Introdução a Servlets

Preparando o Ambiente

- Eclipse
  - http://www.eclipse.org
- Tomcat
  - Servlet and JSP Container
  - http://tomcat.apache.org/
- Sysdeo Tomcat-Eclipse Plugin
  - http://www.sysdeo.com/eclipse/tomcatplugin
- Descompacte o plugin dentro do diretório eclipse/plugins
- Inicie o Eclipse com a opção –clean, para forçar a identificação de novos plugins
  - \eclipse -clean

Instalação do Plugin

- Ativição do Plugin no Eclipse 3.0(ou superior) :
  selecione menu 'Window->Customize Perspective...->Commands', e clique na caixa 'Tomcat' na Tab 'Available command groups'

- Configure o caminho do Tomcat: Window -> Preferences,

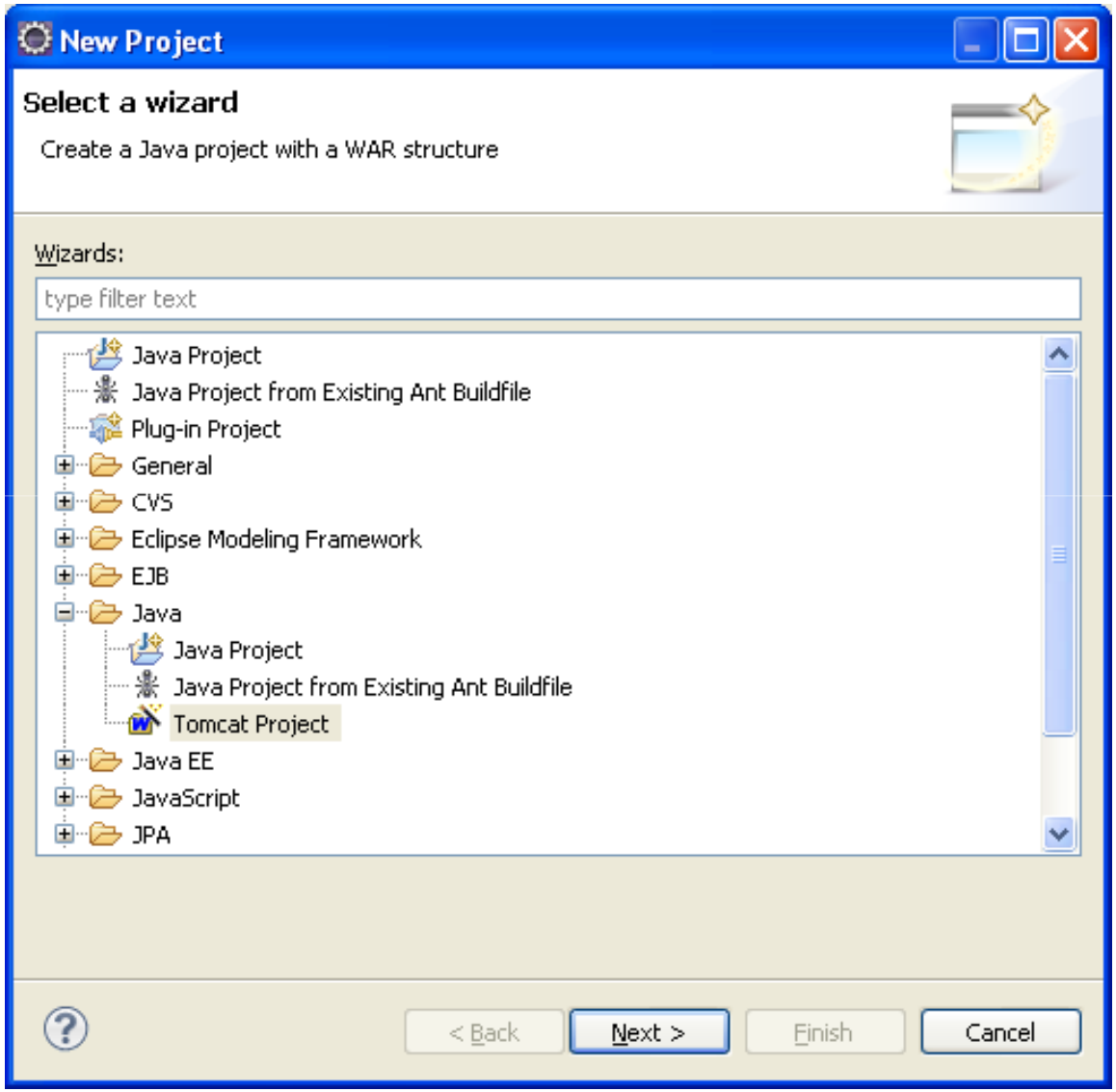
- O plugin inicia o Tomcat usando a JRE default definida na janela de Preferências do Eclipe.

