Integrallis

Ideas, Implemented.

STRUTS 2

By Joseph Faisal Nusairat

Part I

Java Development Education Series

www.integrallis.com

A publication of Integrallis Software, LLC.
www.integrallis.com/en/training
info@integrallis.com

OVERVIEW & OBJECTIVES

- Part 1 reviews and refreshes the students knowledge of Java Web Development starting with Servlets and JSPs and the evolution of the Model-View-Controller architectural pattern
- Part 1 introduces Struts 2 and walks the students over configuring and deploying a simple application
- Part 1 will also cover every component of a Struts 2 application as they apply to the MVC pattern and the creation of a typical CRUD application

2

OVERVIEW & OBJECTIVES

- Part 1 covers:
  - Java Web Foundations and the MVC architectural pattern
  - Introduction to Struts 2
  - Components of a Struts application
  - Struts configuration details
  - Dynamic content display techniques
  - Building CRUD applications with Struts

MATERIAL CONVENTIONS

- Each training day consists of a mix of lecture time, discussion time and Q & A, guided exercises and labs

- For the guided exercises you will see a green "follow along" sign on the slides

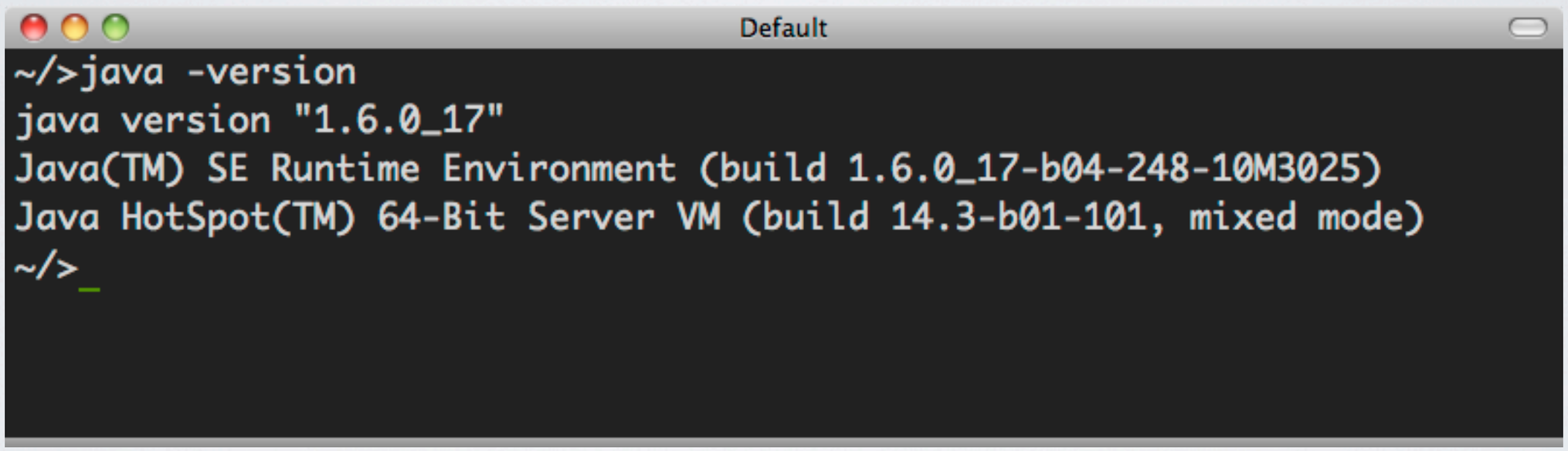
- For the labs you'll see an orange sign with the lab number

4

PREPARATION & SETUP

CORE REQUIREMENTS

- Testing Java version:



Any version equal to or higher than **1.5** should suffice

PREPARATION & SETUP

CORE REQUIREMENTS

- Testing Maven version:

```
Default
~/>mvn -v
Apache Maven 2.2.0 (r788681; 2009-06-26 06:04:01-0700)
Java version: 1.6.0_17
Java home: /System/Library/Frameworks/JavaVM.framework/Versions/1.6.0/Home
Default locale: en_US, platform encoding: MacRoman
OS name: "mac os x" version: "10.6.2" arch: "x86_64" Family: "mac"
~/>
```

Any version equal to or higher than **2.0.8** should suffice

6

PREPARATION & SETUP

CORE REQUIREMENTS

- Testing Eclipse by launching:

7

M2ECLIPSE

MAVEN PLUGIN FOR ECLIPSE

M2ECLIPSE

ECLIPSE / MAVEN INTEGRATION

- M2Eclipse provides comprehensive Maven integration for Eclipse
  - Manage both simple and multi-module Maven projects
  - Execute Maven builds from within Eclipse
  - Search and integrate Maven Repositories

9

M2ECLIPSE INSTALL

ECLIPSE / MAVEN INTEGRATION

- M2Eclipse, like most Eclipse plugins is installed via the Eclipse Update Sites

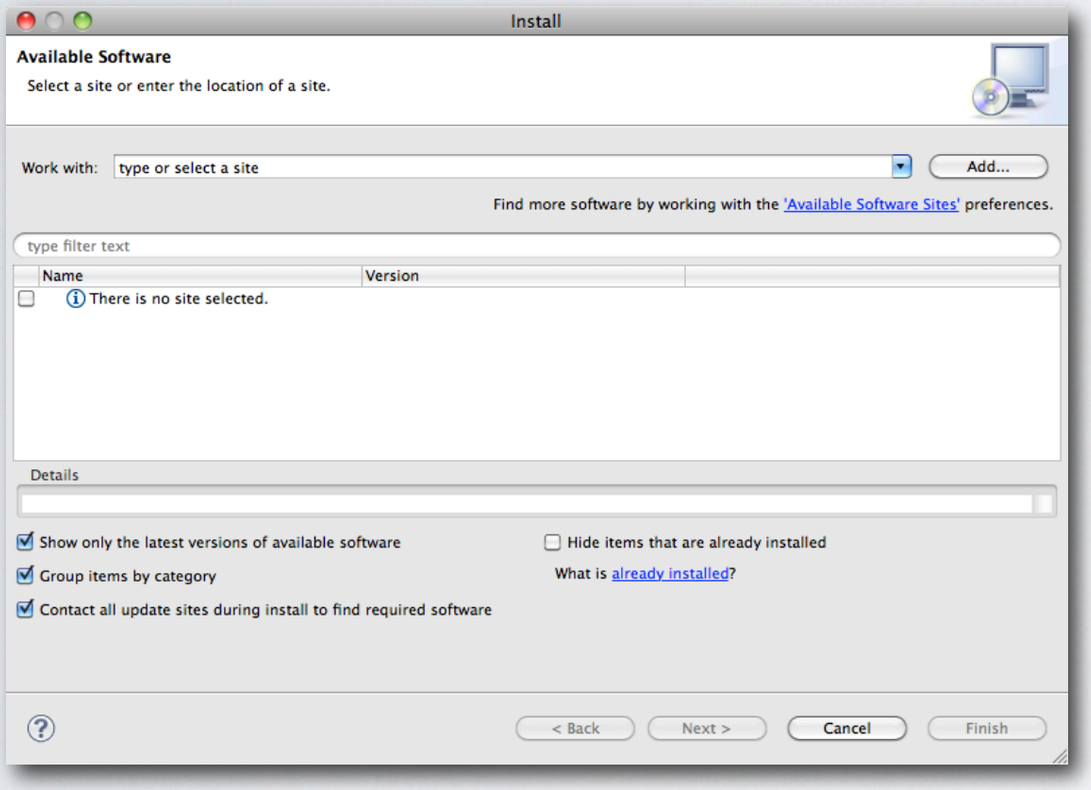
1. Select "Help" >> "Install New Software..."
2. Next to the "Work with:" field, select "Add..."
3. Enter "http://m2eclipse.sonatype.org/sites/m2e" with a name like "M2Eclipse"
4. Select the check box for "Maven Integration for Eclipse"

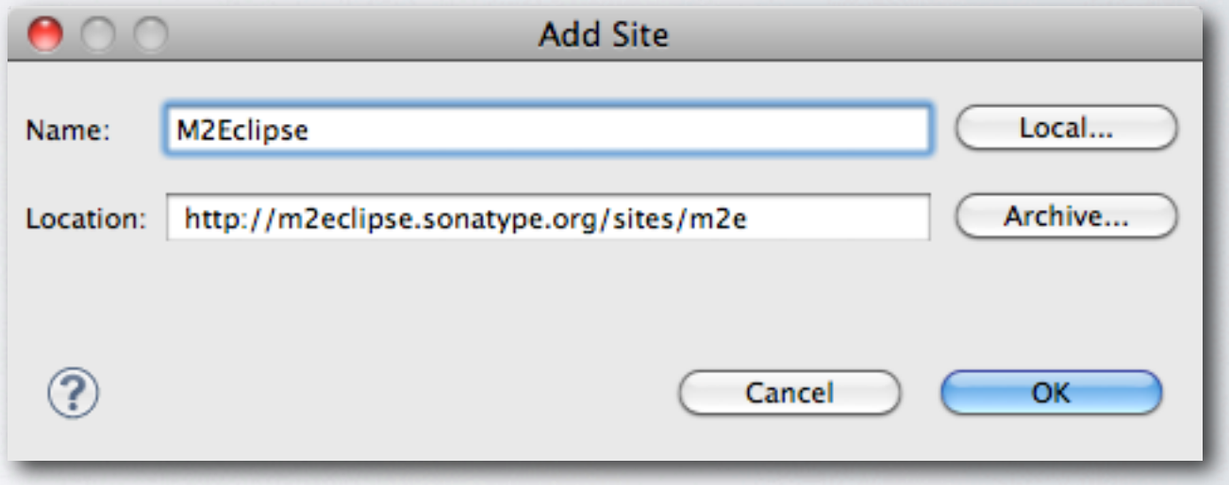
10

M2ECLIPSE INSTALL

ECLIPSE / MAVEN INTEGRATION

- Add a site with the M2Eclipse update site URL:

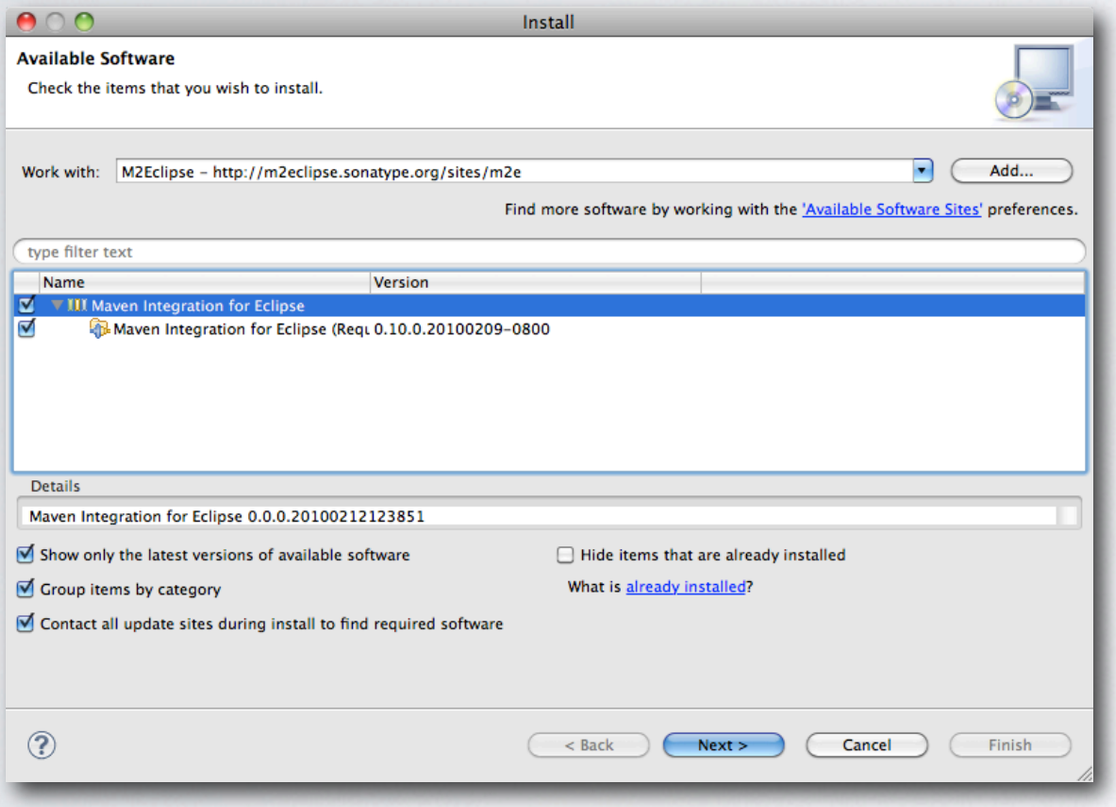


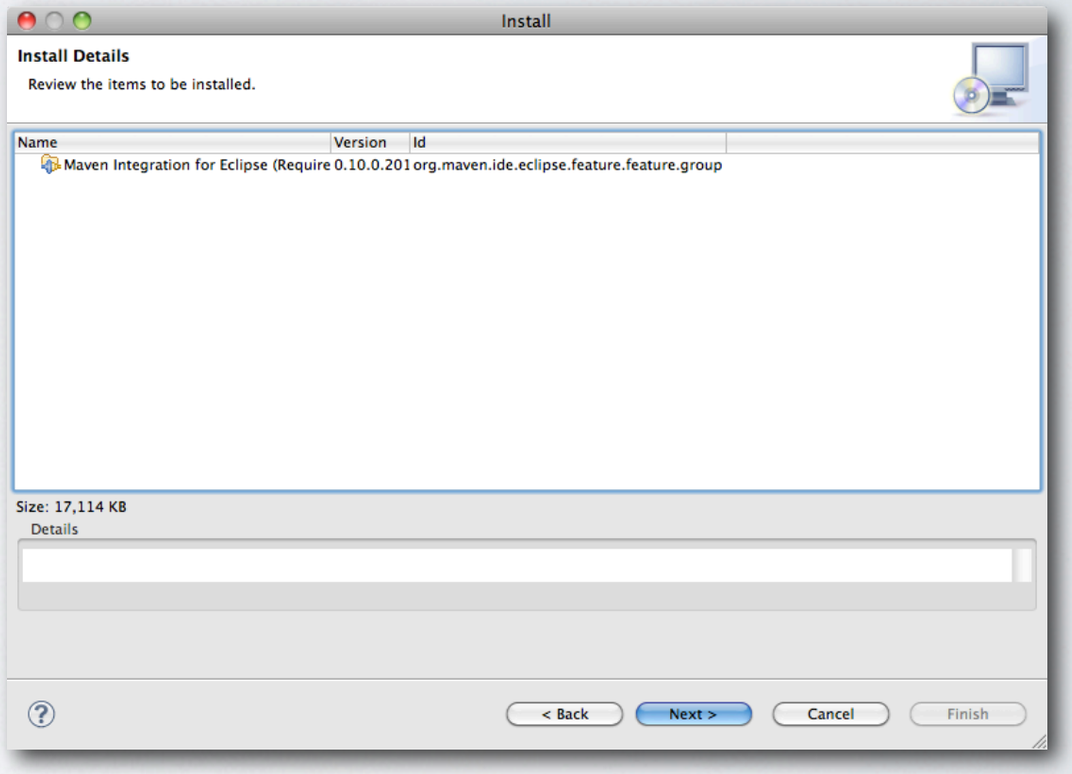


M2ECLIPSE INSTALL

ECLIPSE / MAVEN INTEGRATION

- Select the "Maven Integration for Eclipse"



