Developing JSPs and Servlets with Netbeans

Nick Shrine

<N.R.Shrine@cs.bham.ac.uk>

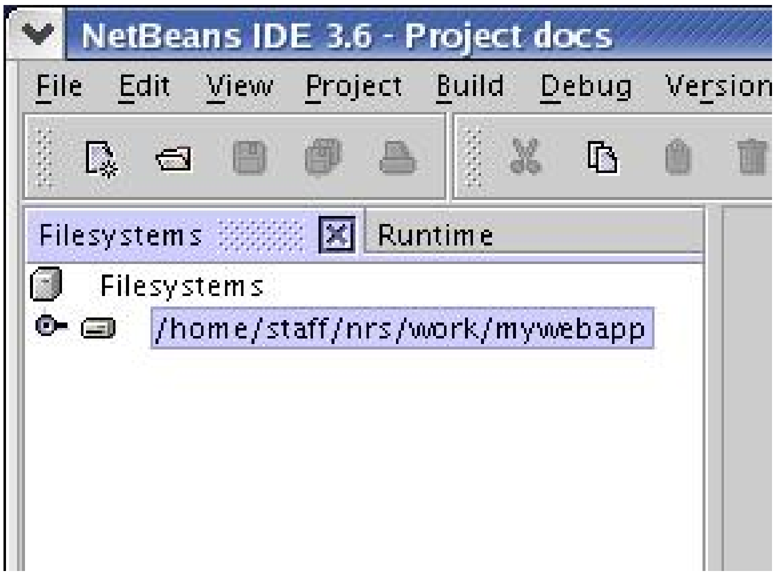
Printable (pdf) version [netbeans-webapps.pdf], XML source [netbeans-webapps.xml]

Netbeans comes with a built-in Tomcat server for development of JSPs and Servlets. It also has templates for Web Applications, JSP pages and Servlet classes and automatically updates your web application delplyoment descriptor (`web.xml`) when you add new servlets. It is also possible to run web applications in the debugger.

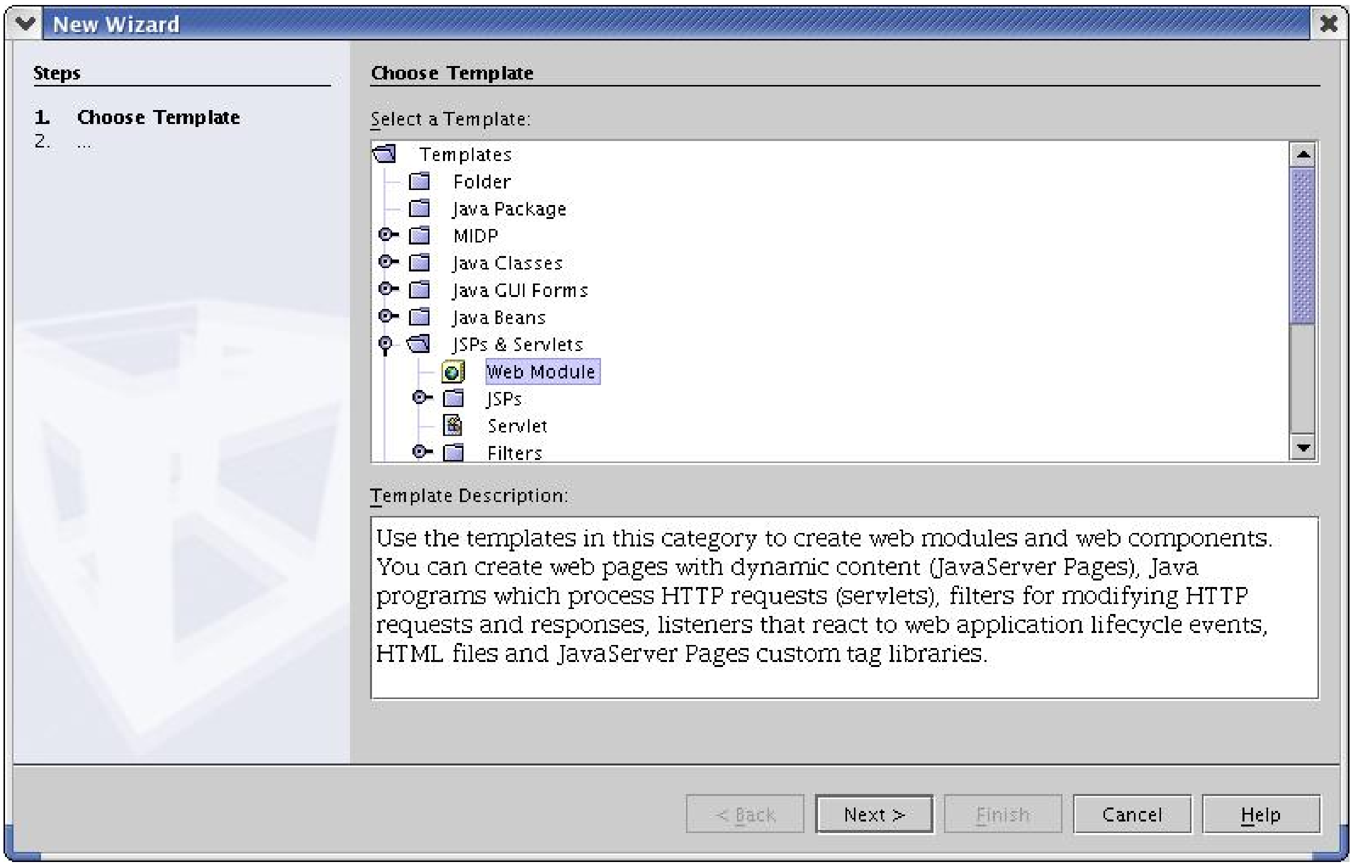
Table of Contents

1. Creating a new Web Application

1. Start a new project and create and mount a directory for your web application.

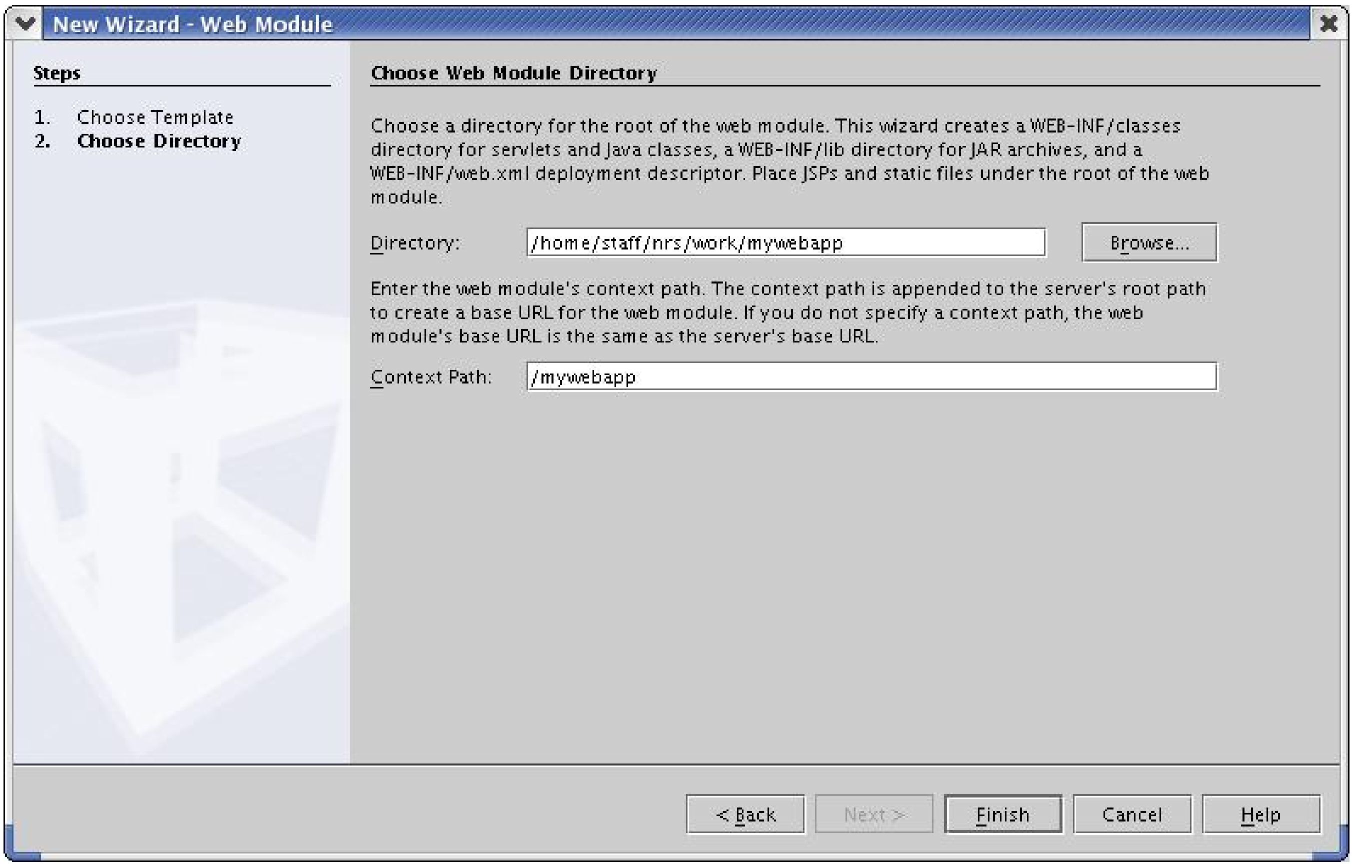


2. With the above directory selected, select: `File # New # JSPs & Servlets # Web Module`.



Use the templates in this category to create web modules and web components. You can create web pages with dynamic content (JavaServer Pages), Java programs which process HTTP requests (servlets), filters for modifying HTTP requests and responses, listeners that react to web application lifecycle events, HTML files and JavaServer Pages custom tag libraries.