- Para definir uma JDK como JRE default para o Eclipse, abra a janela de preferências: Window -> Preferences -> Java -> Installed JREs.
  A JRE deve ser uma JDK (Isto é prerequisito do Tomcat).

- O plugin configura o classpath e bootclasspath do Tomcat, caso precise de configuração específica. Você pode altera-la em:
  Preferences -> Tomcat ->JVM Settings

Criação de Projeto TomCat

- Acesse o Menu File | New | Project
- Opção:
  Java | Tomcat Project



7.1.3.1 Setting Up the Apache Tomcat Server (Cont.).

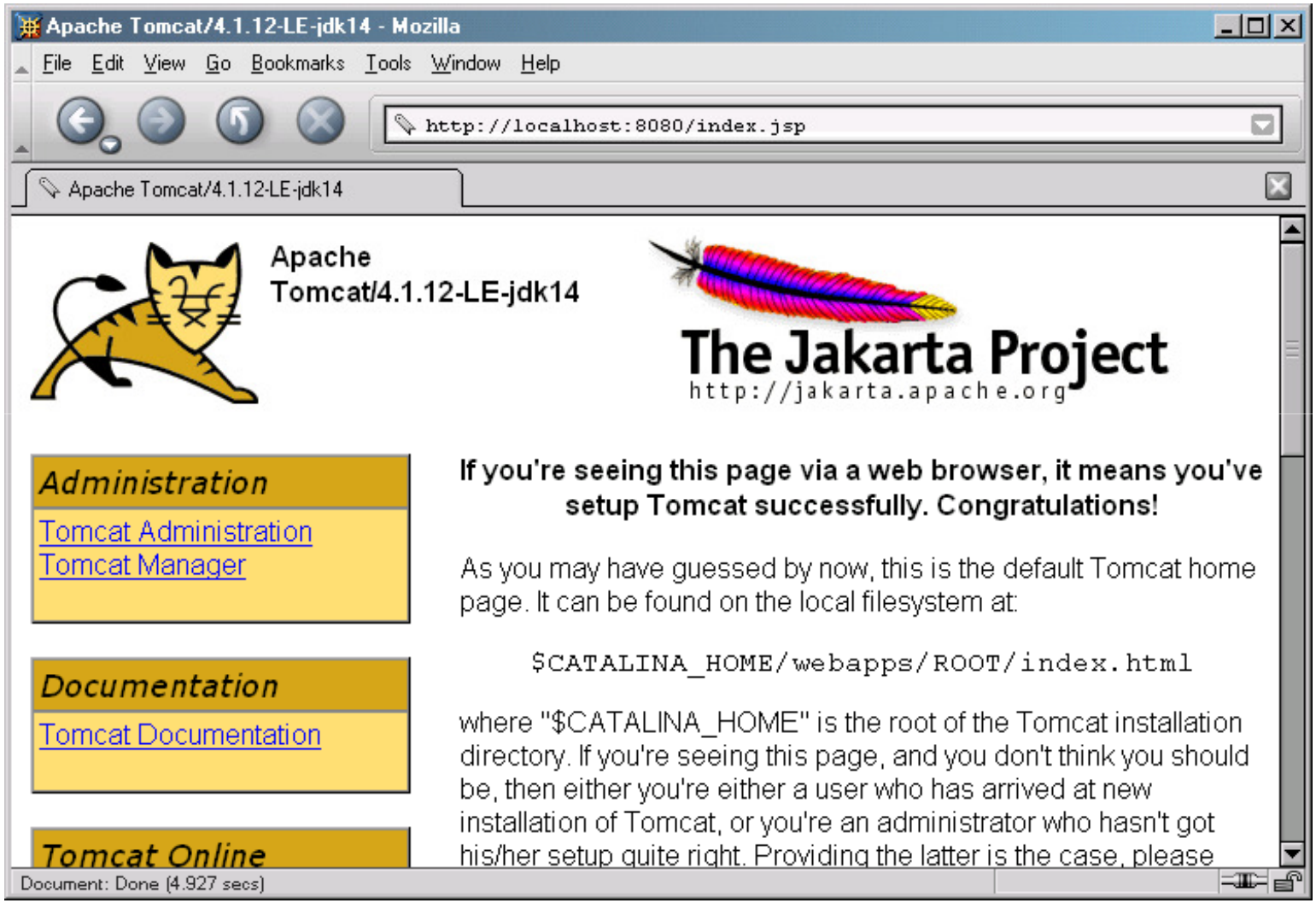


Fig. 7.1.7 Tomcat documentation home page. (Courtesy of The Apache Software Foundation.)

Usuários Tomcat

- Usuários Tomcat são definidos em
  - **`[TOMCAT_HOME]\conf\tomcat-users.xml`**
- Adicione a linha abaixo para criar um administrador com senha admin:

<user username="admin" password="admin" roles="standard,manager"/>

7.1.3.2 Deploying a Web Application

- Web applications
  - JSPs, servlets and their supporting files
- Deploying a Web application
  - Directory structure
    - Context root
  - Web application archive file (WAR file)
  - Deployment descriptor
    - **`web.xml`**

7.1.1 Introduction

- Java networking capabilities
  - Socket-based and packet-based communications
    - Package **`java.net`**
  - Remote Method Invocation (RMI)
    - Package **`java.rmi`**
  - Servlets and Java Server Pages (JSP)
    - Request-response model
    - Packages **`javax.servlet`**
      `javax.servlet.http`
      `javax.servlet.jsp`
      `javax.servlet.tagext`
    - Form the Web tier of J2EE

7.1.1 Introduction (Cont.)

- Servlets
  - Thin clients
  - Request/response mechanism
  - redirection
- Tomcat
  - Jakarta project
  - Official reference implementation of the JSP and servlet standards

7.1.2 Servlet Overview and Architecture

- Servlet container (servlet engine)
  - Server that executes a servlet
- Web servers and application servers
  - Sun ONE Application Server
  - Microsoft's Internet Information Server (IIS)