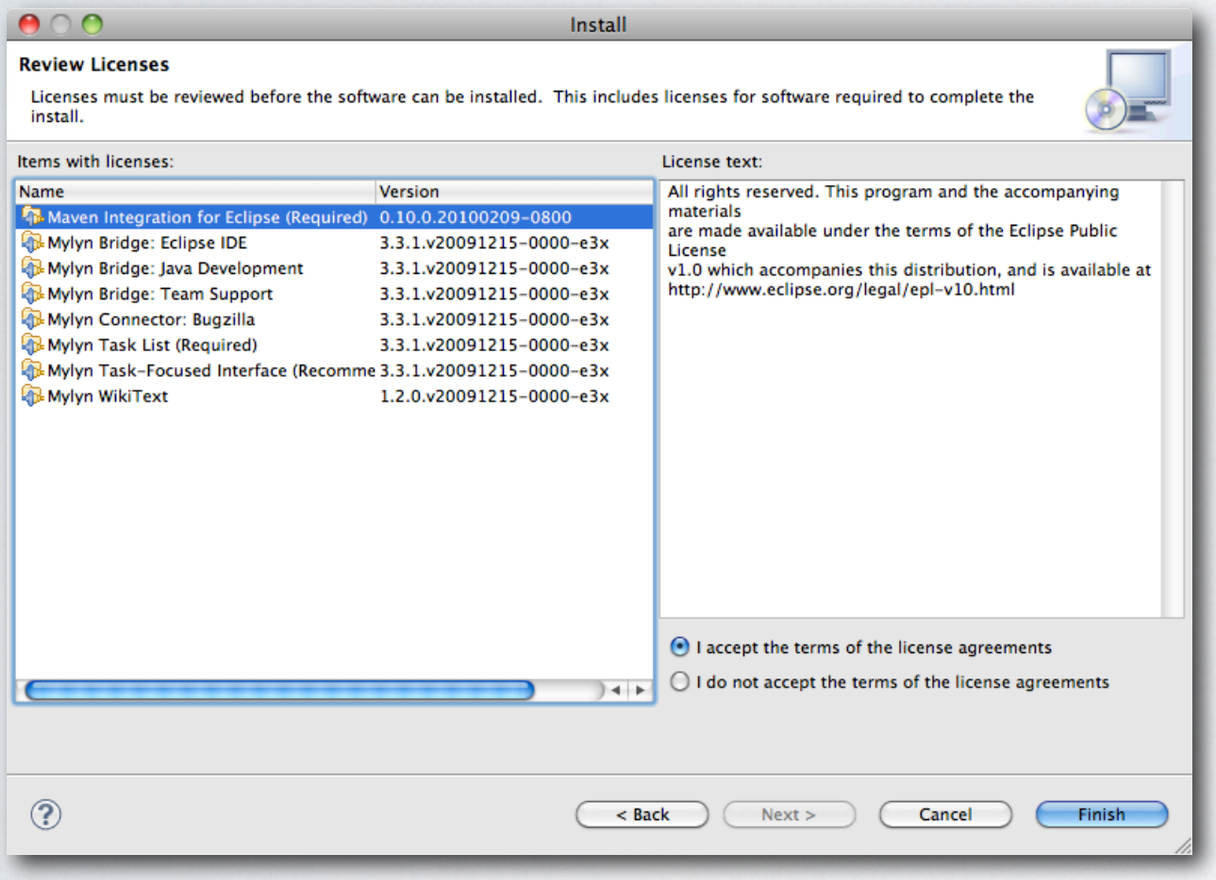


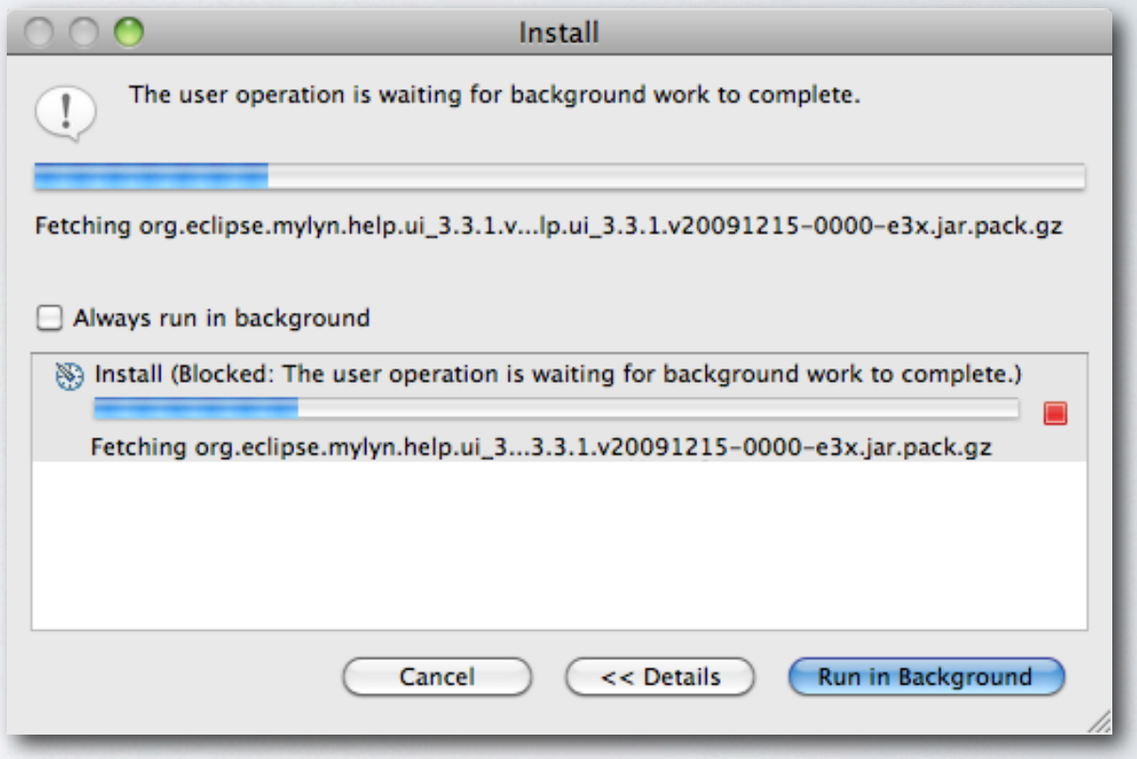
12

M2ECLIPSE INSTALL

ECLIPSE / MAVEN INTEGRATION

- Complete the installation by selecting "Finish". Restart Eclipse!





13

JAVA WEB FOUNDATIONS

SERVLETS AND JAVA SERVER PAGES

MODEL IMPLEMENTATIONS

MVC REFRESHER

- Early Java lacked a framework for the structured development of web applications
- Before the advent of JavaServer Pages (JSP), Java web applications were Servlet applications in which routing, business logic and presentation logic where tightly coupled
- JSP inverted the problem by leading to the creation of applications in which the markup was littered with snippets of Java code
- The open source community reacted by creating several "template" engines to separate the view rendering from the logic

15

MODEL 1

JSP MODEL 1

- Sun pushed forward a more object-oriented approach using JSPs which we now call the JSP Model 1 Architecture
- In the Model 1 architecture the JSP page is responsible for processing the incoming request and responding to the client
- Separation of concerns is achieved by delegating all data access/ business operations using JavaBeans
- A side-effect of Model 1 is the proliferation of scriptlets of Java code embedded within JSP pages which makes page maintenance difficult, specially for non-programmers (designers)

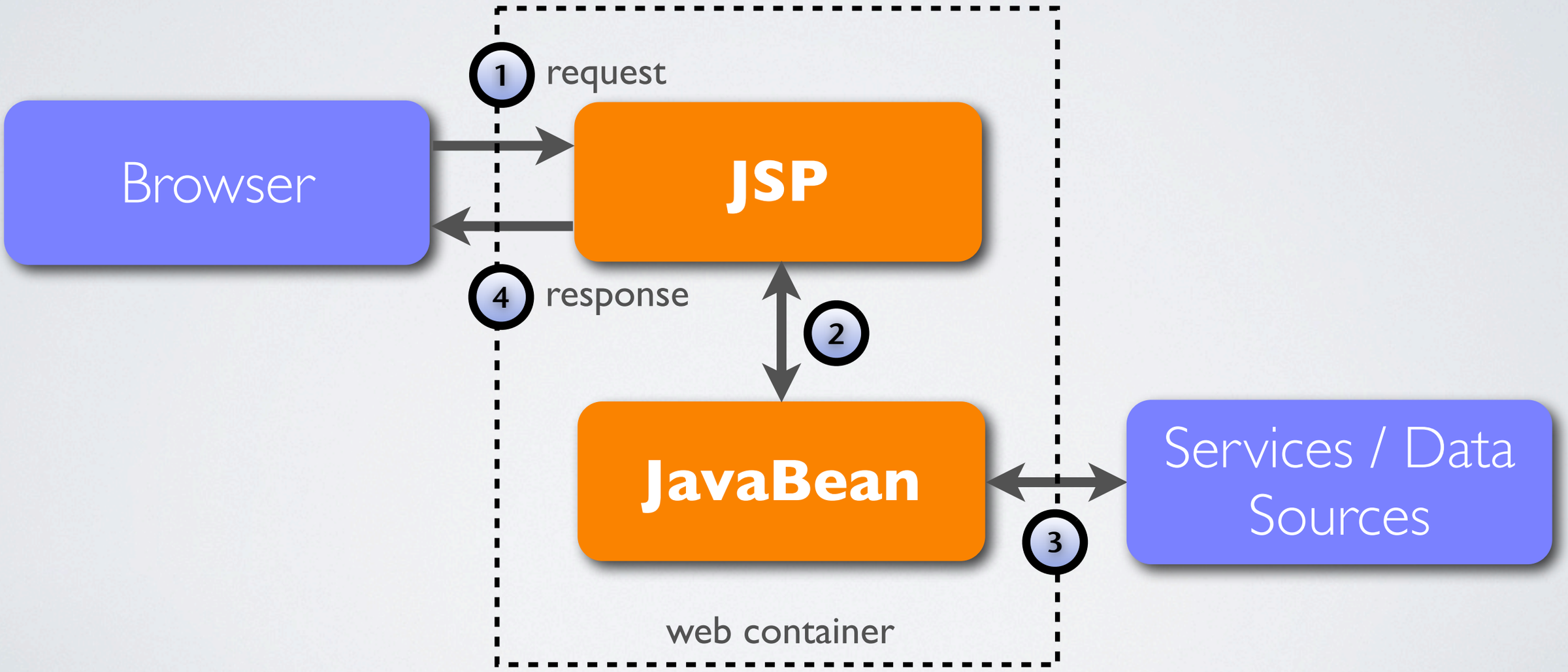
16

From Wikipedia:
Model 1 is a simple pattern whereby the code responsible for the display of content is intermixed with logic. Model 1 is only recommended for small applications and is mostly obsolete in modern development practices.

