

RZ 3747  
Computer Science

(# 99757)  
44 pages

08/10/2009

# Research Report

## Starting Hamlets

René Pawlitzek

IBM Research GmbH  
Zurich Research Laboratory  
8803 Rüschlikon  
Switzerland

### LIMITED DISTRIBUTION NOTICE

This report will be distributed outside of IBM up to one year after the IBM publication date.  
Some reports are available at <http://domino.watson.ibm.com/library/Cyberdig.nsf/home>.

 **Research**  
Almaden • Austin • Beijing • Delhi • Haifa • T.J. Watson • Tokyo • Zurich

# **Starting Hamlets – How to write your first web-based application in Java**

by  
René Pawlitzek  
IBM Research - Zurich  
[rpa@zurich.ibm.com](mailto:rpa@zurich.ibm.com)

July 31<sup>st</sup>, 2009

In today's internet-driven world, web-based applications are increasingly replacing traditional stand-alone applications, and for good reasons. You no longer need to install and upgrade web-based applications. They are always up-to-date. Simply point your browser to a particular site and you are ready to read your e-mail, to get driving directions, to book your vacation, and to do your tax return, to name just a few examples. Web-based applications can run on any platform that provides a browser. Moreover, thanks to new programming techniques such as AJAX, they have become more interactive and easier to use.

Developing web-based applications can be done using a variety of tools and languages. However, the Java<sup>™</sup> programming language and Java servlets are the ideal choice because of a number of attractive features, namely, portability, efficiency, safety, extensibility, and flexibility. Few viable alternatives exist that can be considered equally powerful.

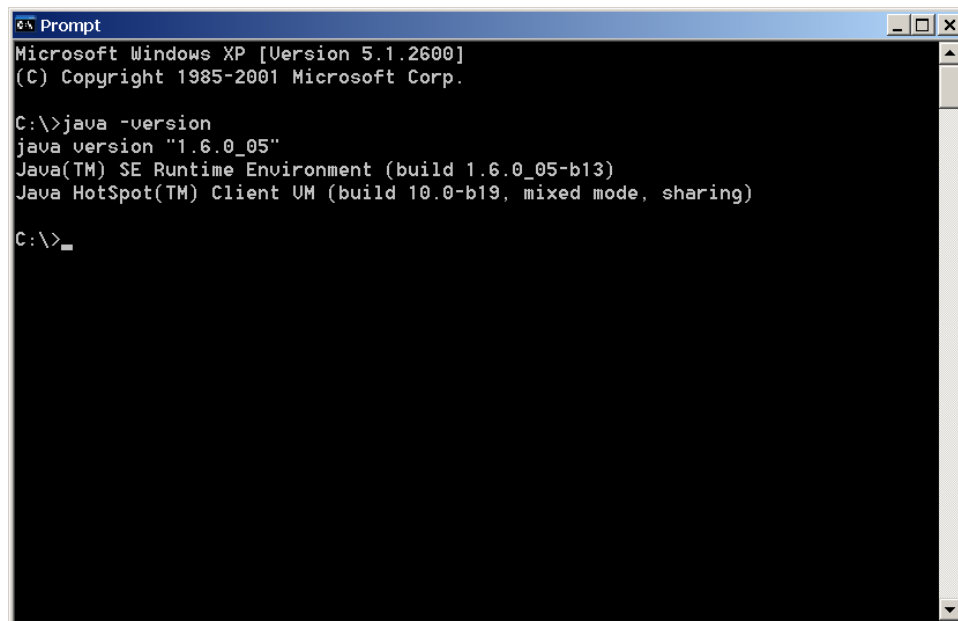
Over the past few years, I have developed an easy-to-use, easy-to-understand framework based on Java servlets to facilitate the development of web-based applications. The framework (called Hamlets) is the result of a radical simplification effort. In this article I show you step by step how to write your first web-based application in Java using Hamlets. At the end you will have enough knowledge to create small web-based applications on your own.

# 1. Setting up the development environment

Before you can start developing your first Java-based web application, you need to set up your development environment. This article assumes that your machine does not contain any development tools at all. The following steps demonstrate how to set up an efficient development environment to write Java-based applications for the web.

## 1.1 Installing Java

It is very likely that at least one Java Runtime Environment (JRE) is already installed on your computer. By entering 'java -version' on the command line (see Figure 1) you can see the system's default Java environment.



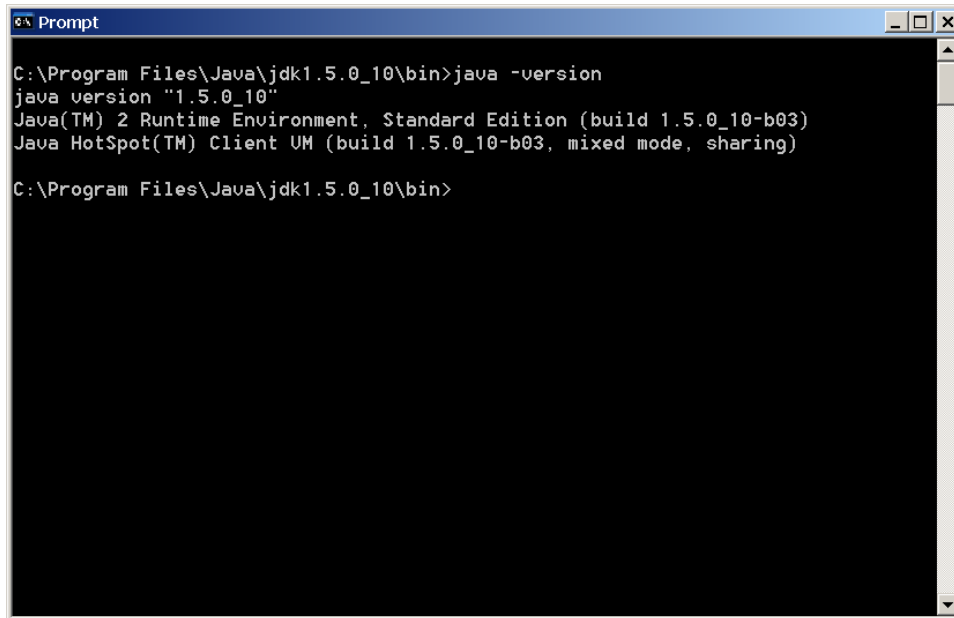
```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\>java -version
java version "1.6.0_05"
Java(TM) SE Runtime Environment (build 1.6.0_05-b13)
Java HotSpot(TM) Client VM (build 10.0-b19, mixed mode, sharing)

C:\>_
```

Figure 1: Checking for the default Java Runtime Environment (JRE)

Additional (older or newer) Java Runtime Environments (JREs) might be 'hiding' in other directories on your hard disk (see Figure 2):



```
C:\Program Files\Java\jdk1.5.0_10\bin>java -version
java version "1.5.0_10"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_10-b03)
Java HotSpot(TM) Client VM (build 1.5.0_10-b03, mixed mode, sharing)

C:\Program Files\Java\jdk1.5.0_10\bin>
```

**Figure 2: Additional Java Runtime Environment (JRE)**

To develop Java software, you need a Java Development Kit (JDK). A JDK not only contains a Java Runtime Environment (JRE) but also a number of command-line development tools to facilitate software construction. The installation is straightforward and thus not described here.

For the example in this article JDK version 1.6 is recommended. The Resources section below contains a link to the Sun Java website, from where you can download a suitable version of Java for your machine.

## ***1.2 Installing Tomcat***

A web-based application is displayed by a browser (Microsoft® Internet Explorer®, Mozilla Firefox, Opera, Safari, etc.) but runs on a web application server. Tomcat is such a web application server. It consists of a servlet container (to run Java code in form of servlets) and a web server (to provide static content (e.g.: html, css, gif, jpg, png files)). Several other web application servers are available on the market with various feature sets. Tomcat is an excellent choice, because it is reliable, lightweight, and open source. It is the reference implementation for servlets.

Like Java, Tomcat exists in several versions. To develop Java-based web application using Hamlets 1.5 you need at least version 5.5. The Resources section contains a download link for Tomcat. The installation is straightforward and should not cause any problems. Note that for software development, I do not recommend to install Tomcat as a service on a Windows® NT/2000/XP machine. After you have successfully installed and started Tomcat (see Figure 3), you should see the following screen (see Figure 4) in your browser when you point it to <http://localhost:8080/>.

```
Local Prompt
C:\Program Files\Apache Software Foundation\apache-tomcat-5.5.26\bin>startup
Using CATALINA_BASE: C:\Program Files\Apache Software Foundation\apache-tomcat-5.5.26
Using CATALINA_HOME: C:\Program Files\Apache Software Foundation\apache-tomcat-5.5.26
Using CATALINA_TMPDIR: C:\Program Files\Apache Software Foundation\apache-tomcat-5.5.26\temp
Using JRE_HOME: C:\JBUILDER2007\jre
C:\Program Files\Apache Software Foundation\apache-tomcat-5.5.26\bin>startup
```

Figure 3: Starting Tomcat from the command line with startup

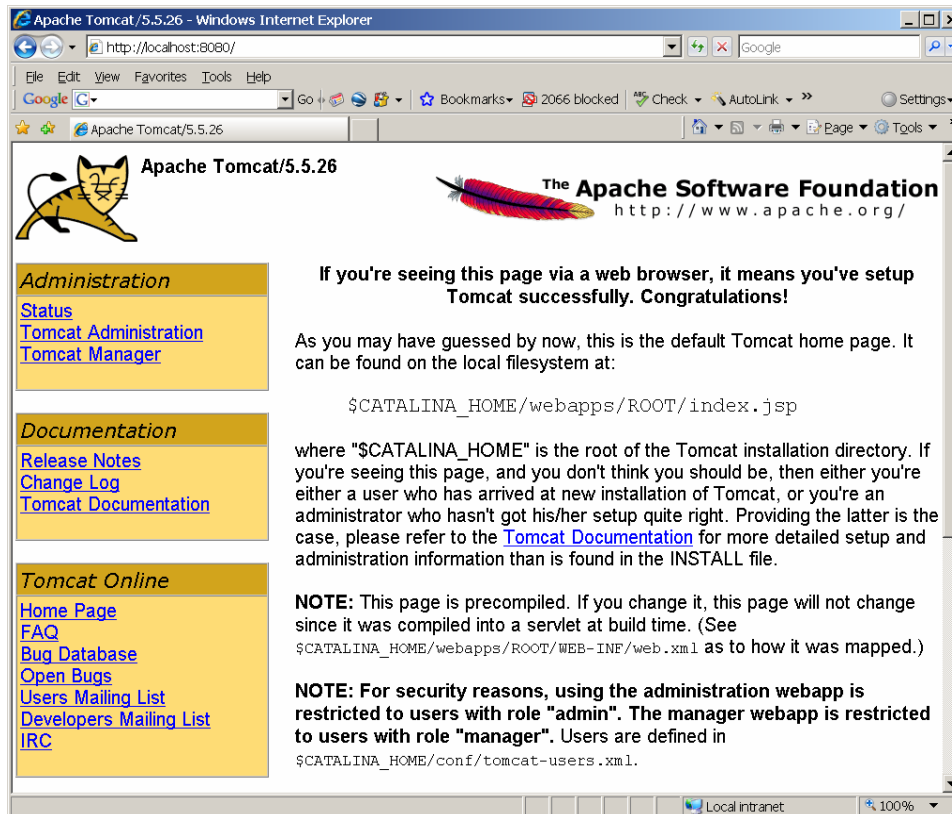


Figure 4: Apache Tomcat/5.5.26 main screen

## 1.3 Installing Eclipse

Next, you need to install an Integrated Development Environment (IDE) to facilitate your development efforts. As one would expect, a number of IDEs are available to choose from. For this article, Eclipse Classic Version 3.4.1 (Ganymede) is used. See the Resources section for a download link and a link to several video tutorials. According to its developers, Eclipse is not just a Java IDE but “an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle”.

After a successful installation, you should see the following screen when you start Eclipse:

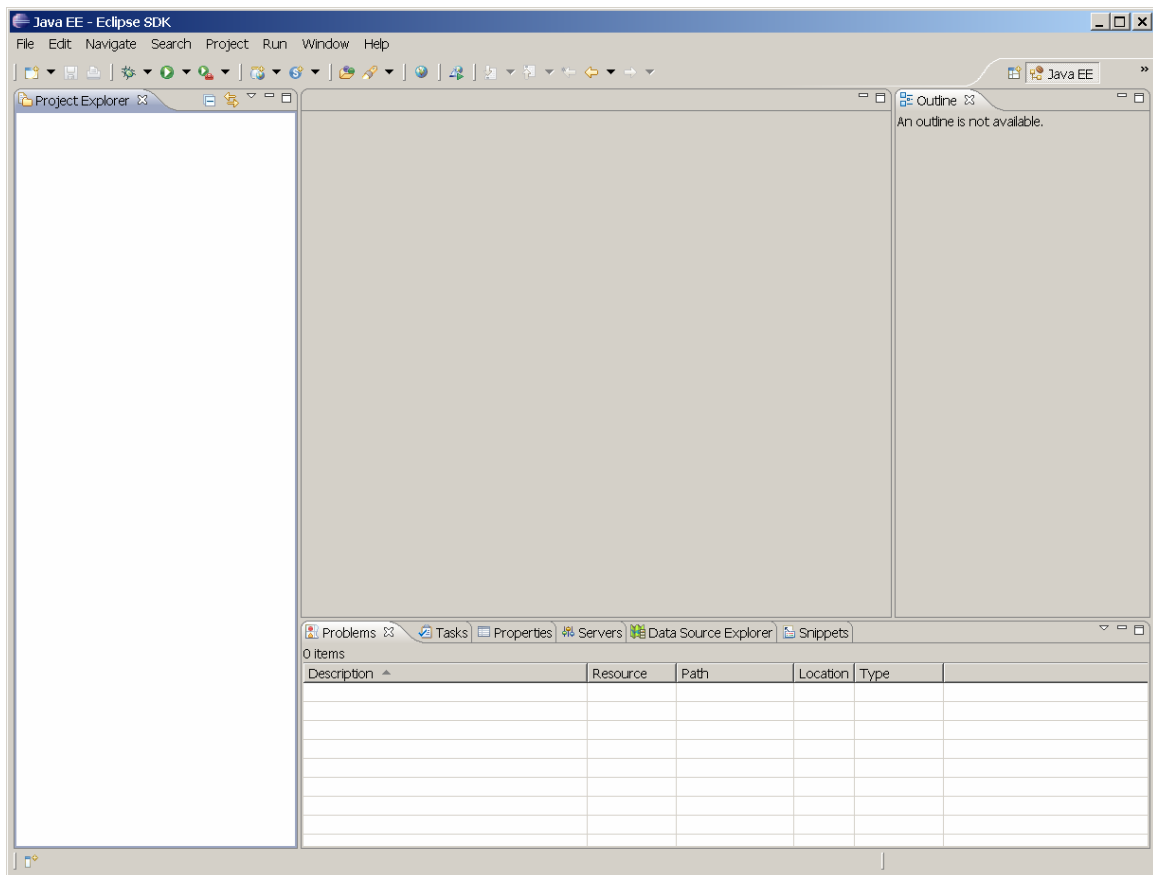
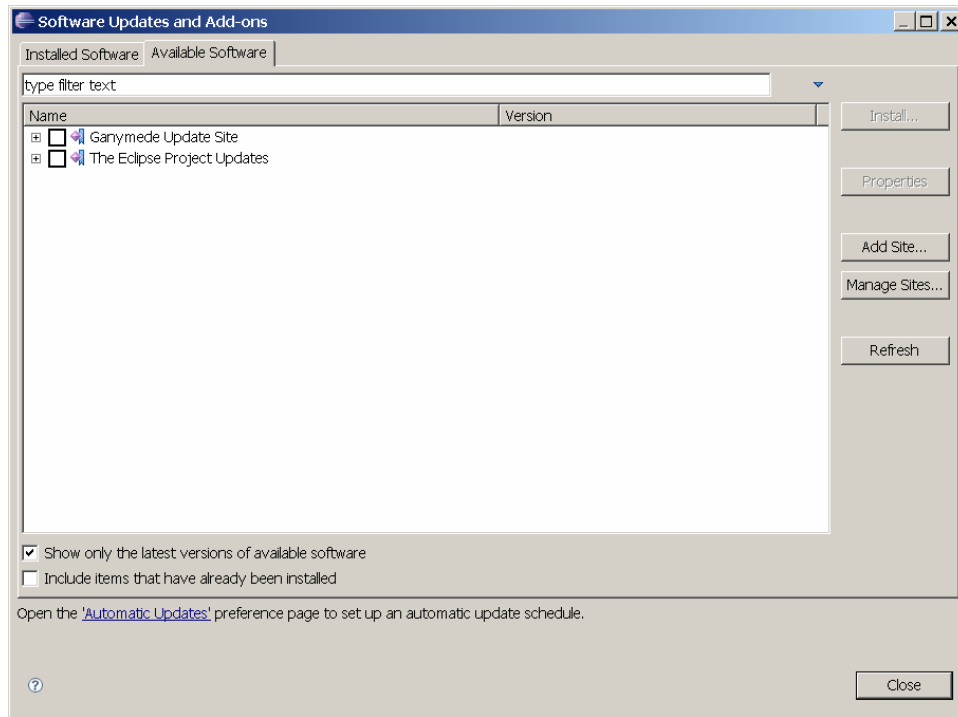