3. Hit **Next**, the target directory in the subsequent dialog should be the one you mounted above.



Choose a directory for the root of the web module. This wizard creates a WEB-INF/classes directory for servlets and Java classes, a WEB-INF/lib directory for JAR archives, and a WEB-INF/web.xml deployment descriptor. Place JSPs and static files under the root of the web module.

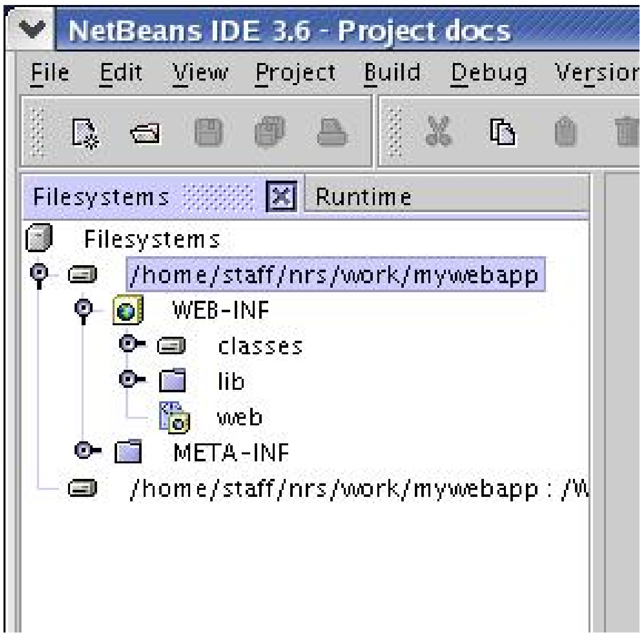
Directory: /home/staff/nrs/work/mywebapp

Enter the web module's context path. The context path is appended to the server's root path to create a base URL for the web module. If you do not specify a context path, the web module's base URL is the same as the server's base URL.

Context Path: /mywebapp

Now hit **Finish**.

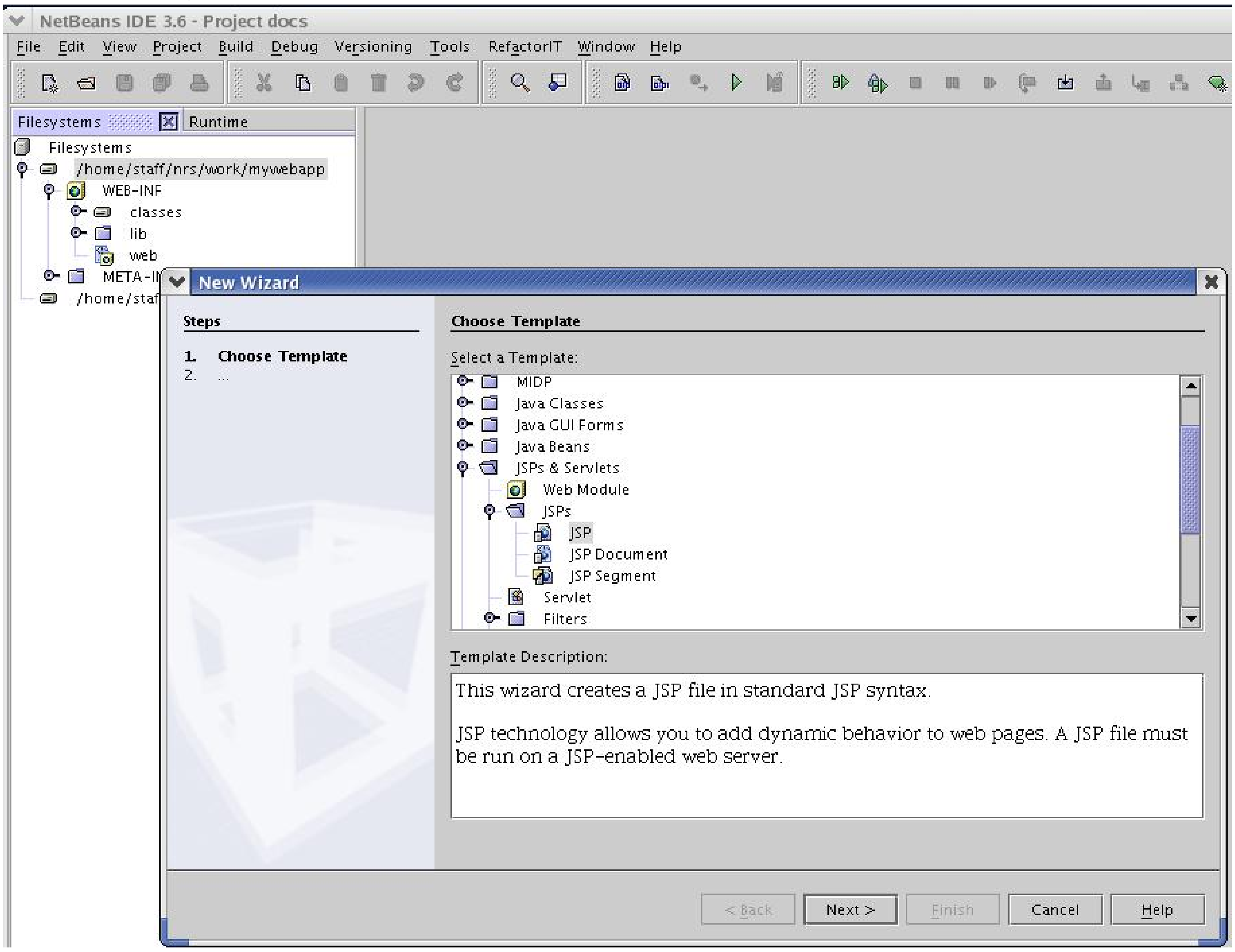
4. Now you should have a view in your Filesystem explorer like this:


NetBeans IDE 3.6 - Project docs
File Edit View Project Build Debug Version
Filesystems
Runtime
Filesystems
/home/staff/nrs/work/mywebapp
WEB-INF
classes
lib
web
META-INF
/home/staff/nrs/work/mywebapp : /W

- You put your `.html` and `.jsp` files in the top level directory, any file references in your JSP/servlets are relative to this directory.
- `WEB-INF/classes` is where you put your servlet classes.
- `WEB-INF/lib` is where you put any `.jar` files required by your application e.g. `postgresql.jar` for web applications that access a Postgresql database.
- `WEB-INF/web` is your `web.xml` file (double-click to edit).
- You can ignore the `META-INF` directory.
- Netbeans also remounts `WEB-INF/classes` at the bottom to give you a shortcut to your servlet package hierarchy.

