Лабораторная работа №4. Аутентификация и работа с сессиями

Цель работы

Необходимо развернуть и настроить web-сервер. Необходимо реализовать аутентификацию пользователя на сайте, познакомиться с механизмом сессий.

Введение

Разработка и распространение веб-приложений под управлением Tomcat

Создание структуры каталогов и размещение веб-страниц

Все веб-приложения размещаются в каталоге webapps (/srv/tomcat/webapps). Каждое приложение размещается в собственном, одноименном, каталоге с определенной вложенной структурой (webapps/youappname/WEB-INF/classes).
Для нового веб-приложения (например, myapp) эту структуру можно создать, например, так:

```
// WEB-INF - предопределенное и регистрозависимое имя каталога
aag@stilo:~> sudo mkdir -p /srv/tomcat/webapps/myapp/WEB-INF/classes
```

Назначение созданных каталогов следующее:

- myapp: Корневой каталог веб-приложения. Здесь размещаются HTML-страницы и прочие ресурсы (таблицы стилей (CSS), изображения, клиентские скрипты (javascript), JSP и т.п.), доступные веб-клиентам.
- myapp/WEB-INF: Этот каталог, недоступный веб-пользователям, содержит описание веб-приложения и его параметры в файле web.xml.
- myapp/WEB-INF/classes: В этом каталоге размещаются все файлы Java-классов сервлетов.

Поместим в корневой каталог создаваемого веб-приложения файл index.html следующего содержания:

```
<!-- Сохранить как /srv/tomcat/webapps/myapp/index.html -->
<html>
<head>
        <meta charset="utf-8" />
        <title>Static HTML sample</title>
</head>
<body>
        <h1>Здравствуй, мир!</h1>
```

```
</body>
</html>
```

После перезапуска сервера к создаваемому веб-приложению можно будет обратиться по адресу http://localhost:8080/myapp/(рис. 2).

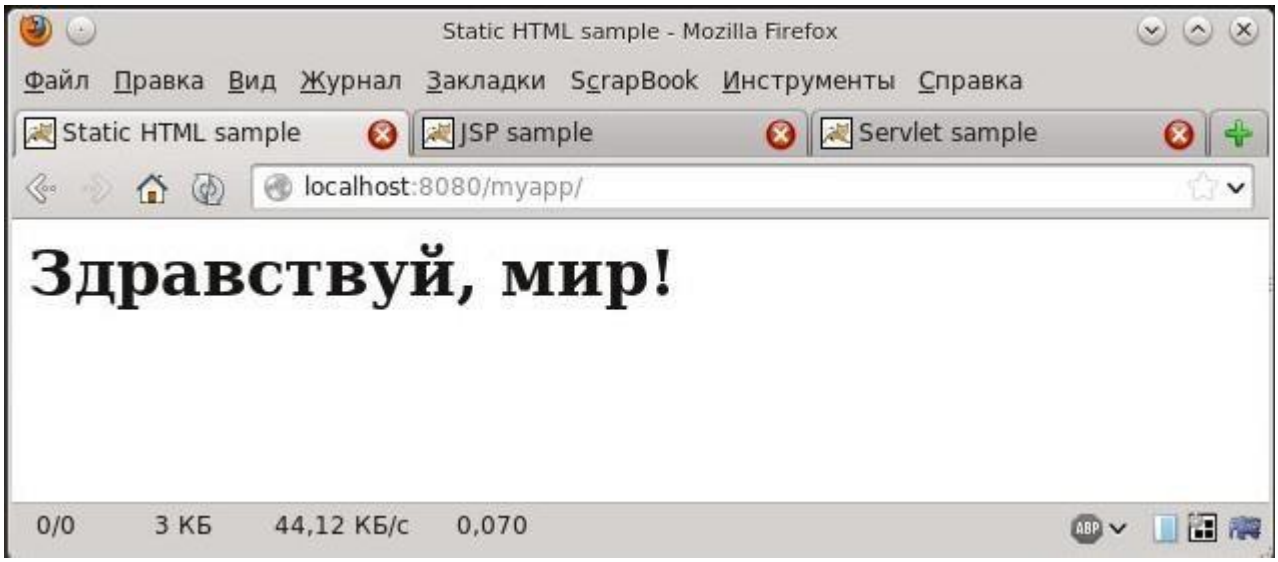


Рисунок 1 - Отображение статических страниц

Java Server Pages

Java Server Pages (JSP) - динамические веб-страницы, содержащие, помимо HTML-разметки, инструкции на языке Java
(подобно PHP или ASP). Такие ресурсы размещаются и используются так же, как и статические ресурсы. Пример JSP приведен в листинге 1, а результат на рис. 3.