Figure 5: Starting Eclipse 3.4.1 for the first time

## 1.4 Installing WTP

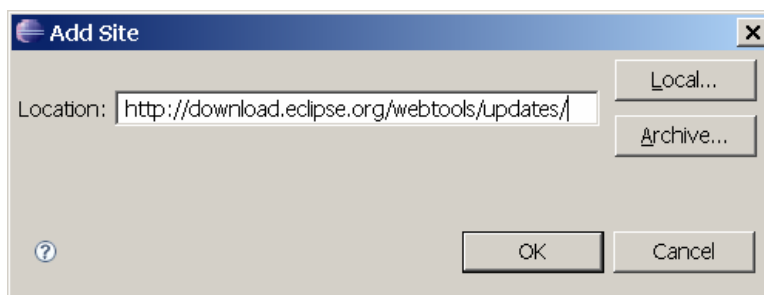
The Eclipse software you just downloaded does not yet contain all functionality necessary to develop web-based applications. An additional package is required: the Web Tools Platform ([WTP](#)). You can use Eclipse’s updating mechanism to retrieve and install

WTP. Select Help->Software Updates, and you will see the following screen (see Figure 6).



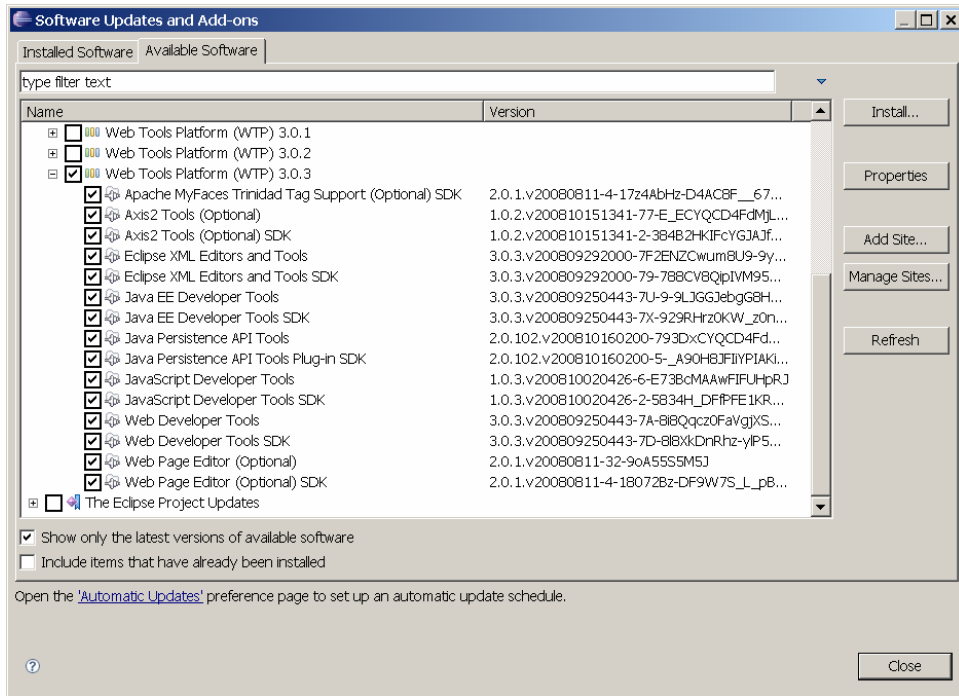
**Figure 6: Eclipse software update screen**

Next, add a new download site by pressing the 'Add Site...' button and specify its location <http://download.eclipse.org/webtools/updates/> (see Figure 7).



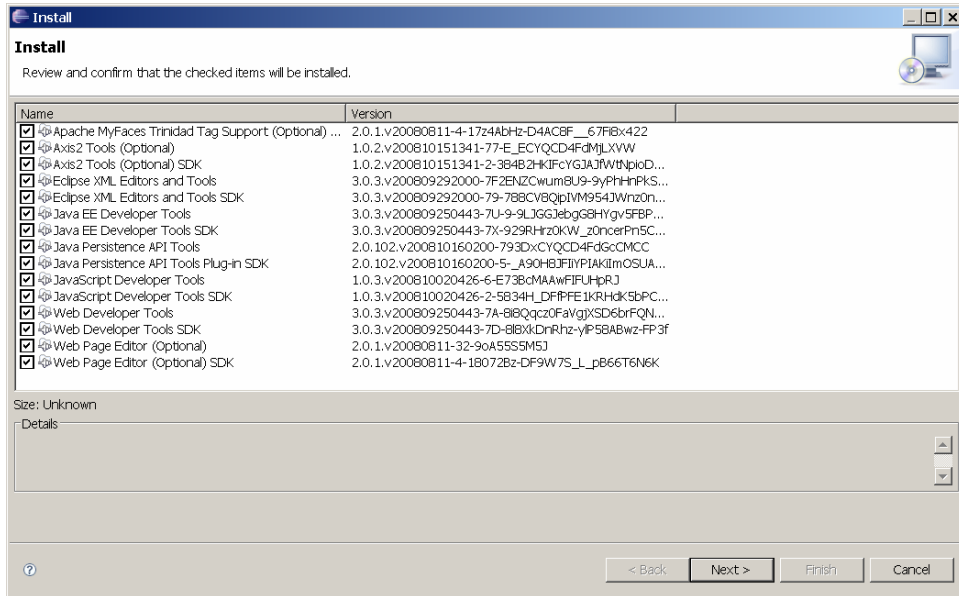
**Figure 7: Adding the WTP update site**

Press 'OK' to return to the previous screen, which now contains a list of available WTP packages. Select the Web Tools Platform (WTP) 3.0.3 package (see Figure 8).



**Figure 8: Selecting WTP from the software update screen**

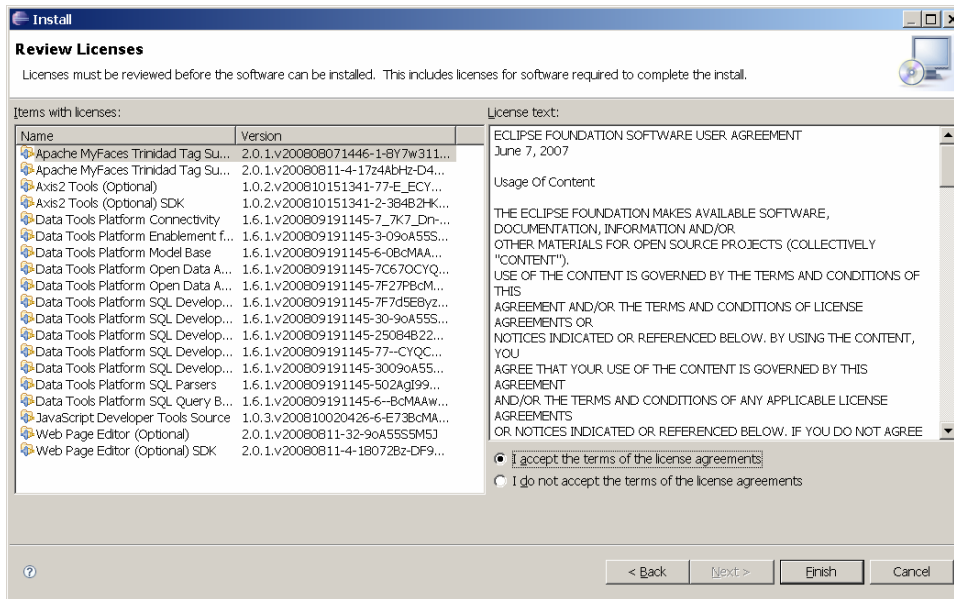
Next, press ‘Install...’ to bring up the installation screen (see Figure 9).



**Figure 9: Installing WTP**

Press ‘Next’, and accept the terms of the license agreements (see Figure 10).



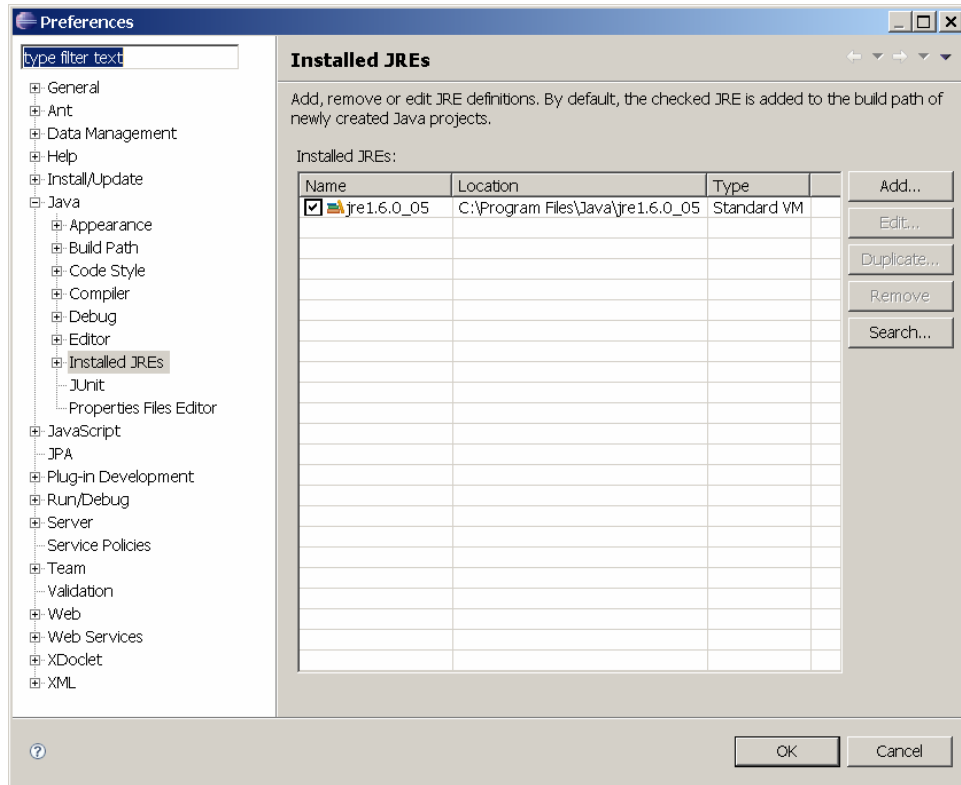


**Figure 10: Accepting the terms of the license agreements.**

The Eclipse system will now install the WTP 3.0.3 package and prompt you to restart Eclipse when it has finished the installation.

## 1.5 Configuring Java for Eclipse

The Java Runtime Environment (JRE) will most likely already have been configured properly during the installation of Eclipse. To verify whether this is indeed the case, you can bring up the Preferences dialog (with Window->Preferences) and display a list of JREs installed by selecting 'Installed JREs' from the tree view on the left (see Figure 11).

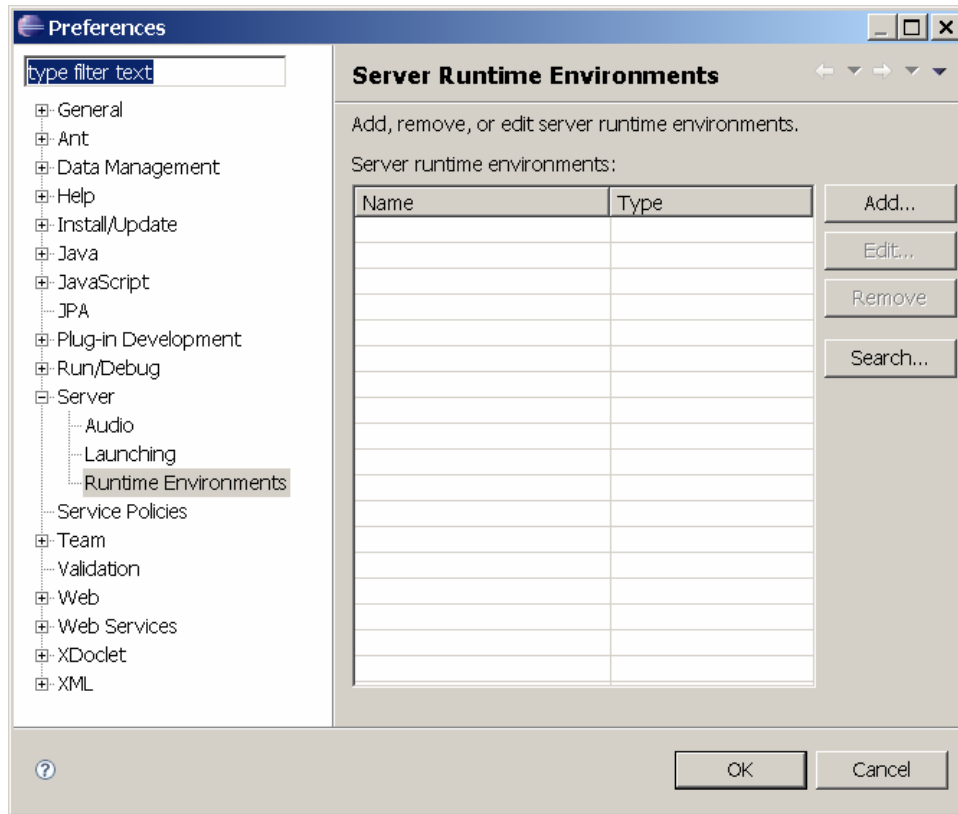


**Figure 11: Displaying all JREs installed**

This is also the dialog to be used for adding, removing or editing JREs.

