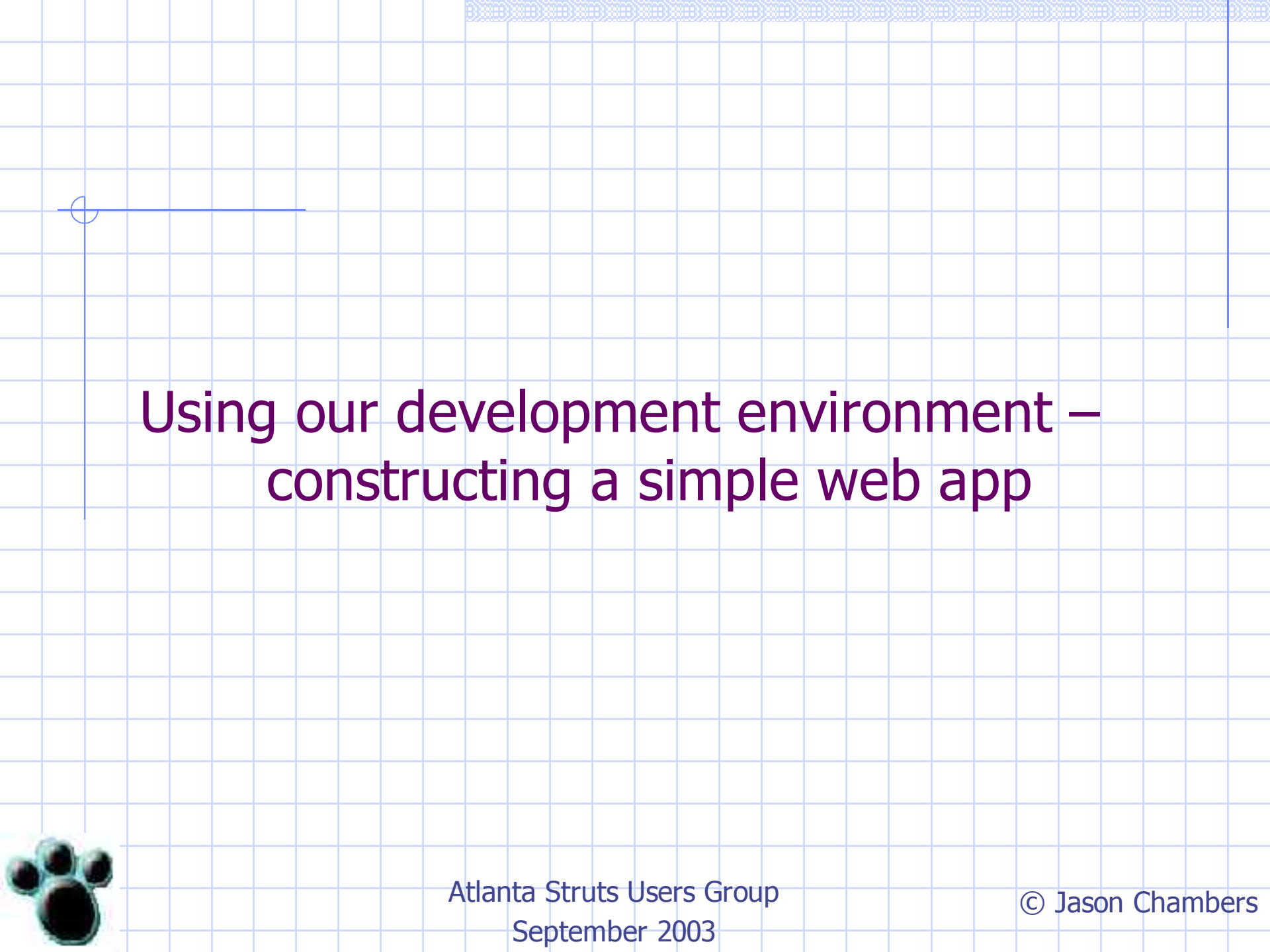
- ◆ Eclipse 2.1.1. comes with Ant 1.5.3. pre-installed
- ◆ It is possible to upgrade Eclipse to use a different version of Ant.
- ◆ Recommend you do this only when you start running into problems
- ◆ Eventually (sooner rather than later) you will want to automate the builds which means you will want to use Ant from outside of the Eclipse environment.
- ◆ If you start having build behavior differences between your build in the IDE and automated Ant build, consider synching the Ant versions.