2. Creating a new JSP page

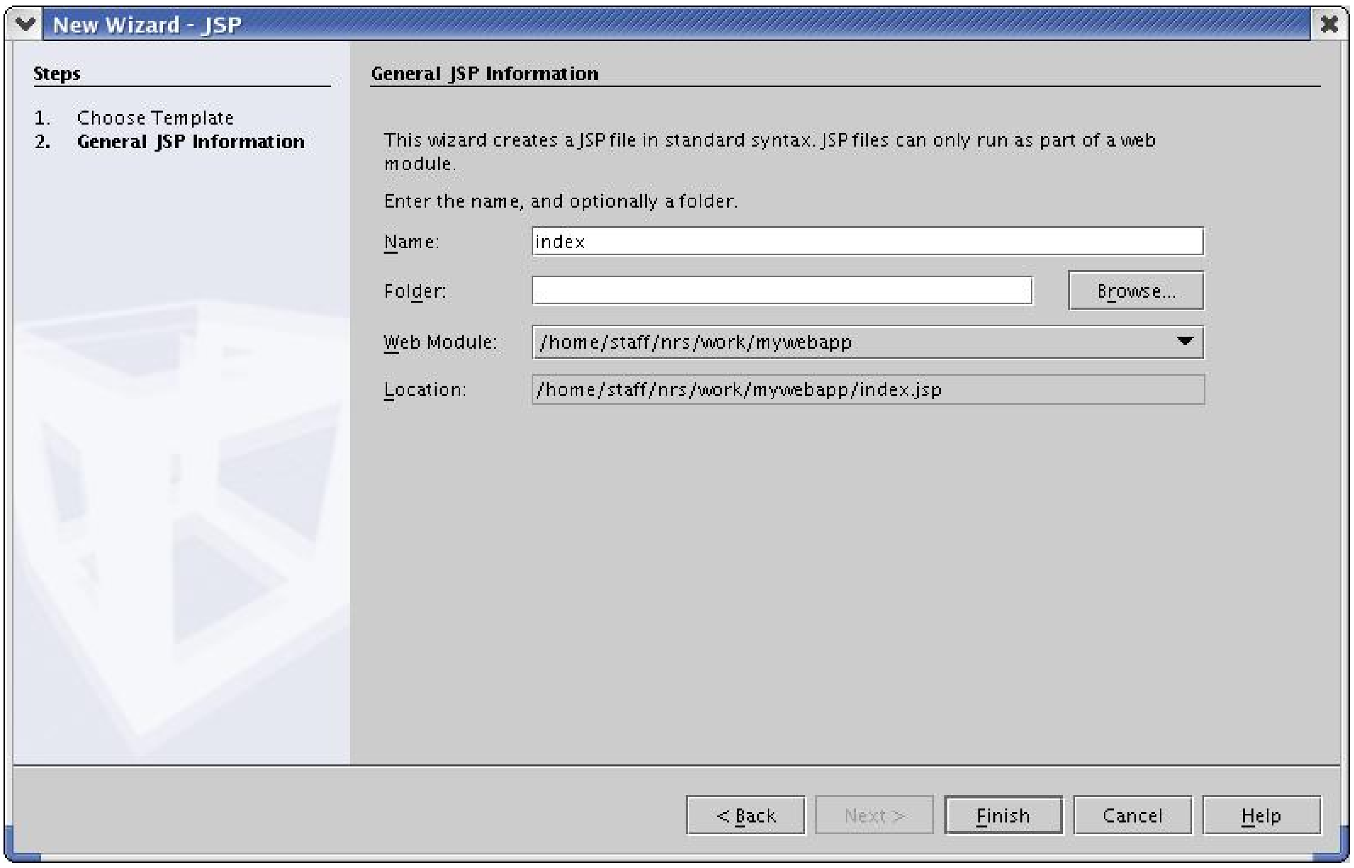
1. To create a JSP page select the top level folder in the explorer then do `File # New # JSPs & Servlets # JSP`.



This wizard creates a JSP file in standard JSP syntax.

JSP technology allows you to add dynamic behavior to web pages. A JSP file must be run on a JSP-enabled web server.

2. Hit **Next**.



This wizard creates a JSP file in standard syntax. JSP files can only run as part of a web module.

Enter the name, and optionally a folder.

Name: index

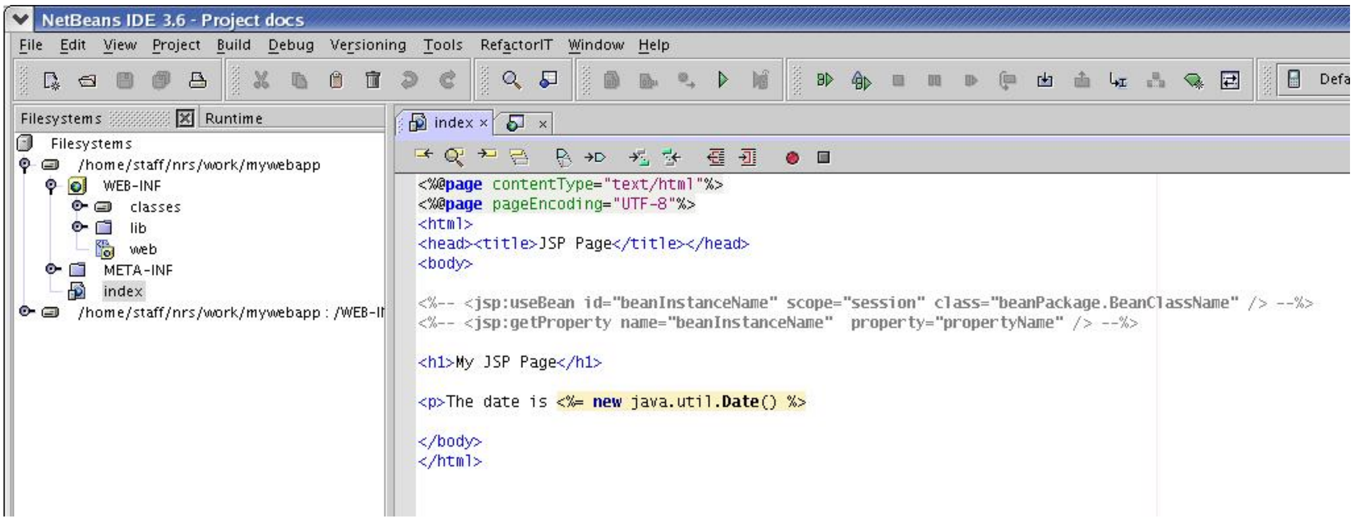
Folder:

Web Module: /home/staff/nrs/work/mywebapp

Location: /home/staff/nrs/work/mywebapp/index.jsp

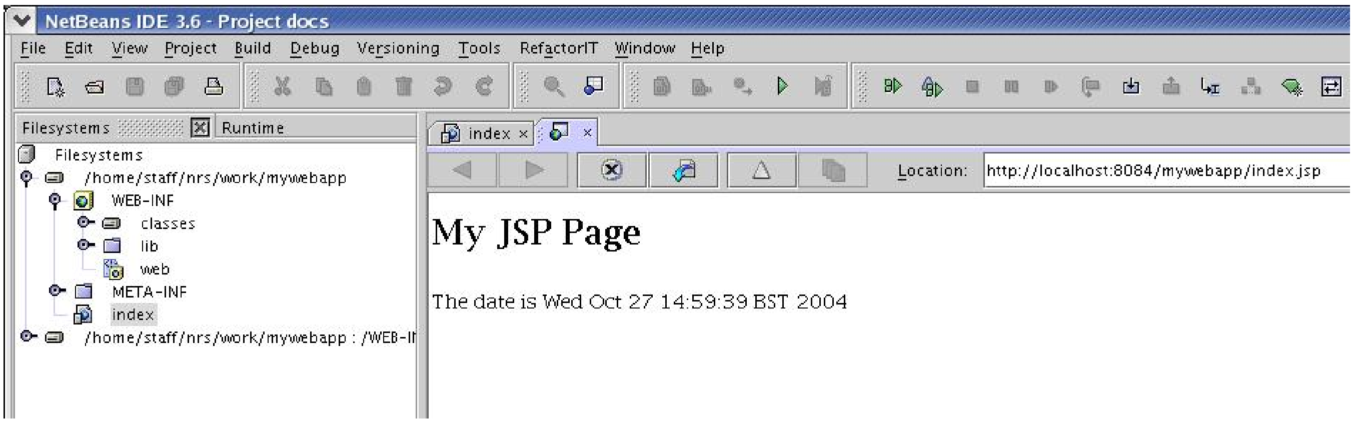
Enter a name for your JSP (without .jsp extension). You can select a subfolder to put it in if you want to organise your pages into subfolders. Then hit **Finish**:

3. Netbeans creates a skeleton JSP page comprising of little more than the `<head>` and `<body>` tags and a couple of commented-out sample bean directives. I have added the `<h1>` and `<p>` lines in the screenshot below.


NetBeans IDE 3.6 - Project docs
File Edit View Project Build Debug Versioning Tools RefactorIT Window Help
Filesystems
Runtime
Filesystems
/home/staff/nrs/work/mywebapp
WEB-INF
classes
lib
web
META-INF
index
/home/staff/nrs/work/mywebapp : /WEB-I
index

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
<head><title>JSP Page</title></head>
<body>

<%-- <jsp:useBean id="beanInstanceName" scope="session" class="beanPackage.BeanClassName" /> --%>
<%-- <jsp:getProperty name="beanInstanceName"  property="propertyName" /> --%>