MODEL 1

JSP MODEL 1

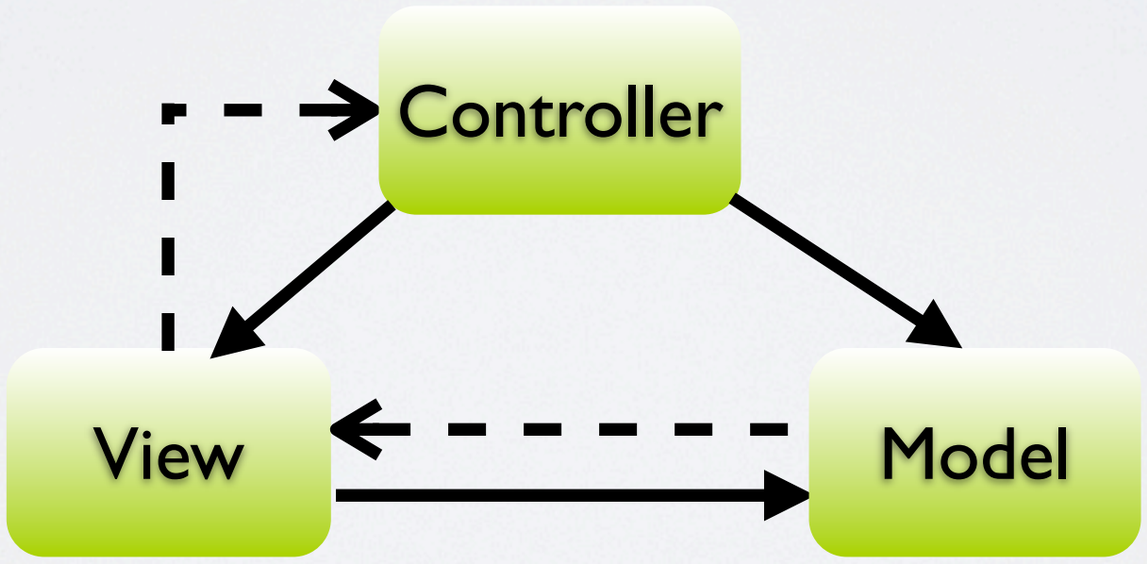
- The JSP Model 1 Architecture:



MODEL-VIEW-CONTROLLER

MVC ARCHITECTURAL PATTERN

- Model-View-Controller (MVC) is an architectural pattern that separates the domain objects and logic (the model) from the user interface input and presentation (the view) by using an intermediary (the controller) to coordinate their interaction



18

MODEL 2

JSP MODEL 2

- JSP Model 2 is hybrid approach that combines the Servlets and JSPs
- JSP Model 2 is an implementation of the MVC pattern using JSPs as the view and servlets as the controller
- In Model 2, servlets handle request processing, creation of beans/objects used by the JSP pages and handle the routing/ redirection or flow of the application
- Model 2 creates a cleaner separation of concerns and responsibilities while increasing the level of complexity

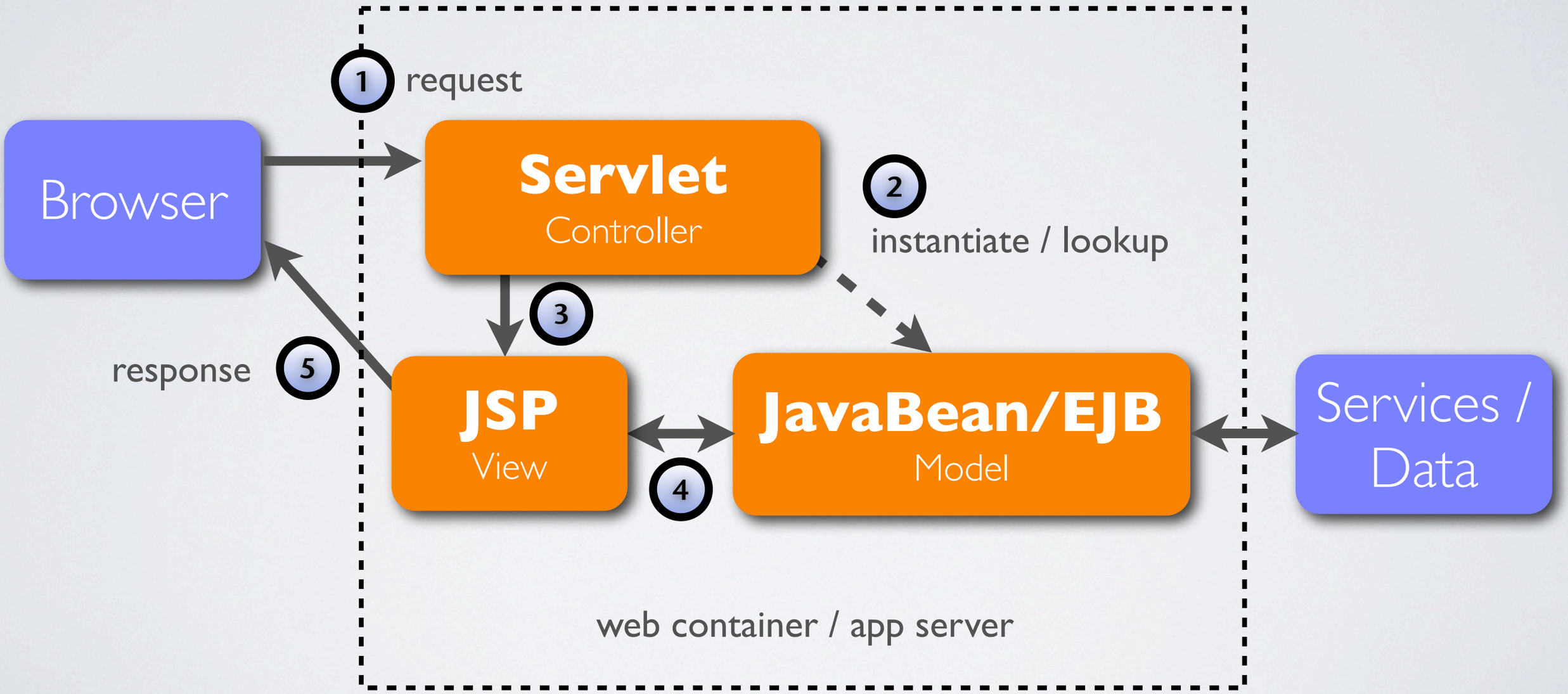
19

From wikipedia:
Model 2 is a more complex design pattern that separates the display of content from the logic used to obtain and manipulate the content. Since Model 2 drives a separation between logic and display, it is usually associated with the Model–View–Controller(MVC) paradigm. While the exact form of the MVC "Model" was never specified by the Model 2 design, a number of publications recommend a formalized layer to contain MVC Model code. The Java BluePrints, for example, originally recommended using EJBs to encapsulate the MVC Model.

MODEL 2

JSP MODEL 2

- The JSP Model 2 Architecture:



JAVA SERVLETS

JAVA MEETS HTTP

JAVA SERVLETS

JAVA MEETS HTTP

- A Java Servlet is a software component that understands the HTTP protocol and can respond to requests sent by a client
- A Servlet can receive data explicitly from a client as form data or implicitly as request headers
- A Servlet can explicitly return content back to a client (Plain Text, XML, HTML, etc) or implicitly as status codes and response headers

22

JAVA SERVLETS

JAVA MEETS HTTP

- Servlets are managed components that live in a Web Container
- Servlets consists of a combination of compiled Java classes (byte code) and XML configuration files packaged in a Web Application aRchive (WAR)
- When deployed to a Web Container, Servlet instances are created, initialized and destroyed by the container.

23

JAVA SERVLETS

JAVA MEETS HTTP

- The Servlet Life-cycle consists of:
  - If there is not an instance of a Servlet available, the Web container will load the servlet class
  - Create an instance of the servlet class
  - Initializes the server by calling the init method
  - Invokes the service method, passing a request and response object
  - Container calls the servlet destroy method when deleting

24

JAVA SERVLETS

A SERVLET THAT RETURNS PLAIN TEXT

- Let's examine a Servlet that can send a simple text response to a client:

```
package com.integrallis.modernjee.servlets;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class HelloWorldServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```


25

JAVA SERVLETS

A SERVLET THAT RETURNS PLAIN TEXT

- Dissecting the simple "Hello World" servlet:

We extend HTTPServlet which provides event handlers for HTTP verbs (GET, POST, PUT, DELETE)

The HTTPServletRequest is passed to a Servlet's service methods (like doGet) and contain information like the path information, headers, authentication information, query string, etc.

```
public class HelloWorldServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```

From the response object we can obtain a PrintWriter instance which we can write out content to be returned in the response object

The HTTPServletResponse is where we put our response back to the client

26

MAVEN

MAVEN AND CREATING OUR FIRST APP

MAVEN

CREATING & BUILDING JAVA APPLICATIONS

MAVEN

BUILDING & MANAGING DEPENDENCIES

- Java EE projects are complex in terms of their requirements. A minimal Java EE project requires at least a dozen JAR files.
- Environment libraries are populated from the system/boot classpath and application classpath. Classloader hierarchies further complicate things.
- The Java JAR format fell short in terms of mandating metadata requirements. The result is what most of us refer to as JAR-Hell see http://en.wikipedia.org/wiki/JAR_hell#JAR_hell

29

MAVEN

BUILDING & MANAGING DEPENDENCIES

- At a high level Maven provides:
  - A way to manage your application dependencies with a clear cut way to discern between environment dependencies (development, test, production, server provided, etc.)
  - A way to quickly get started by providing archetypes that generate pre-configured, IDE-independent Java/Java EE projects

30

There are 3 possible ways to satisfy an application's dependencies in Java:
Old School: Use the old style of dependency management and add a /lib directory to your project. Download all the dependencies, first researching the compatibility amongst the different JARs. Repeat as upgrades are required.
Use Maven: Maven is a build tool that provides dependency management capabilities. With Maven you provide the version of a library and its associated dependencies are automatically downloaded and managed
Use Ant+Ivy: The Ant build tool can be used in conjunction with the Apache Ivy project to provide automated dependency management.

For the labs in this course we would simplify the project setup by using a Maven based build

MAVEN

CREATING JAVA PROJECTS WITH MAVEN

- A skeleton Maven project can be created from within Eclipse by installing the Maven Integration plugin or directly from the command line
- Maven provides command-line project creation wizards called archetypes

31

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- To create a simple Java application using Maven we can use the **maven-archetype-quickstart**

- The quickstart archetype requires that you provide a **groupId** (a root Java package for you application) and an **artifactId** (the name of the JAR, WAR or EAR to be created)

32

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- For example, to use the **maven-archetype-quickstart** at the command line enter (CD to the directory where your project folder will reside): `mvn archetype:generate`

33

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- The archetype:generate command will display a list of available archetypes. Press enter or enter 15 to select the **maven-archetype-quickstart**:

```
36: internal -> myfaces-archetype-helloworld (A simple archetype using MyFaces)
37: internal -> myfaces-archetype-helloworld-facelets (A simple archetype using MyFa
ces and facelets)
38: internal -> myfaces-archetype-trinidad (A simple archetype using Myfaces and Tri
nidad)
39: internal -> myfaces-archetype-jsfcomponents (A simple archetype for create custo
m JSF components using MyFaces)
40: internal -> gmaven-archetype-basic (Groovy basic archetype)
41: internal -> gmaven-archetype-mojo (Groovy mojo archetype)
Choose a number:  (1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/18/19/20/21/22/23/24/25
/26/27/28/29/30/31/32/33/34/35/36/37/38/39/40/41) 15: :
```


34

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- Enter the values shown below for groupId, artifactId, version and package:

```
Choose a number:  (1/2/3/4/5/6/7/8/9/10/11/12/13/14/15/16/17/18/19/20/21/22/23/24/25
/26/27/28/29/30/31/32/33/34/35/36/37/38/39/40/41) 15: :
Define value for groupId: : com.integrallis.mjee
Define value for artifactId: : tddwjunit
Define value for version:  1.0-SNAPSHOT: :
Define value for package:  com.integrallis.mjee: :
Confirm properties configuration:
groupId: com.integrallis.mjee
artifactId: tddwjunit
version: 1.0-SNAPSHOT
package: com.integrallis.mjee
 Y: :
```


35

In the above example a project called tddwjunit with a root package of com.integrallis.mjee is created

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- Once the archetype generation goal completes, you should see the BUILD SUCCESSFUL message:

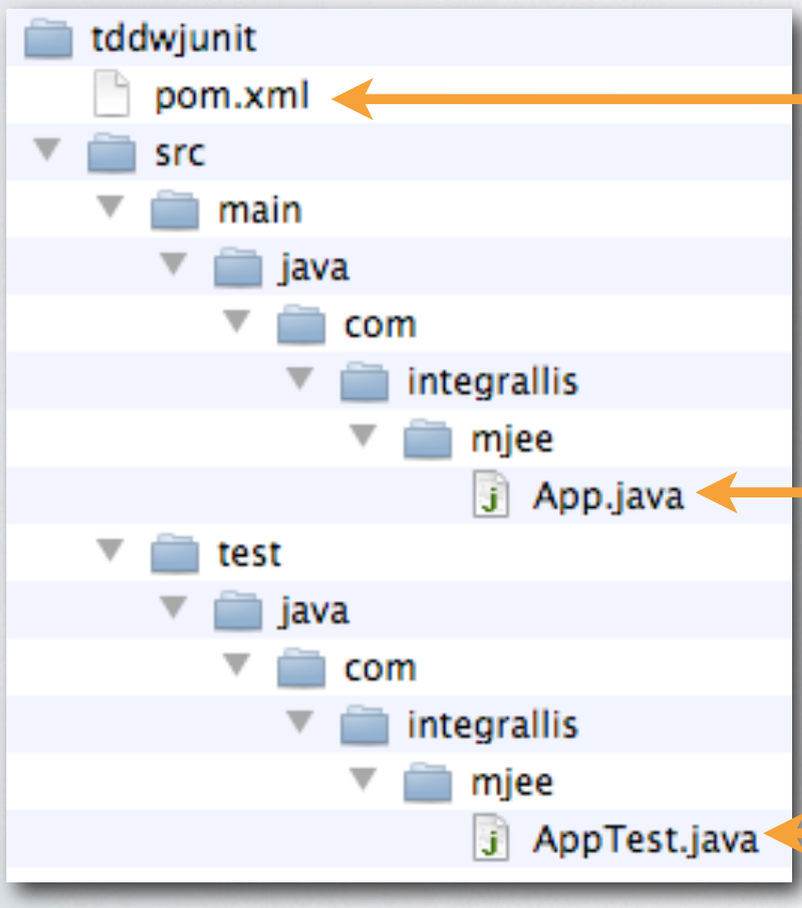
```
[INFO] ------------------------------------------------------------------------
[INFO] Using following parameters for creating OldArchetype: maven-archetype-quickstart:RELEASE
[INFO] ------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: com.integrallis.mjee
[INFO] Parameter: packageName, Value: com.integrallis.mjee
[INFO] Parameter: package, Value: com.integrallis.mjee
[INFO] Parameter: artifactId, Value: tddwjunit
[INFO] Parameter: basedir, Value: /Users/bsbodden/Documents/Training/Modern JavaEE/day 1/code
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] ********************* End of debug info from resources from generated POM ***************
********
[INFO] OldArchetype created in dir: /Users/bsbodden/Documents/Training/Modern JavaEE/day 1/code/
tddwjunit
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESSFUL
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 5 minutes 2 seconds
[INFO] Finished at: Wed Mar 10 23:55:47 MST 2010
[INFO] Final Memory: 13M/79M
[INFO] ------------------------------------------------------------------------
```


36

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- The resulting project structure is shown below:



Maven POM (Project Object Model) describes the goals of the build

Sample Java class. Contained in the /src/main/java directory.

