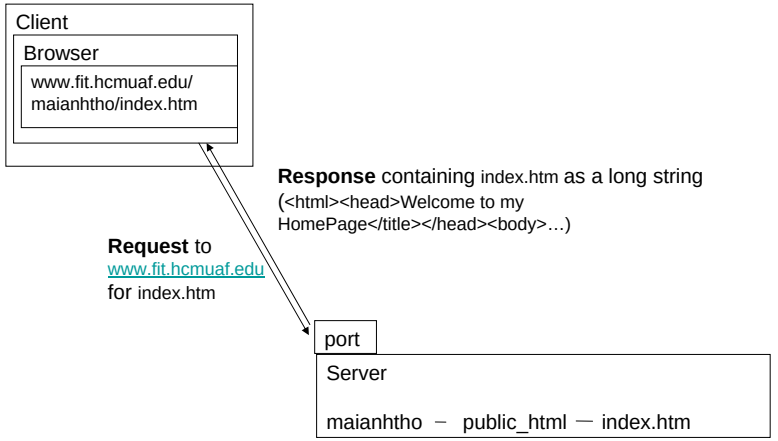
Server-side Web Development and Programming

Lecture 2:
The Jakarta Tomcat Web Container and the NetBeans IDE

Client-Server Web Architecture

- Client browser sends request for page to server
- Server sends response page and sends to client



Form Handling

Server must:

- Listen on port for requests
- Parse request to determine values of parameters
- Generate appropriate response page based on parameter values
- Send response page back to client

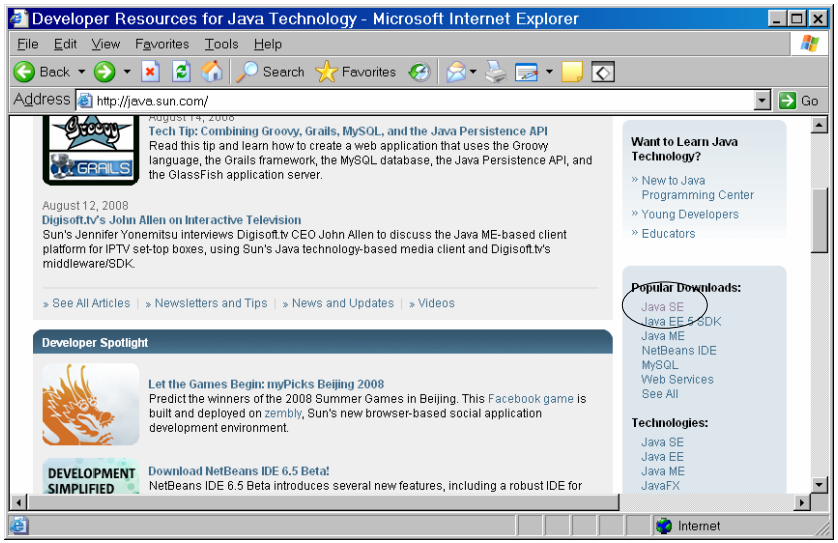
Web Containers

- Program running continuously on server
- Runs code to handle requests
- Built-in methods for parsing requests, generating responses
- Handles other important functions:
  - Session tracking
  - Database access
  - Email generation
  - Security and encryption