<h1>My JSP Page</h1>

<p>The date is <%= new java.util.Date() %>

</body>
</html>
```

4. To run your JSP page select it in the explorer or source editor and hit **F6** or the button. You will either see the page in Netbeans internal browser:

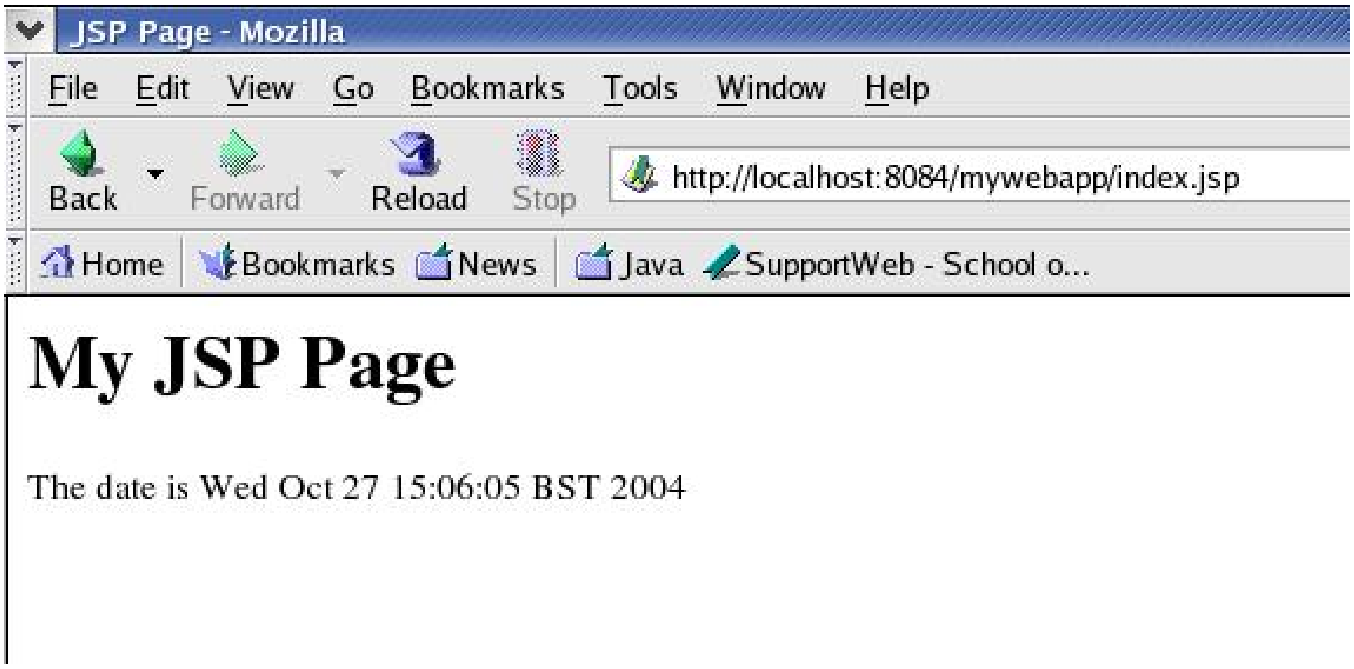

NetBeans IDE 3.6 - Project docs
File Edit View Project Build Debug Versioning Tools RefactorIT Window Help
Filesystems
Runtime
Filesystems
/home/staff/nrs/work/mywebapp
WEB-INF
classes
lib
web
META-INF
index
/home/staff/nrs/work/mywebapp : /WEB-I
index
Location:

http://localhost:8084/mywebapp/index.jsp

My JSP Page

The date is Wed Oct 27 14:59:39 BST 2004

or you can point Mozilla at the relevent URL:



http://localhost:8084/mywebapp/index.jsp

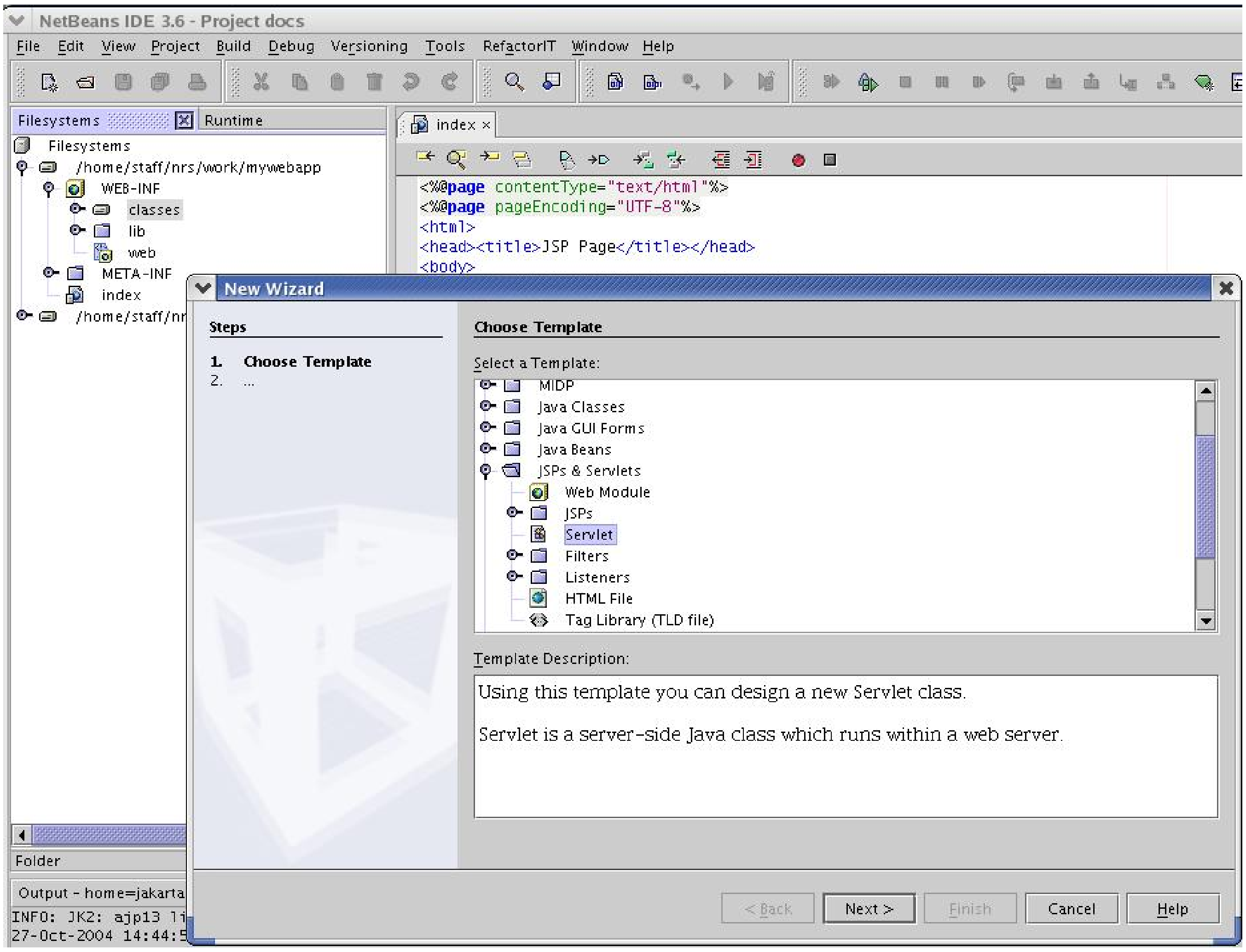
My JSP Page

The date is Wed Oct 27 15:06:05 BST 2004

Netbeans 3.6 uses port 8084 for Tomcat so the URL will be of the format `http://localhost:8084/<your web app>/<your jsp or servlet>`

3. Creating a new Servlet

1. To create a servlet, select the `WEB-INF/classes` folder or the corresponding mount at the bottom, then do `File # New # JSPs & Servlets # Servlet`.