  - Apache HTTP Server
  - BEA's WebLogic Application Server
  - IBM's WebSphere Application Server
  - World Wide Web Consortium's Jigsaw Web Server

7.1.2.1 Interface `Servlet` and the Servlet Life Cycle

- Interface **`Servlet`**
  - All servlets must implement this interface
  - All methods of interface **`Servlet`** are invoked by servlet container
- Servlet life cycle
  - Servlet container invokes the servlet's **`init`** method
  - Servlet's **`service`** method handles requests
  - Servlet's **`destroy`** method releases servlet resources when the servlet container terminates the servlet
- Servlet implementation
  - **`GenericServlet`**
  - **`HttpServlet`**

7.1.2.1 Interface Servlet and the Servlet Life Cycle (Cont.)

Method	Description
`void init(ServletConfig config )`	The servlet container calls this method once during a servlet's execution cycle to initialize the servlet. The `ServletConfig` argument is supplied by the servlet container that executes the servlet.
`ServletConfig getServletConfig()`	This method returns a reference to an object that implements interface `ServletConfig`. This object provides access to the servlet's configuration information such as servlet initialization parameters and the servlet's `ServletContext`, which provides the servlet with access to its environment (i.e., the servlet container in which the servlet executes).
`String getServletInfo()`	This method is defined by a servlet programmer to return a string containing servlet information such as the servlet's author and version.
`void service( ServletRequest request,ServletResponse response )`	The servlet container calls this method to respond to a client request to the servlet.
`void destroy()`	This "cleanup" method is called when a servlet is terminated by its servlet container. Resources used by the servlet, such as an open file or an open database connection, should be deallocated here.

Fig. 24.1 Methods of interface `Servlet` (package `javax.servlet`).