Листинг 1. Пример разметки JSP

```
<!-- Сохранить как /srv/tomcat/webapps/myapp/hello.jsp -->
<html>
<head>
        <meta charset="utf-8" />
        <title>JSP sample</title>
</head>
<body>
        <%= new String("<h1>Hello, world!</h1>") %>
</body>
</html>
```

Рисунок 2 - Вывод JSP-страниц

Примечание: Если при обращении к JSP-странице вы получаете сообщение об ошибке вида:

```
org.apache.jasper.JasperException: java.lang.IllegalStateException: No output folder....,
```

это скорее всего означает, что каталог tomcat/work не доступен для записи.

Сервлеты

Все сервлеты (серверные приложения на Java) размещаются в подкаталоге WEB-INF/classes/. Рассмотрим использование сервлетов на примере приложения, выводящего некоторую информацию о сервере (листинг 1).

Листинг 1. Исходный код Java-сервлета

```
// Сохранить как /srv/tomcat/webapps/myapp/WEB-INF/classes/MyServlet.java
 import java.io.*;
import javax.servlet.*;
import
javax.servlet.http.*;
  public class MyServlet extends
HttpServlet {
   @Override    public void doGet(HttpServletRequest request,
HttpServletResponse response)
                 throws IOException, ServletException {


      // установить MIME-type и кодировку ответа
response.setContentType("text/html; charset=UTF8");
      PrintWriter out = response.getWriter();
```

```
      // Отправка веб-страницы
                  try {
         out.println("<html>");          out.println("<head><title>Servlet
sample</title></head>");          out.println("<body>");
out.println("<p>Запрошенный ресурс: " + request.getRequestURI() + "</p>");
out.println("<p>Протокол: " + request.getProtocol() + "</p>");
out.println("<p>Адрес сервера: " + request.getRemoteAddr() + "</p>");
out.println("</body></html>");
      } finally {          out.close();  //
Всегда закрывать Writer
      }
   }
}
```

Следующий этап – компиляция. Это можно сделать из той среды разработки, которую вы используете (Eclipse, NetBeans, IntelliJ и т.п.) или из командной строки. В случае использования openJDK и Tomcat это будет выглядеть примерно так:

```
javac -classpath /usr/share/tomcat/lib/tomcat-servlet-3.0-api.jar:classes
/srv/tomcat/webapps/myapp/WEB-INF/classes/MyServlet.java
```

В результате компиляции в каталоге WEB-INF/classes будет создан файл MyServlet.class. Теперь нужно сконфигурировать Tomcat для его использования.

Настройка доступа к сервлету

С точки зрения пользователя сервлет – это обычный веб-ресурс, адресуемый URI и обращение к сервлету выполняется через адресную строку браузера. Однако, чтобы сервлет стал доступным для клиентов, необходимо выполнить его настройку в файле WEB-INF/web.xml веб-приложения. Структура этого файла для рассматриваемого примера будет примерно такой:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Сохранить как "myapp/WEB-INF/web.xml" -->
<web-app version="3.0"
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_3_0.xsd">


   <servlet>
      <servlet-name>aboutServer</servlet-name>       <servlet-
class>MyServlet</servlet-class>
   </servlet>
```