JUnit integration

- ◆ JUnit with Eclipse
 - ◆ Eclipse 2.1.1. comes with JUnit 3.8.1. pre-installed
 - ◆ JUnit integration with Eclipse is first class
 - ◆ No further work to be done
- ◆ JUnit with Ant
 - ◆ Eventually (sooner rather than later) you will want to automate your unit tests
 - ◆ Weave unit testing into the build process
 - ◆ To integrate JUnit with Ant, checkout <http://ant.apache.org/manual/OptionalTasks/junit.html>





Using our development environment – constructing a simple web app



Building your first web app

- ◆ How do I organize my code? Where do I store content? What about config files?
- ◆ Recommendation – don't develop directly under the Tomcat installation (webapps directory)
- ◆ Keep development separate from deployment
- ◆ Deployment structure enforced by J2EE Servlet specification
- ◆ Development structure of your web application is largely up to you
- ◆ Recommendation – refer to the Tomcat Application Developer's Guide



Application structure (dev)

- ◆ Tomcat Application Developer's Guide recommendations
<http://localhost:8080/tomcat-docs/appdev/source.html>
- ◆ Each application has it's own directory
- ◆ Under each application:
 - docs/ - Documentation for your app
 - src/ - Java source files organized into packages
 - web/ - View components – HTML, JSP, CSS, JPG JS etc.
 - web/WEB-INF - Deployment descriptor, TLDs etc.
 - build.xml - Ant build file
 - build.properties - Ant build properties
- ◆ Create a new (Java) project using Eclipse and create the above directories
- ◆ Recommendation: Put this tree under version control



Structure comparison



Application structure -
development



Application structure -
deployment



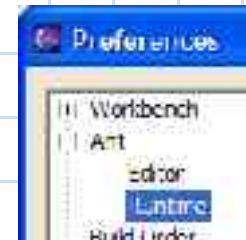
Add a dash of Ant (1)

- ◆ Using the structure enables you to use the template build.xml
<http://localhost:8080/tomcat-docs/appdev/build.xml.txt>
- ◆ In Eclipse, create a new file called build.xml in the project/application root
- ◆ Copy/paste from the template into your build.xml
- ◆ In Eclipse, create a new file called build.properties in the same location as build.xml
- ◆ In build.xml:
 - ◆ set "app.name" property with your app name
 - ◆ set "catalina.home" property to point to the Tomcat installation
- ◆ In build.properties
 - ◆ manager.username=manager
 - ◆ manager.password=manager



Add a dash of Ant (2)

- ◆ Need to tell Eclipse's Ant about the Catalina Ant tasks to enable you to control deployment of your application to Tomcat from Eclipse
- ◆ Eclipse: Windows->Preferences
- ◆ Add `$CATALINA_HOME/server/lib/catalina-ant.jar` to the Ant Runtime Classpath



A very simple web app

- ◆ Add a new file called hello.jsp in web/ contents:

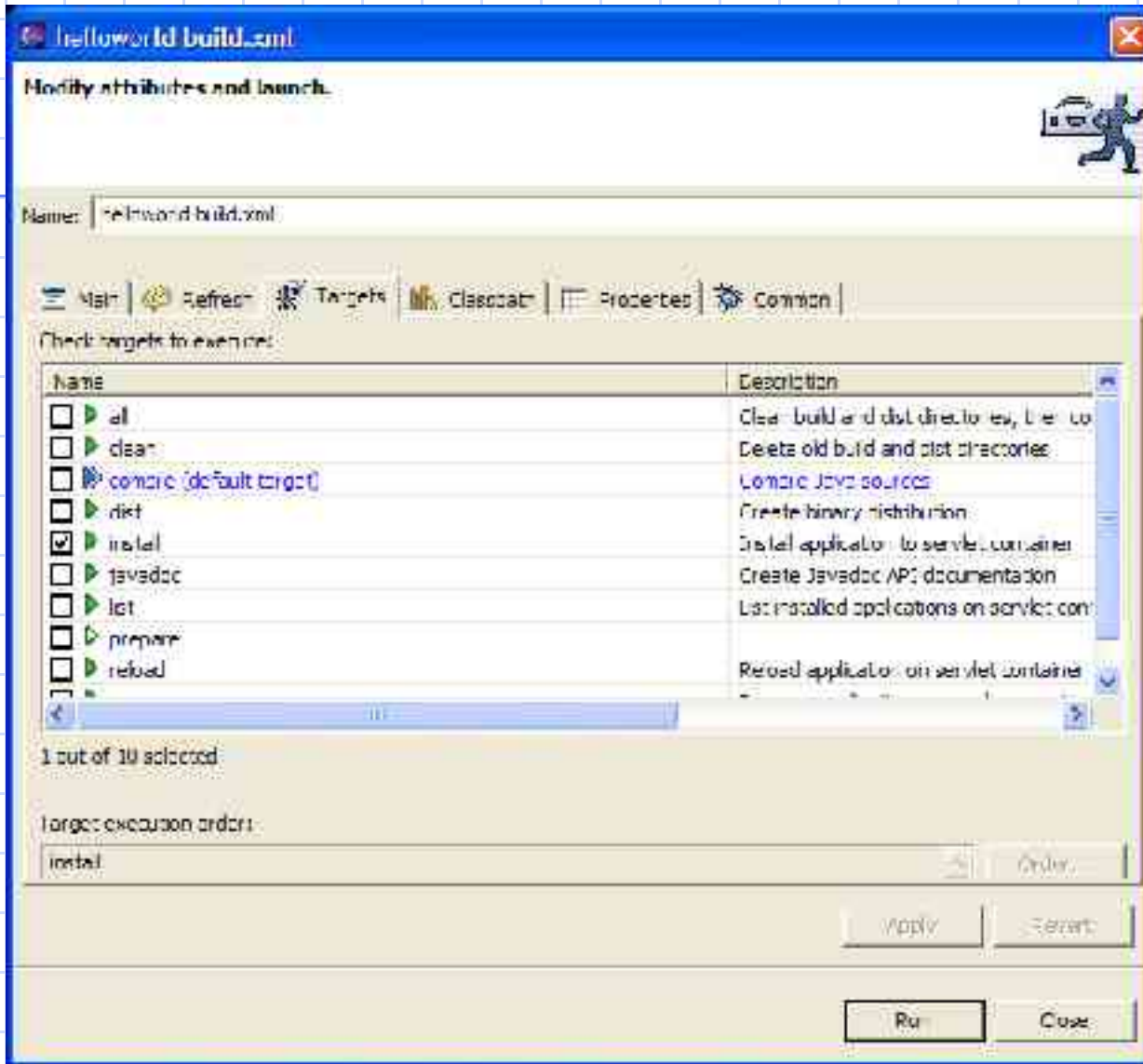
```
<html>
<head>
</head>
<body>
Hello there. The time is <%= new java.util.Date() %>
</body>
</html>
```



A taste test

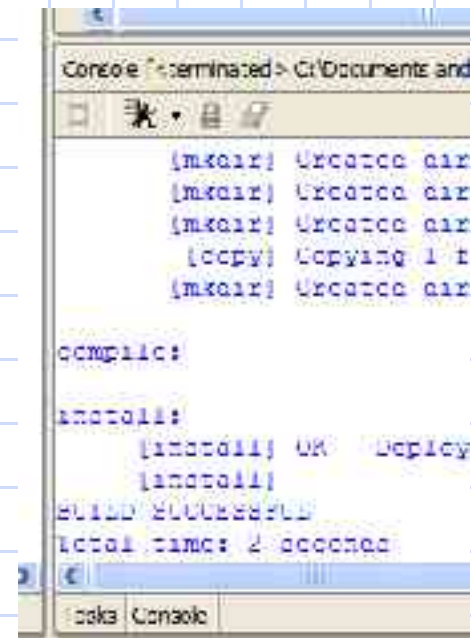
- ◆ Ensure Tomcat is running
- ◆ Locate your build.xml
- ◆ Right mouse-click
- ◆ Select "Run Ant"
- ◆ Check the install target
- ◆ Click Run





A taste test

- ◆ Look for BUILD SUCCESSFUL
- ◆ Point your browser to <http://localhost:8080/helloworld/hello.jsp>
- ◆ For subsequent changes, use the reload target



A screenshot of a console window titled "Console [terminated] - C:\Documents and Settings\...". The window shows the following output:

```
[mkdir] Created dir  
[mkdir] Created dir  
[mkdir] Created dir  
[copy] Copying 1 fi  
[mkdir] Created dir  
  
compile:  
  
install:  
[install] OK Deploy  
[install]  
BUILD SUCCESSFUL  
Total time: 2 seconds
```

The console window has a "Tasks Console" tab at the bottom.





Demo time





Using our development environment – constructing a Struts web app



Add a dash of Struts for good measure

- ◆ Unzip the Struts installation
- ◆ Copy struts-documentation.war into \$CATALINA_HOME/webapps
- ◆ To access Struts documentation locally, <http://localhost:8080/struts-documentation>



Install struts-example into our devenv

◆ Setting up the Eclipse project

- ◆ Extract the files from struts-example.war
- ◆ In Eclipse, create a new project as before (src, web, web/WEB-INF structure)
- ◆ File->Import into src/ from the WEB-INF/classes directory of struts-example
- ◆ Right mouse click struts-example->Properties
- ◆ Add External JARs struts.jar (Struts), servlet-api.jar (Tomcat) etc.
- ◆ Keep adding until you get no more compilation issues
- ◆ The list should reflect static dependencies only – everything you need to compile not necessarily everything you need to run the application (dynamic dependencies)
- ◆ Do not copy dependencies, refer to their correctly installed location
- ◆ Plan to copy dependencies at deploy-time (assembly process)
- ◆ File->Import into web/ the view components from struts-example (JSPs etc.)
- ◆ File->Import into web/WEB-INF the xml config files from WEB-INF (web.xml, struts-config.xml etc.)