Sample Java test class. Contained in the /src/test/java directory.

37

The Maven basic archetype creates a simple Java project with an src directory for the Java files and a test directory for the JUnit/TestNG tests
Notice that the root package is the groupId and the project name is the artifactId
The sample Java files App.java and AppTest.java can be removed after you add your own

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- The basic POM will compile and execute tests:

```
<project
  xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd"
  >
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.integrallis.mjee</groupId>
  <artifactId>tddwjunit</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>tddwjunit</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

</project>
```

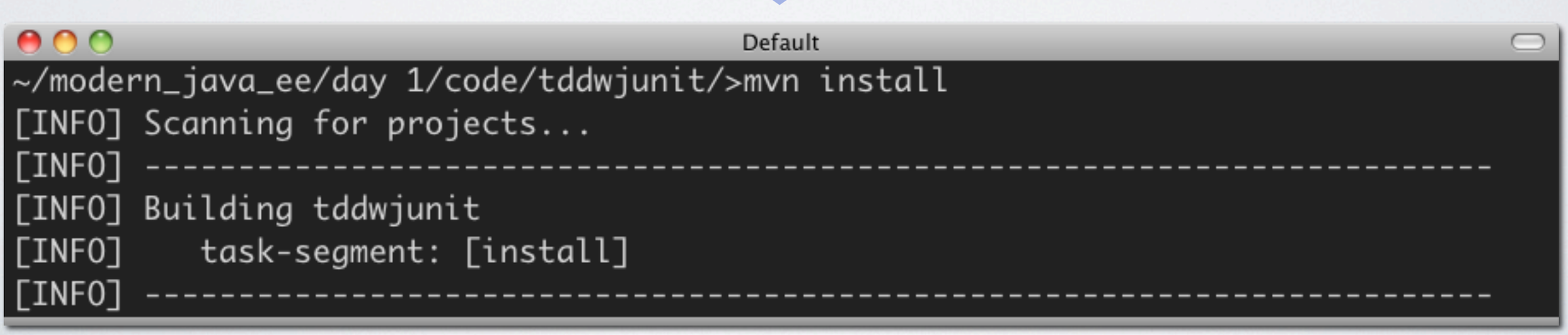

38

The POM XML file main points of interest as it relates to this class are the dependencies element and the build plugins (which we will see later in this section)
As you can see from the POM, Maven allows you to declare the dependencies of the artifact being built. In this case the artifact to be created is tddwjunit.jar (see artifactId and packaging elements)
The current dependencies are JUnit 3.8.1 in the scope of testing. The dependency element for JUnit can be read as: “for running the tests of this project I need the junit-3.8.1.jar and its dependencies in the classpath”

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- To build the application change directories to the newly created application directory (`tddwjunit`) and enter: `mvn install`



39

Maven is launched via the mvn command. The command mvn install is a lifecycle command that installs the JAR or WAR file in the local Maven repository which we will learn about shortly. Other mvn life-cycle commands are:

- mvn clean: cleans all generated files
- mvn compile: compiles Java sources
- mvn test-compile: compiles all JUnit/TestNG test classes
- mvn test: executes all tests
- mvn package: builds the target artifact (JAR/WAR/EAR)

Other type of Maven commands are plugin commands which we will explore later. For a complete list of Maven commands see http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- As seen below the build should execute the single JUnit 3.8.1 test contained within.

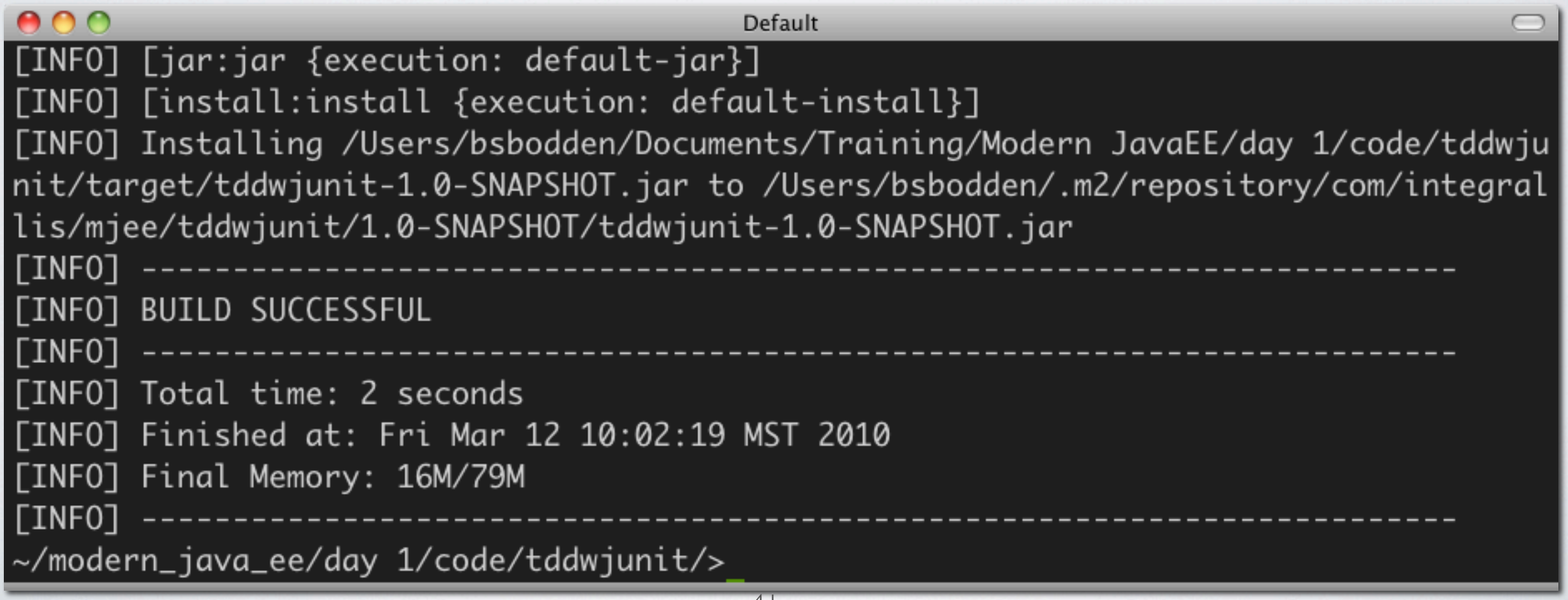
```
Default
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.integrallis.mjee.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.029 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
```


40

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- Maven builds are artifact-driven. The default target artifact is a JAR file. In the case of the `tddwjunit` project the resulting JAR file is `tddwjunit-1.0-SNAPSHOT.jar`



41

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- We need to update the POM, starting with the VM used to compile the code
- Add the maven-compiler-plugin set to 1.5 or 1.6

```
<build>
  <plugins>
      <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>2.1</version>
          <configuration>
              <source>1.6</source>
              <target>1.6</target>
          </configuration>
      </plugin>
  </plugins>
</build>
```


42

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- We also need to update the POM's dependencies
- Update the JUnit dependency to JUnit 4 as shown below:

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.8.1</version>
    <scope>test</scope>
</dependency>
```


43

MAVEN

MAVEN ARCHETYPES FROM THE COMMAND LINE

- After running the build with mvn install, two obvious questions come to mind:

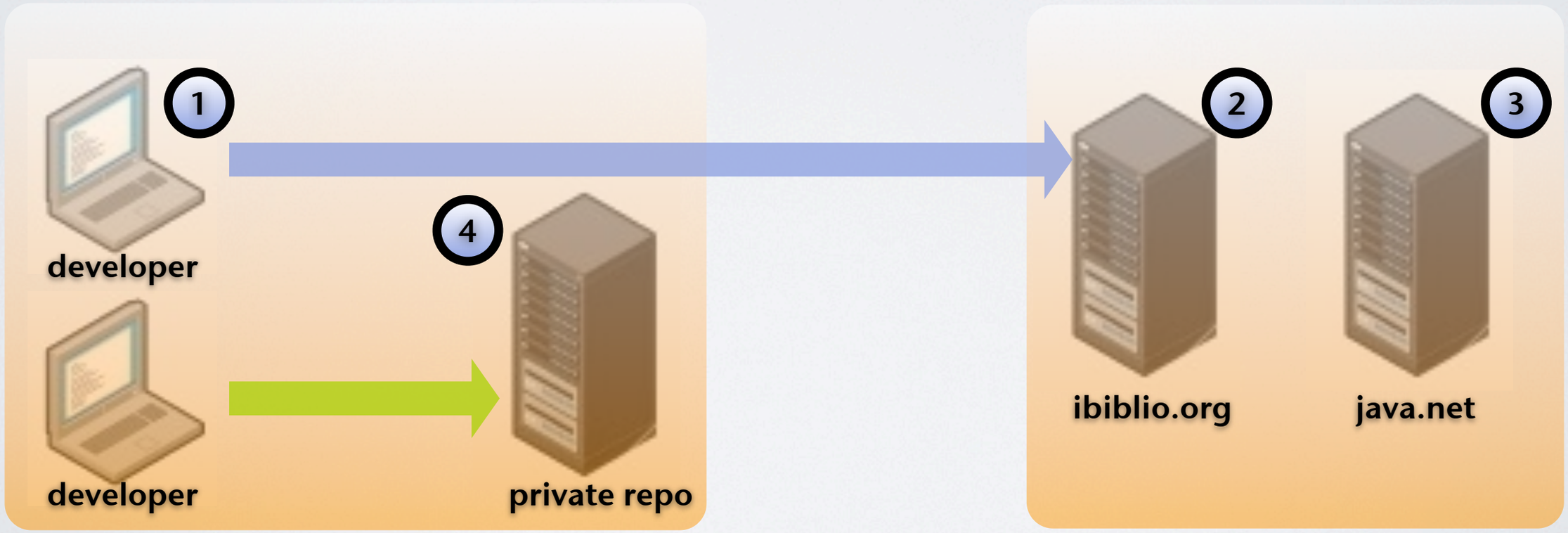
1. Where do the dependencies (JAR files) come from?
2. Where do the build artifacts end up?

44

MAVEN

LOCAL AND REMOTE REPOSITORIES

- A repository is defined as a network accessible location where artifacts and their metadata are stored



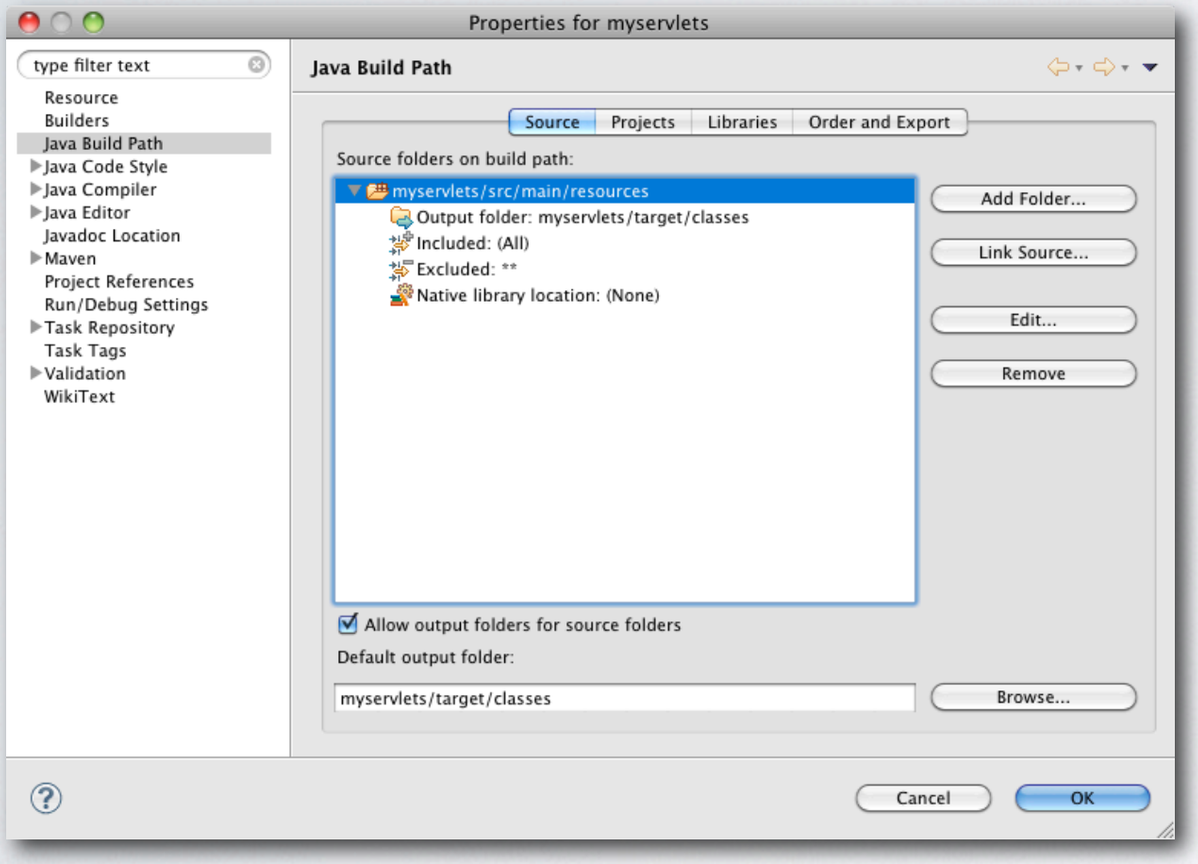
45

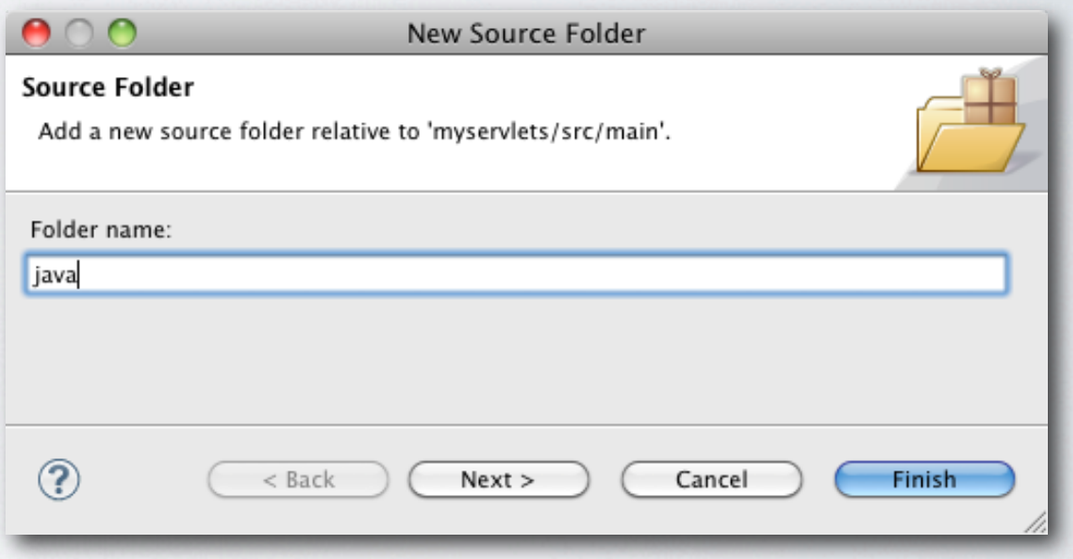
The diagram above shows how Maven operates. When you build locally Maven will first check if your local system (1) has the required dependencies, if not it will try a list of external repositories. By default Maven is configured to search ibiblio.org (2), one of the largest and the official repository for Maven. There are countless other repositories on the web that you can configure Maven to search, for example the open source site java.net has a Maven repository available (3).
Enterprises can also have a private internal Maven repository (4) for artifacts that you do not want to expose to the public and also as a mirror for “approved” artifacts from other public repositories.
Locally Maven keeps a repository under you home directory in the .m2/repository directory.

A SERVLET PROJECT

CREATING A MAVEN SERVLET PROJECT IN ECLIPSE

- Add a "java" source folder under /src/main



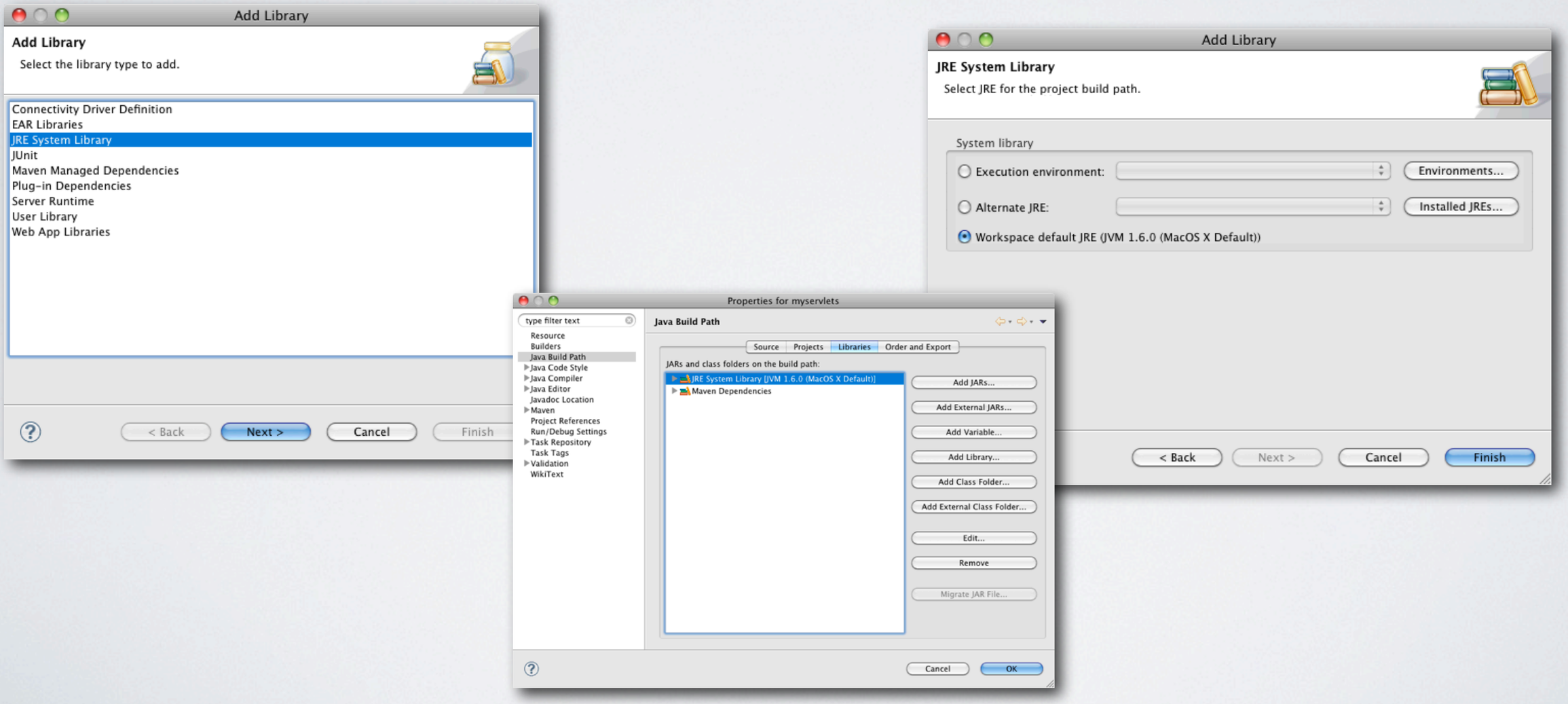


46

A SERVLET PROJECT

CREATING A MAVEN SERVLET PROJECT IN ECLIPSE

- Upgrade the JRE System Library to 1.5 or 1.6 if necessary

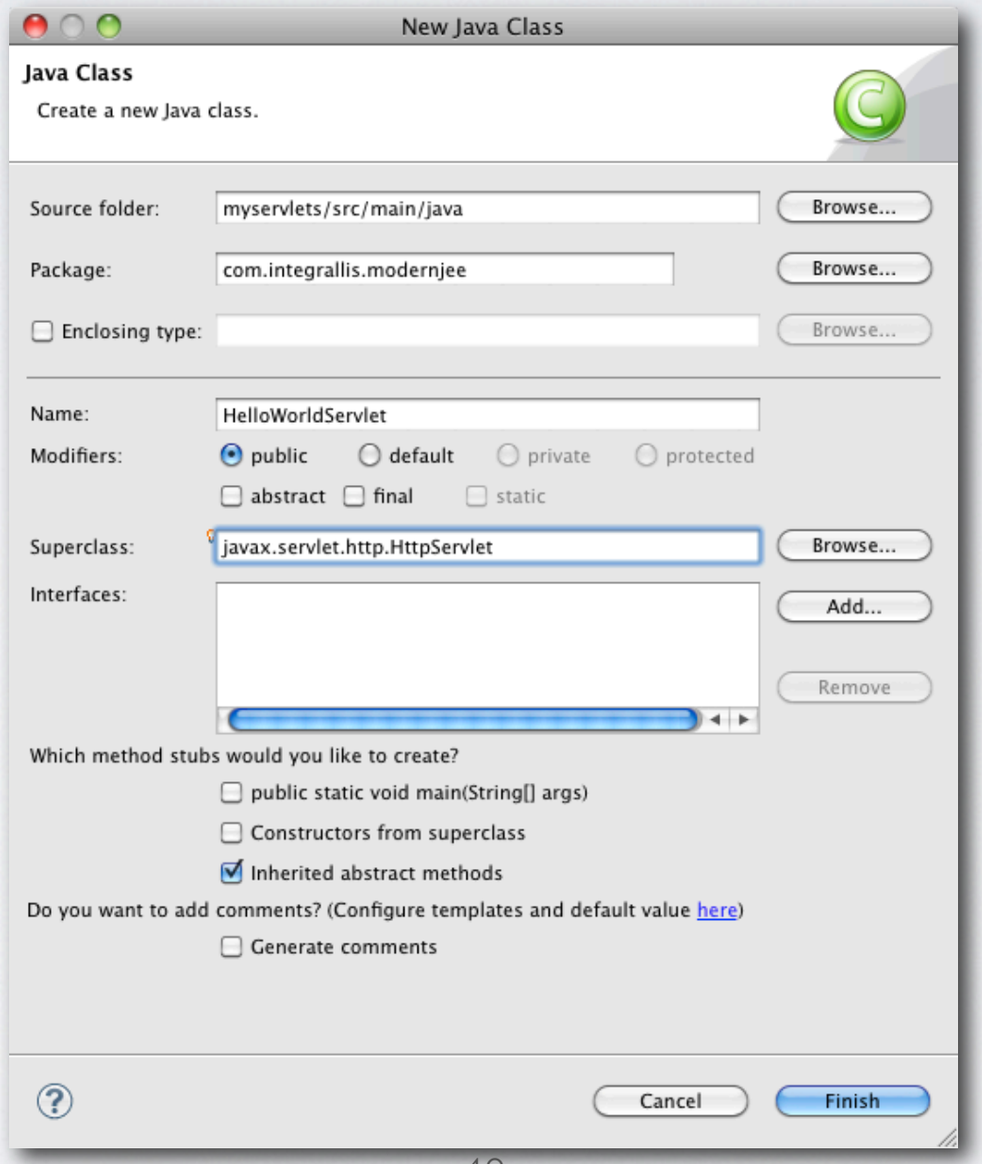


47

A SERVLET PROJECT

CREATING A MAVEN SERVLET PROJECT IN ECLIPSE

- Create a new Java class "`HelloWorldServlet`" extending `HTTPServlet`



48

JAVA SERVLETS

A SERVLET THAT RETURNS PLAIN TEXT

- Implement the HelloWorldServlet as shown below:

```
package com.integrallis.modernjee;

import javax.servlet.http.HttpServlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class HelloWorldServlet extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.println("Hello World");
    }
}
```


49

A SERVLET PROJECT

CREATING A MAVEN SERVLET PROJECT IN ECLIPSE

- Start by removing the JUnit dependency and adding the Servlet and JSP API libraries

```
<dependencies>
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.5</version>
        <scope>provided</scope>
    </dependency>

    <dependency>
        <groupId>javax.servlet</groupId>
         <artifactId>jsp-api</artifactId>
         <version>2.0</version>
         <scope>provided</scope>
    </dependency>
</dependencies>
```


50

A SERVLET PROJECT

CREATING A MAVEN SERVLET PROJECT IN ECLIPSE

- In the **<plugins>** section add the Jetty Web Container plugin
- This will enable us to quickly launch and test our Servlet app

```
<plugin>
    <groupId>org.mortbay.jetty</groupId>
    <artifactId>maven-jetty-plugin</artifactId>
    <configuration>
        <contextPath>${project.build.finalName}</contextPath>
        <scanIntervalSeconds>3</scanIntervalSeconds>
    </configuration>
</plugin>
```


51

A SERVLET PROJECT

CREATING A MAVEN SERVLET PROJECT IN ECLIPSE

- At the command line, CD to the "**myservlets**" project and enter "**mvn jetty:run**"

```
Default
~/Projects/workspace/myservlets/>mvn jetty:run
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building myservlets Maven Webapp
[INFO]    task-segment: [jetty:run]
[INFO] ------------------------------------------------------------------------
[INFO] Preparing jetty:run
[INFO] [resources:resources {execution: default-resources}]
[WARNING] Using platform encoding (MacRoman actually) to copy filtered resources
, i.e. build is platform dependen
[INFO] Copying 0 resource
```

```
Default
app/WEB-INF/web.xml
[INFO] Webapp directory = /Users/bsbodden/Projects/workspace/myservlets/src/main
/webapp
[INFO] Starting jetty 6.1.22 ...
2010-03-14 09:39:17.254:INFO::jetty-6.1.22
2010-03-14 09:39:17.339:INFO::No Transaction manager found - if your webapp requ
ires one, please configure one.
2010-03-14 09:39:17.559:INFO::Started SelectChannelConnector@0.0.0.0:8080
[INFO] Started Jetty Server
[INFO] Starting scanner at interval of 3 seconds.
```


52

A SERVLET PROJECT

CREATING A MAVEN SERVLET PROJECT IN ECLIPSE

- Point your web browser to http://localhost:8080/myservlets

53

JAVA SERVLETS

A SERVLET THAT RETURNS HTML

- To write a Servlet that returns HTML we first need to tell the client in the response that we are indeed returning the MIME type "text/html"

```
public class HelloWorldInHTMLServlet extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html");
        StringBuilder sb = new StringBuilder();
        sb.append("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01 Transitional//EN\"");
        sb.append("\"http://www.w3.org/TR/html4/loose.dtd\">");
        sb.append("<html>");
        sb.append("    <body>");
        sb.append("    <h1>Hello in HTML</h1>");
        sb.append("    </body>");
        sb.append("</html>");