```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
<head><title>JSP Page</title></head>
<body>
```

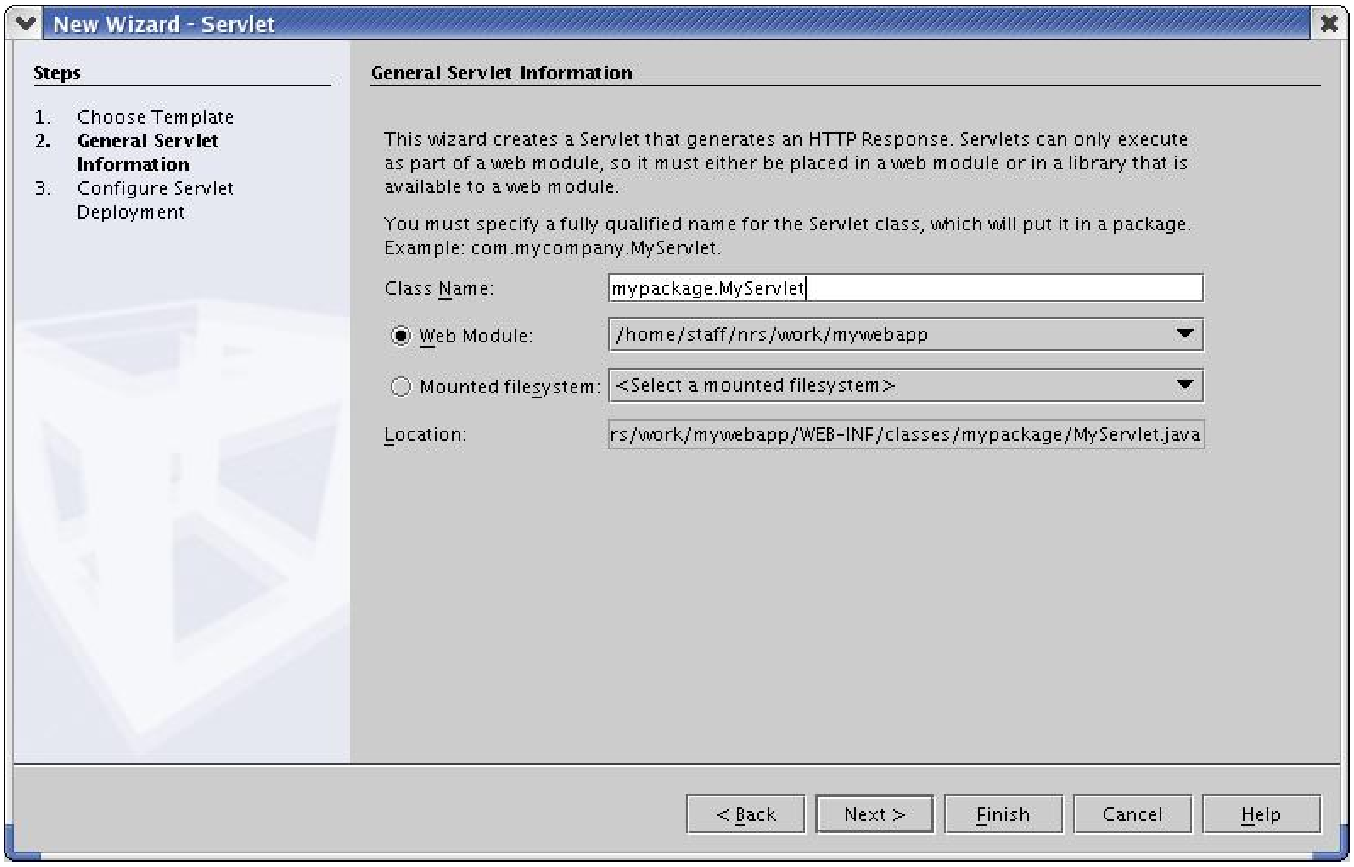
Template Description:

Using this template you can design a new Servlet class.

Servlet is a server-side Java class which runs within a web server.

Hit **Next**

2. Enter a name for your servlet. You must specify a package for your servlet classes.



New Wizard - Servlet
Steps
1. Choose Template
2. General Servlet Information
3. Configure Servlet Deployment
General Servlet Information
< Back
Next >
Finish
Cancel
Help

This wizard creates a Servlet that generates an HTTP Response. Servlets can only execute as part of a web module, so it must either be placed in a web module or in a library that is available to a web module.

You must specify a fully qualified name for the Servlet class, which will put it in a package. Example: com.mycompany.MyServlet.

Class Name: mypackage.MyServlet

(●) Web Module: /home/staff/nrs/work/mywebapp

( ) Mounted filesystem: <Select a mounted filesystem>

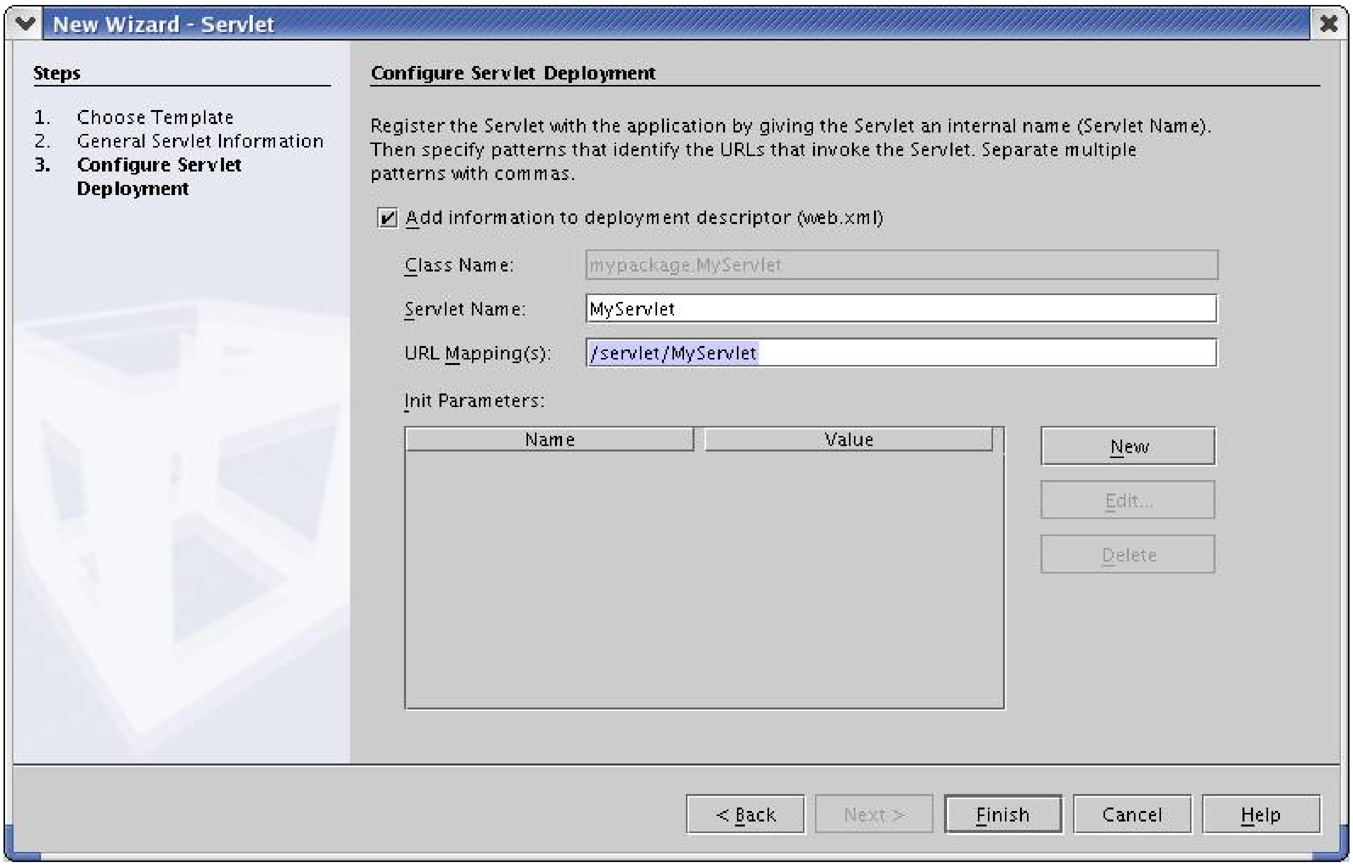
Location: rs/work/mywebapp/WEB-INF/classes/mypackage/MyServlet.java

Hit **Next**.

3. Specify a URL mapping for your servlet.

   It can be useful to prefix servlet URLs with `/servlet/` for later deployment on web servers such as Apache where this prefix can be used for deciding which pages to forward to web container such as Tomcat for processing.

   You can also specify any servlet initialisation parameters that you can then access from the servlet's `init()` method.



New Wizard - Servlet

Steps

1. Choose Template
2. General Servlet Information
3. **Configure Servlet Deployment**

Configure Servlet Deployment

Register the Servlet with the application by giving the Servlet an internal name (Servlet Name). Then specify patterns that identify the URLs that invoke the Servlet. Separate multiple patterns with commas.

☑ Add information to deployment descriptor (web.xml)

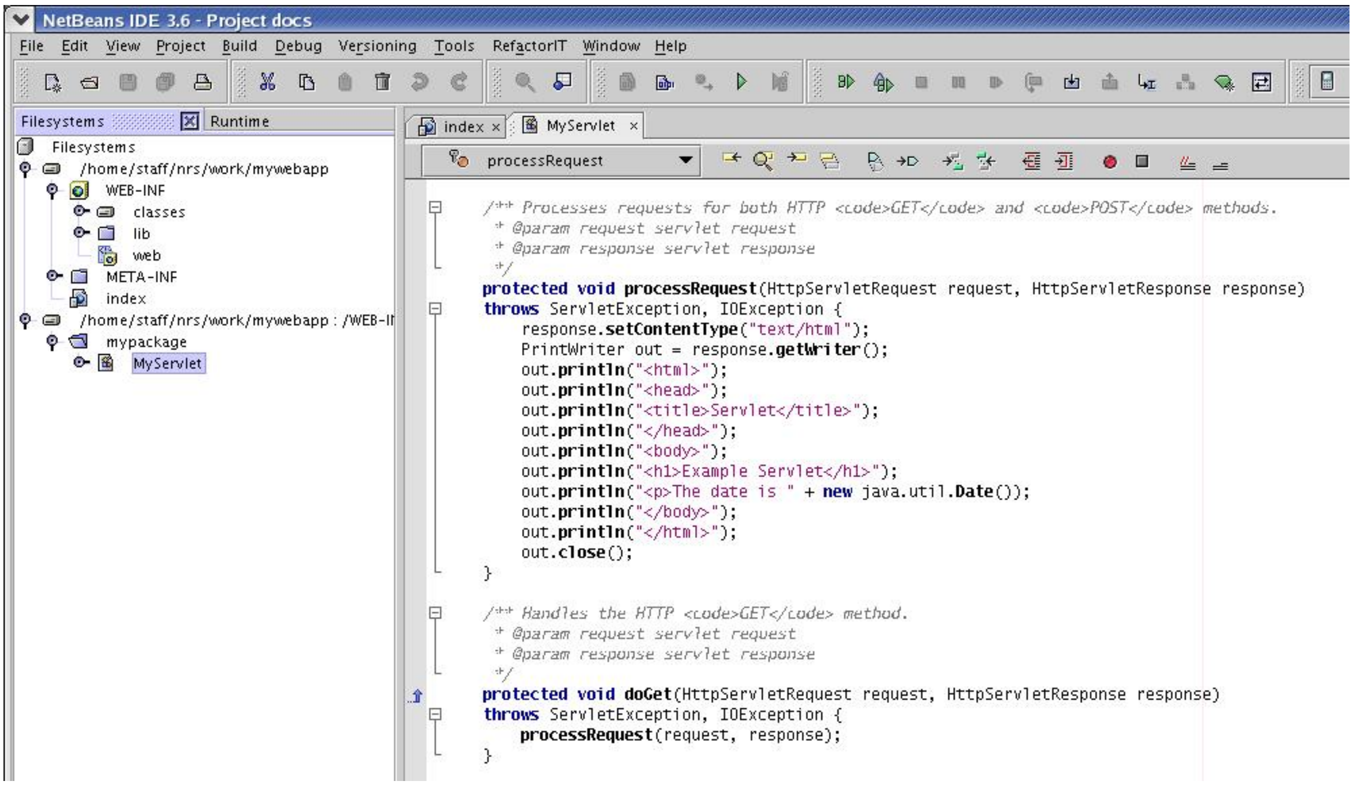
Class Name:	mypackage.MyServlet
Servlet Name:	MyServlet
URL Mapping(s):	/servlet/MyServlet

Init Parameters:

Name	Value

Hit **Finish**

4. Netbeans creates a skeleton servlet with `init(), destroy(), doGet(), doPost() and getServletInfo()` methods.



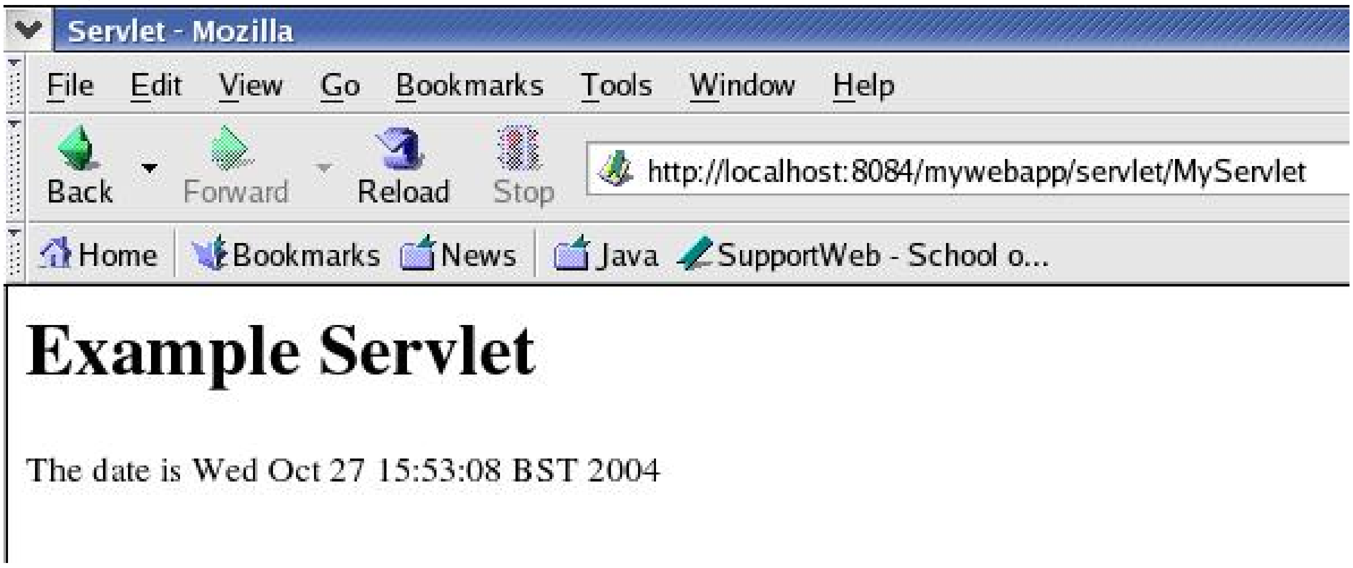
NetBeans IDE 3.6 - Project docs
File Edit View Project Build Debug Versioning Tools RefactorIT Window Help
Filesystems
Runtime
index
MyServlet
processRequest