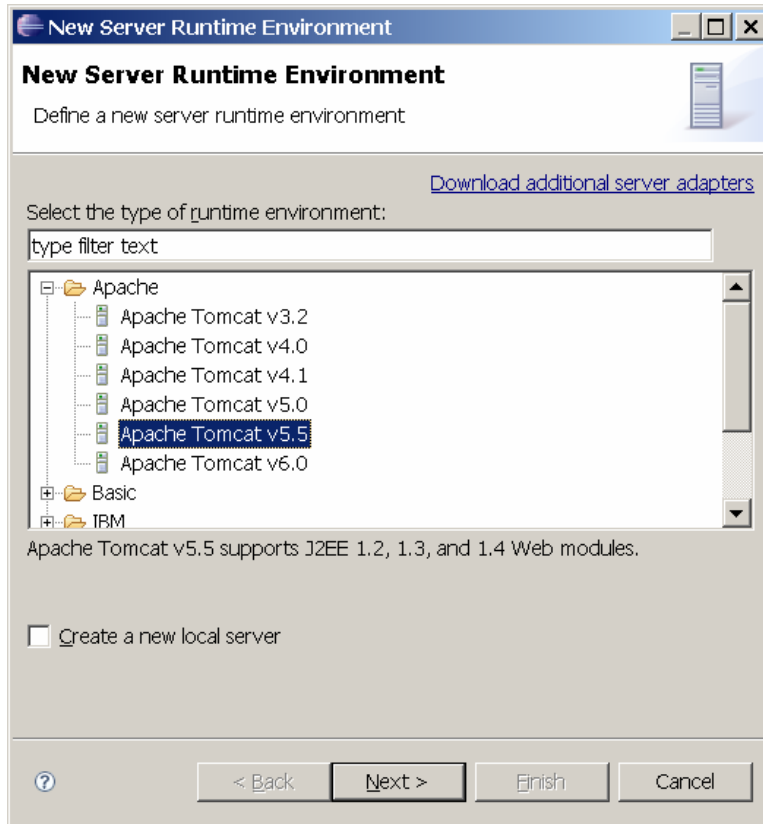
## **1.6 Configuring Tomcat for Eclipse**

The final step in setting up the development environment for programming web-based applications is to configure Tomcat for Eclipse. Use Window->Preferences to bring up the Preferences dialog screen and select the 'Runtime Environments' node from the tree control on the left-hand side of the dialog (see Figure 12).



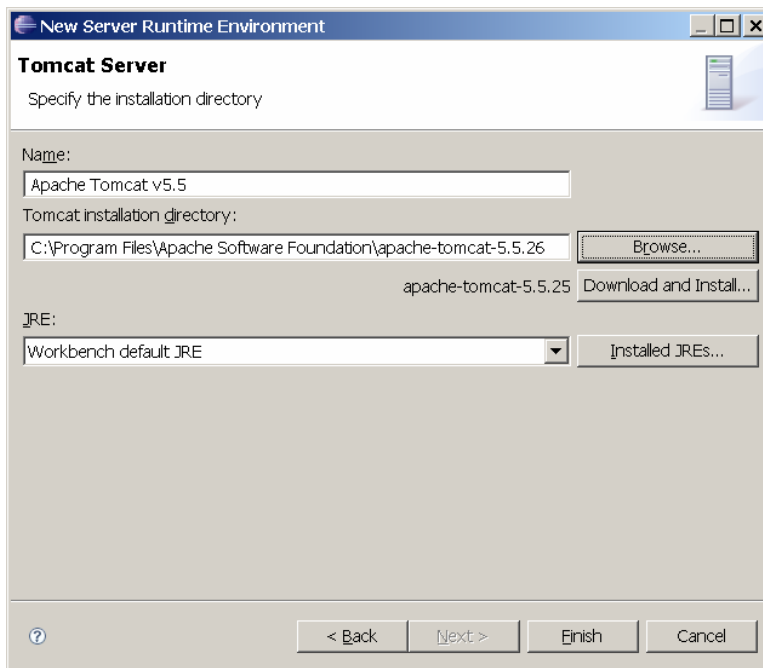
**Figure 12: Server Runtime Environments**

Next, press ‘Add...’ to define a new server runtime environment and select the Tomcat version that you have just installed, i.e., version 5.5.x if you followed the instructions herein (see Figure 13).



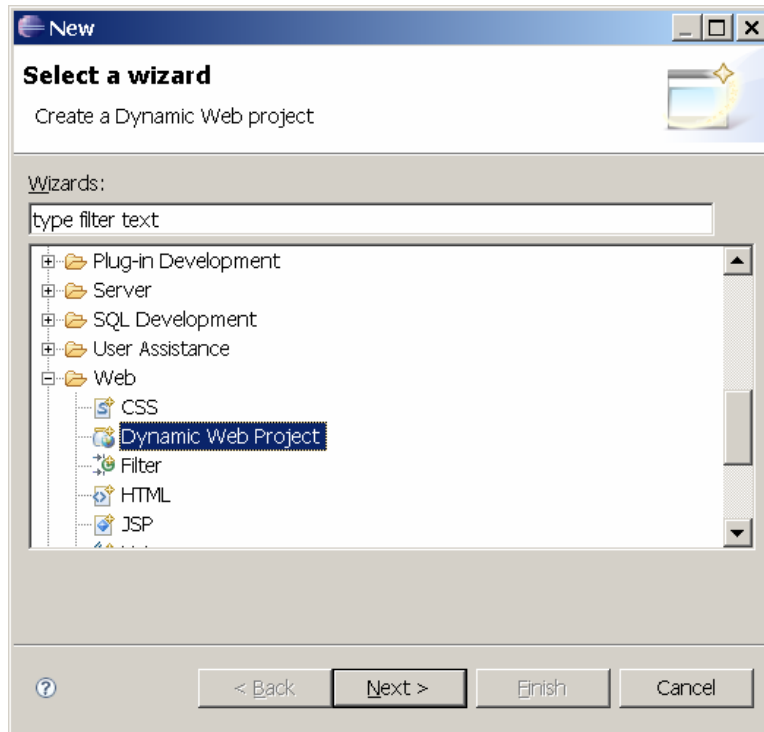
**Figure 13: Defining a new server runtime environment**

Press 'Next' and specify the Tomcat installation directory (see Figure 14).



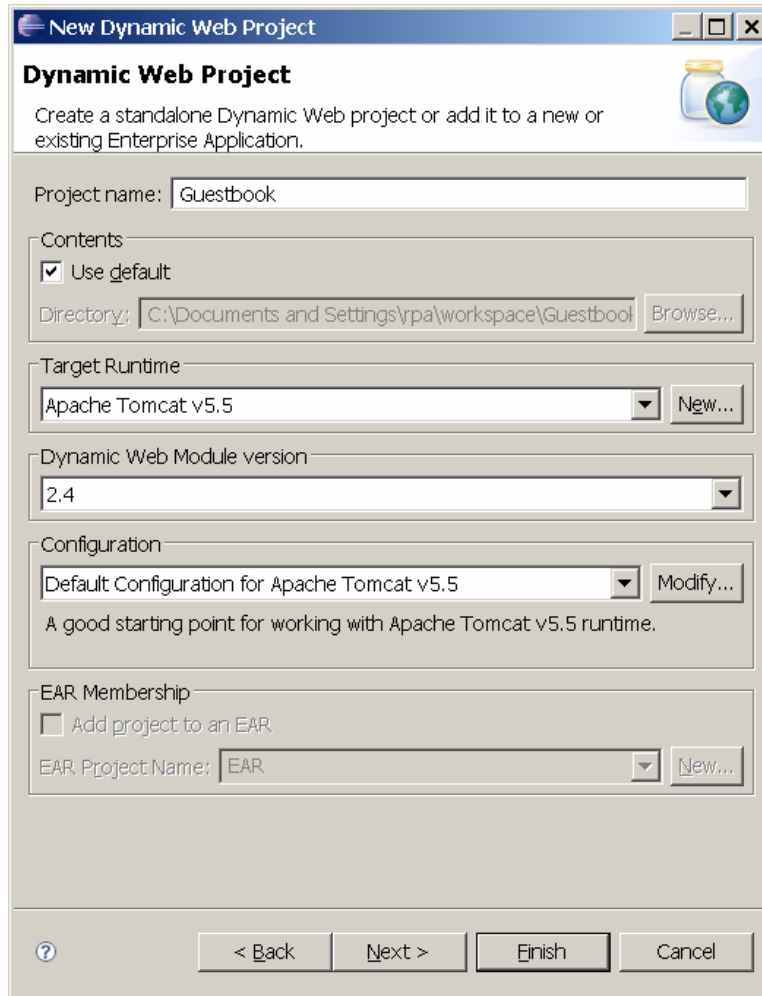
**Figure 14: Specifying the Tomcat installation directory**





**Figure 16: Creating a dynamic web project**

Press 'Next' and enter the project name Guestbook (see Figure 17).



**Figure 17: Providing the project name**

Press 'Finish' to start the creation of the dynamic web project called Guestbook. Once the project has been created, it will show up in the Project Explorer (see Figure 18).

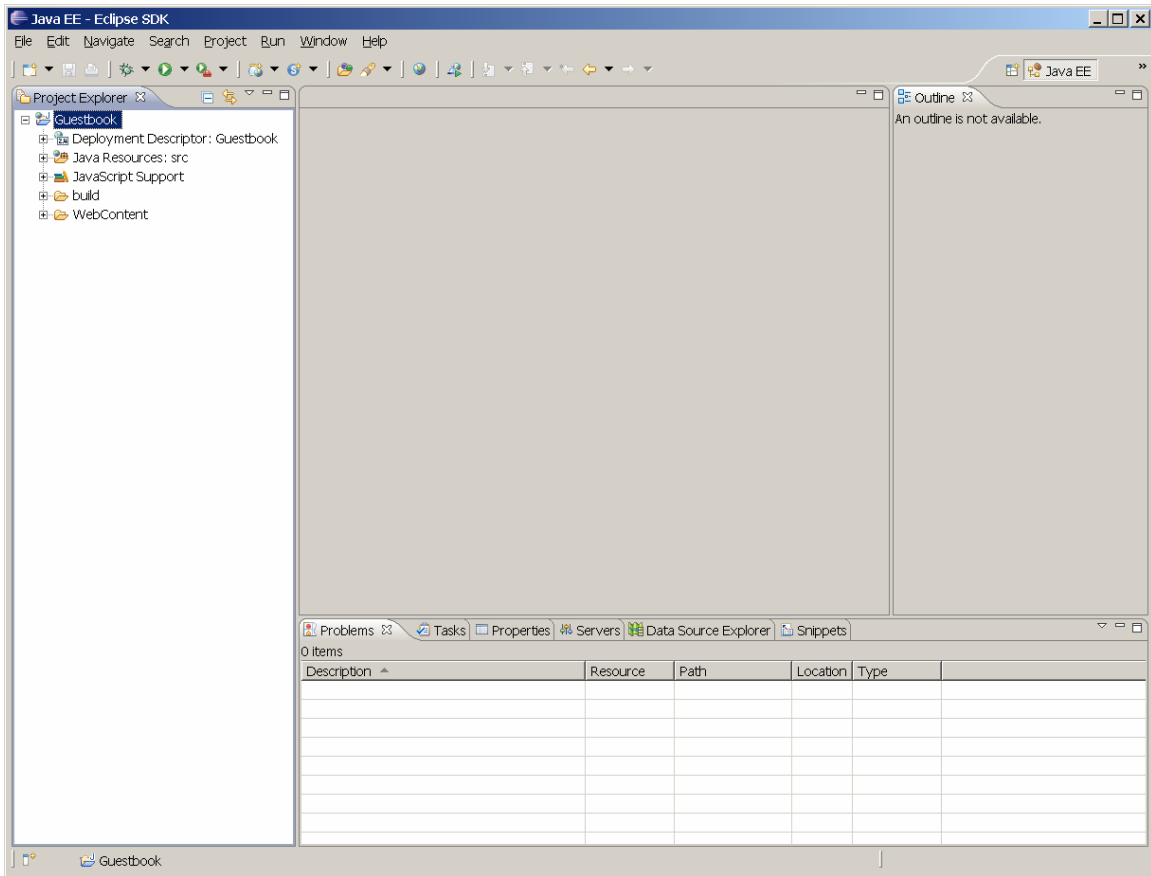


Figure 18: Guestbook project showing up in the Project Explorer

## 2.2 Adding libraries

Most applications depend on a number of libraries, which have to be added to the project so that Eclipse and the Java compiler know about them. For the Guestbook application, you will need three libraries:

- a) The hamlet-1.5.jar library provides the Hamlets framework to the Guestbook application. It enforces the separation of code (Java) and presentation (html) in an easy-to-use and easy-to-understand fashion. Download hamlets-bin.zip from <http://hamlets.sourceforge.net> and unzip the package. Next, look for the hamlet-1.5.jar library and simply drag and drop it into the lib folder of the Guestbook project.

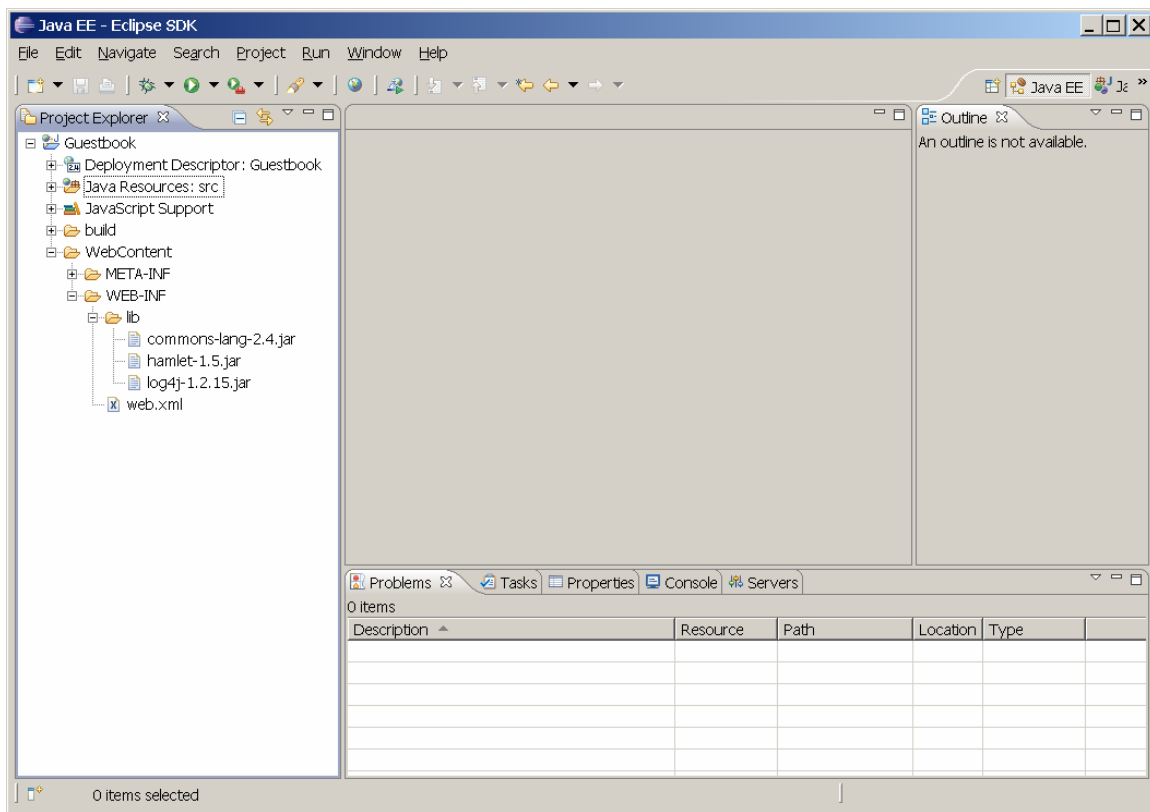
Note that the lib folder is part of the WEB-INF folder, which in turn is part of the WebContent folder (see Figure 19).<sup>1</sup>

<sup>1</sup> You may also use the older hamlet.jar library for this project, but if you do, you will have to change serveDoc (req, res, "GuestBookTemplate.html", handler); to serveDoc (req, res, "/GuestBookTemplate.html", handler); in the GuestBook.java file. Note the additional "/" in front of "GuestBookTemplate.html".



- b) The log4j-1.2.15.jar library is part of the Apache logging services and is required by the Hamlets framework. Download apache-log4j-1.2.15.zip from <http://logging.apache.org/log4j/1.2/download.html> and unzip the package. Next, look for the log4j-1.2.15.jar library and add it to the lib folder using drag and drop.
- c) The commons-lang-2.4-bin.zip library is part of the Apache Commons project, which focuses on all aspects of reusable Java components. Download commons-lang-2.4-bin.zip from [http://commons.apache.org/downloads/download\\_lang.cgi](http://commons.apache.org/downloads/download_lang.cgi), unzip the package, and add the commons-lang-2.4.jar library to the lib folder using drag and drop (see Figure 19).