7.1.2.2 HttpServlet Class

- Overrides method **`service`**
- Two most common HTTP request types
  - **`get`** requests
  - **`post`** requests
- Method **`doGet`** responds to **`get`** requests
- Method **`doPost`** responds to **`post`** requests
- **`HttpServletRequest`** and **`HttpServletResponse`** objects

7.1.2.2 `HttpServlet` Class (Cont.)

Method	Description
`doDelete`	Called in response to an HTTP *`delete`* request. Such a request is normally used to delete a file from a server. This may not be available on some servers, because of its inherent security risks (e.g., the client could delete a file that is critical to the execution of the server or an application).
`doHead`	Called in response to an HTTP *`head`* request. Such a request is normally used when the client only wants the headers of a response, such as the content type and content length of the response.
`doOptions`	Called in response to an HTTP *`options`* request. This returns information to the client indicating the HTTP options supported by the server, such as the version of HTTP (1.0 or 1.1) and the request methods the server supports.
`doPut`	Called in response to an HTTP *`put`* request. Such a request is normally used to store a file on the server. This may not be available on some servers, because of its inherent security risks (e.g., the client could place an executable application on the server, which, if executed, could damage the server—perhaps by deleting critical files or occupying resources).
`doTrace`	Called in response to an HTTP *`trace`* request. Such a request is normally used for debugging. The implementation of this method automatically returns an HTML document to the client containing the request header information (data sent by the browser as part of the request).

Fig. 7.1.2 Other methods of class `HttpServlet`.

7.1.2.3 `HttpServletRequest` Interface

- Web server
  - creates an **`HttpServletRequest`** object
  - passes it to the servlet's **`service`** method
- **`HttpServletRequest`** object contains the request from the client

7.1.2.3 `HttpServletRequest` Interface (Cont.)

Method	Description
`String getParameter( String name )`	Obtains the value of a parameter sent to the servlet as part of a `get` or `post` request. The `name` argument represents the parameter name.
`Enumeration getParameterNames()`	Returns the names of all the parameters sent to the servlet as part of a `post` request.
`String[] getParameterValues( String name )`	For a parameter with multiple values, this method returns an array of strings containing the values for a specified servlet parameter.
`Cookie[] getCookies()`	Returns an array of `Cookie` objects stored on the client by the server. `Cookie` objects can be used to uniquely identify clients to the servlet.
`HttpSession getSession( boolean create )`	Returns an `HttpSession` object associated with the client's current browsing session. This method can create an `HttpSession` object (`true` argument) if one does not already exist for the client. `HttpSession` objects are used in similar ways to `Cookie`s for uniquely identifying clients.

Fig. 24.3 Some methods of interface `HttpServletRequest`.

7.1.2.4 `HttpServletResponse` Interface

- Web server
  - creates an **`HttpServletResponse`** object
  - passes it to the servlet's **`service`** method

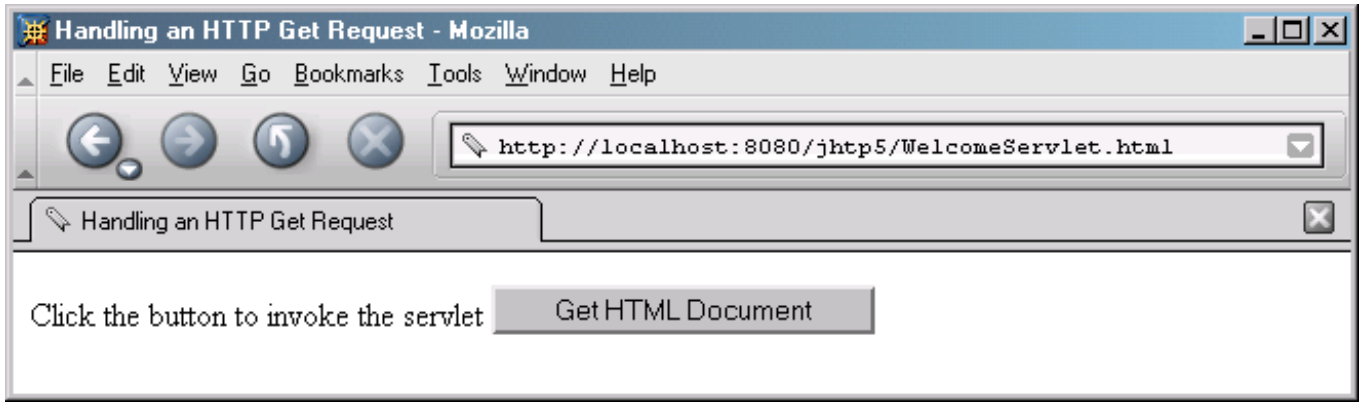
7.1.2.4 `HttpServletResponse` Interface (Cont.)