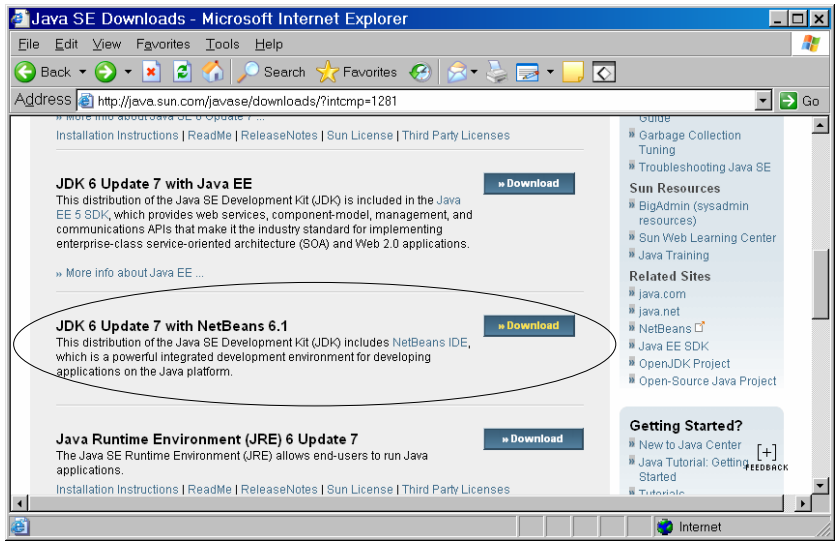
Web Containers

- Jakarta Tomcat
  - Written in Java
  - NetBeans IDE
  - Acts as engine for Java Server Pages and servlets
- Microsoft IIS
  - Visual Basic/Visual C++
  - Active Server Pages