Libraries added to the lib folder of a dynamic web project are automatically deployed to the web application server when you run the web application. In addition, they are also automatically included in the war (Web Archive) file (see Section 4.1).

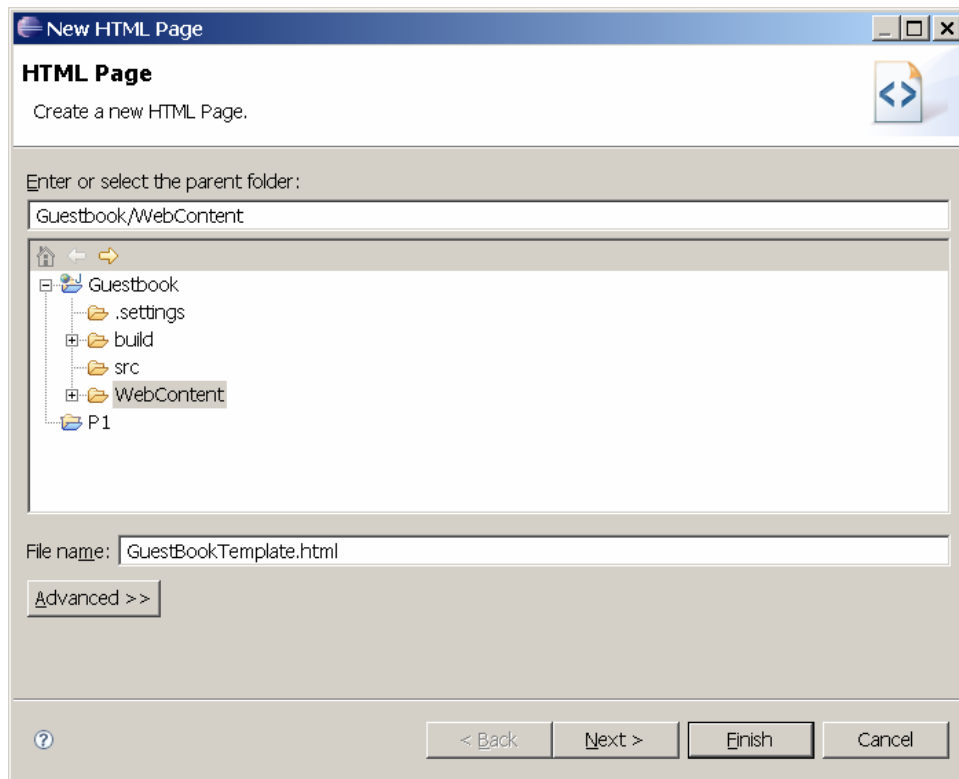


**Figure 19: Libraries added to the Guestbook project**

## ***2.3 Writing the template to provide the static content***

Next, we will develop the static content of the Guestbook application. This content consists primarily of an HTML template which will be filled in with dynamic content (the feedback left by visitors to your website) when the application is run.

To create the template, select the Guestbook project in the Project Explorer, and right-click. Next, choose New->HTML. This brings up the following dialog box, in which you can enter the template's name: GuestBookTemplate.html (see Figure 20).



**Figure 20: Creating the GuestBookTemplate.html file**

After pressing 'Finish', you will see the following screen (see Figure 21).

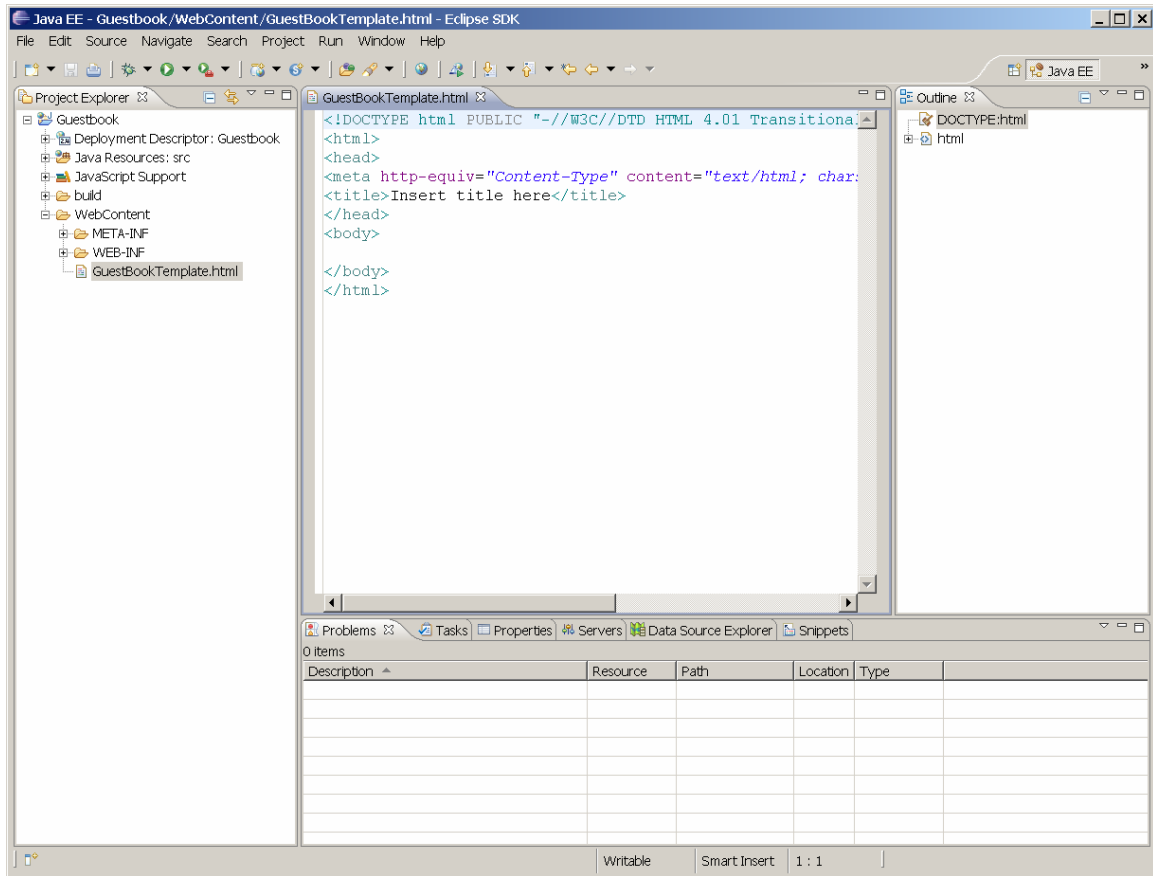


Figure 21: GuestBookTemplate.html after creation

Replace the auto-generated content with the following HTML code:

```
<HTML>
<HEAD>
  <TITLE>Guestbook</TITLE>
  <LINK REL="stylesheet" TYPE="text/css" HREF="GuestBook.css" MEDIA="all" />
</HEAD>
<BODY>
  <FORM ACTION="GuestBook.html" METHOD="POST">
    <DIV CLASS="Title">Guestbook</DIV>
    <DIV CLASS="Text">Add your comment here:</DIV>
    <DIV CLASS="Text"><TEXTAREA CLASS="entryfield" NAME="Comment"></TEXTAREA></DIV>
    <DIV CLASS="Text"><INPUT CLASS="button" TYPE="SUBMIT" VALUE="Add" /></DIV>
  </FORM>
  <DIV CLASS="Text">
    <REPEAT ID="Comments">
      <DIV CLASS="Text">Date: <REPLACE ID="Date">26.9.2007</REPLACE></DIV>
      <DIV CLASS="Text"><REPLACE ID="Comment">Hello world!</REPLACE></DIV>
      <BR/>
    </REPEAT>
  </DIV>
  <HR CLASS="Separator" />
  <INCLUDE SRC="Copyright.html" />
</BODY>
</HTML>
```

Your screen should now look as follows:

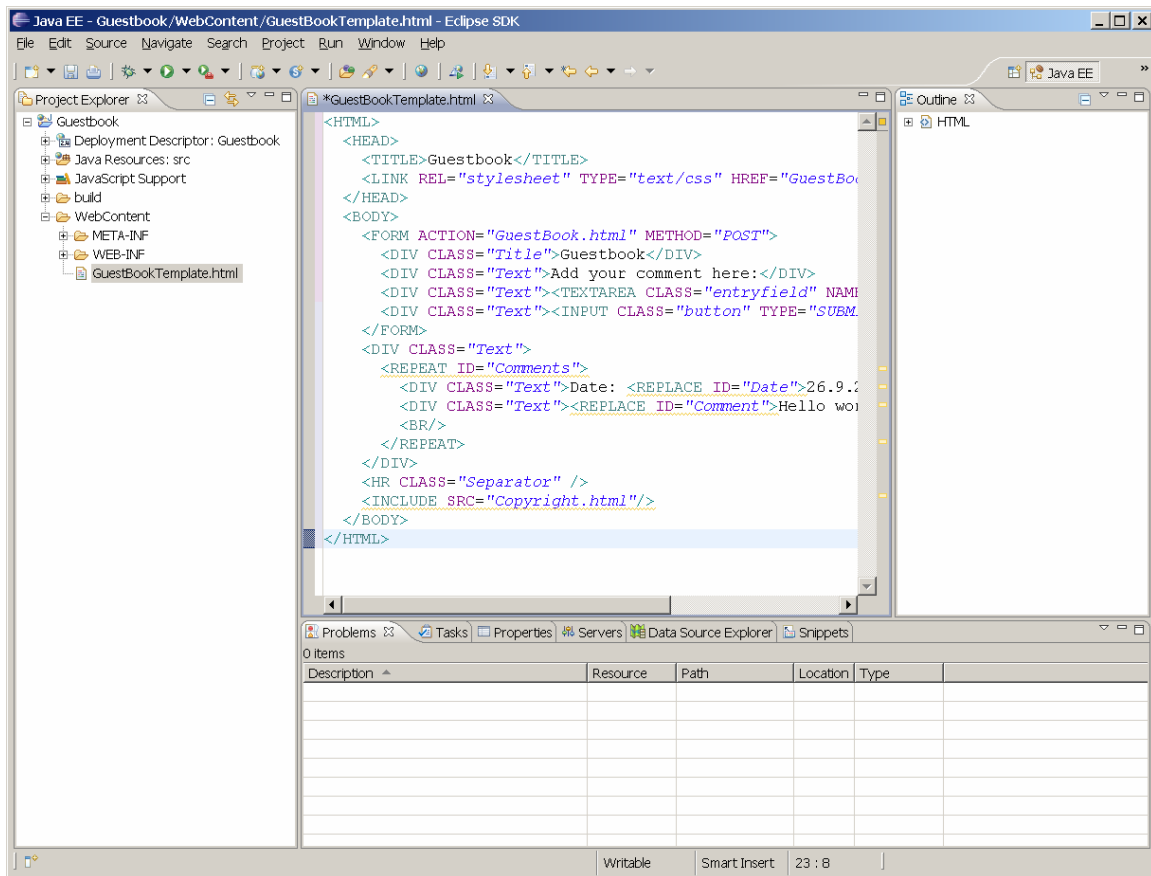


Figure 22: GuestBookTemplate.html in its finished form

A careful look at GuestBookTemplate.html reveals that this file references two additional files: GuestBook.css and Copyright.html.

Add Copyright.html in the same way you added GuestBookTemplate.html and enter the following HTML code for Copyright.html:

```

<DIV CLASS="Copyright">
(C) Copyright IBM Corp. 2007, 2009.
</DIV>

```

The GuestBook.css file is a cascading style sheet (CSS). It contains style information (font and color specifications) to give the Guestbook application an appealing look.

GuestBook.css contains the following content:

```

div.title {
font-family: Helvetica, sans-serif;
font-size: 14pt;
font-weight: bold;
text-align: left;
background-color: lightblue;
margin-bottom: 10px;
}

```

```

div.text {
    font-family: Helvetica, sans-serif;
    font-size: 10pt;
    text-align: left;
}

div.copyright {
    font-family: Helvetica, sans-serif;
    font-size: 8pt;
    text-align: left;
}

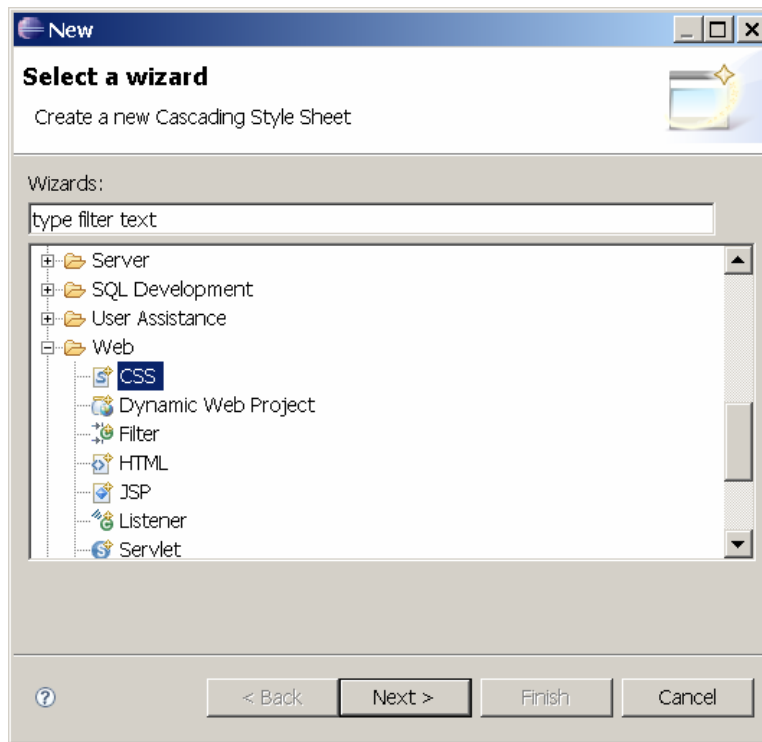
textarea.entryfield {
    font-family: Helvetica, sans-serif;
    font-size: 10pt;
    text-align: left;
    width: 400px;
    height: 80px;
}

hr.separator {
    height: 1px;
    color: black;
}

input.button {
    font-family: Helvetica, sans-serif;
    font-size: 10pt;
}

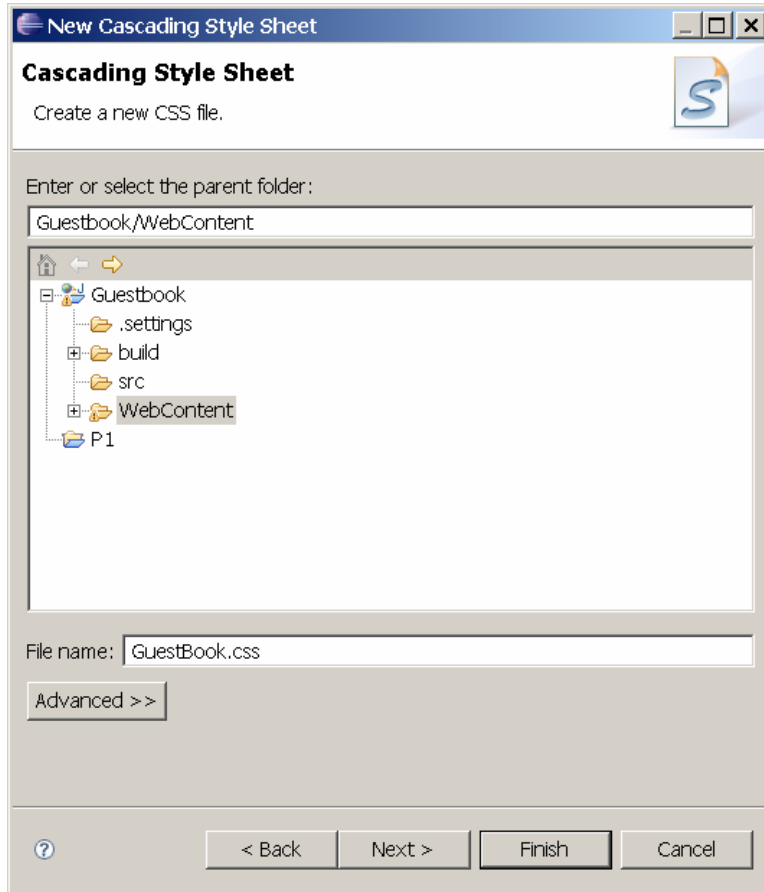
```

To add a CSS file to your project, proceed as follows. Select the Guestbook project in the Project Explorer and right-click to display the context-sensitive popup menu. From the menu, select New->Other to bring up a dialog box, in which you select the CSS wizard (see Figure 23).



