Tomcat

The Apache Jakarta Project

- The Apache Jakarta Project "creates and maintains open source solutions on the Java platform for distribution to the public at no charge"
- Apache Jakarta Tomcat--or just "Tomcat"--is one of those projects
- Tomcat is a container for servlets
  - Tomcat can act as a simple standalone server for Web applications that use HTML, servlets, and JSP
  - Apache is an industrial-strength, highly optimized server that can be extended with Tomcat

Getting Tomcat

- The Apache Jakarta website is hard to navigate
- If you want to get Tomcat, one reasonable download site is http://mirrors.xtria.com/apache/jakarta/tomcat-5/v5.0.29/bin/
- You would need the whole "tarball", which will have a name such as jakarta-tomcat-5.0.29.tar.gz
- An excellent tutorial site is *Configuring & Using Apache Tomcat,* http://www.coreservlets.com/Apache-Tomcat-Tutorial/
    - This site also contains many examples you can use to test your installation
- Installing Tomcat by itself is much easier than installing Apache and then adding Tomcat to it

Web apps

- A web application is basically a web site that:
  - "Knows who you are"--it doesn't just give you static pages, it interacts with you
  - Can permanently change data (such as in a database)
- A web application can consist of multiple pieces
  - Static web pages (possibly containing forms)
  - Servlets
  - JSP
- Tomcat organizes all these parts into a single directory structure for each web application
  - ...but you have to help with the organization

Directories

- To create servlets, you really should have two directory structures:
  - A development directory, in which you can write and partially debug your code
  - A deployment directory, in which you put "live" code
- Tomcat requires a particular set of directories for your web application
  - It is extremely picky about having everything in the right place!
- Since your web application must typically co-exist with other web applications, you should use packages to avoid name conflicts
  - This further complicates the Tomcat directory structure

Packages

- A package statement in Java must be the very first line of code in the file
- Example:

  -
    ```
    package com.example.model;
    import javax.servlet.*;
    import javax.servlet.http.*;
    import java.io.*;

    public class MyServlet extends HttpServlet { ... }
    ```
- This implies that
  - This program is in a file named `MyServlet.java`, which is
    - in a directory named `model`, which is
    - in a directory named `example`, which is
    - in a directory named `com`

Tomcat directory structure

```
myApplicationDirectory/ -- this is your top level directory
  myWebForm.html
  myJspPage.jsp
  WEB-INF/ -- must have this directory, named exactly like this
    lib/ -- mostly for external .jar files
    classes/ -- must have this directory, named exactly like this
      com/ -- The com.example.model package directory
        example/
          model/
            myModel.class -- in package com.example.model;
          web/
            myServlet.class --in package com.example.web;
      web.xml -- this is the deployment descriptor, it must have this name
```

My files

- myWebForm.html
  - This is the web page with a form that starts up the servlet
- com/example/web/myServlet.class
  - This is the servlet I intend to use; it will use the myModel class, but to do this it needs an import statement:
    import com.example.model.myModel;
- com/example/model/myModel.class
  - This does the “business logic” it is good form to keep it separate
- myJspPage.jsp
  - The (optional) JSP page to create the HTML output (could be done directly by myServlet)
- web.xml
  - A file *required* by Tomcat to tell it what class to start with and how to refer to that class

myWebForm.html

```
<html>
  ...
  <body>
    ...
    <form method="POST" action="NameSeenByUser.do">
      ...various form elements...
    </form>
    ...
  </body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation=
             "http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
         version="2.4">

  <servlet>
    <servlet-name>Some internal name</servlet-name>
    <servlet-class>com.example.web.MyServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Some internal name</servlet-name>
    <url-pattern>/NameSeenByUser.do</url-pattern>
  </servlet-mapping>

</web-app>
```

Servlet without JSP

```
public class MyServlet extends HttpServlet {

    public void doPost(HttpServletRequest request,
                       HttpServletResponse response)
        throws IOException, ServletException {

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String value = request.getParameter("name");
      out.println("<html><body>I got: " + name + " = " +
                    value + "</body></html>");
    }
}
```

Servlet with JSP

```
public class MyServlet extends HttpServlet {

    public void doPost(HttpServletRequest request,
                                        HttpServletResponse response)
            throws IOException, ServletException {

        String value = request.getParameter("name");
        ...computation resulting in some Object obj...
        request.setAttribute(”objName", obj);
        RequestDispatcher view =
            request.getRequestDispatcher("result.jsp");
        view.forward(request, response);
    }
}
```

JSP (result.jsp)

```
<%@ page import="java.util.*" %>
<html>
<head><title>Your results</title></head>
<body>

<%
   MyObject object =
         (MyObject)request.getAttribute("objName");
   String someResult = ...computations using object...
   out.print("<br>And the answer is: " + someResult);
%>

</body>
</html>
```

Flow

- The user submits an HTML form
- Tomcat finds the servlet based on the URL and the deployment descriptor (`web.xml`) and passes the request to the servlet
- The servlet computes a response
- Either:
  - The servlet writes an HTML page containing the response
- Or:
  - The servlet forwards the response to the JSP
  - The JSP embeds the response in an HTML page
- Tomcat returns the HTML page to the user

Alternatives to Tomcat

- Sun’s Java Web Server
  - Old, no longer being developed, all in Java
- Java Web Server Development Kit (JWSDK)
  - Official reference implementation
  - Difficult to install and configure
- JBoss
  - Open source
  - Opinions vary on how easy it is to install
  - Comes with built-in database

The End