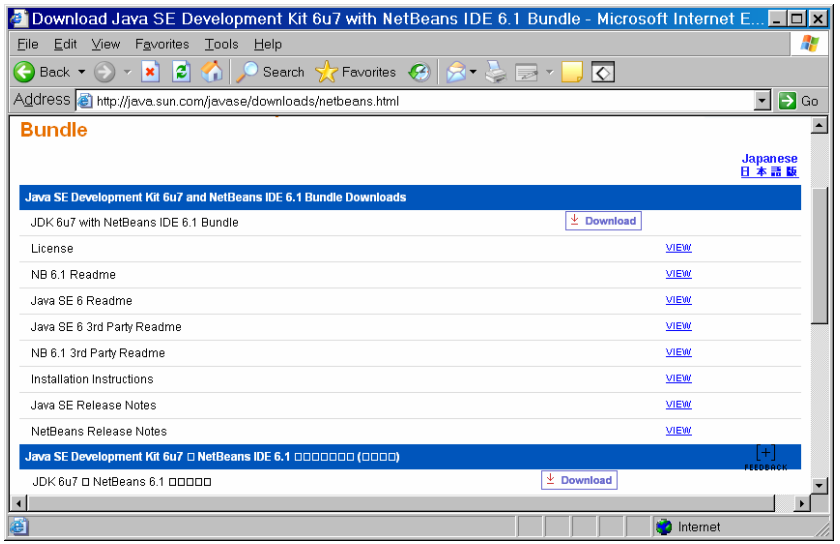
Downloading the Java SDK



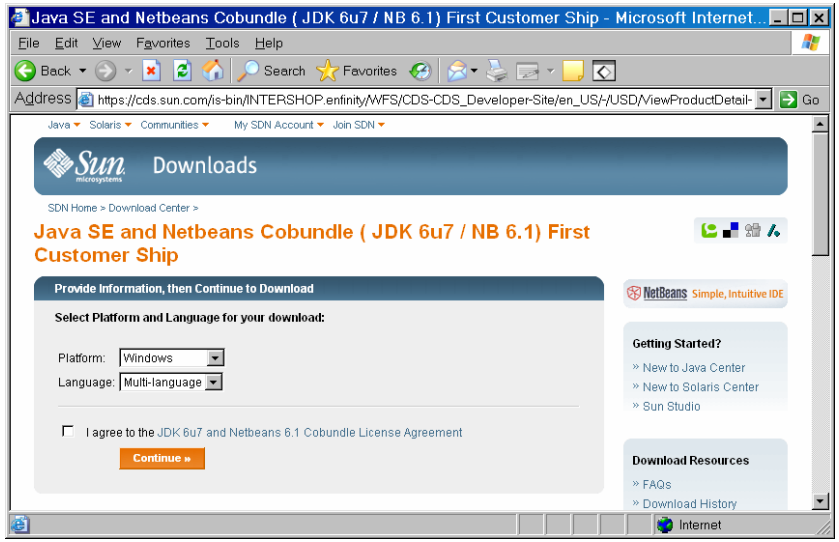
Downloading the Java SDK



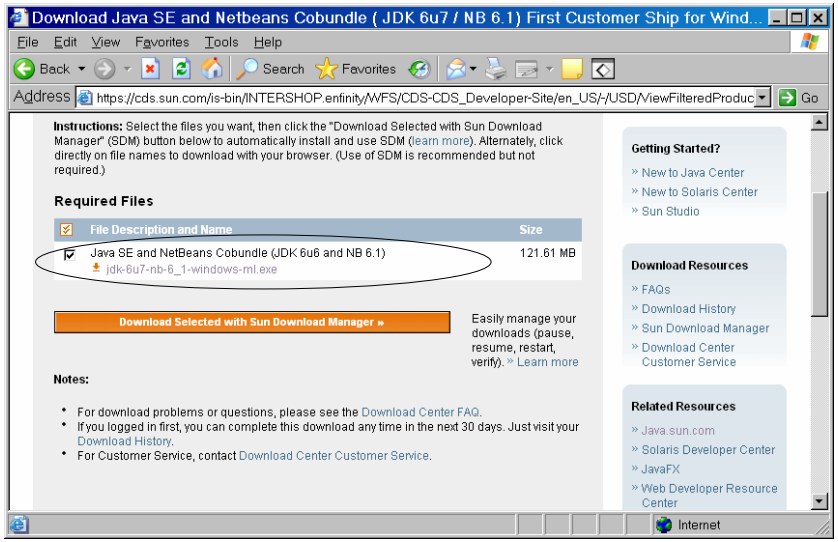
Downloading the Java SDK



Downloading the Java SDK

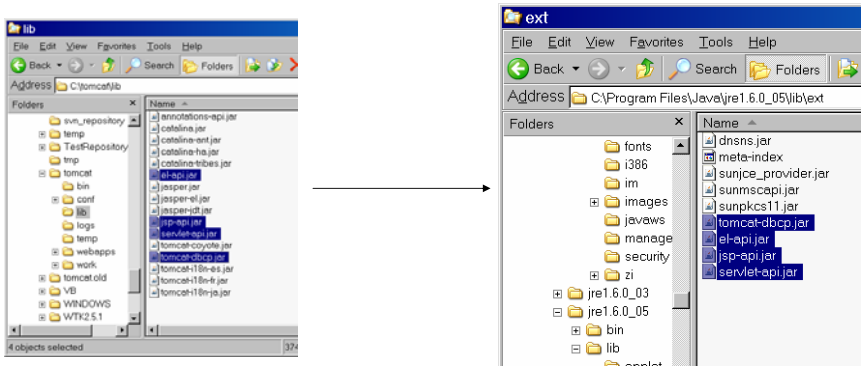


Downloading the Java SDK



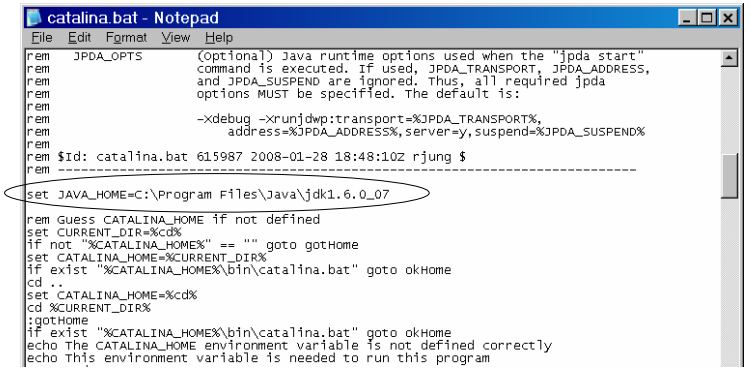
Installing Tomcat

- Detailed Instructions in Chapter 2 (page 31)
- Copy JAR files from Tomcat to Java Runtime Environment
  - Necessary for JSPs and servlets to compile



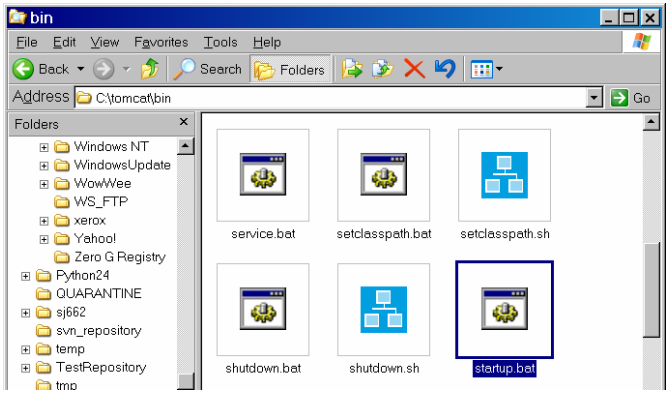
Installing Tomcat

- Tell Tomcat where to look for Java (page 34)
  - Edit catalina.bat file in bin directory of Tomcat