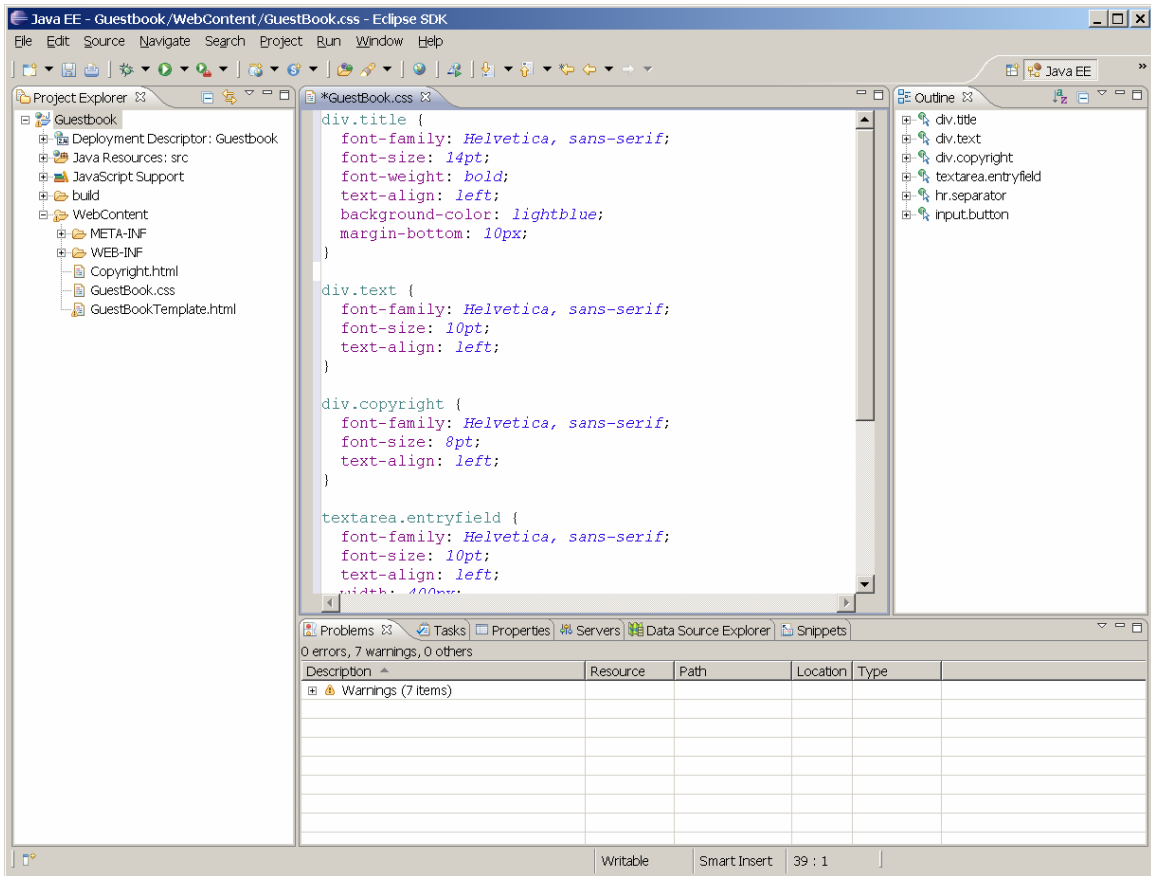
**Figure 23: Adding a CSS file**

Press 'Next' and provide the name for the cascading style sheet:



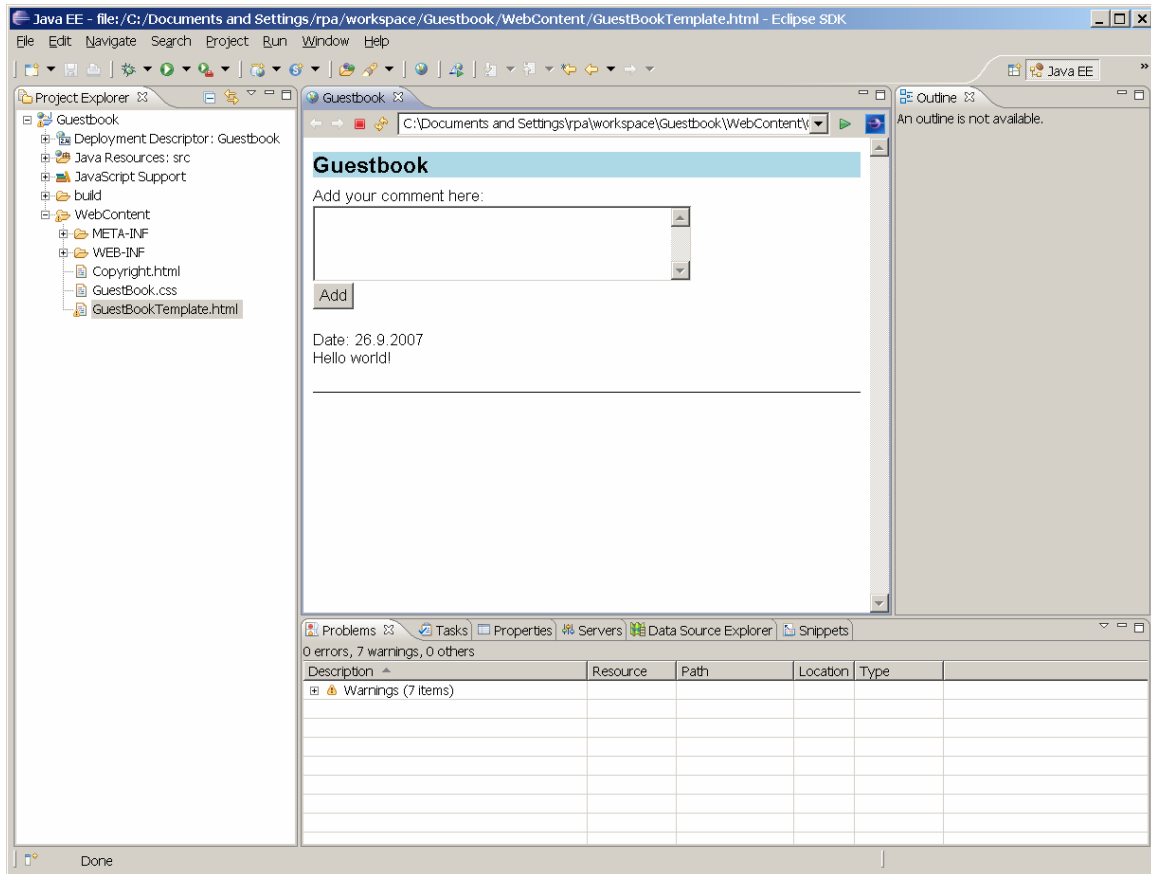
**Figure 24: Naming the Cascading Style Sheet (CSS)**

Press 'Finish' and enter the GuestBook.css content. Your screen should look now like this:



**Figure 25: GuestBook.css in its completed form**

Now it is time for a sneak preview of the Guestbook web application, even though it is not yet finished. Select the GuestBookTemplate.html file from the Project Explorer and right-click to display the popup menu. Select Open with -> Web Browser to see a preview of the Guestbook application containing some placeholder content (Date: 26.9.2007 'Hello world!') for the time being (see Figure 26).



**Figure 26: Preview of the Guestbook**

This preview (of GuestBookTemplate.html) gives you a first impression of how the application will look once it is finished. The Hamlets framework used for this web project enforces the complete separation of code and presentation. In other words, you can start developing an application by designing a number of static screens containing placeholder information and then bring these screens to life later by developing the Java code that provides the actual dynamic content. As you can see, the Guestbook application contains a first entry (Date: 26.9.2007 Hello world!), some dummy content, as this will facilitate the layout during the development of the template. It will later be replaced by actual content once the application has been finished and goes live.

## ***2.4 Providing dynamic content by writing a hamlet***

Next, we will develop the logic (the Java code) for the Guestbook application. The Java code will not only fill in the GuestBook template (GuestBookTemplate.html) with the actual dynamic content, but also store and retrieve the comments with the help of a file (comments.dat). To create this logic, select the Guestbook project in the Project Explorer, and right-click to bring up the context-sensitive popup menu. From the menu, select New->Class, which displays a dialog box to create a new Java class. Provide a class name (GuestBook) and a package name (com.ibm.webui) (see Figure 27).





**Figure 27: Creating the GuestBook.java class**

Press 'Finish' and enter the following Java code for GuestBook.java:

```
/**
 *
 * © Copyright International Business Machines Corporation 2007, 2009.
 * All rights reserved.
 *
 * The 'GuestBook' hamlet implements a simple guest book.
 *
 * File      : GuestBook.java
 * Created   : 2007/08/27
 *
 * @author   Rene Pawlitzek (rpa@zurich.ibm.com)
 * @version  1.00, 2009/02/13
 * @since    JDK 1.5
 *
 * History   : 2007/08/27, rpa, new file
 *             2009/02/13, rpa, code review
 */

package com.ibm.webui;

import java.io.*;
import java.util.*;
import java.text.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.ibm.hamlet.*;

import org.apache.commons.lang.*;
import org.apache.log4j.*;
import org.xml.sax.*;
```

```

public class GuestBook extends Hamlet {

    private static final long serialVersionUID = 5264892082158101750L;

    private static final String fileName = "Comments.dat";

    private static Logger logger = Logger.getLogger (GuestBook.class.getName ());

    private static String      filePath;
    private static Vector<Comment> comments;

    private static class Comment implements Serializable {

        private static final long serialVersionUID = -1199467372679306131L;

        public Date    date;
        public String  text;

        public Comment (Date date, String text) {
            this.date = date;
            this.text = text;
        } // Comment
    } // Comment

    private static class GuestBookHandler extends HamletHandler {

        private int      i = 0;
        private Comment  curComment;
        private Vector<Comment> comments;
        private SimpleDateFormat format;

        public GuestBookHandler (Hamlet hamlet, Vector<Comment> comments) {
            super (hamlet);
            this.comments = comments;
            format = new SimpleDateFormat ("yyyy-MM-dd   HH:mm:ss");
        } // GuestBookHandler

        public int getElementRepeatCount (String id, String name, Attributes atts) {
            return comments.size ();
        } // getElementRepeatCount

        public String getElementReplacement (String id, String name, Attributes atts)
            throws Exception {
            if (id.equals ("Date")) {
                curComment = (Comment) comments.elementAt (i);
                return format.format (curComment.date);
            } else if (id.equals ("Comment")) {
                i++;
                return curComment.text;
            } // if
            return "?";
        } // getElementReplacement
    } // GuestBookHandler

    @SuppressWarnings("unchecked")
    public void init () {
        BasicConfigurator.configure ();
        logger.debug ("init");
        filePath = getServletContext ().getRealPath ("/");
        logger.debug (filePath);
        try {
            ObjectInputStream in =
                new ObjectInputStream (new FileInputStream (filePath + fileName));
            comments = (Vector<Comment>) in.readObject ();
        } catch (Exception e) {
            comments = new Vector<Comment> ();
        } // try
    } // init

    public void doPost (HttpServletRequest req, HttpServletResponse res)
        throws ServletException {
        try {
            logger.debug ("doPost");
            String text = req.getParameter ("Comment");

```

```

        if (text != null) {
            text = StringEscapeUtils.escapeHtml (text);
            Comment comment = new Comment (new Date (), text);
            comments.add (0, comment);
            ObjectOutputStream out =
                new ObjectOutputStream (new FileOutputStream (filePath + fileName));
            out.writeObject (comments);
        } // if
        doGet (req, res);
    } catch (Exception e) {
        logger.error ("", e);
    } // try
} // doPost

public void doGet (HttpServletRequest req, HttpServletResponse res)
throws ServletException {
    try {
        logger.debug ("doGet");
        HamletHandler handler = new GuestBookHandler (this, comments);
        serveDoc (req, res, "GuestBookTemplate.html", handler);
    } catch (Exception e) {
        logger.error ("", e);
    } // try
} // doGet

} // Guestbook

/* ----- End of File ----- */

```

When you have finished entering the Java source code, your screen will look like this (see Figure 28). Don't forget to save your work with File->Save.

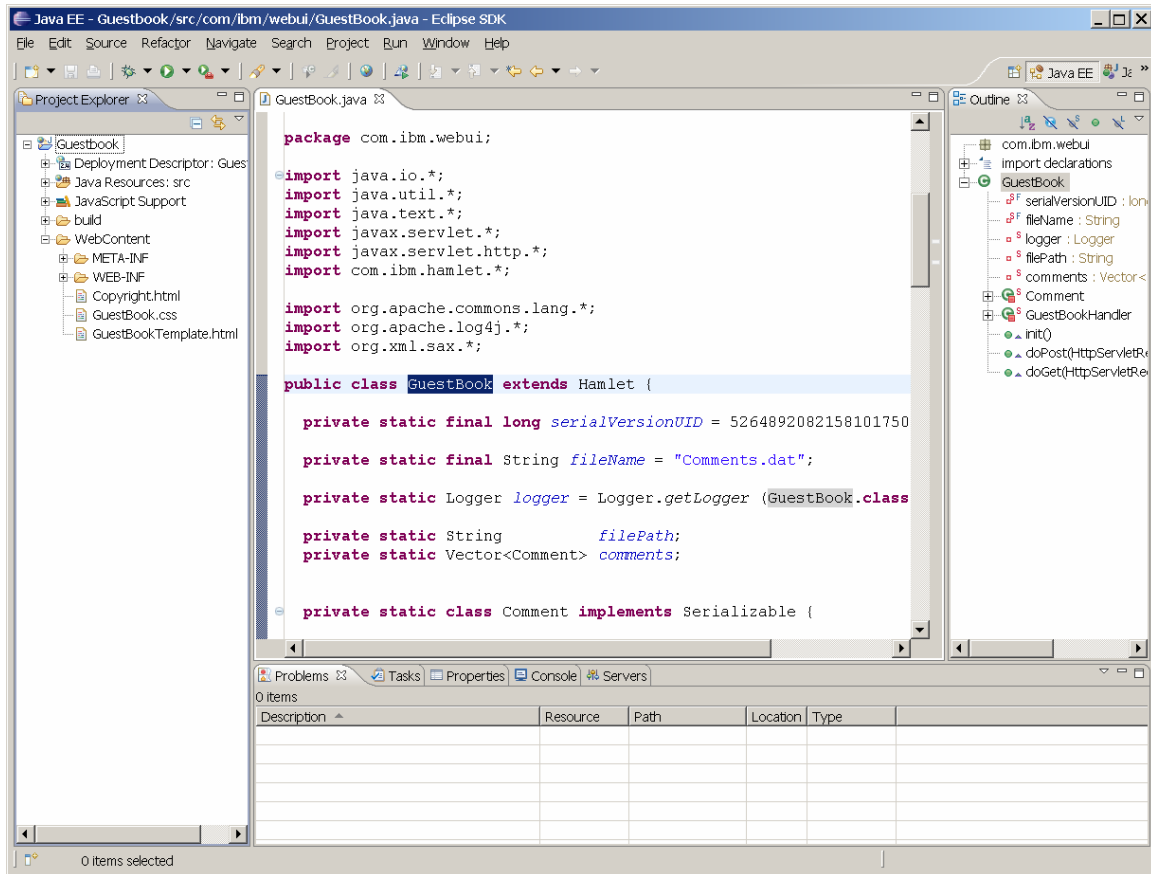


Figure 28: GuestBook.java source code

## 2.5 Running the web application

There is one more, small thing that you have to do before you can run the Guestbook application. You need to provide the content for the web.xml file. This file configures the Guestbook application to run properly on an application server. To change the web.xml file already created, select it in the Project Explorer and right-click to display the popup menu. Select Open with->Text Editor from the menu. Next, overwrite the existing content with following lines:

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" 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 http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">

    <display-name>Guestbook</display-name>

    <servlet>
        <description>A simple guest book hamlet</description>
        <display-name>GuestBook hamlet</display-name>
        <servlet-name>GuestBook</servlet-name>
        <servlet-class>com.ibm.webui.GuestBook</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>GuestBook</servlet-name>
        <url-pattern>/index.html</url-pattern>

```

```

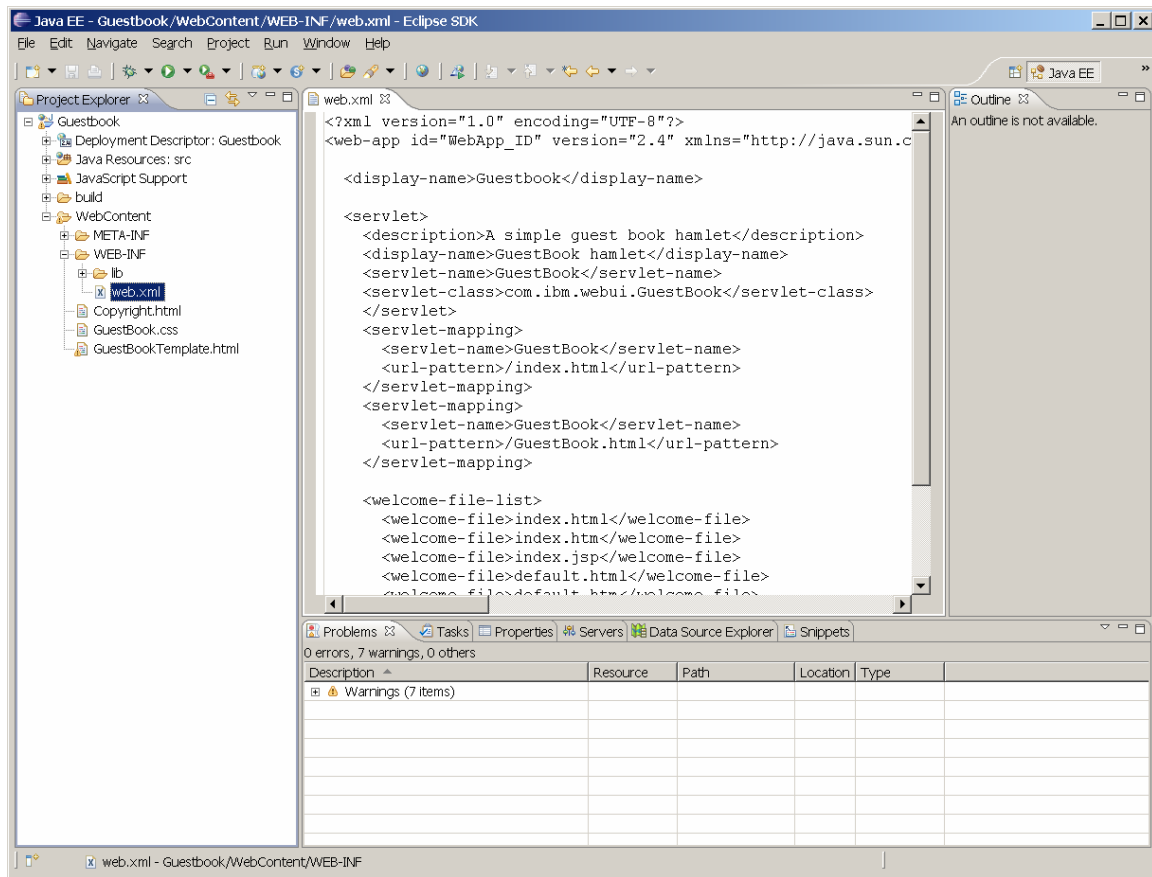
</servlet-mapping>
<servlet-mapping>
  <servlet-name>GuestBook</servlet-name>
  <url-pattern>/GuestBook.html</url-pattern>
</servlet-mapping>

<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
</welcome-file-list>

</web-app>

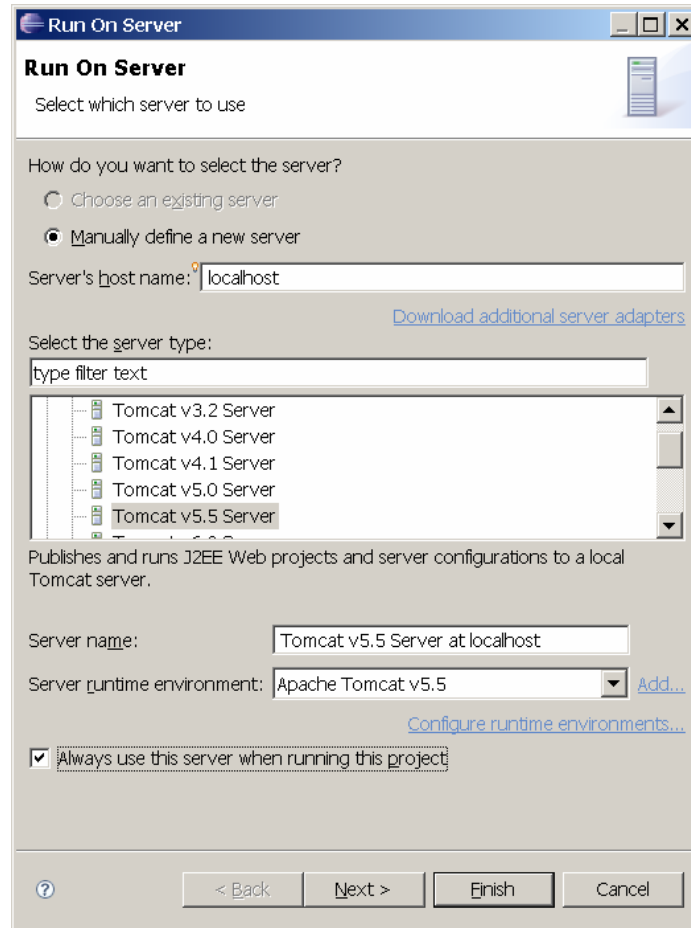
```

When you have done this, your screen will look like this:



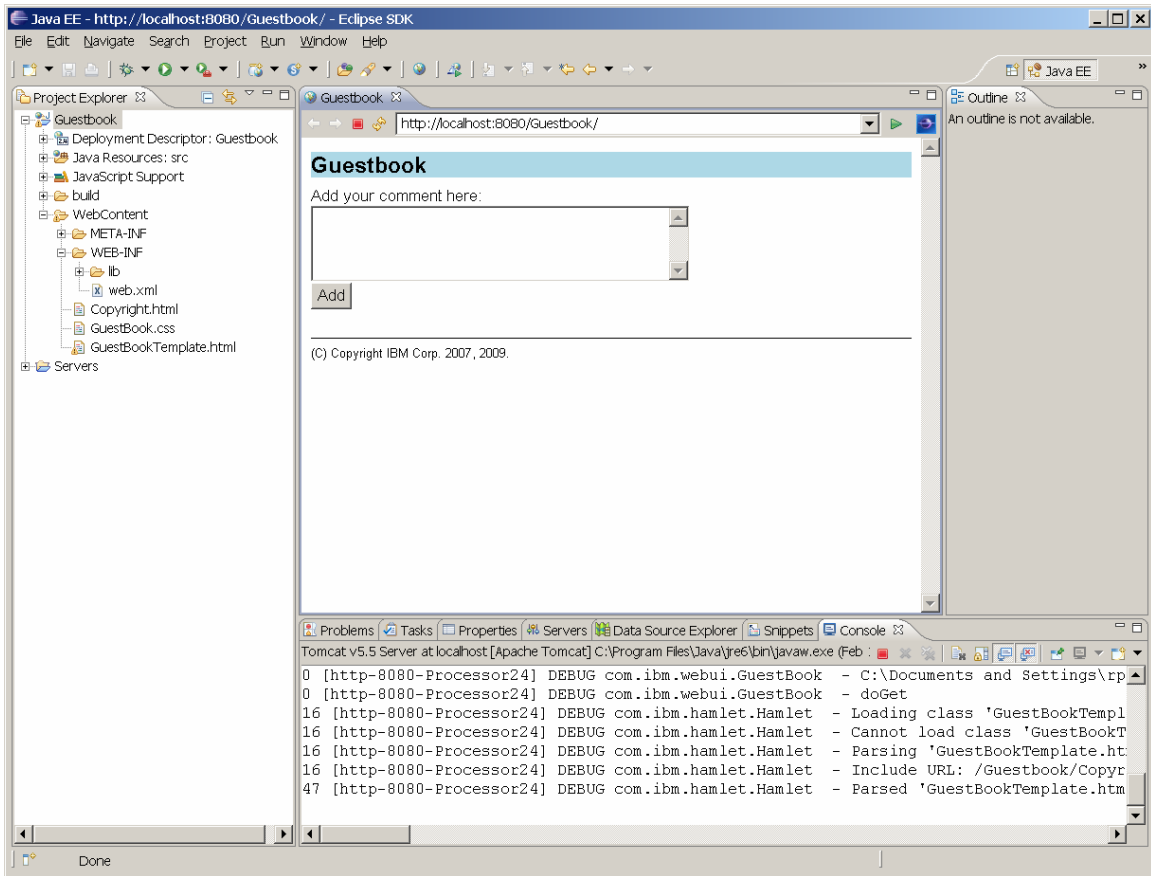
**Figure 29: Providing the content for web.xml**

Now comes the big moment. You are ready to run the Guestbook application. Select the Guestbook project in the Project Explorer, right-click to display the popup menu then select Run As->Run on Server. You will see the following screen:



**Figure 30: Running the Guestbook application**

Select 'Tomcat v5.5 Server' and check 'Always use this server when running this project'. Next, press 'Finish', and wait a short moment for the Guestbook application to appear (see Figure 31). Congratulations! You have successfully developed and deployed your first web-based application using the Hamlets framework in Eclipse. Now enter a comment in the guestbook to test it.



**Figure 31: Running Guestbook application**

Press 'Add' to add your comment to the page (see Figure 32).

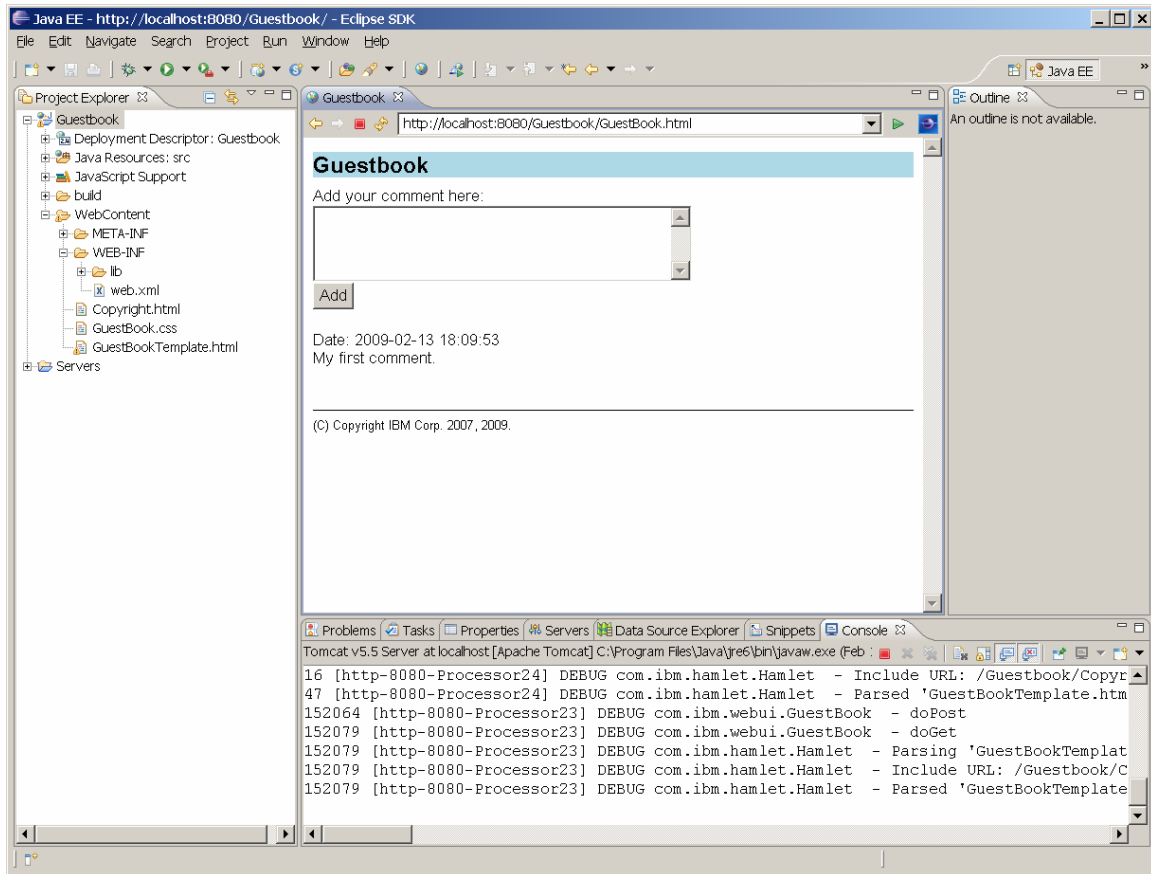


Figure 32: Guestbook web application displaying a first comment

As you can see, magic still exists! In the next section, I lift the secrets of the Java code of the Guestbook application.

### 3. Explaining the web application

It is time now to explain how the Guestbook application works. Let's start with the data structure. The Java class `Comment` is an inner class used to hold the user comments in memory.

```
private static class Comment implements Serializable {
    public Date    date;
    public String  text;

    public Comment (Date date, String text) {
        this.date = date;
        this.text = text;
    } // Comment
} // Comment
```

A comment in the Guestbook application consists of a date (to store the time when the comment was made) and a string (to store the user comment content). All comments are stored in a vector called `comments` throughout the lifetime of the application.



```
private static Vector<Comment> comments;
```

The Guestbook application itself is implemented by the GuestBook class, which extends the Hamlet class. It contains three methods: `init()`, `doPost()` and `doGet()`.

```
public class GuestBook extends Hamlet {  
  
    public void init    ()    { ... };  
    public void doPost (... ) { ... };  
    public void doGet  (... ) { ... };  
  
} // GuestBook
```

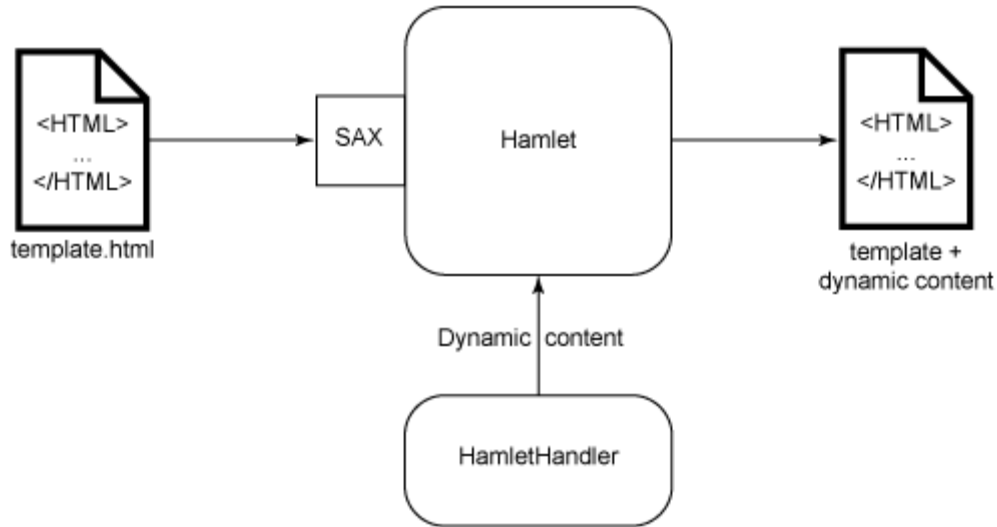
When the application is loaded onto an application server, the `init()` function is called once. The purpose of this function is to retrieve previously saved comments from a file (`comments.dat`) by calling `readObject` on an `ObjectInputStream`.