* - I use SuSE 8.1 at home for development



Install struts-example into our devenv

◆ Setting up Ant

- ◆ In Eclipse, add tools.jar to the Ant run-time properties
- ◆ Import build.properties and build.xml as before
- ◆ build.properties:

```
manager.username=manager  
manager.password=manager
```

```
lib.dir=C:/Documents and Settings/Jason/My Documents/packages
```

```
# Struts  
struts.version=1.1  
struts.home=${lib.dir}/jakarta-struts-${struts.version}
```

* - I use SuSE 8.1 at home for development



Ant: Getting struts-example to compile

```
<property name="app.name" value="struts-example"/>
```

```
<!-- Added by Jason -->
```

```
<path id="struts.classpath">
```

```
<!-- Include all elements that Struts exposes to applications -->
```

```
<fileset dir="{struts.home}/lib">
```

```
<include name="*.jar"/>
```

```
</fileset>
```

```
</path>
```

```
....
```

```
<path id="compile.classpath">
```

```
<!-- Include all JAR files that will be included in /WEB-INF/lib -->
```

```
<!-- *** CUSTOMIZE HERE AS REQUIRED BY YOUR APPLICATION *** -->
```

```
<!-- Added by Jason -->
```

```
<path refid="struts.classpath"/>
```

* - I use SuSE 8.1 at home for development



Ant: Getting struts-example to build

Need to make sure the Struts jars are in the payload of the built application. In the prepare target:

```
<copy todir="${build.home}/WEB-INF">  
  <fileset dir="${struts.home}/lib">  
    <include name="*.tld"/>  
  </fileset>  
</copy>
```

```
<!-- Copy Struts jars into WEB-INF/lib -->  
<copy todir="${build.home}/WEB-INF/lib">  
  <fileset dir="${struts.home}/lib">  
    <include name="*.jar"/>  
  </fileset>  
</copy>
```

* - I use SuSE 8.1 at home for development





Demo time





Alternatives



Alternatives

- ◆ Windows XP*¹
- ◆ JDK: JDK 1.4.2_01 – java.sun.com
- ◆ Web container: Tomcat v5 – jakarta.apache.org
- ◆ Build tool: Ant 1.5.4 – ant.apache.org
- ◆ Unit testing tool: JUnit 3.8.1 – junit.org
- ◆ IDE: Eclipse 3.0M3 – eclipse.org
- ◆ IDE plug-ins: Sysdeo Eclipse Tomcat Launcher plug-in 2.1.1 – sysdeo.com
MyEclipse IDE – myeclipseide.com
- ◆ Modeling tool: none
- ◆ Version control: none
- ◆ Cost*²: \$0
\$29.95 / year

- ◆ And the icing on the cake: Struts 1.1 - <http://jakarta.apache.org/struts/>



MyEclipse

- ◆ Highly productive extension to Eclipse
- ◆ HTML editor
- ◆ JSP Development – syntax highlighting, code completion, debugging, highlights errors before deployment
- ◆ Application Server connectors – Tomcat, JBOSS etc. Stop/start
- ◆ For web application development and EJB development
- ◆ XML editor
- ◆ And so on...
- ◆ Works great on Linux too!
- ◆ myeclipseide.com
- ◆ 30 day free trial





Demo time (on Linux)



Summary

- ◆ The fundamentals
- ◆ Designing our development environment
- ◆ Building our development environment
- ◆ Using our development environment – basics
- ◆ Demo
- ◆ Using our development environment – Struts app development
- ◆ Demo
- ◆ Alternatives - MyEclipse



Further reading

- ◆ Eclipse <http://www.eclipse.org>
- ◆ Eclipse plug-ins <http://eclipse-plugins.2y.net>
- ◆ Ant <http://ant.apache.org>
- ◆ JUnit <http://www.junit.org>
- ◆ MyEclipse <http://www.myeclipseide.com>
- ◆ Sysdeo Tomcat plug-in <http://www.sysdeo.com/eclipse/tomcatPlugin.html>
- ◆ Java Technology on Linux <http://java.sun.com/linux>
- ◆ SuSE Linux <http://www.suse.com>