```
Filesystems
  /home/staff/nrs/work/mywebapp
    WEB-INF
      classes
      lib
      web
    META-INF
    index
  /home/staff/nrs/work/mywebapp : /WEB-I
    mypackage
      MyServlet
```

```
/** Processes requests for both HTTP <code>GET</code> and <code>POST</code> methods.
 * @param request servlet request
 * @param response servlet response
 */
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    out.println("<title>Servlet</title>");
    out.println("</head>");
    out.println("<body>");
    out.println("<h1>Example Servlet</h1>");
    out.println("<p>The date is " + new java.util.Date());
    out.println("</body>");
    out.println("</html>");
    out.close();
}

/** Handles the HTTP <code>GET</code> method.
 * @param request servlet request
 * @param response servlet response
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    processRequest(request, response);
}
```

By default the familiar `doGet() and doPost()` methods are both forwarded to a single common `processRequest()` method as shown above. But you can delete this if you want and code the appropriate `doGet()` and `doPost()` method bodies as appropriate.

5. To run your servlet select it in the explorer or the source editor and hit **F6** or the button and Netbeans should start Tomcat. Fire up your browser and point it at the relevant URL thus:


Servlet - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://localhost:8084/mywebapp/servlet/MyServlet
Home Bookmarks News Java SupportWeb - School o...

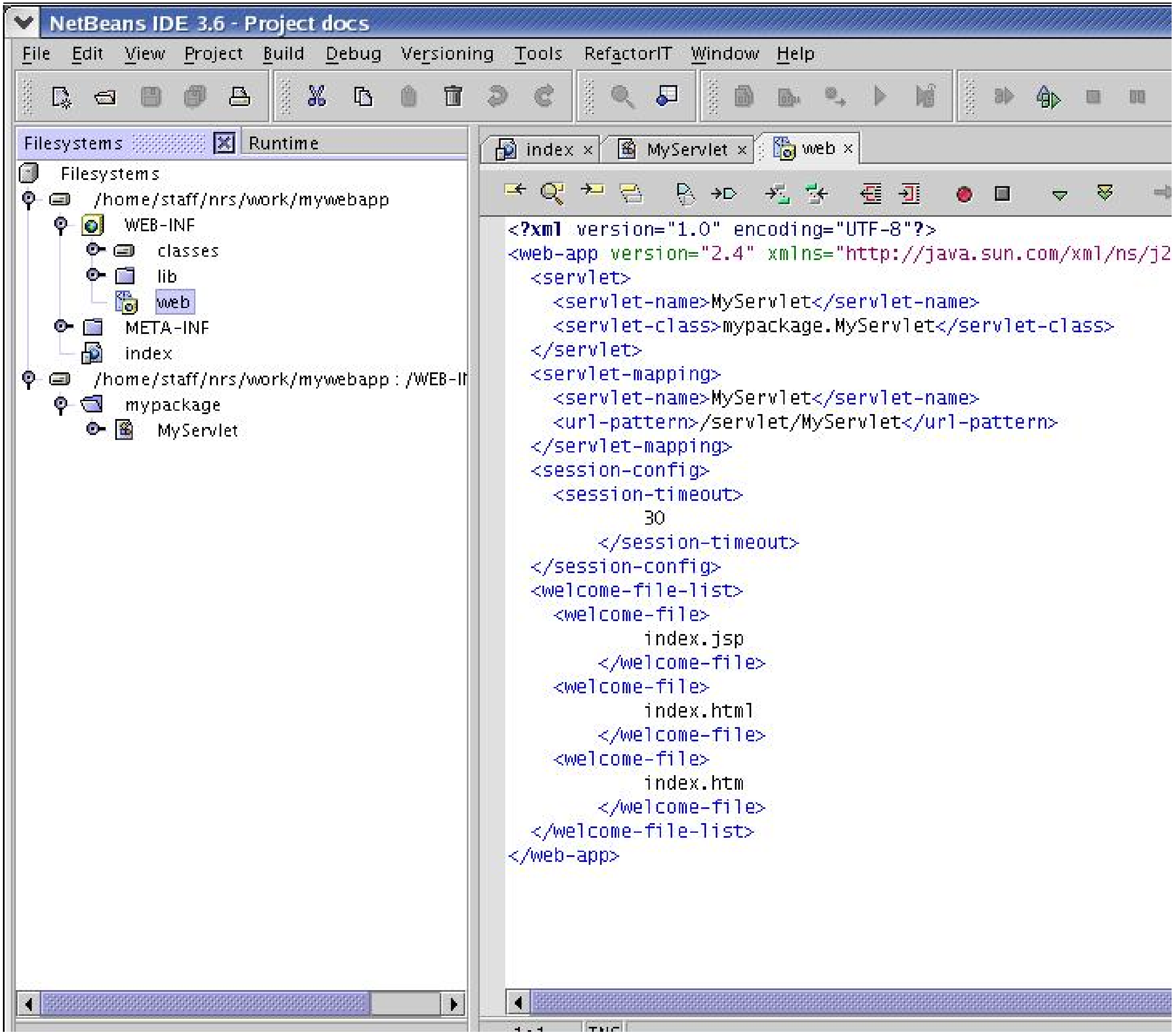
Example Servlet

The date is Wed Oct 27 15:53:08 BST 2004

4. The `web.xml` file

The `web.xml` file is the *Web Application Deployment Descriptor*, which defines which servlets should be run for certain URLs and some other parameters of your web application.

Netbeans creates one for you when you create a new web application and it looks something like this:


NetBeans IDE 3.6 - Project docs
File Edit View Project Build Debug Versioning Tools RefactorIT Window Help
Filesystems
Runtime
Filesystems
/home/staff/nrs/work/mywebapp
WEB-INF
classes
lib
web
META-INF
index
/home/staff/nrs/work/mywebapp : /WEB-I
mypackage
MyServlet
index
MyServlet
web

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2
  <servlet>
    <servlet-name>MyServlet</servlet-name>
    <servlet-class>mypackage.MyServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServlet</servlet-name>
    <url-pattern>/servlet/MyServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
            30
        </session-timeout>
  </session-config>
  <welcome-file-list>
    <welcome-file>
            index.jsp
        </welcome-file>
    <welcome-file>
            index.html
        </welcome-file>
    <welcome-file>
            index.htm
        </welcome-file>
  </welcome-file-list>
</web-app>
```