        PrintWriter out = response.getWriter();
        out.println(sb.toString());
    }
}
```


54

JAVA SERVLETS

CONFIGURING THE WEB APPLICATION

- To be able to access our servlet we need to tell the Web Container how to map request URLs to servlets
- Servlet mappings and other settings are specified in the web.xml deployment descriptor
- To map a Servlet to a particular request URL we need to use two XML elements; the <servlet> and the <servlet-mapping>

55

JAVA SERVLETS

WEB.XML DEPLOYMENT DESCRIPTOR

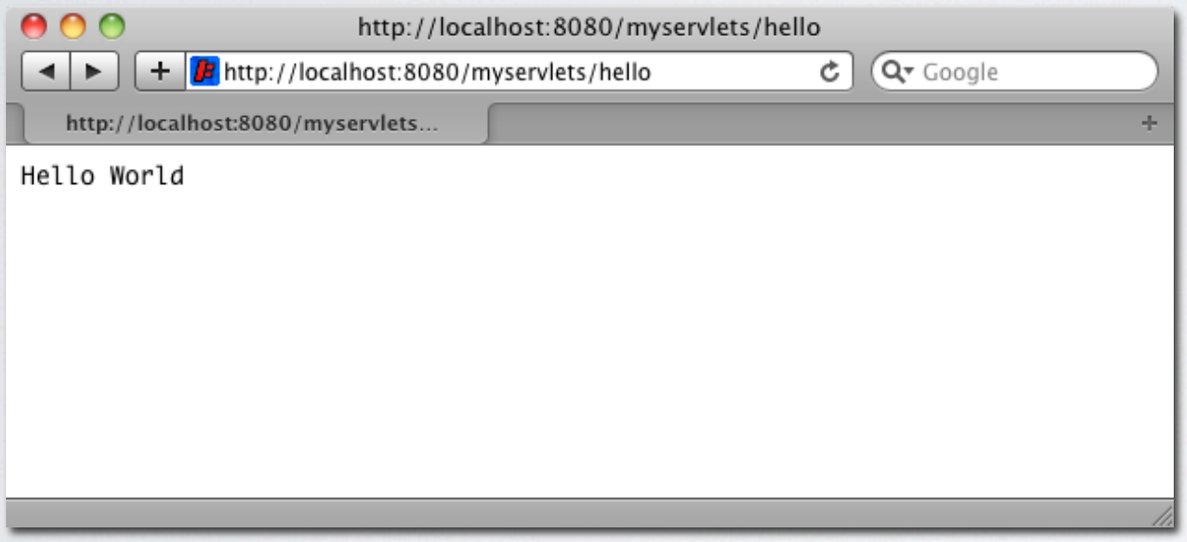
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5"
   xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

   <servlet>
      <servlet-name>HelloWorldServlet</servlet-name>
      <servlet-class>com.integrallis.modernjee.HelloWorldServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

    <servlet>
      <servlet-name>HelloWorldInHTMLServlet</servlet-name>
      <servlet-class>com.integrallis.modernjee.HelloWorldInHTMLServlet</servlet-class>
      <load-on-startup>1</load-on-startup>
   </servlet>

    <servlet-mapping>
        <servlet-name>HelloWorldServlet</servlet-name>
        <url-pattern>/hello/*</url-pattern>
        <url-pattern>/*</url-pattern>
    </servlet-mapping>

    <servlet-mapping>
        <servlet-name>HelloWorldInHTMLServlet</servlet-name>
        <url-pattern>/hello_in_html/*</url-pattern>
    </servlet-mapping>

</web-app>
```


56

JAVA SERVLETS

CONFIGURING THE WEB APPLICATION

- The root of the web app still resolved to the `HelloWorldServlet`



- /myservlets/hello

- /myservlets/hello_in_html

57

MAVEN

OUR FIRST APP

- The first application will have you create a maven web application with an index page with input boxes and a list page.
- The basic create / list you will be creating is for a Book with an Author. You will receive a copy of these classes in class.
- The services will throw an exception so make sure to try and catch them.

58

LAB 1.0

SERVLETS

- You will create 2 servlets, one for adding and one for the list page. You will use the `HTTPServletRequest` with `getParameter` method to retrieve both URL encoded parameters from the form.

- In the end you will have 3 pages. An index page with a link to add and list, the add page, and then the list page.

59

STRUTS 2

INTRODUCTION

JAVA WEB IN LATE 90S

JAVA WEB FRAMEWORK BEGINNINGS

- One of the major problems with the Servlet architecture was not the architecture itself but was how it was used. In the late 90s there were no Sun guidelines. There was the spec which mostly included servlets, filters, and JSPs.

- This was the same problem that plagued Enterprise Java Beans (EJB) spec up until EJB 3. Developers were given the tools by Sun and told how they work but not any guidelines on how to use them correctly and most effectively.

61

DEVELOPMENT LATE 90S

MVC REFRESHER

- Every development shop wrote their own unique custom wrappers around the spec, creating their own frameworks.
- Most groups based their frameworks off of the MVC architecture but this leads to a difficulty in attracting developers who can use your framework fast.
- Many corporations wrapped there own sometimes obscene architectures (like one that was 100% XML driven)

62

START OF STRUTS

STRUTS

- Developers that created these in house frameworks wanted to share their ideas, their code and wanting to gain notoriety started publishing there frameworks for consumption.

- Many developers also got together to combine ideas, and create new web frameworks to handle ever increasing demands.

- One of the early web frameworks was Apache Struts.

63

APACHE PROJECT

APACHE

- The Apache project is a group of open source software for the Java platform licensed under the Apache license.
- Apache Software Foundation has led community projects since 1999.
  - There are over 5000 non-ASF projects in SourceForge.net
  - In addition 25% of projects hosted on Google Code use Apache License as well.

64

As of July 2009 for Sourceforge.
Google code was taken from a May 2008 blog

STRUTS 1 POPULARITY

STRUTS 1

- The original Struts project was successful in gaining contributors, users, and essentially unifying Java developers on a web framework in early to mid 2000s.
- The accomplishment was successful enough that until about 2005 almost every Java web job listing listed Struts 1. (many still do in fact)
- Struts 1 was the ubiquitous Java web framework.

65

MISTAKES OF STRUTS 1

STRUTS 1

- Creating an application in Struts while faster than creating an application by hand still could take at least a day to set up and get running from scratch.

- This seemed the norm for most developers until a video was posted showing how to create a CRUD framework from scratch in ~10 minutes ... Ruby on Rails

- Struts gave us some convention over configuration but it was still slower and some concepts by 2005 were dated.

66

BLOAT OF STRUTS

MISTAKES OF STRUTS 1

- Action classes required explicit forms to be passed in, entire mappings to passing, request and response always passed in. Much of this information can either be derived from simpler return calls and much of it was not always needed. Responses are very rarely needed unless you are having to do redirects or writing to it directly..

- Action classes in Struts 1 were Singletons, meaning any global variables would be accessed by any callers.

67

<<Refer to in class or ask if you need any more questions>>

BLOAT OF STRUTS

MISTAKES OF STRUTS 1

- Most of the work that drove Struts 1 Actions was the RequestProcessor. The processor handled transformations, validation, calling of the action and finally the forwarding of the data.

- The RequestProcessor was not a class that was EASY to change, there were very few natural hooks into the class itself. Altering it required extending the class and copying much of the code.

- View code was hard coded to always need JSPs if you wanted to make use of Struts components.

68

CHANGES NEEDED

MISTAKES OF STRUTS 1

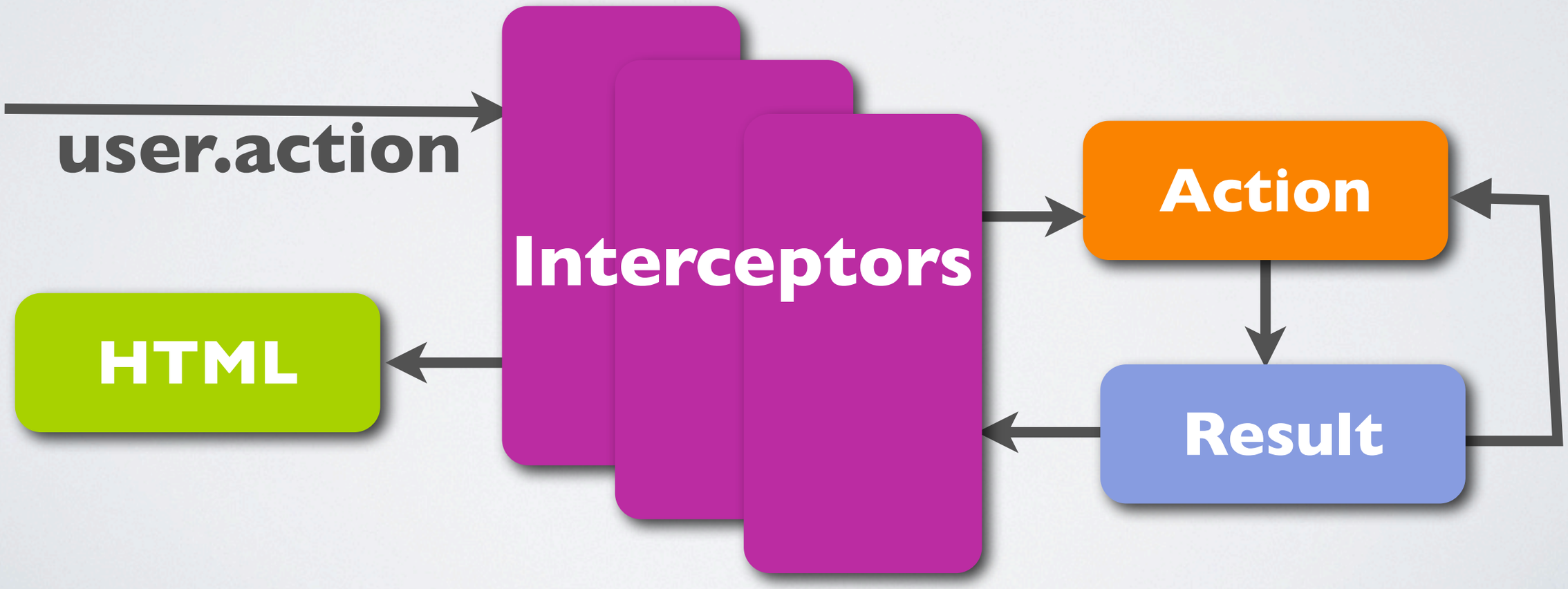
- 3 Big Areas needed to be addressed by Struts 2 that Struts 1 did not.
  - Allowing an easier way to alter the request processing flow.
  - Lighten the weight of the Action class
  - Provide more customization and flexibility with the results and view.

69

STRUTS LIFE CYCLE

STRUTS

- Basic lifecycle of the flow of Struts 2.
- Interceptors replaced much of the Request Processor.



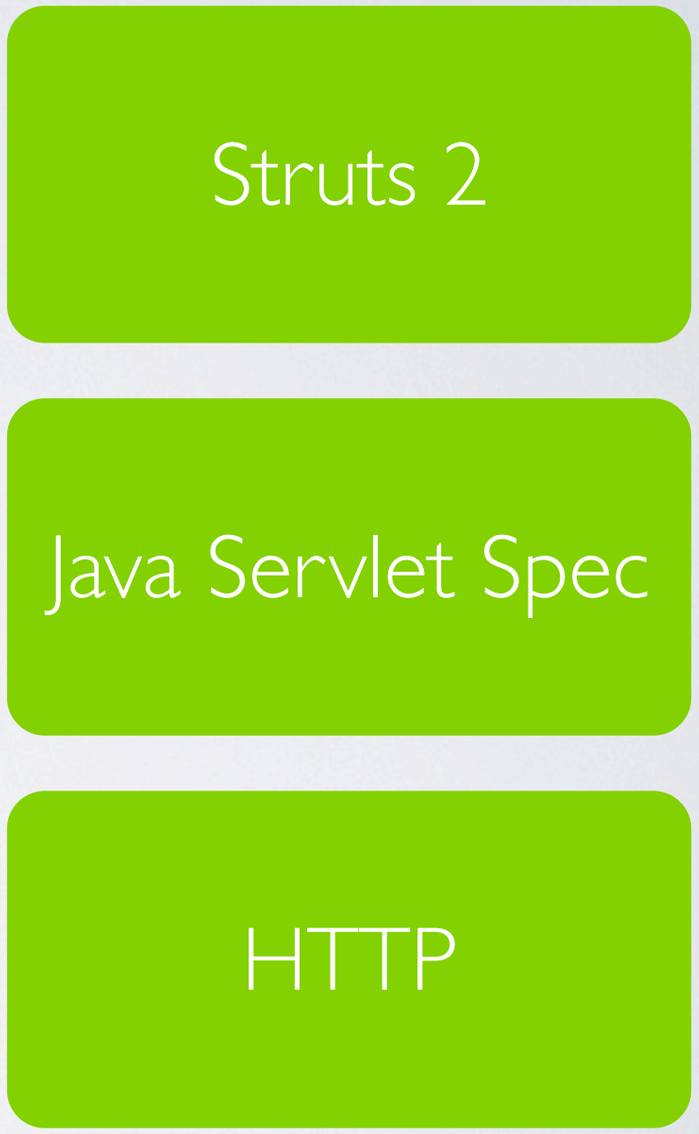
70

Beginning of Struts 2 Explanation

STRUTS STACK

STRUTS MVC COMPONENTS

- Struts sits on top of the Java Servlet Spec and the HTTP spec. This makes it so all the underlying programming of interactions is done for us.
- The JSS sits on top of HTTP programming it and Struts 2 uses the Servlet API directly so that except in rare cases we don't have to.



HTTP was not really designed for a stateless implementation. And a text based implementation. The either of its web and dynamic nature has been given some issues. We will next view the various areas of Struts/

HTTP STRUCTURE

STRUTS MVC COMPONENTS

- The web and HTTP is an entirely text based structure. And our interactions with it have to be text based.
- This has presented challenges as Java developers where we deal with actual types. This is because we need to convert the type from the web into a type that Java will understand.
- This requires the framework to perform data binding, validation, etc.