```
public void init () {  
    BasicConfigurator.configure ();  
    logger.debug ("init");  
    filePath = getServletContext ().getRealPath ("/");  
    logger.debug (filePath);  
    try {  
        ObjectInputStream in =  
            new ObjectInputStream (new FileInputStream (filePath + fileName));  
        comments = (Vector<Comment>) in.readObject ();  
    } catch (Exception e) {  
        comments = new Vector<Comment> ();  
    } // try  
} // init
```

The `doGet()` function is called when a user points his or her browser to the Guestbook application. It is responsible for displaying the actual content of the page.

```
public void doGet (HttpServletRequest req, HttpServletResponse res)  
    throws ServletException {  
    try {  
        logger.debug ("doGet");  
        HamletHandler handler = new GuestBookHandler (this, comments);  
        serveDoc (req, res, "GuestBookTemplate.html", handler);  
    } catch (Exception e) {  
        logger.error ("", e);  
    } // try  
} // doGet
```

`doGet()` creates an instance of `GuestBookHandler` with `new GuestBookHandler (...)` and passes this instance together with the name of a template (`GuestBookTemplate.html`) to the `serveDoc()` method. Now the Hamlet framework takes action. It uses a SAX parser to read the XHTML code from the template. In addition to the usual HTML tags, a template can contain a number of 'special' tags, namely `<REPLACE>`, `<REPEAT>`, and `<INCLUDE>`. Whenever one of these tags is found in the template, a method of the `GuestBookHandler` class is invoked to provide dynamic content, which the Hamlets framework combines with the static XHTML content from the template (see Figure 33).



**Figure 33: A Hamlet uses SAX for reading content from a template and calls a HamletHandler to add dynamic content**

The GuestBookTemplate.html contains the following XHTML sequence:

```

<REPEAT ID="Comments">
  <DIV CLASS="Text">Date: <REPLACE ID="Date">26.9.2007</REPLACE></DIV>
  <DIV CLASS="Text"><REPLACE ID="Comment">Hello world!</REPLACE></DIV>
  <BR/>
</REPEAT>
  
```

The Hamlet framework encounters the `<REPEAT ID="Comments">` tag during template parsing and records all subsequent XHTML content in memory. When the `</REPEAT>` tag is encountered, it stops the recording and invokes the `getElementRepeatCount()` method of the `GuestBookHandler` class:

```

private static class GuestBookHandler extends HamletHandler {
    ...
    public int getElementRepeatCount (String id, String name, Attributes atts) {
        return comments.size ();
    } // getElementRepeatCount
    ...
} // GuestBookHandler
  
```

The code in this method returns the number of comments (e.g.,  $N = 2$ ). Subsequently, the recorded content is played back  $N$  times, producing the following sequence:

```

<DIV CLASS="Text">Date: <REPLACE ID="Date">26.9.2007</REPLACE></DIV>
<DIV CLASS="Text"><REPLACE ID="Comment">Hello world!</REPLACE></DIV>
<BR/>
<DIV CLASS="Text">Date: <REPLACE ID="Date">26.9.2007</REPLACE></DIV>
<DIV CLASS="Text"><REPLACE ID="Comment">Hello world!</REPLACE></DIV>
<BR/>
  
```

Next, the Hamlet framework parses this content. When the `<REPLACE ID="Date">` tag is encountered, the `getElementReplacement()` method is called:

```

private static class GuestBookHandler extends HamletHandler {

    ...
    public String getElementReplacement (String id, String name, Attributes atts)
    throws Exception {
        if (id.equals ("Date")) {
            curComment = (Comment) comments.elementAt (i);
            return format.format (curComment.date);
        } else if (id.equals ("Comment")) {
            i++;
            return curComment.text;
        } // if
        return "?";
    } // getElementReplacement
    ...
} // GuestBookHandler

```

This function checks the tag's ID attribute (`ID="Date"`) and replaces the placeholder value `26.9.2007` between the `<REPLACE>` and `</REPLACE>` tags with the date of the  $n$ -th comment. Similarly, the `getElementReplacement()` will replace the other placeholder values with the content of all comments. In the end, the sequence

```

<DIV CLASS="Text">Date: <REPLACE ID="Date">26.9.2007</REPLACE></DIV>
<DIV CLASS="Text"><REPLACE ID="Comment">Hello world!</REPLACE></DIV>
<BR/>
<DIV CLASS="Text">Date: <REPLACE ID="Date">26.9.2007</REPLACE></DIV>
<DIV CLASS="Text"><REPLACE ID="Comment">Hello world!</REPLACE></DIV>
<BR/>

```

will turn into:

```

<DIV CLASS="Text">Date: <REPLACE ID="Date">13.3.2009</REPLACE></DIV>
<DIV CLASS="Text"><REPLACE ID="Comment">second comment</REPLACE></DIV>
<BR/>
<DIV CLASS="Text">Date: <REPLACE ID="Date">1.1.2009</REPLACE></DIV>
<DIV CLASS="Text"><REPLACE ID="Comment">first comment</REPLACE></DIV>
<BR/>

```

And finally, the Hamlets framework replaces the `<INCLUDE SRC="Copyright.html"/>` tag in the template with the content of the `copyright.html` file. Attributes of a tag can be added, deleted, and modified with the `getElementAttributes()` method. However, for this example we did not need to do that.

The Guestbook application's `doPost()` method is invoked when a user comment is entered and submitted by pressing the 'Add' button in the browser.