The format of the URL-to-servlet mappings is described here [http://supportweb.cs.bham.ac.uk/documentation/java/servlets/socs-tomcat/#id213976].

You can see that Netbeans has automatically added `<servlet>` and `<servlet-mapping>` entries for `MyServlet` we just created above.

The `<session-config>` contains a definition for the time in minutes before a user's session times out (in this case 30 minutes) and a <welcome-file-list> section describing which files will be loaded as the default home page for the web application. In this case it will try `index.jsp`, `index.html` and finally if these don't exist `index.htm`.

When you create a new servlet you can set up URL mappings in the New-Servlet dialogs and Netbeans will add the appropriate entries to your `web.xml`, but you are free to edit these by hand.

5. Adding `.jar` files to your application

If your web applications needs additional libraries such as for database access then copy the corresponding `.jar` file to your project's `WEB-INF/lib` directory (e.g. `postgresql.jar`) and then right-click on `WEB-INF/lib` and select **Refresh Folder** and the file will be added to your web application's classpath.

I have not found an easy way to copy files from within Netbeans. If you copy and paste a jar file between mounts it tends to unpack the jar file, so I usually copy them in by hand e.g.:

```
cp /bham/common/java/lib/postgresql.jar ~/work/mywebapp/WEB-INF/lib/
```

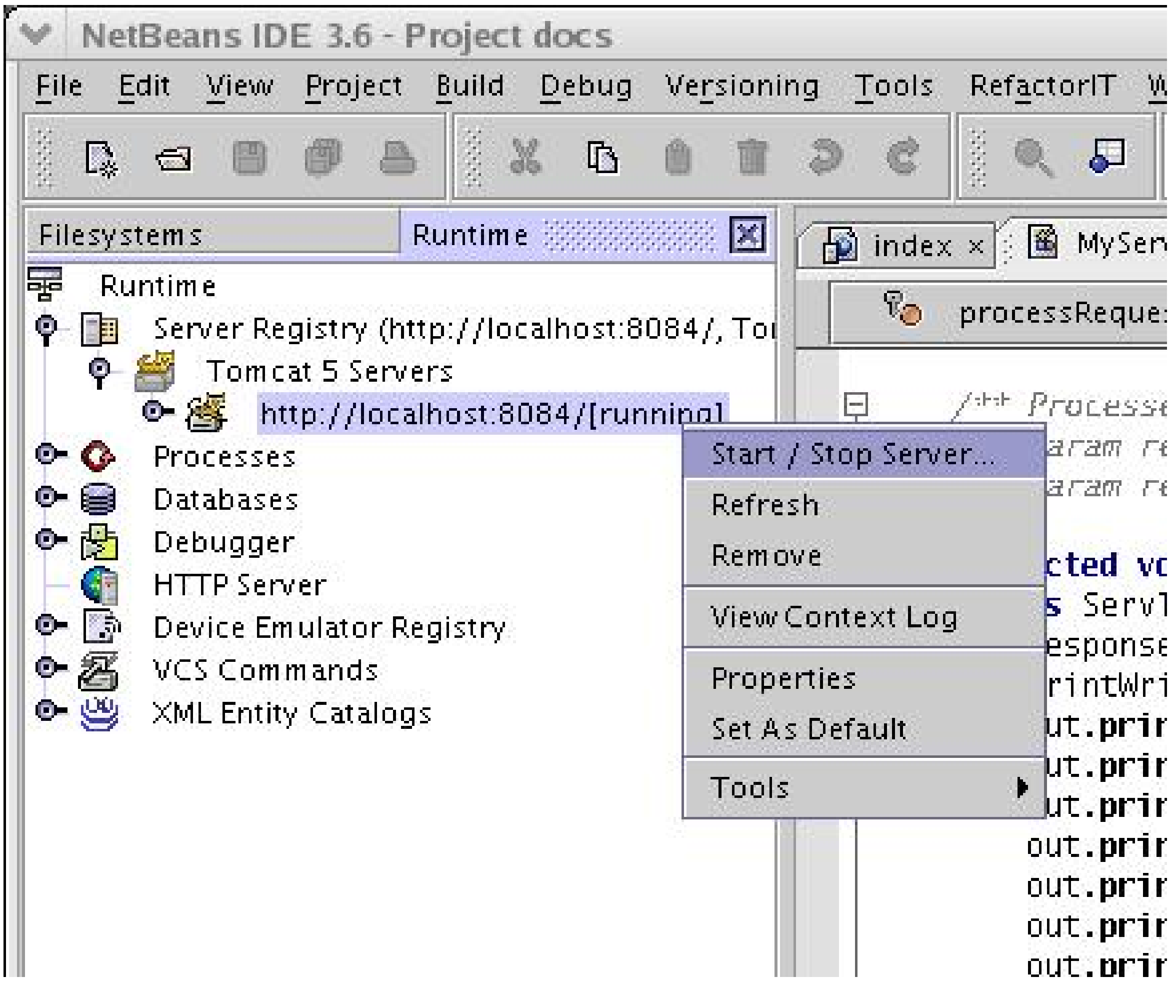
Then subsequently, right-click on `WEB-INF/lib` and select **Refresh Folder**

6. Restarting Tomcat

1. If you edit your code, hitting (run) again will recompile and restart Tomcat.

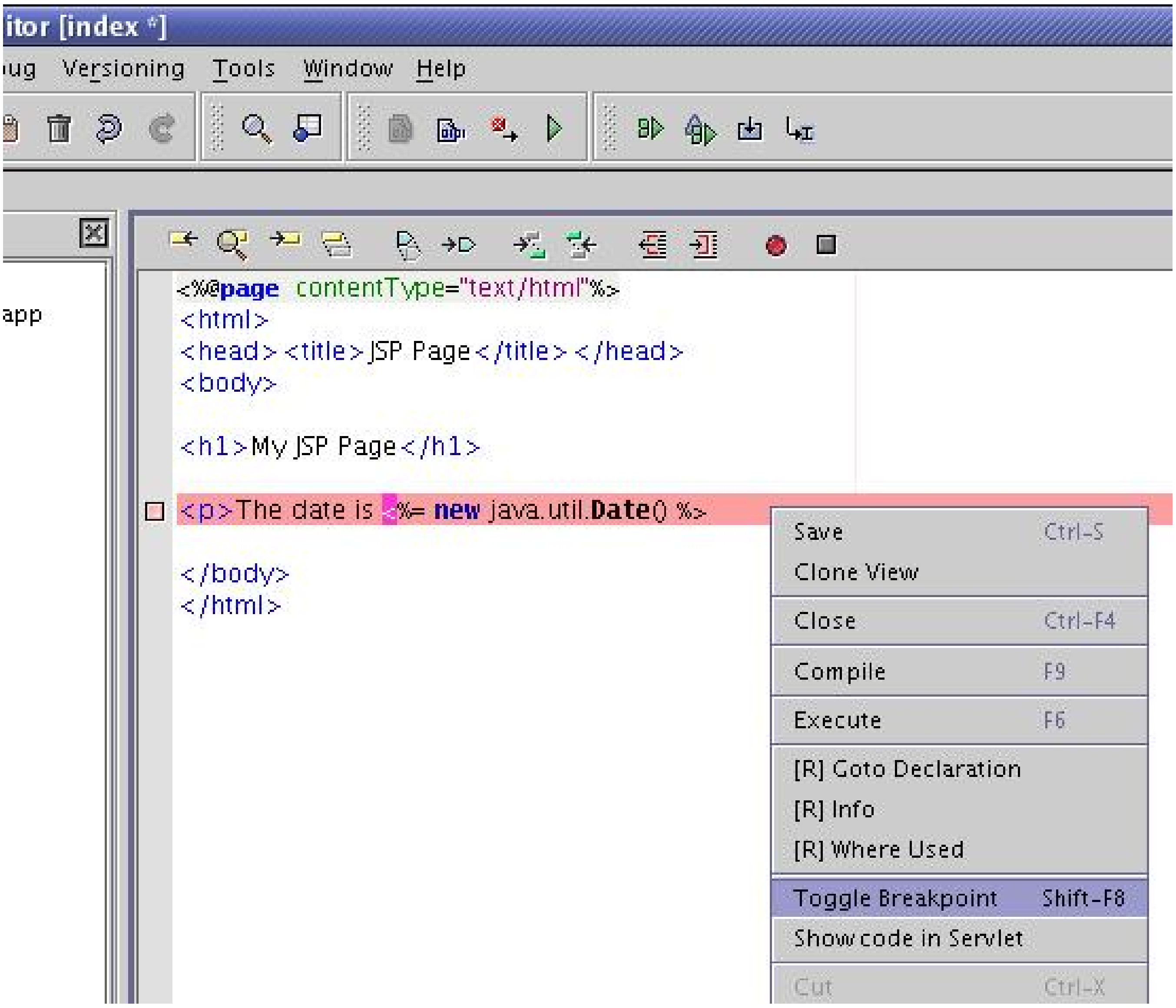
2. To restart the whole web application select `WEB-INF` in the explorer and hit (run). Any altered JSP/servlets will be recompiled and redeployed.