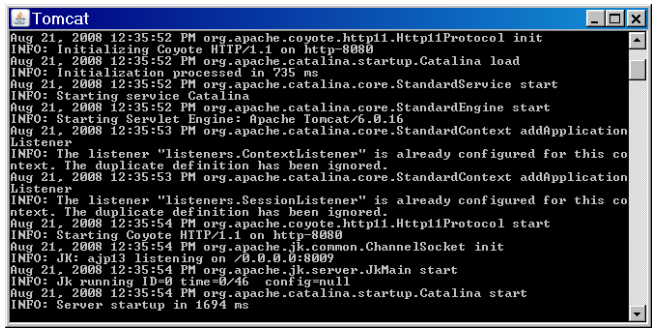
Testing Tomcat

- Start Tomcat
  - Execute statup.bat in bin directory



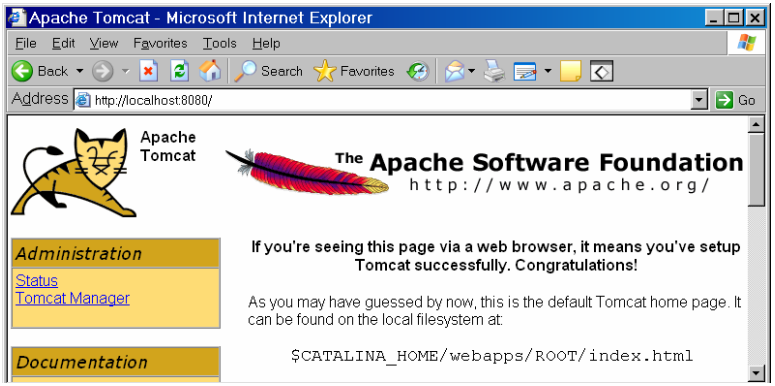
Testing Tomcat

- This will open Tomcat control window



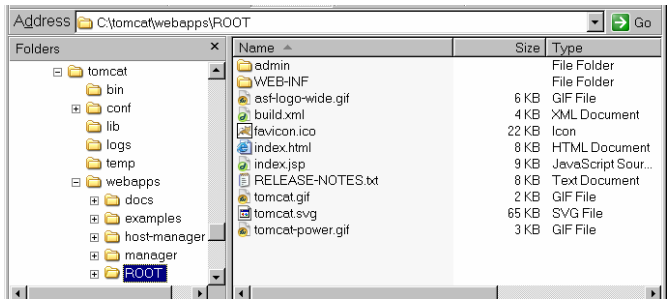
Testing Tomcat

- Tomcat is now listening for requests!
  - Listening at port 8080 by default
- Test: enter http://localhost:8080/ in your browser



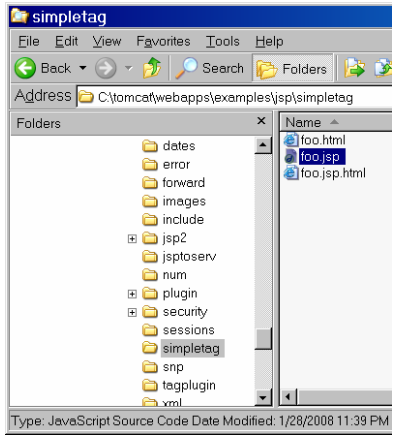
Tomcat Structure

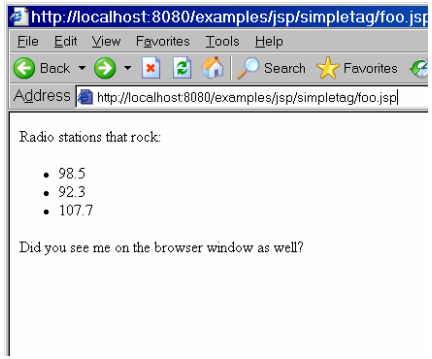
- Listens on port 8080 for requests
- Retrieves page requested
  - Must be part of its file structure in webapps directory
- Example: http://localhost:8080/
  - Sends request to this machine for index.html file in ROOT subdirectory of webapps