```
    <servlet-mapping>
        <servlet-name>aboutServer</servlet-name>
        <url-pattern>/about</url-pattern>
    </servlet-mapping>
</web-app>
```

В такой конфигурации сервлет из файла *MyServlet.class* будет выполняться при обращении по адресу [server:port]/myapp/*about*. Для КАЖДОГО сервлета должна быть описана пара <servlet> и <servlet-mapping>, связанная по элементу <servlet-name>.

Для того, чтобы изменения вступили в силу, требуется перезапустить сервер. Результат выполнения сервлета приведен на рис. 4.

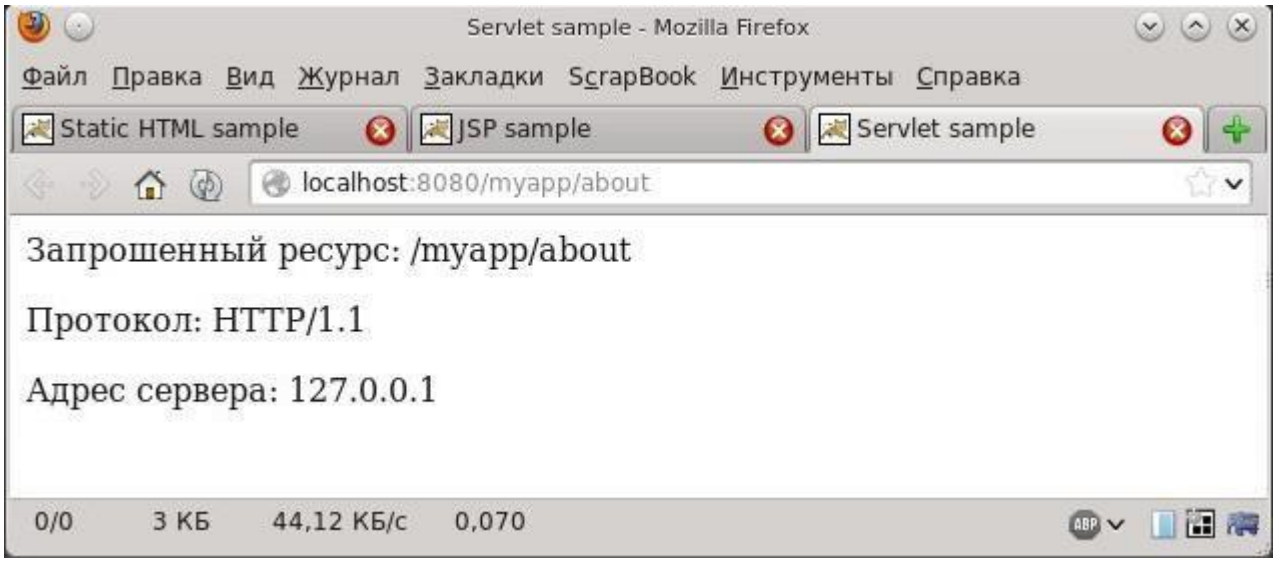


Рисунок 3 - Выполнение сервлета

Создание приложения

1. File → New → Dynamic Web Project
2. **Dynamic Web Project** Указываем имя проекта, остальные поля по умолчанию. Отметим, что в качесте Target runtime выбран созданный на предыдущем этапе сервер. Также можно изменить версию спецификации сервлетов в Dynamic web module version.

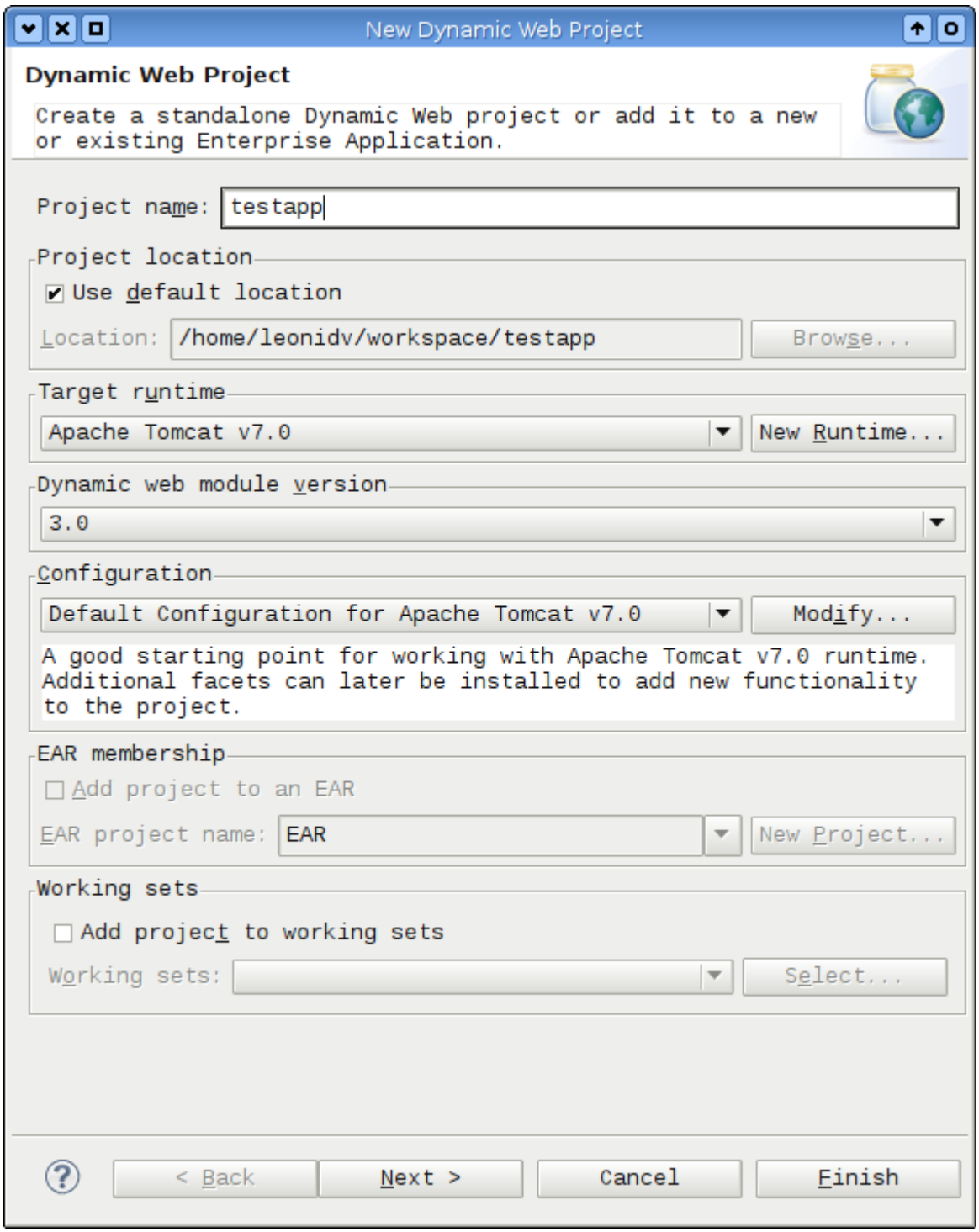
3. 

Рисунок 4 - Настройки проекта