3. Occasionally not all changes will be registered, such as if you manually edit `web.xml` or edit tag libraries. In which case I manually restart Tomcat by going to the Runtime tab and right-click on the node `Server Registry # Tomcat 5 Servers # http://localhost:8084` and select **Start / Stop Server**



7. Debugging JSPs and Servlets

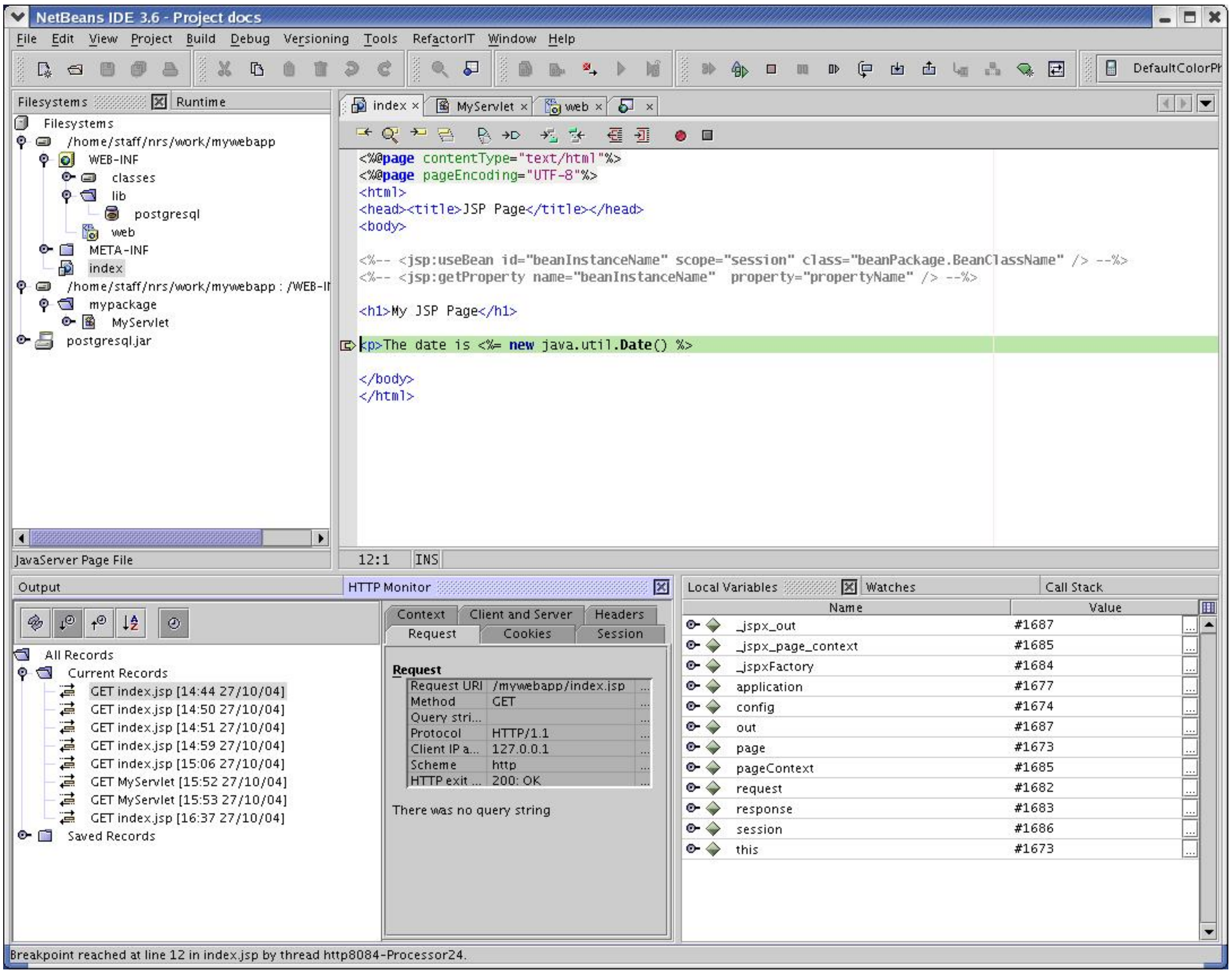
1. Set any breakpoints where you would like execution to halt in your JSP or Servlet code by moving the cursor to the appropriate line in the editor and either hit **Shift-F8** or right-click # `Toggle Breakpoint`.



```
<%@page contentType="text/html"%>
<html>
<head><title>JSP Page</title></head>
<body>

<h1>My JSP Page</h1>

<p>The date is <%= new java.util.Date() %>

</body>
</html>
```

2. To debug the current JSP or Servlet that you are editing hit the button whilst the cursor is in the editor pane. To debug the whole web application hit the button whilst `WEB-INF` is selected in the Explorer.

   The application will run until a breakpoint is encountered.


NetBeans IDE 3.6 - Project docs
File Edit View Project Build Debug Versioning Tools RefactorIT Window Help
DefaultColorPr
Filesystems Runtime
Filesystems
/home/staff/nrs/work/mywebapp
WEB-INF
classes
lib
postgresql
web
META-INF
index
/home/staff/nrs/work/mywebapp : /WEB-I
mypackage
MyServlet
postgresql.jar
JavaServer Page File
index
MyServlet
web
12:1 INS
Output
HTTP Monitor
Context Client and Server Headers
Request Cookies Session
Local Variables Watches Call Stack

```
<%@page contentType="text/html"%>
<%@page pageEncoding="UTF-8"%>
<html>
<head><title>JSP Page</title></head>
<body>

<%-- <jsp:useBean id="beanInstanceName" scope="session" class="beanPackage.BeanClassName" /> --%>
<%-- <jsp:getProperty name="beanInstanceName"  property="propertyName" /> --%>

<h1>My JSP Page</h1>

<p>The date is <%= new java.util.Date() %>

</body>
</html>
```

```
All Records
  Current Records
    GET index.jsp [14:44 27/10/04]
    GET index.jsp [14:50 27/10/04]
    GET index.jsp [14:51 27/10/04]
    GET index.jsp [14:59 27/10/04]
    GET index.jsp [15:06 27/10/04]
    GET MyServlet [15:52 27/10/04]
    GET MyServlet [15:53 27/10/04]
    GET index.jsp [16:37 27/10/04]
  Saved Records
```

Request

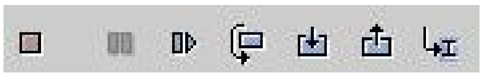
Request URI	/mywebapp/index.jsp
Method	GET
Query stri...	
Protocol	HTTP/1.1
Client IP a...	127.0.0.1
Scheme	http
HTTP exit ...	200: OK

There was no query string

Name	Value
_jspx_out	#1687
_jspx_page_context	#1685
_jspxFactory	#1684
application	#1677
config	#1674
out	#1687
page	#1673
pageContext	#1685
request	#1682
response	#1683
session	#1686
this	#1673

Breakpoint reached at line 12 in index.jsp by thread http8084-Processor24.

The green line shows the current line to be executed and the execution can be controlled by the buttons at the top:



The buttons left to right are `Stop, Pause, Continue, Step Over, Step Into, Step Out and Run to Cursor`. (`Pause` is greyed out as the program is already paused at the breakpoint).

In the panes at the bottom you can see the Call Stack, the values of Local Variables (you can expand objects to look at member variables) and the values of any Watches you have set. You can set a new watch on a variable by right-clicking on it and selecting `New Watch`.

For debugging web applications Netbeans also provides a HTTP Monitor so that you can look at the values of parameters in the actual HTTP requests.

3. To see the servlet code generated for a JSP page (remember JSPs are just templates for servlets created automatically by the container), right click anywhere in the source of the JSP and select **View Servlet**.

8. Exporting your Web Application

To create a web application `.war` file to deploy on external JSP/Servlet containers, right-click on `WEB-INF` and select **Export WAR file...** then give it a name in the file dialog.