Tomcat Structure

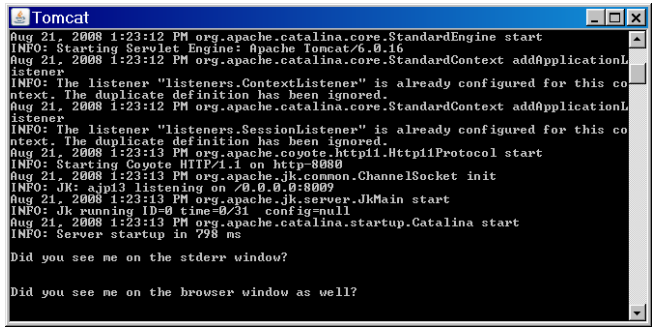
- Another example:
  http://localhost:8080/examples/jsp/simpletag/foo.jsp





Tomcat Structure

- Side point:
  JSPs/servlets can display messages in Tomcat window (often used for diagnostics)



Tomcat Structure

- Meaning of this URL:
  http://localhost:8080/examples/jsp/simpletag/foo.jsp

Invoke server listening on port 8080 of this machine

Access this file in this subdirectory of the tomcat/webapps directory

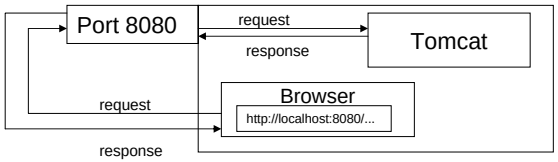
Tomcat Structure

- Side note: Usually refer to server on another machine
  http://www.hcmuaf.edu.vn/examples/jsp/simpletag/foo.jsp

Invoke server at this remote URL

Access this file in this subdirectory of the tomcat/webapps directory

- For testing, often run client and server on same machine



Tomcat Structure

- If requested page is server page/servlet, executes code embedded in page to generate corresponding html page
- Final html page sent as response

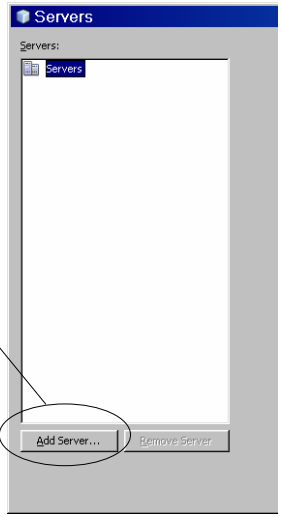
```
<HTML>
<HEAD><TITLE>cgi-bin
response</TITLE></HEAD>
<BODY>
<P>
Thank you for your order of
<%= request.getParameter("quantity") %>
widgets!
</P>
</BODY>
</HTML>
```

The NetBeans IDE

- Integrated Development Environment for Java Programming
  - Editing code (with hints)
  - Compiling code
  - Running code
- Good for JSP/servlet development
  - Allows easy development of web applications
  - Automatically interacts with Tomcat
  - No need to manipulate Tomcat file structure