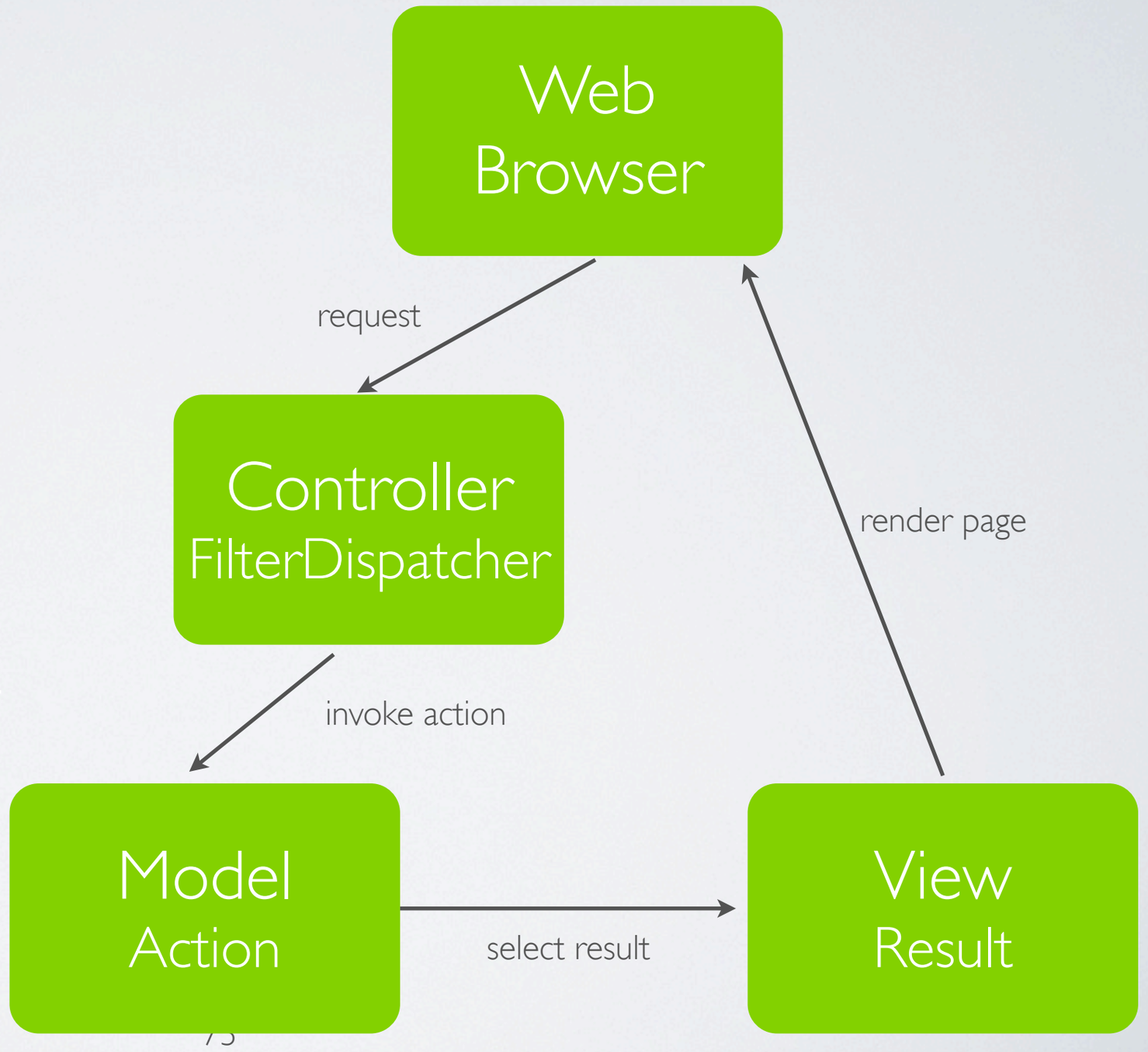
72

Challenge in creating a framework.

STRUTS 2 MVC PARTS

STRUTS MVC COMPONENTS

- model - action
- view - result
- controller - FilterDispatcher



FILTER DISPATCHER

STRUTS CONTROLLER

- In the original Sun master blueprint this is considered the front controller.
- The job of the controller is to take in requests from the HTTP and via configurations determine the appropriate action to route the request to.
- Its import that this layer only performs routing of requests and NOT business logic.

74

ACTION

STRUTS MODEL

- The model is the harder part to encapsulate how we use it, this is the internal state of the application. It handles 2 major items.

- 1st: it encapsulates the business logic into a single unit of work

- 2nd: the action is used to transfer the data between the layers.

75

ACTION

STRUTS MODEL

- The Action class in Struts 2 bears little resemblance to the Action class in Struts 1. The only similarity is they are both the controllers.

- The Action's main goal is to receive data from the view, validate if necessary, then send it to the business tier for processing, and finally return to the framework which view to render and the model information to render with.

76

RESULT

STRUTS VIEW

- The view is probably the easiest of the three items to not only understand but to see. The result tells the framework which display results to render.
- In Struts and other Java technologies this is a JSP page (or some derivate of it), Velocity templates, FreeMarker, XSLT, etc.
- With Struts swapping out this view component is fairly simple and we can easily substitute in different result types if needed.

77

Out of box support is provided for JSP, Velocity, FreeMarker, XSLT

STRUTS FRAMEWORK

STRUTS MVC

- The MVC Stack described only represents the MVC portion at the fine level. Struts itself provides other wrappers and integration points in the flow.

- The following will describe how Struts gets incorporated from beginning to end to process a request.

78

STRUTS WORKFLOW

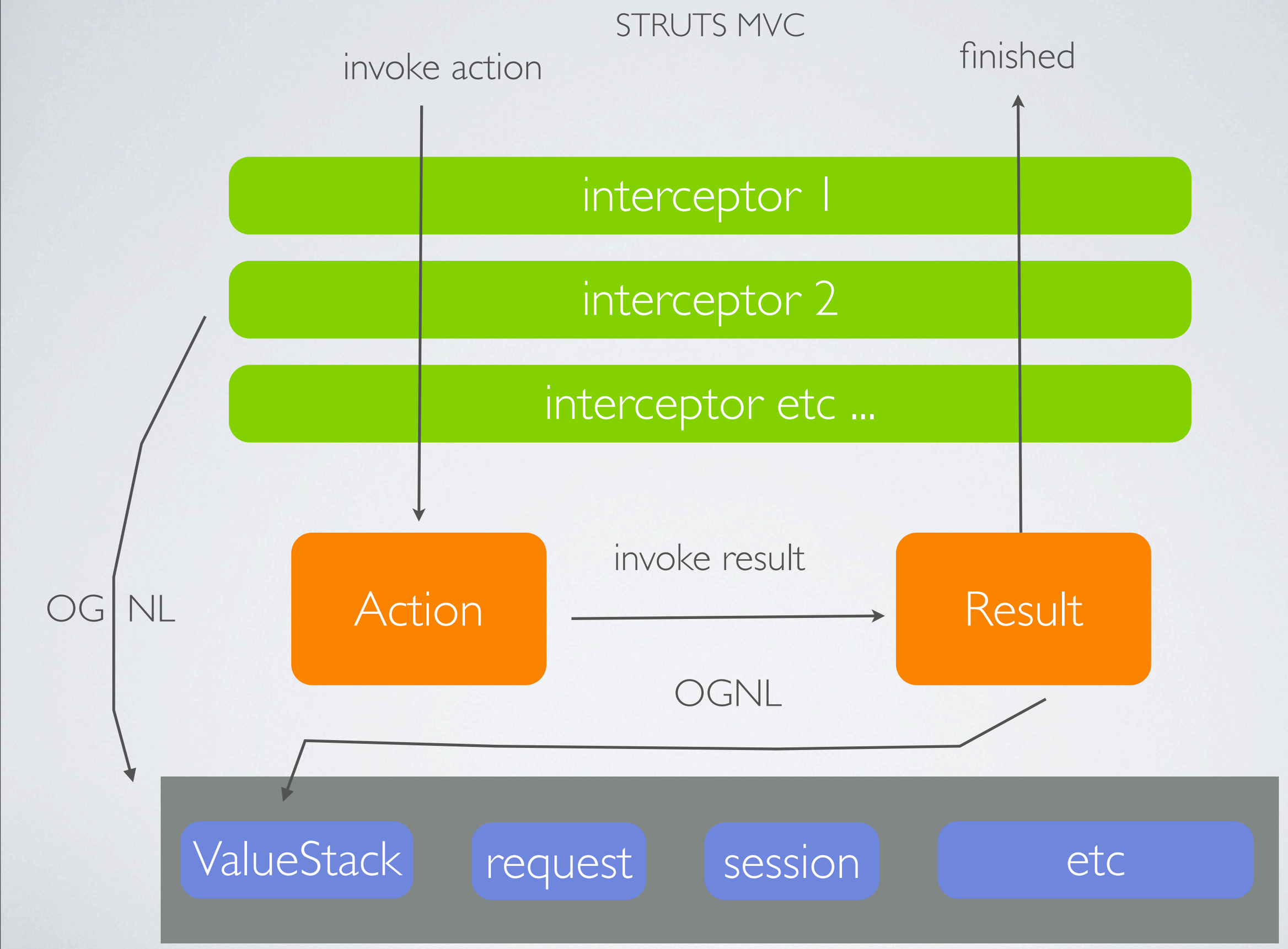


Diagram borrowed from Struts in Action page 15
Each of these areas will be explained in more details now and we will use these areas throughout the class.

INTERCEPTORS

STRUTS MVC

- Pluggable interceptors have become increasingly more popular in the realm of creating java based architecture.

- The interceptors inside of Struts are built on a stack that is fired before and after the action is performed. This allows each interceptor to make changes before and after the action is fired.

- Interceptors are pluggable and changing which interceptors are fired only requires an XML change.

80

INTERCEPTORS

STRUTS MVC

- There are a variety of interceptors provided by Struts, in addition we also have the ability to create our own interceptors.
- A few of the built in interceptors are :
  - exception, servletConfig, checkbox, logging, conversionError, workflow
- Interceptors will be covered in greater depth in Part 3.

81

The type of items that should be in interceptors are items that are generally soo common they should not be in the action itself.

VALUE STACK

STRUTS MVC

- ValueStack holds all the data associated with the request.
- Data is stored inside of the ValueStack for all application domain data that will be needed for a request.
- If you are coming from Struts 1 you may find the ValueStack and the OGNL components to be a significant change from the way data was handled previously.

82

OGNL

STRUTS MVC

- Object Graphical Navigation Library (OGNL) is the component that will access the data. The OGNL can use expression languages to access the data outside of it (much like in Tapestry 4).

- You will use the OGNL in various parts of the application to access the data in the ValueStack for display or computation use.

83

FIRST STRUTS 2 APP

STRUTS MVC

- Lets now step through and see how we add Struts 2 to an existing web application project.
- We will use Maven to step through the process of adding Struts 2 to an application.

84

MAVEN FOR STRUTS

STRUTS

- The Struts 2 components will first have to be added to our application via the pom.xml

- Add the Struts 2 dependency

```
<dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts2-core</artifactId>
    <version>${struts2.version}</version>
</dependency>
```


85

WEB.XML

STRUTS

- Filters are Java core supported features for the web that allow for all or predefined URLs to be intercepted.
- The filters will allow the Struts application to gain a hook into the Java web flow.
- This is the common architecture for Java web happens.
- Thus you will have to define a Struts filter dispatcher in the web.xml.

86

FILTER MAPPING

WEB.XML


```
<filter>
   <filter-name>struts2</filter-name>
   <filter-class>
   org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
   </filter-class>

   <init-param>
      <param-name>actionPackages</param-name>
      <param-value>com.integrallis</param-value>
   </init-param>
</filter>

<filter-mapping>
   <filter-name>struts2</filter-name>
   <url-pattern>/*</url-pattern>>
</filter-mapping>
```


87

You can see in here we use the filter dispatcher.
/* will have EVERY item intercepted by the mapping.
ActionPackages is only necessary if you want to use annotations.
Include whatever other servlets or filters u want in here.
StrutsPrepareAndExecuteFilter is new since 2.1.3

FILTER DISPATCHER

STRUTS

- This is the central point to intercepting all of the requests.
- This will intercept both Struts and non struts requests in the way we defined it.
- Requests marked with /struts/** will be sent and looked up via static content.
- With Struts it will forward anything mark with *.action to the Action classes. (this can be changed)

88

The framework itself will add the .action to the end when looking at requests

STRUTS.XML

STRUTS

- The struts.xml file is the core configuration file that is specific and needed for struts.
- The file itself can contain one large file with all the necessary configurations, or we can break up parts of it into sub files. The sub files can be good for very large applications.
- This file will define all the configurations needed for the application.

89

CONTENTS OF STRUTS XML

STRUTS.XML

- Constants - these are constants that can be used in the application or configurations.
- Interceptors - the interceptors are items that will fire before the Action
- Action - definitions of the action classes and all the options that exist for the action classes.
- Result Types - any additional global result types.

90

Ehere are about 4 items really defined in here.
There are of course more items, include files etc.

BASIC STRUTS FILE

STRUTS.XML

- The basic struts file contains the struts definition and maybe a few configuration items:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts
Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>
   <constant name="struts.devMode" value="true" />

   <!-- Package Here -->
</struts>
```


91

We will add our actions and other interceptors there

PACKAGE DEFINITION

STRUTS.XML

- Add a package definition. This is going to define two things, the namespace to put the class in and the package it extends.
- For right now we will extend "struts-default" this is a predefined package that comes with Struts 2.

```
<package name="default"
         namespace="/" extends="struts-default">
    <!-- Action-->
</package>
```


92

IMPLEMENTING ACTIONS

CREATING AN ACTION CLASS

ACTIONS

- Actions are the core components for interacting between the web and the services tier.
- Actions and the interceptors is where most of the customized web code will reside.
- These will provide all the tools for interacting with Services and sending back to the view.

94

MOST BASIC ACTION

STRUTS

- The most basic action is an action that does nothing but forwards onto a JSP page. This type of action is 100% defined in the XML and no other class is needed.

- This is how you would forward to a hello world page with the url /Hello

```
<action name="Hello">
    <result>/pages/hello-world.jsp</result>
</action>
```


95

URL STRUCTURE

STRUTS

- The combination of our package and action name will define the URL that will be needed to access the action.