4. **Java**. Стандартные настройки Java-приложения. Предлагается возможность указать пути к папкам с исходными кодами и папку с скомпилированными классами. Нажимаем next.
5. **Web Module** Здесь можно ничего не менять, но важно понимать значения полей:
    1. **Context root** - контекст, по которому будет доступно ваше приложение в браузере. Полный путь будет http://localhost:8080/test-app (где test-app - значение поля).
    2. **Content directory** - корневая папка для war-архива. В ней будут созданы WEB-INF/, META-INF. Все файлы в ней (за исключением служебных) будут доступны клиентам.

3. **Generate web.xml** - начиная со спецификации Servlets 3.0 web-приложения могут обходится без файла web.xml. Выбранная опция создает файл в любом случае.

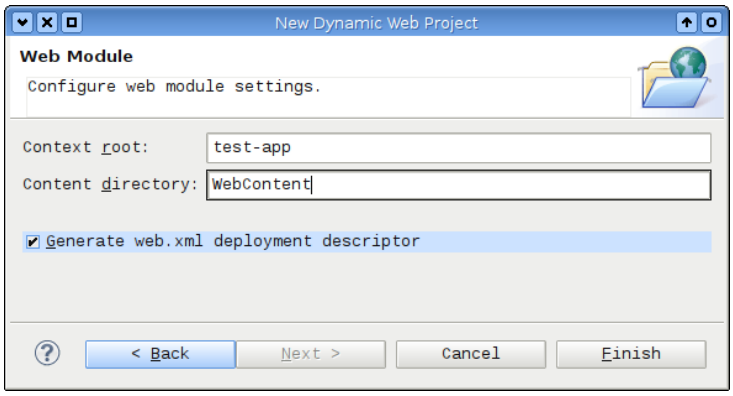


Рисунок 5 - Задание директории контента

Для запуска приложения щелкните по нему правой клавишей мышки и выбирите Run As → Run on Server. В появившемся окне отметьте опцию Always use this server и нажимите Finish. Приложение появится во вкладке Server в виде узла сервера и во встроенном браузере. Чтобы оно запускалось в другом браузере необходимо настроить Eclipse: Window → Preferences → General → Web Browser и определить браузеры.