```

public void doPost (HttpServletRequest req, HttpServletResponse res)
throws ServletException {
    try {
        logger.debug ("doPost");
        String text = req.getParameter ("Comment");
        if (text != null) {
            text = StringEscapeUtils.escapeHtml (text);
            Comment comment = new Comment (new Date (), text);
            comments.add (0, comment);
            ObjectOutputStream out =
                new ObjectOutputStream (new FileOutputStream (filePath + fileName));
            out.writeObject (comments);
        } // if
        doGet (req, res);
    } catch (Exception e) {
        logger.error ("", e);
    } // try
} // doPost

```

The web application server retrieves the comment in the Guestbook application by calling `req.getParameter ("Comment")`. Next, the comment is HTML-escaped with `StringEscapeUtils.escapeHtml (text)` from the Apache Commons Lang library and added to the vector of comments as the first element with `comments.add (0, comment)`. Finally, all comments are saved to the disk with `out.writeObject (comments)` before `doGet (req, res)` is called to return the content of the updated page.

As you can see, the Hamlets framework enforces the separation of content (provided by the Java code) and presentation (XHTML code) so that Java and XHTML code do not end up intermixed in the same source file. This separation is highly desirable because developers writing Java code and designers writing XHTML code can work independently of each other, and changes in the XHTML code will not require any retesting of the Java code. Maintaining the application becomes a lot easier. More information on Hamlets can be found in the "Introducing Hamlets" paper listed in the Resources section. A summary of the full Hamlets functionality is given in Table 1.

Template content	HamletHandler callback method	Result	Description
<REPLACE ID="Name"> Joe Average </REPLACE>	public String getElementReplacement (String id, String name, Attributes atts) throws Exception { if (id.equals ("Name")) return "Rene Pawlitzek"; return "?"; } // getElementReplacement	Rene Pawlitzek	The <REPLACE> tag and the getElementReplacement() callback replace content in the template.
<REPEAT ID="Rows"> <TR><TD>Row</TD></TR> </REPEAT>	public int getElementRepeatCount (String id, String name, Attributes atts) throws Exception { if (id.equals ("Rows")) return 5; return 0; } // getElementRepeatCount	<TR><TD>Row</TD></TR> <TR><TD>Row</TD></TR> <TR><TD>Row</TD></TR> <TR><TD>Row</TD></TR>	The <REPEAT> tag and the getElementRepeatCount() callback repeat sections in the template.
<TR ID="Color" BGCOLOR="#FFFFFF" >	public Attributes getElementAttributes (String id, String name, Attributes atts) throws Exception { if (id.equals ("Color")) atts = Helpers.getAttributes (atts, "BGCOLOR", "#FCFCFC"); return atts; } // getElementAttributes	<TR ID="Color" BGCOLOR="#FCFCFC" >	The getElementAttributes() callback changes tag attributes in the template.
<TR ID="Color" BGCOLOR="#FFFFFF" >	public Attributes getElementAttributes (String id, String name, Attributes atts) throws Exception { if (id.equals ("Color")) atts = Helpers.getAttributes (atts, "BGCOLOR", null); return atts; } // getElementAttributes	<TR ID="Color">	The getElementAttributes() callback removes tag attributes in the template.
<TR ID="Color">	public Attributes getElementAttributes (String id, String name, Attributes atts) throws Exception { if (id.equals ("Color")) atts = Helpers.getAttributes (atts, "BGCOLOR", "#FCFCFC"); return atts; } // getElementAttributes	<TR ID="Color" BGCOLOR="#FCFCFC" >	The getElementAttributes() callback adds tag attributes in the template.
<INCLUDE ID="Copyright" SRC="copyright.html" />		<DIV> © Copyright IBM Corp. 2009. </DIV>	The <INCLUDE> tag inserts content into the template.

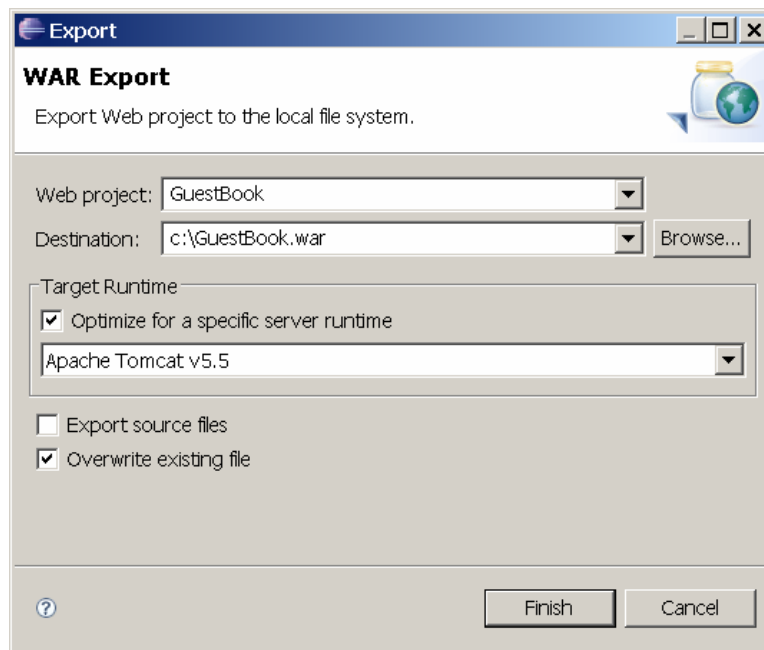
**Table 1: Functionality of the Hamlets framework**

## 4. Deploying the web application

You now have all the information to write (small) web-based applications in Eclipse on your own. Eventually, you will want to run your web apps on the production server of your company. To do so, you must package the application. In other words, you need to produce a WAR file (containing your web application), which you can then install on the web application server of your production system.

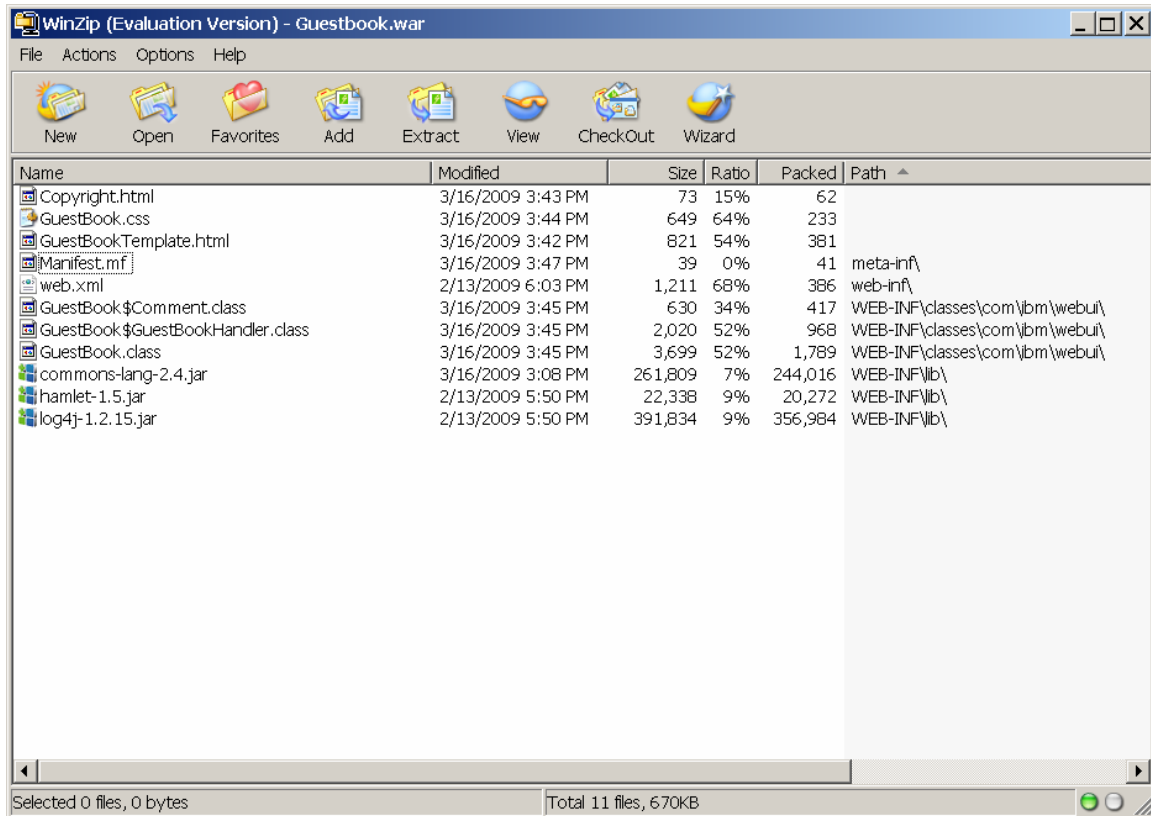
### 4.1 Producing a WAR file by exporting the project

Producing a WAR file is similar to generating an .exe file in a Windows development environment. A WAR file is a web archive; it contains all parts of a web application in a single ZIP file. Instead of using a .zip ending, WAR files use the extension .war. The generation of a WAR file in Eclipse is straightforward. Select the Guestbook application in the Project Explorer and right-click. From the popup menu choose Export->WAR file to display the Export dialog and enter a destination for the WAR file (see Figure 34).



**Figure 34: Creating the GuestBook.war file**

Press 'Finish' to create the GuestBook.war file. You can view the content of this web archive with WinZip or any other tool that can handle zip files (see Figure 35).



**Figure 35: Opening the GuestBook.war file with WinZip**

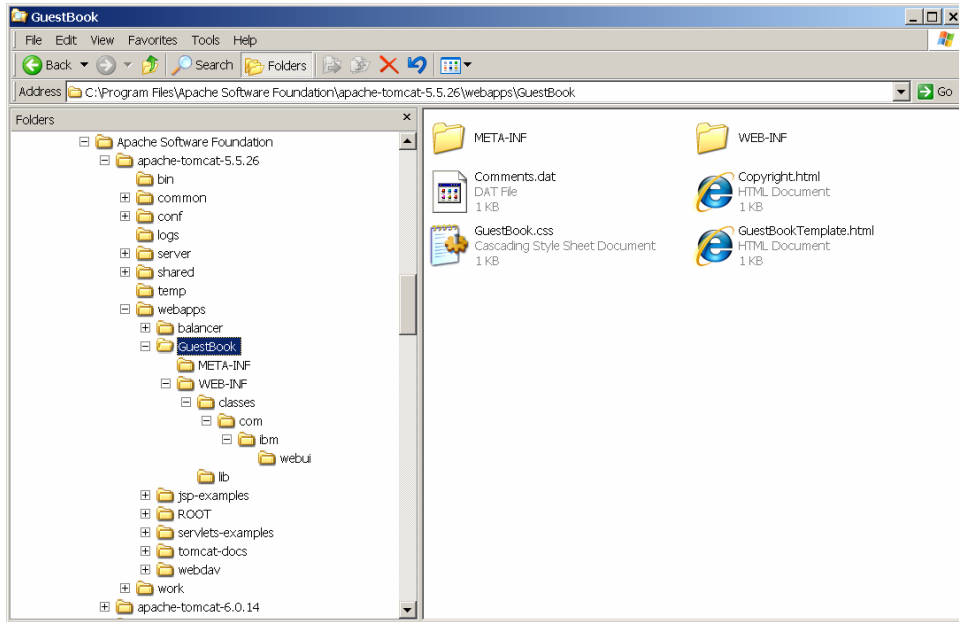
A closer look at GuestBook.war reveals that the web archive contains a number of already familiar items:

- The template GuestBookTemplate.html with placeholder content, the Copyright.html file with the copyright message, and the cascading style sheet GuestBook.css with font and color instructions.
- The three libraries: commons-lang-2.4.jar, hamlet-1.5.jar, and log4j-1.2.15.jar.
- The three class files GuestBook.class, GuestBook\$Comment.class and GuestBook\$GuestBookHandler.class containing the logic (Java bytecode) of the Guestbook application.
- The web.xml file with instructions for the application server on how to run the Guestbook application.

## ***4.2 Installing the WAR file on an application server***

You can give the GuestBook.war file to other people and ask them to have a look at your first web-based application. Anyone who runs the Tomcat web application server on their

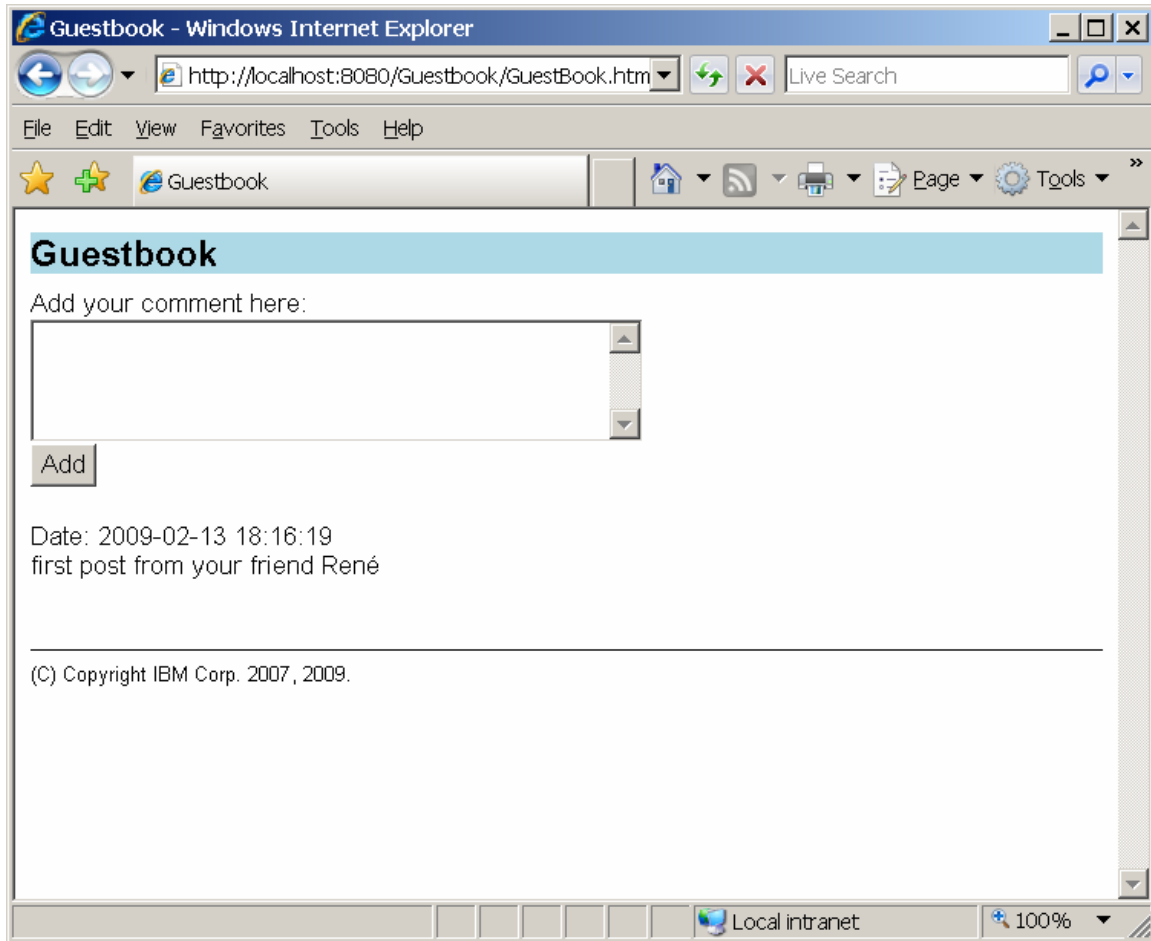
machine can simply extract the GuestBook.war file into a newly created subdirectory of the webapps directory to install your application (see Figure 36).



**Figure 36: Installing GuestBook by extracting GuestBook.war in the GuestBook subdirectory**

After GuestBook.war has been unzipped, your application is ready to be used. To run it, they simply have to point the browser to <http://localhost:8080/GuestBook/GuestBook.html> (see Figure 37).

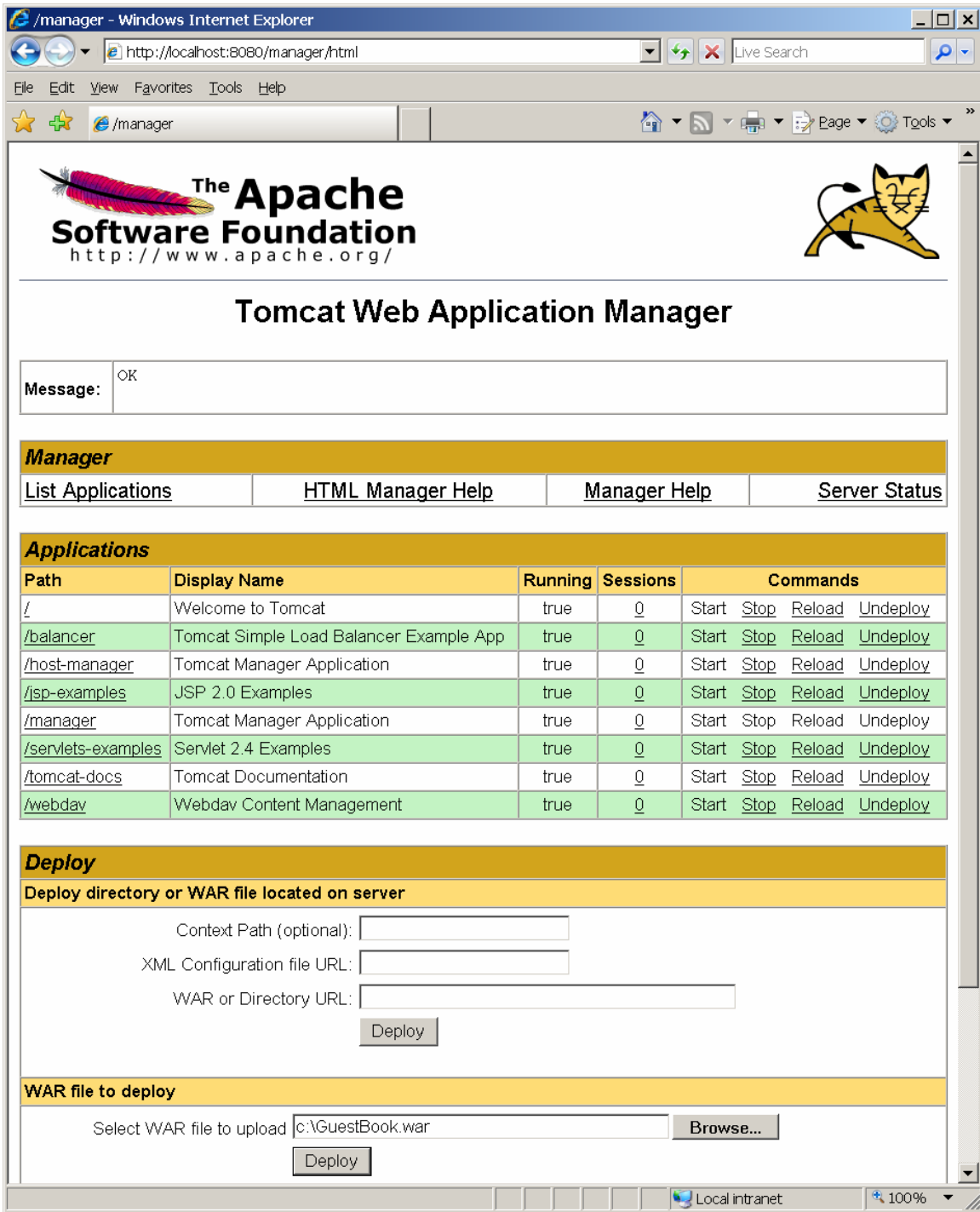




**Figure 37: Guestbook application in action**

There is another (more convenient) way to install a web application. Most web application servers provide a web-based application to install WAR files (even remotely).

Unfortunately, Tomcat's administration web application is no longer installed by default. You need to download and install the "admin" package in order to use it (see Figure 38). For more information check the Resources section.



**Figure 38: Installing GuestBook.war with the Tomcat Web Application Manager**

If you do not see the Guestbook application in the browser and instead encounter an error message (see below), Tomcat is quite likely running with an older Java version.

```
SEVERE: Allocate exception for servlet GuestBook
java.lang.UnsupportedClassVersionError: Bad version number in .class file
at java.lang.ClassLoader.defineClass1(Native Method)
at java.lang.ClassLoader.defineClass(ClassLoader.java:620)
at java.security.SecureClassLoader.defineClass(SecureClassLoader.java:124)
at org.apache.catalina.loader.WebappClassLoader.findClassInternal(WebappClassLoader.java:1853)
at org.apache.catalina.loader.WebappClassLoader.findClass(WebappClassLoader.java:875)
at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1330)
at org.apache.catalina.loader.WebappClassLoader.loadClass(WebappClassLoader.java:1209)
at org.apache.catalina.core.StandardWrapper.loadServlet(StandardWrapper.java:1068)
at org.apache.catalina.core.StandardWrapper.allocate(StandardWrapper.java:791)
at org.apache.catalina.core.StandardWrapperValve.invoke(StandardWrapperValve.java:127)
at org.apache.catalina.core.StandardContextValve.invoke(StandardContextValve.java:174)
at org.apache.catalina.core.StandardHostValve.invoke(StandardHostValve.java:127)
at org.apache.catalina.valves.ErrorReportValve.invoke(ErrorReportValve.java:117)
at org.apache.catalina.core.StandardEngineValve.invoke(StandardEngineValve.java:108)
at org.apache.catalina.connector.CoyoteAdapter.service(CoyoteAdapter.java:174)
at org.apache.coyote.http11.Http11Processor.process(Http11Processor.java:874)
at org.apache.coyote.http11.Http11BaseProtocol$Http11ConnectionHandler.processConnection(Http11BaseProtocol.java:665)
at org.apache.tomcat.util.net.PoolTcpEndpoint.processSocket(PoolTcpEndpoint.java:528)
at org.apache.tomcat.util.net.LeaderFollowerWorkerThread.runIt(LeaderFollowerWorkerThread.java:81)
at org.apache.tomcat.util.threads.ThreadPool$ControlRunnable.run(ThreadPool.java:689)
at java.lang.Thread.run(Thread.java:595)
```

To make things work, you need to upgrade to a more recent Java release (1.5 or 1.6).

You are now able to create Java-based web applications with the Hamlets framework that (thanks to Java) are cross-platform. In other words, your Guestbook application can run on a low-end PC or a high-end mainframe, as well as on everything in between.

If you want to know more about the Hamlets framework, read the other Hamlets articles. I hope you enjoy your journey.

Happy Programming!

## Acknowledgments

I thank Andreas Kind, Mathias Bjoerkqvist, and Xiao-Yu Hu for their valuable comments and suggestions.

## Resources

### For learning:

- [“Introducing Hamlets”](#), René Pawlitzek (developerWorks, March 2005): This article introduces the basic ideas behind Hamlets and describes the very first implementation.
- [“Programming Hamlets”](#), René Pawlitzek (developerWorks, May 2005): This tutorial illustrates various aspects of Hamlet programming as it provides a number of practical Hamlet examples.
- [“Implementing Hamlets”](#), René Pawlitzek (developerWorks, February 2006): This article explains the current implementation of the Hamlets framework.

- “[Compiling Hamlets](#)”, René Pawlitzek (developerWorks, June 2006): This article introduces a method of compiling Hamlet templates to improve application performance.
- “[Embedding Hamlets](#)”, René Pawlitzek (developerWorks, June 2007): This article shows how to write a web-based user interface for embedded devices running OSGi.
- “[Java Servlet Programming](#)”, Jason Hunter, O’Reilly & Associates, Inc.: This is an excellent book on servlets, which are the key components of server-side Java development.
- “[Tomcat – The Definitive Guide](#)”, Jason Brittain and Ian Darwin, O’Reilly & Associates, Inc.: This book describes the Apache Tomcat application server in great detail.
- “[HTML & XHTML – The Definitive Guide](#)”, Chuck Musciano and Bill Kennedy, O’Reilly & Associates, Inc.: This book describes the Hypertext Markup Language (HTML) and the Extensible Hypertext Markup Language (XHTML).
- “[Cascading Style Sheets](#)”, Eric A. Meyer, O’Reilly & Associates, Inc.: This book describes how to control the layout of web pages with Cascading Style Sheets.
- Find out more about [Hamlets](#) and [Web application](#) at Wikipedia.
- Watch these [video tutorials](#) to learn more about Eclipse and Java.

### **To get products and technologies:**

- [Hamlets home page](#): Now that this project is open source, check it out on SourceForge.
- [Java](#): Get your “cup” of Java from Sun Microsystems.
- [Log4j](#): Download this logging package from the Apache Project.
- [Commons](#): Check out this Apache project which focuses on reusable Java components.
- [Tomcat](#): Download this servlet container and Web server from the Apache Project.
- [Eclipse](#): The open source IDE used in this article.

## About the author

[René Pawlitzek](#) is a citizen of Liechtenstein and holds an engineering degree in computer science from the Swiss Federal Institute of Technology (ETH Zürich). René works as a research and development engineer and as a project manager for the Systems Management group at the IBM Zurich Research Laboratory in Switzerland. Before coming to IBM, he worked in California for Hewlett-Packard, WindRiver Systems, and Borland International.

## Trademarks

IBM is trademarks of International Business Machines Corporation in the United States, other countries, or both. Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both. Microsoft, Windows, and Internet Explorer are trademarks of Microsoft Corporation in the United States, other countries, or both. Other company, product, or service names may be trademarks or service marks of others.