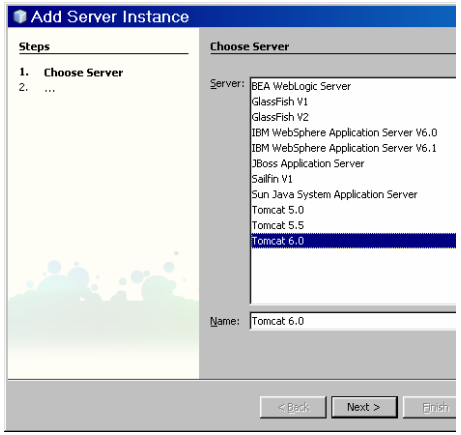
Adding a Tomcat Server

- Tools → Servers
- Press



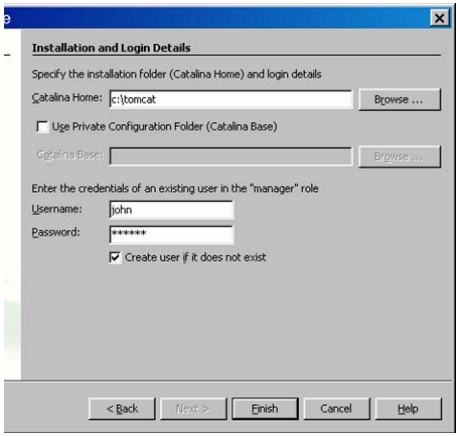
Adding a Tomcat Server

- Select Tomcat 6.0



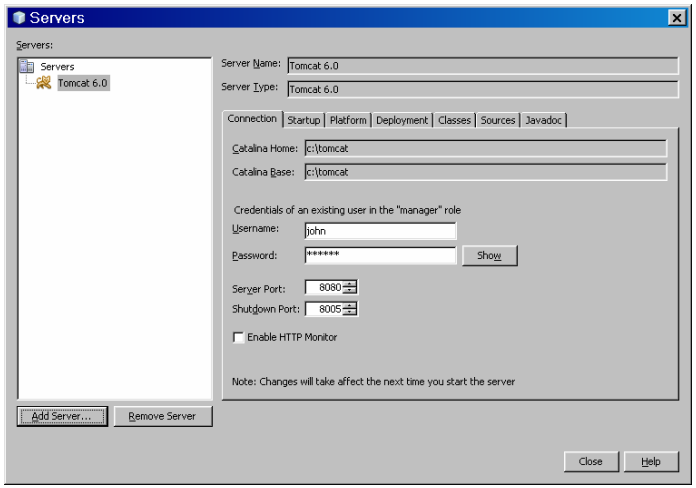
Adding a Tomcat Server

- Enter the directory where you installed Tomcat
- Enter a name and a password for a “manager” role (we will use this more later)



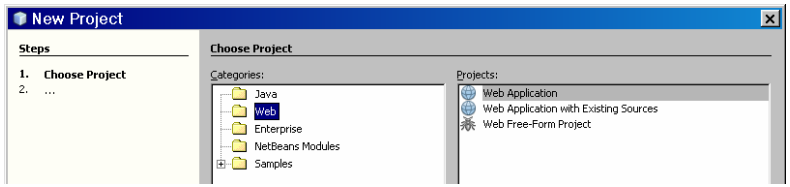
Adding a Tomcat Server

- By default, Tomcat listens at port 8080



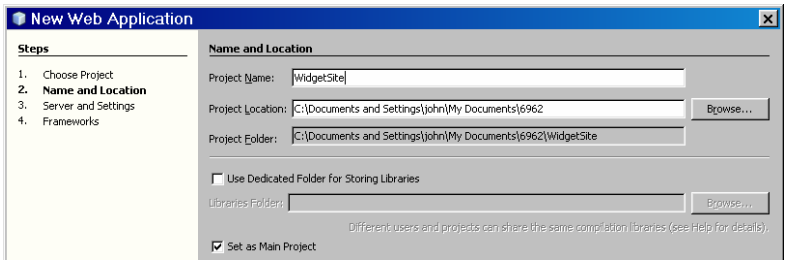
Creating a Web Application

- In NetBeans: File → New Project
- Choose Web and Web Application



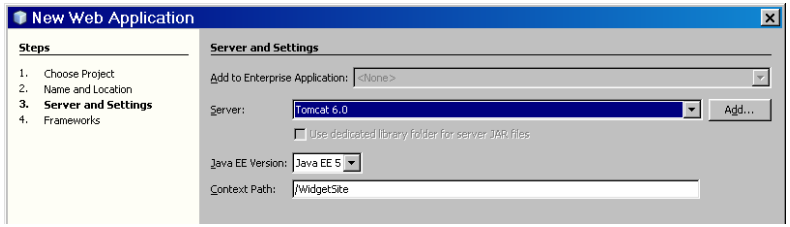
Creating a Web Application

- Give your project a name (I called this one “WidgetSite”)
- Give it a location (I have put it in a directory called 6962)
- Make sure it is set as the Main Project



Creating a Web Application

- The final page shows information (such as which server this project uses)
- You can press “finish” at this point