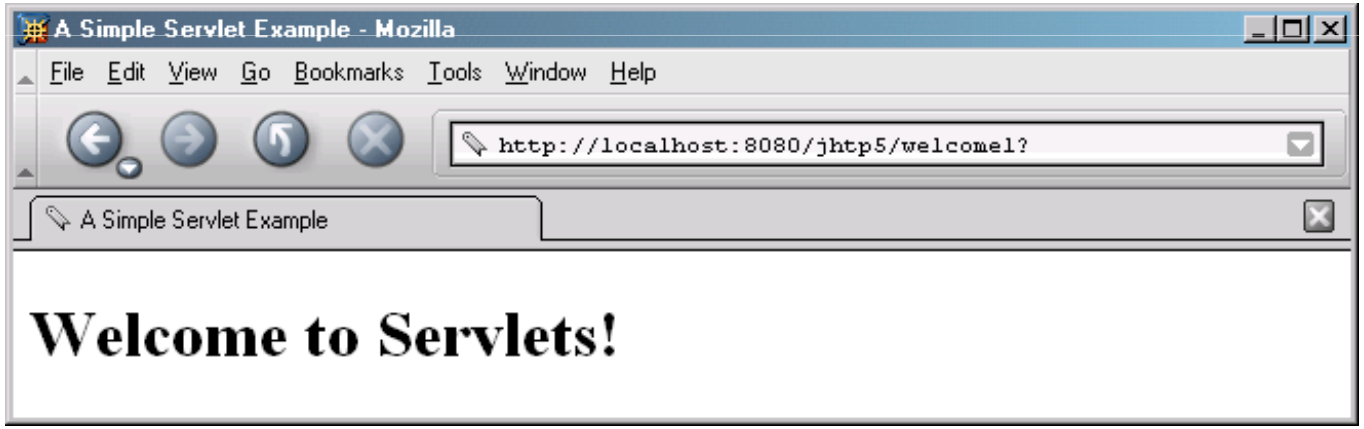
Method	Description
`void addCookie(Cookie cookie )`	Used to add a `Cookie` to the header of the response to the client. The `Cookie`'s maximum age and whether `Cookie`s are enabled on the client determine if `Cookie`s are stored on the client.
`ServletOutputStream getOutputStream()`	Obtains a byte-based output stream for sending binary data to the client.
`PrintWriter getWriter()`	Obtains a character-based output stream for sending text data to the client.
`void setContentType( String type )`	Specifies the MIME type of the response to the browser. The MIME type helps the browser determine how to display the data (or possibly what other application to execute to process the data). For example, MIME type `"text/html"` indicates that the response is an HTML document, so the browser displays the HTML page.

Fig. 24.4 Some methods of interface `HttpServletResponse`.

7.1.3 Handling HTTP `get` Requests

- **`get`** request
  - Retrieve the content of a URL
- Example: **`WelcomeServlet`**
  - a servlet handles HTTP **`get`** requests


Handling an HTTP Get Request - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet.html
Handling an HTTP Get Request
Click the button to invoke the servlet
Get HTML Document


A Simple Servlet Example - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome1?
A Simple Servlet Example
Welcome to Servlets!

```
1  // Fig. 7.1.5: WelcomeServlet.java
2  // A simple servlet to process get requests.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet extends HttpServlet {
9
10    // process "get" requests from clients
11    protected void doGet( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       response.setContentType( "text/html" );
16       PrintWriter out = response.getWriter();
17
18       // send XHTML page to client
19
20       // start XHTML document
21       out.println( "<?xml version = \"1.0\"?>" );
22
23       out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
24          "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
25          "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
26
```

Import the `javax.servlet` and `javax.servlet.http` packages.

Extends `HttpServlet` to handle HTTP `get` requests

Override method `doGet` to provide custom get request processing.