```
<package name="default" extends="struts-default"
         namespace="/example">
  <action name=”Hello”>
    <result>/pages/hello-world.jsp</result>
  </action>
</package>
```

- The corresponding URL will be a combination of the namespace + the action name, so in this case the URL would be :

```
/example/Hello.action
```


96

please note in addition before that whatever you deployed the war as and set up as would also prefix this.

MOST BASIC ACTION

STRUTS

- The most basic action is an action that does nothing but forwards onto a JSP page. This type of action is 100% defined in the XML and no other class is needed.

- This is how you would forward to a hello world page with the url /Hello

```
<action name="Hello">
    <result>/pages/hello-world.jsp</result>
</action>
```


97

OGNL DATA

STRUTS

- Data can be shared between the JSP and the Action class. This allows us to send data to the Action and to retrieve the data from the Action. All of this is part of the OGNL flow of data discussed earlier.

- Input boxes and links are the used to send the data to the Action.

- The way these correlate is by the name of the input.

98

INPUT DATA

STRUTS

- Data on the JSP can be inputted and retrieved from the Action class.
- Struts provides many components to aid in ge
- This is how you would forward to a hello world page with the url /Hello

```
<action name="Hello">
    <result>/pages/hello-world.jsp</result>
</action>
```


99

zift

DEFINING AN ACTION

STRUTS

- In Struts the Actions are defined by implementing the Action interface. The interface is an interface with one method and five static Strings. (**`com.opensymphony.xwork2.Action`**)

- The method "String execute()" takes no arguments and returns a String which is the result type. The result type is defined in the struts.xml and can be used to determine where to forward the next page to .

100

DEFAULT RESULT TYPES

STRUTS

- The 5 result types given are not the only result types that can be used, but they are common names to be used for result types.

  SUCCESS- used when the execution was successful
  ERROR - used when an execution caused an error
  INPUT - used when an execution was missing an input
  LOGIN - used when the user was not logged in
  NONE - used when action was successful but no view to forward to

101

CREATE ACTION

STRUTS

- To build an action we have to define the following:
  - The configuration files : struts.xml : needs to be updated to reference the Action class
  - The JSP file : that defines the input page to interaction with the action.
  - Also a variety of other XML and message property files can be defined as well.

102

FORM DATA

STRUTS

- The name of the input should then match a global level variable on the form complete with getters and setters.
- Struts provides a set of tag libraries to help in inputting the data with.
- These tag libraries will provide wrappers for the form, input, and the save.

103

These tag will become more useful tomorrow when we attach validation to them.

FORM DATA

STRUTS

Defines the URI for the tag library (struts-tag) is built into the system. Also defines the prefix

```
<%@ taglib prefix="s" uri="/struts-tags" %>

<s:form action="save">
    <s:textfield key="name"/>
    <s:submit/>
</s:form>
```

Action name defined in the struts.xml

The name on the key will correspond to a property on the action.

104

These tag will become more useful tomorrow when we attach validation to them.

STRUTS DEFINITION

STRUTS

The name that was referenced in the form.

```
<action name="save"
   class="com.integrallis.TodoAction">
   <result name="input">
     /todo/input.jsp
   </result>
   <result>/todo/finish.jsp</result>
</action>
```

The results that return from the action.

105

These tag will become more useful tomorrow when we attach validation to them.

Struts Action class imported and implemented


```
import com.opensymphony.xwork2.Action;

public class TodoAction implements Action {

  String name;
  TodoService service = new TodoService();

  public String execute() {
    Todo todo = new Todo();
    todo.setName(name);
    service.addTodo(todo);
    return SUCCESS;
  }

  // Getters and setters necessary
  public String getName() {
    return name;
  }

  public void setName(String s) {
    name = s;
  }
```

The return defines what result to forward to.

Name referenced from the JSP (note same key)

106

BASIC STRUTS APP

ACTIONS

- This application will build upon the existing application we built in lab 1. You will convert the existing application into a Struts application.
- The index page will now be used by a forwarding action.
- 2 additional actions will be created for add and list.

107

MORE ACTION CONFIGURATIONS

TESTING THE ACTION

ACTIONS

- When developing complete applications the parts that test the areas accessing outside systems are always harder to test.
- The main reason for this is either receiving or saving data you are accessing a system you do not have control over and therefor you will in some way have to mock out items.
- This is more so the case in web tiers because unless you use a tool like Selenium you can't access the web tier.

109

Please note you SHOULD mock with services the demand for that is not as much

TRADITIONAL MOCKING

TESTING THE ACTION

- With traditional mocking tools like JMock, EasyMock, Mockito we can test the Action class for its logic and that it returns the correct items.
- The downfall is if you create an Action correctly it generally won't be doing much besides an if/else for validation and return. And with Struts this is compounded more that much of the interesting aspects of what happens in the action happens in the Interceptors.

Question to ask yourself : So what do we do then?

STRUTS 2 JUNIT PLUGIN

TESTING THE ACTION

- In order to adequately test the Action tier without having to rely on a web testing container like Selenium we need to use an integration testing mechanism that is built into JUnit

- The Struts 2 JUnit plugin provides a testing framework that will test your Action and interceptors for a given URL.

- The plugin will read the struts.xml and load up the contexts. So that when you call the action class all the interceptors will be called as well.

SETTING UP IN MAVEN

TESTING THE ACTION

- Configuring the Junit plugin for struts in Maven.

```
<dependency>
    <groupId>org.apache.struts</groupId>
    <artifactId>struts2-junit-plugin
    </artifactId>
    <version>STRUTS_VERSION</version>
</dependency>
```


112

CREATE UNIT TEST

ACTIONS

- Then JUnit test case will extend StrutsTestCase (or StrutsSpringTestCase if you are using Spring service beans and want to test those in conjunction)

- The next item is to set the request parameters as if you were coming from the web:

```
request.setParameter("bean.userName", "Joseph")
request.setParameter("bean.lastName", "Nusairat")
```

- Then you create the proxy call by passing in the URL

```
ActionProxy proxy = getActionProxy("/finduser")
```


113

CREATE UNIT TEST

ACTIONS

- Now we can find the action for it and execute the action.

```
UserAction userAction
        = (UserAction) proxy.getAction()
String result = proxy.execute()
```

- With this information we can pick up a few items. For one we can determine the result and if the correct result was returned.

- In addition we can access the field errors by using: `userAction.getFieldErrors()`

HANDLING EXCEPTIONS

ACTIONS

- Exceptions in Struts 2 are handled via the interceptor layer. The interceptors are defined in the XML. The default stack we have chosen for now will use the exception handling interceptors. Greater detail of interceptors is handled in Part 2.

- The exceptions are configured in the struts.xml.

- The way to handle exceptions is by altering the result type that is returned. We can customize this result type on the action.

115

HANDLE EXCEPTIONS

STRUTS

- Configured exception handling globally :

```
<global-exception-mappings>
  <exception-mapping
    exception="java.lang.Exception" result="error"/>
</global-exception-mappings>
```

- Configured per action:

```
<action name="test">
  <interceptor-ref name="exception"/>
  <interceptor-ref name="basicStack"/>
  <exception-mapping
   exception="com.integrallis.CustomException"
   result="custom_error"/>
  <result name="custom_error">custom_error.ftl</result>
</action>
```


In here you notice we had the exception intercepotr toh te basic stack (its in the default stack)
Define the exception and the error for it.

YOUR TURN

EXCEPTIONS AND UNIT TEST

- Instead of trying and catching your service call, let it propagate out and handle it via the exception handling in the configurations.
- Also create unit test of that class. Unit test should test for two things.
- 1. Good case when everything is entered properly.
- 2. Bad case when the unit test fails and throws an exception

117

ACTION SUPPORT

ACTION SUPPORT

ACTIONS

- The ActionSupport class is a convenience class that provides default implementations of the Action interface. As well as a few other interfaces.

- Think of this as an Action+ class. This will provide some default implementations as well as some more advance items.

- This class should be used throughout the application instead of Action when wanting to implement all but the most basic concepts.

119

ANATOMY OF THE CLASS

ACTIONSUPPORT

- Action - a given since its a must
- Validetable - provides our calls for validation
- ValidetableAware - used to keep track of our error messages
- TextProvider - provides access to the resource bundle
- LocaleProvider - used if you want to override the locale, useful for localization

120

Serializable is also defined as well.
Many of the features here we will dive into tomorrow in more depth when we get into MEssages and Validation

ADDITIONAL INTERFACES

ACTION INTERFACES

- In general Struts provides support for classes by applying interfaces to the classes. The interfaces will signify to the container additional input needed for the class.

- As you will see on Day 3 we can make use of these interceptors for added behaviour and even create a few of our as well.

- But for right now the additional interfaces expose us more components in the Action.

121

Serializable is also defined as well.
Many of the features here we will dive into tomorrow in more depth when we get into MEssages and Validation

AWARE INTERFACES

ACTION INTERFACES

- These are a group of interfaces that end in Aware that expose various objects from the Servlet API
  ServletContextAware - Sets the ServletContext
  ServletRequestAware - Sets the HttpServletRequest
  ServletResponseAware - Sets the HttpServletResponse
  ParameterAware - Sets a map of the request parameters
  RequestAware - Sets a map of the request attributes
  SessionAware - Sets a map of the session attributes
  ApplicationAware - Sets a map of the application properties
  PrincipalAware - Sets the Principal object (for security)

122

These are injected via the default stack exposing various items on the ServletAPi

USING THE INTERFACES

ACTION INTERFACES

- Using the interfaces is fairly straight forward. In most cases they will pass in an object and its up to you to set it to a global variable.

```
public Example extends ActionSupport
   implements SessionAware {
     public String execute() { ... }

     Map<String,Object> sessionData;

     public void setSession(Map<String, Object> sess) {
       sessionDate = sess;
     }
}
```


123

These are injected via the default stack exposing various items on the ServletAPI.

ACTION SUPPORT

STRUTS

- Let's build ourselves a basic ActionSupport class and see what it provides for us over using just a regular Action class.
- I will show how to build a simple todo CRUD piece.
- In here implement one of the interfaces to access the session and use it to store the last Todo saved.
- Finally on the index page, always show that last Todo added.

124

MAVEN STRUTS ARCHETYPES

MAVEN ARCHETYPE

STRUTS

- Sometimes setting up the architecture and framework can be very difficult. It's one of the problems with Java enterprise frameworks we have to assemble our jars, wars, ears correctly.

- So to over come this problem we use convention over configuration.

- The most common way to do this in Java is with Maven.

126

MAVEN

STRUTS

- Sometimes setting up the architecture and framework can be very difficult. It's one of the problems with Java enterprise frameworks we have to assemble our jars, wars, ears correctly.
- So to over come this problem we use convention over configuration.
- The most common way to do this in Java is with Maven.
- Struts has multiple Maven Archetypes

127

Documentation :
http://struts.apache.org/2.2.1/docs/struts-2-maven-archetypes.html

WHATS AN ARCHETYPE?

MAVEN

- By default when maven creates a project it creates a generic architecture that leaves what XMLs and jars to add to your project. This still leaves quite a bit of work for yourself.

- Archetypes are plugins for maven that allow you to create a maven 2 project from an existing template.

- The templates then can put in the correct XMLs, JARs, etc that are needed for a project allowing you to focus on development.

128

STRUTS ARCHETYPE

MAVEN

- There can be various archetypes for a framework.
- There are a variety of different archetypes for struts, we will look at a few to figure out which one would work best for our application.

129

4 DIFFERENT ARCHETYPES

MAVEN

- There can be various archetypes for a framework.
- There are a variety of different archetypes for struts, we will look at a few to figure out which one would work best for our application.

130

BLANK ARECHETYPE

"STRUTS2-ARCHETYPE-BLANK"

- This is your most basic and minimal configuration possible
- This will provide all the XML configurations and validations.
- It will also provide a few example actions and unit testing framework for it.

131

STARTER ARECHETYPE

"STRUTS2-ARCHETYPE-STARTER"

- This essentially provides the same as the previous starter items but more.
- Also provides Sitemesh and Spring integration.
- Resource bundles for all levels.
- In addition this will provide validation and conversion examples.

PORTLET ARECHETYPE

"STRUTS2-ARCHETYPE-PORTLET"

- The portlet archetype is used to create a Struts based portlet system.
- This will create a minimally configured JSR 168 styled portlet system.
- This will be able to be deployed as either a servlet or portlet application.

133

PORTLET DB ARECHETYPE

"STRUTS2-ARCHETYPE-DBPORTLET"

- This will be like the previous item but will incorporate the ability to use Spring and Hsql for real database queries.
- This also provides built in caching mechanisms.
- And can be deployed as servlet or portlet as well.

134

BLANK ARECHETYPE

"STRUTS2-ARCHETYPE-CONVENTION"

- This is for convention based validation.
- This provides basic action examples.
- In addition the blank convention is made to be Google AppEngine aware.

135

PLUGIN ARECHETYPE

"STRUTS2-ARCHETYPE-PLUGIN"

- This archetype is used to create a plugin.
- This creates examples for new result types and XML-based configurations.

136

SHOW

MAVEN

- Lets take a look at what an Archetype creates and the resulting infrastructure from it.

```
mvn archetype:generate -B \
  -DgroupId=tutorial \
  -DartifactId=tutorial \
  -DarchetypeGroupId=org.apache.struts \
  -DarchetypeArtifactId=struts2-archetype-blank
  -DarchetypeVersion=<version>
```


137

Will probably need to run the command :

`mvn archetype:generate -DarchetypeCatalog=http://struts.apache.org/`

Depending on if we have the the settings.xml setp corectly

TOUR

MAVEN

- We will take a tour of the different archetypes and the different archetypes artifacts that they create.

138