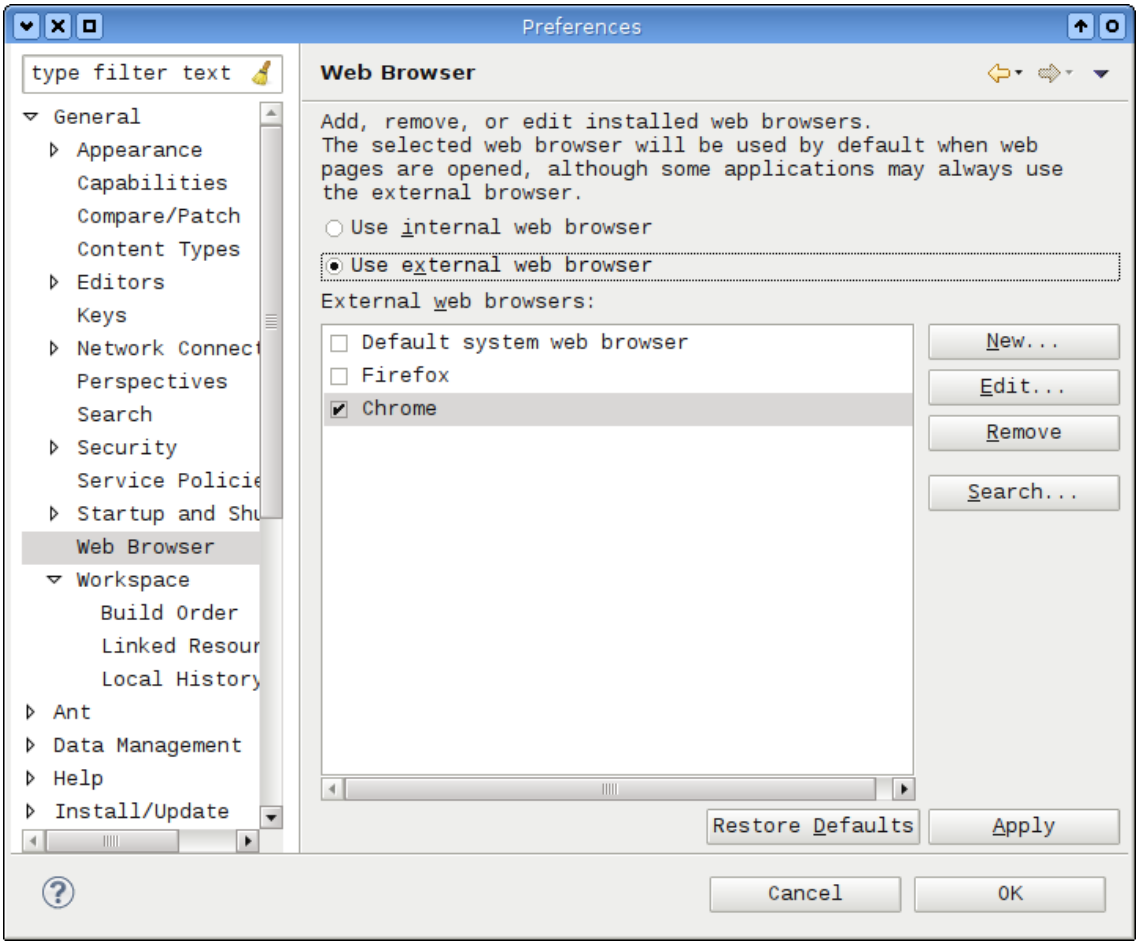


Рисунок 6 - Выбор браузера Обратите внимание, что необходимые для разработки библиотеки tomcat'a Eclipse автоматически подключит к проекту.

Разработка приложения

После запуска вы увидите ошибку 404, это нормально, так как приложение пустое и tomcat не знает, что отдать пользователю. Начнем заполнение приложения с создания простого html-файла:

1. Правой клавишей по папке проекта WebContent: New → Html File, назовите его index.html (названия взято из секции welcome в web.xml). В следующем окне выберите подходящий шаблон страницы, например, New HTML File (5) и нажмите Finish.
2. Добавим в файле какую-нибудь надпись, например, Hello world и запусти приложение снова (см. конец предыдущего этапа) или просто обновите ссылку в браузере. Вы должны увидеть созданную страницу.