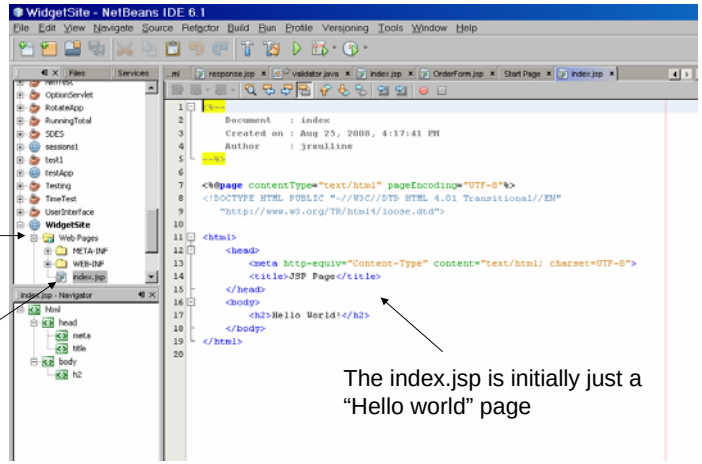


Creating a Web Application

NetBeans creates an initial web site

Structure shown in the project window

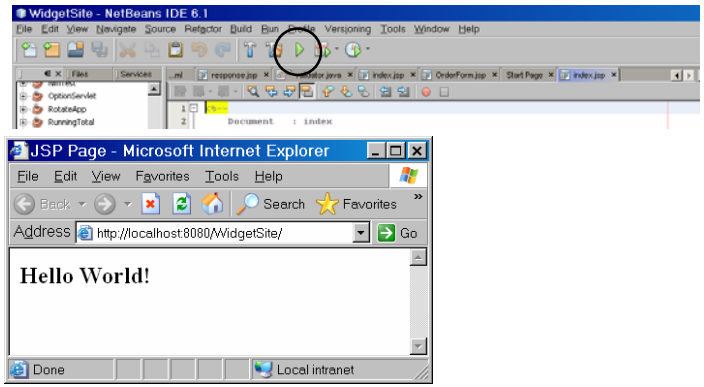
Creates an initial index.jsp page (default home page of the new site)



The index.jsp is initially just a “Hello world” page

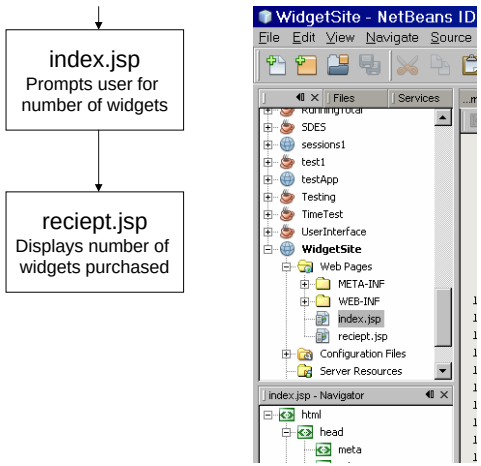
Running a Web Application

- Running the site opens the index.jsp page
  - Can choose browser (and should test with all!)