Uses the `response` object's `getWriter` method to obtain a reference to the `PrintWriter` object that enables the servlet to send content to the client

Create the XHTML document by writing strings with the `out` object's `println` method.

```
27          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
28
29          // head section of document
30          out.println( "<head>" );
31          out.println( "<title>A Simple Servlet Example</title>" );
32          out.println( "</head>" );
33
34          // body section of document
35          out.println( "<body>" );
36          out.println( "<h1>Welcome to Servlets!</h1>" );
37          out.println( "</body>" );
38
39          // end XHTML document
40          out.println( "</html>" );
41          out.close();  // close stream to complete the page
42       }
43    }
```

Closes the output stream, flushes the output buffer and sends the information to the client.

```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.1.6: WelcomeServlet.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Handling an HTTP Get Request</title>
10 </head>
11
12 <body>
13    <form action = "/jhtp5/welcome1" method = "get">
14
15       <p><label>Click the button to invoke the servlet
16          <input type = "submit" value = "Get HTML Document" />
17       </label></p>
18
19    </form>
20 </body>
21 </html>
```

7.1.3.1 Setting Up the Apache Tomcat Server

- Download Tomcat
- Define environment variables
  - JAVA_HOME
  - CATALINA_HOME
- Start the Tomcat server
  - startup.bat
- Connect to the Tomcat server using a Web browser
  - http://localhost:8080/

7.1.3.2 Deploying a Web Application (Cont.)

Directory	Description
context root	This is the root directory for the Web application. All the JSPs, HTML documents, servlets and supporting files such as images and class files reside in this directory or its subdirectories. The name of this directory is specified by the Web application creator. To provide structure in a Web application, subdirectories can be placed in the context root. For example, if your application uses many images, you might place an images subdirectory in this directory. The examples of this chapter use `jhtp5` as the context root.
`WEB-INF`	This directory contains the Web application *deployment descriptor* (`web.xml`).
`WEB-INF/classes`	This directory contains the servlet class files and other supporting class files used in a Web application. If the classes are part of a package, the complete package directory structure would begin here.
`WEB-INF/lib`	This directory contains Java archive (JAR) files. The JAR files can contain servlet class files and other supporting class files used in a Web application.

Fig. 24.8 Web application standard directories.

```
1  <!DOCTYPE web-app PUBLIC \
2     "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
3     "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">
4
5  <web-app>
6
7     <!-- General descr
8     <display-name>
9        Java How to Program JSP
10       and Servlet Chapter Examples
11    </display-name>
12
13    <description>
14       This is the web appli
15       demonstrate our JSP a
16    </description>
17
18    <!-- Servlet definitions -->
19    <servlet>
20       <servlet-name>welcome1</servlet-name>
21
22       <description>
23          A simple servlet th                st.
24       </description>
25
```

Element `web-app` defines the configuration of each servlet in the Web application and the servlet mapping for each servlet.

Element `display-name` specifies a name that can be displayed to the administrator of the server on which the Web application is installed.

Element `description` specifies a description of the Web application that might be displayed to the administrator of the server.

Eleme

Element `servlet-name` is the name for the servlet.

t.

Element `description` specifies a description for this particular servlet.

```
26        <servlet-class>
27           WelcomeServlet
28        </servlet-class>
29     </servlet>
30
31     <!-- Servlet mappings -->
32     <servlet-mapping>
33        <servlet-name>welcome1</servlet-name>
34        <url-pattern>/welcome1</url-pattern>
35     </servlet-mapping>
36
37  </web-app>
```

Element `servlet-class` specifies compiled servlet's fully qualified class name.

Element `servlet-mapping` specifies `servlet-name` and `url-pattern` elements.

7.1.3.2 Deploying a Web Application (Cont.)

- Invoke **`WelcomeServlet`** example
  - /aula7/welcome1
    - **`/aula7`** specifies the context root
    - **`/welcome1`** specifies the URL pattern
- URL pattern formats
  - Exact match
    - /aula7/welcome1
  - Path mappings
    - /aula7/example/*
  - Extension mappings
    - *.jsp
  - Default servlet
    - /