3. Создайте новый файл page.html рядом с index.html и заполните его чем-нибудь. Щелкните правой клавишей мышки по СОЗДАННОЙ странице и выбирет Run As → Run on Server. В браузере откроется сервер с новой страницей.

Для создания сервлетов и JSP-страниц используйте соответствующие команды из меню файл. При этом при создании сервлета вам также предложат указать его отображение (mapping) и начальные параметры.

Тонкая настройка проекта

В свойствах проекта можно настроить еще несколько параметров.

 Deployment Assembly - настройки размещения путей проекта (или других каталогов) на сервере приложений. Допустим, у вас есть каталог data, размещенный в корне проекта. Вы хотите, чтобы данные из этого каталога попали в приложения по адресу /data/. Для этого на вкладке Deployment Assembly нужно сделать следующее: Add → Folder, выбрать папку data. После этого изменить в таблице Deploy Path на data/.

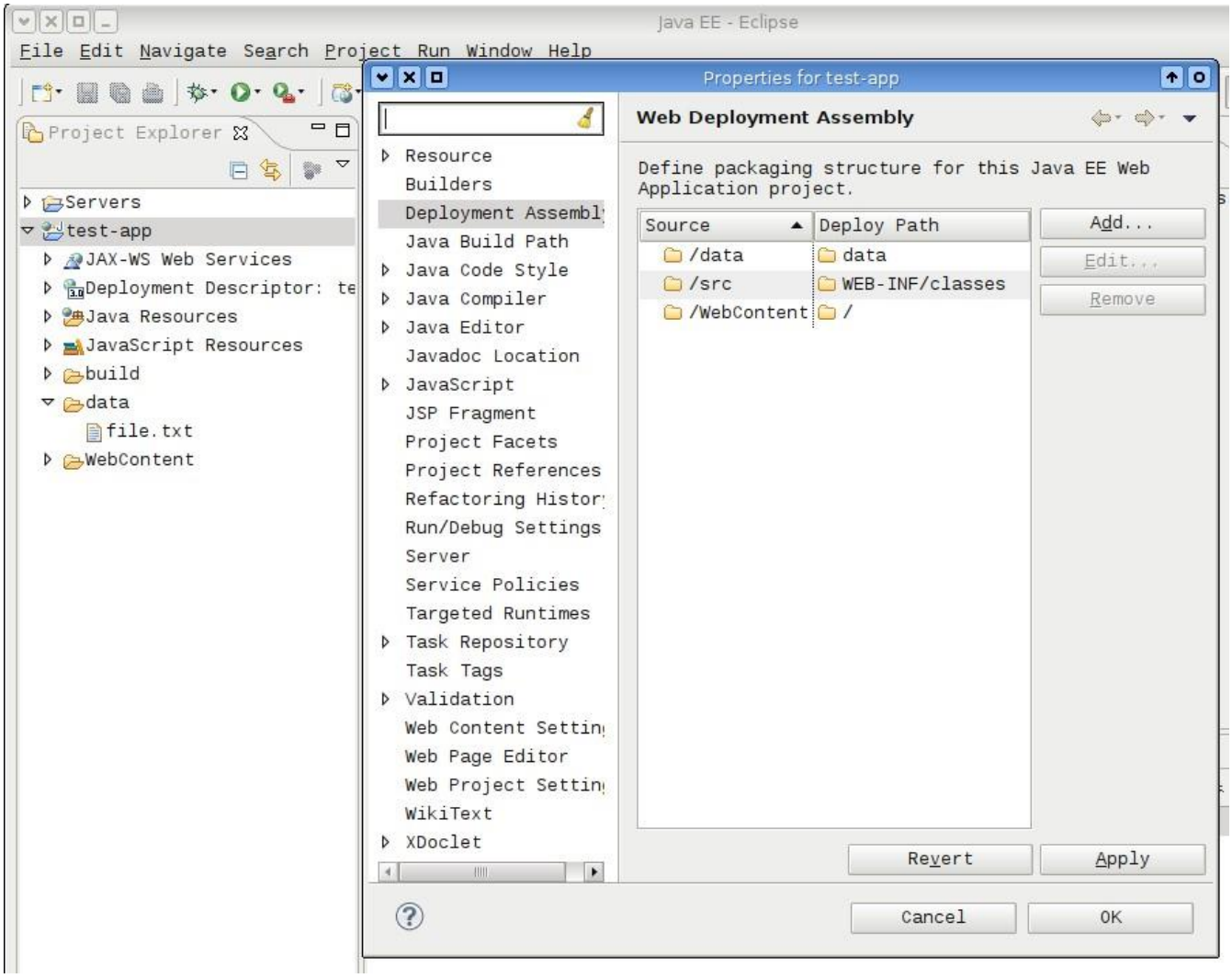


Рисунок 7 - Настройка деплоя

- **Project Facets** - позволяет подключить те или иные фреймворки к проекту.
- **Web Page Editor** - позволяет подключить tag libraries при редактировании JSP.