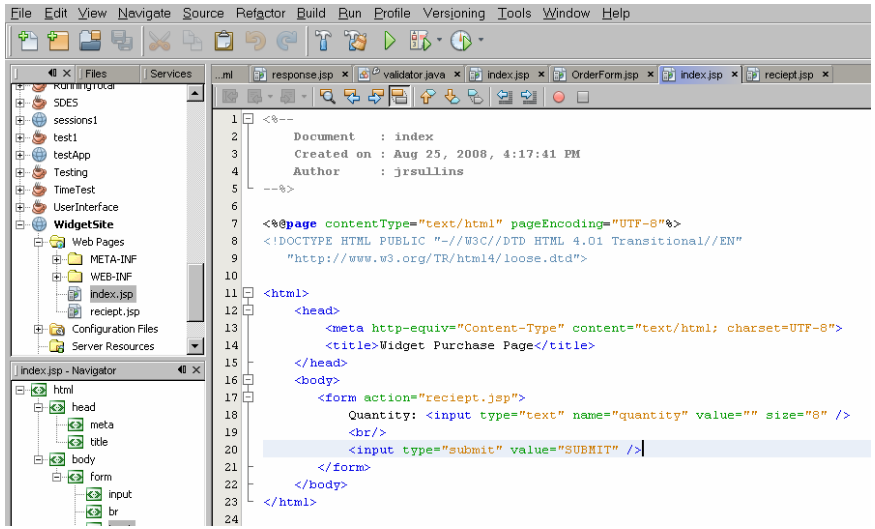


Building a Web Application

- Modify and add files to create web site

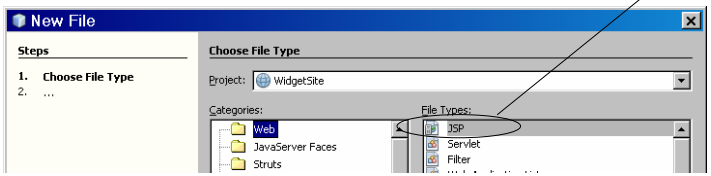


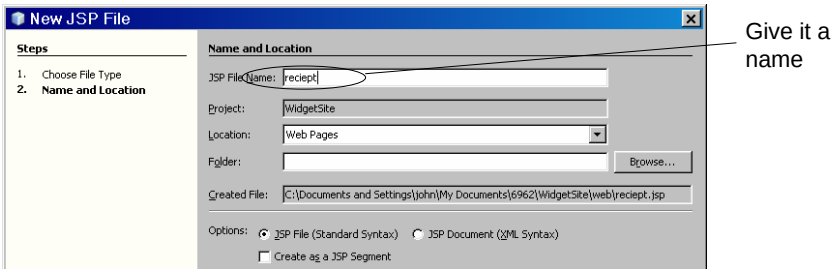
Building a Web Application



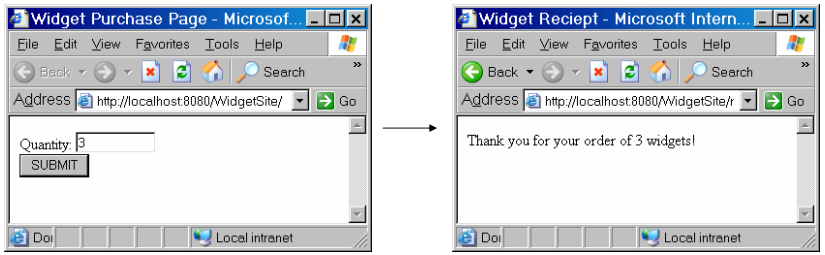
Adding a JSP

- File → New



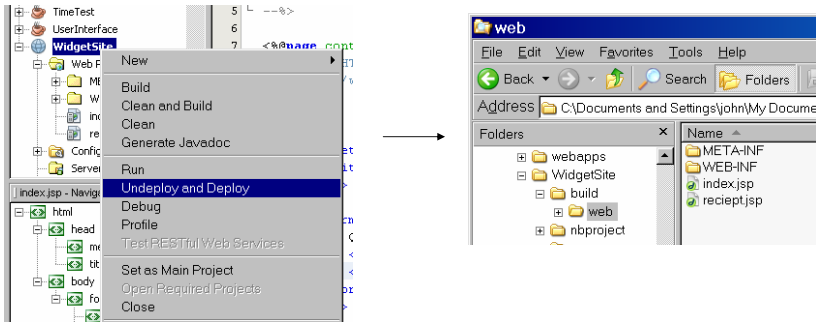


Running the Site



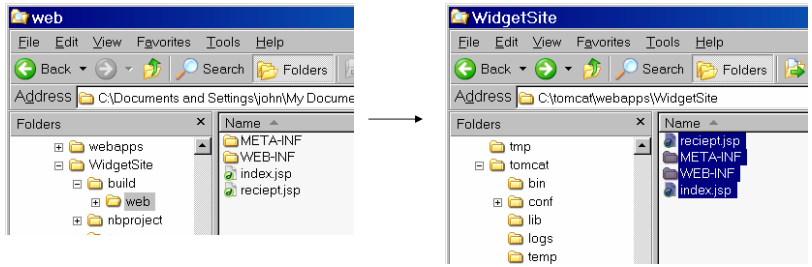
Deploying your Site to Tomcat

- Right-click project and choose “Deploy”
- This copies all web site files into <u>build\web</u> subdirectory



Deploying your Site to Tomcat

- Copy these files into a subdirectory of webapps in Tomcat



Deploying your Site to Tomcat

- Start Tomcat (after closing NetBeans)
- Go to http://localhost:8080/WidgetSite in browser