- **Web Project Settings** - позволяет изменить контекст приложения

Экспорт проекта

Экспорт проекта в war-файл, для размещения в контейнере сервлетов, осуществляется через File → Export → Web → WAR file. Также можно получить доступ быстрее, через контекстное меню проекта Export → WAR. Обратите внимание, что при экспорте учитываются настройке Web Deployment.

Задание

1. Установите и настройте web-сервер TomCat
2. Подготовить JSP (можно использовать различный фреймворки) с информацией, согласно предметной области. Добавить счетчик посещений и отображение текущей даты и времени, запрашиваемых с сервера. Счетчик должен сохранять свое состояние после перезагрузки сервера. Обязательное условие – использование языка Java для разработки бекэнда.
3. Разработать набор классов и JSP-страниц для аутентификации пользователя и хранении информации в сессии.
4. Предусмотреть работу на портале 3 группами пользователей: администратор, модератор, посетитель(пользователь).
5. Реализовать кабинет пользователя. Реализовать возможность загрузки изображения на портал
6. Подготовьте отчет о проделанной работе.

   Варианты тематик:

   1. сайт администрации города
   2. сайт знакомств
   3. сайт университета
   4. сайт прогноза погоды
   5. сайт новостей
   6. сайт библиотеки
   7. социальную сеть
   8. автомобильный портал
   9. интернет магазин бытовой техники
   10. сайт кинотеатра
   11. сайт аукциона
   12. игровой портал
   13. сайт больницы
   14. сайт бензоколонки
   15. портал объявлений

Приложение.

Установка TomCat

Разархивируйте файл apache-tomcat-8.5.6-windows-x86.zip в каталог на локальном диске.

Запустите консоль.

Перейдите в каталог \apache-tomcat\bin\

Выполните команду startup

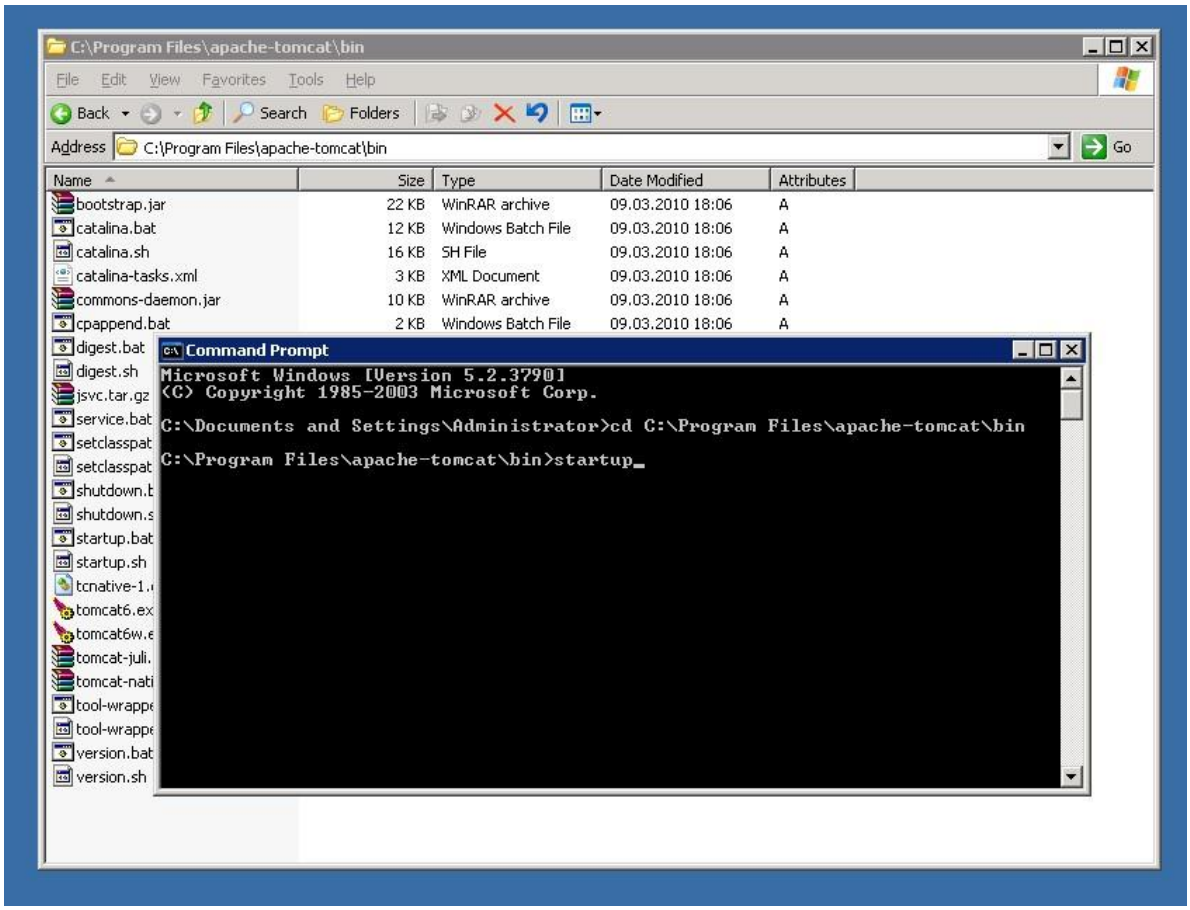


Рисунок 8 - Консоль перед запуском web-сервера

Результат:

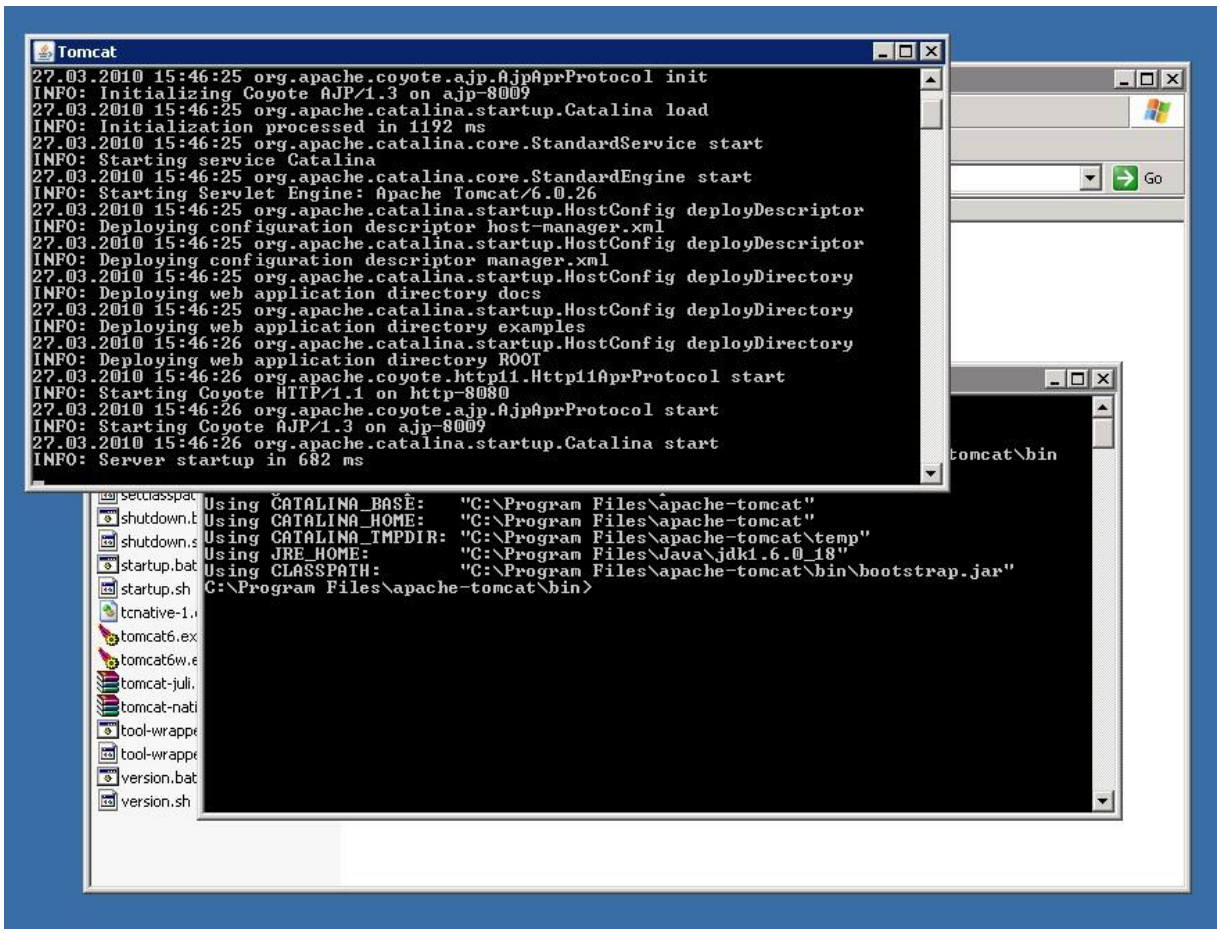


Рисунок 9 - Лог запуска web-сервера

В строке адреса браузера наберите http://localhost:8080/ (таким образом проверяем, работает ли наш WebServer)

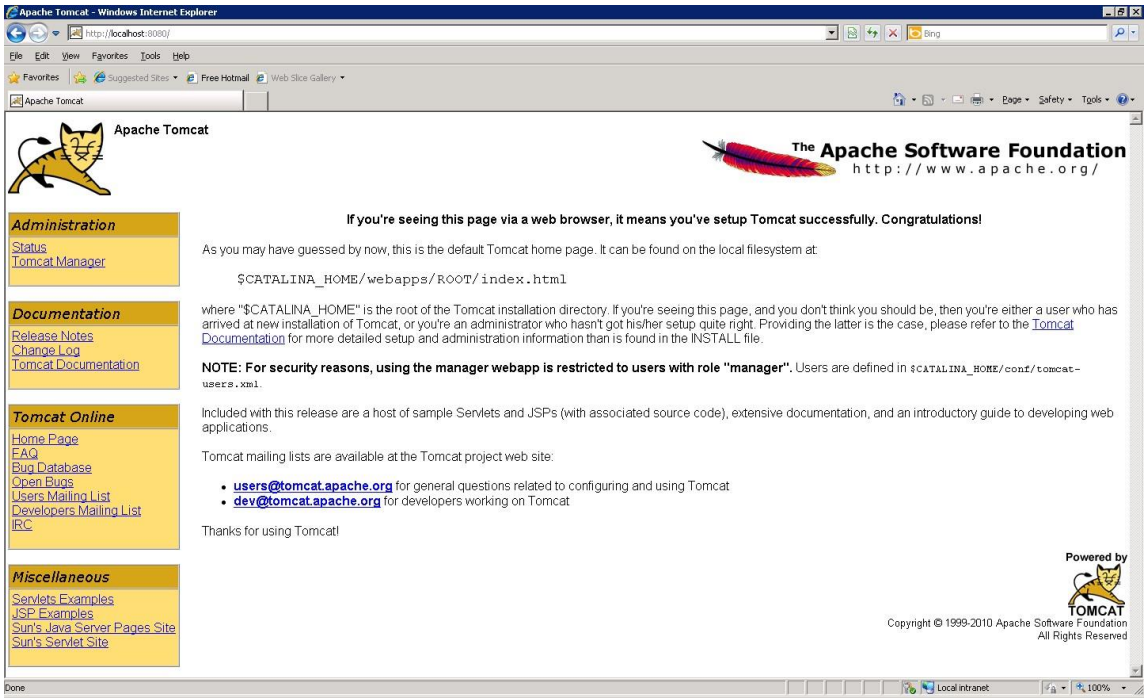


Рисунок 10 - Отображение стартовой страницы сервера