7.1.3.2 Deploying a Web Application (Cont.)

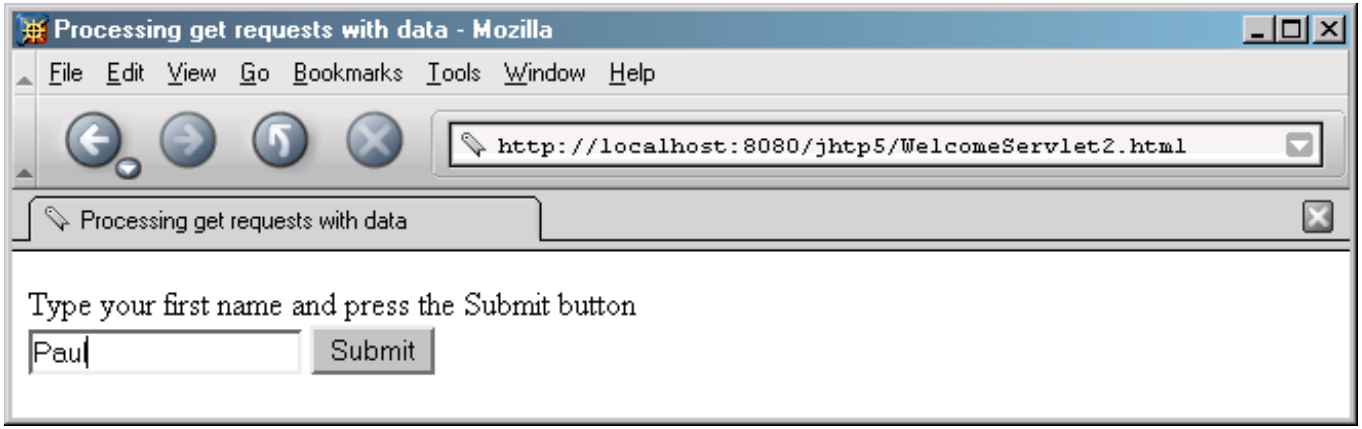
`WelcomeServlet` Web application directory and file structure

```
jhtp5
   servlets
      WelcomeServlet.html
   WEB-INF
      web.xml
      classes
         WelcomeServlet.class
```

Fig. 24.10 Web application directory and file structure for `WelcomeServlet`.

7.1.4 Handling HTTP `get` Requests Containing Data

- Servlet **`WelcomeServlet2`**
  - Responds to a **`get`** request that contains data


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet2.html
Processing get requests with data
Type your first name and press the Submit button
Paul
Submit


Processing get requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome2?firstname=Paul
Processing get requests with data
Hello Paul,
Welcome to Servlets!

```
1  // Fig. 7.1.11: WelcomeServlet2.java
2  // Processing HTTP get requests containing data.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet2 extends HttpServlet {
9
10    // process "get" request from client
11    protected void doGet( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       String firstName = request.getParameter( "firstname" );
16
17       response.setContentType( "text/html" );
18       PrintWriter out = response.getWriter();
19
20       // send XHTML document to client
21
22       // start XHTML document
23       out.println( "<?xml version = \"1.0\"?>" );
24
```

The `request` object's `getParameter` method receives the parameter name and returns the corresponding `String` value.

```
25          out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
26             "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
27             "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
28
29          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
31          // head section of document
32          out.println( "<head>" );
33          out.println(
34             "<title>Processing get requests with data</title>" );
35          out.println( "</head>" );
36
37          // body section of document
38          out.println( "<body>" );
39          out.println( "<h1>Hello " + firstName + ",<br />" );
40          out.println( "Welcome to Servlets!</h1>" );
41          out.println( "</body>" );
42
43          // end XHTML document
44          out.println( "</html>" );
45          out.close();  // close stream to complete the page
46       }
47    }
```

Uses the result of line 16 as part of the response to the client.

```
1   <?xml version = "1.0"?>
2   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5   <!-- Fig. 7.1.12: WelcomeServlet2.html -->
6
7   <html xmlns = "http://www.w3.org/1999/xhtml">
8   <head>
9      <title>Processing get requests with data</title>
10  </head>
11
12  <body>
13     <form action = "/aula7/welcome2" method = "get">
14
15        <p><label>
16           Type your first name and press the Submit button
17           <br /><input type = "text" name = "firstname" />
18           <input type = "submit" value = "Submit" />
19        </p></label>
20
21     </form>
22  </body>
23  </html>
```

Get the first name from the user.

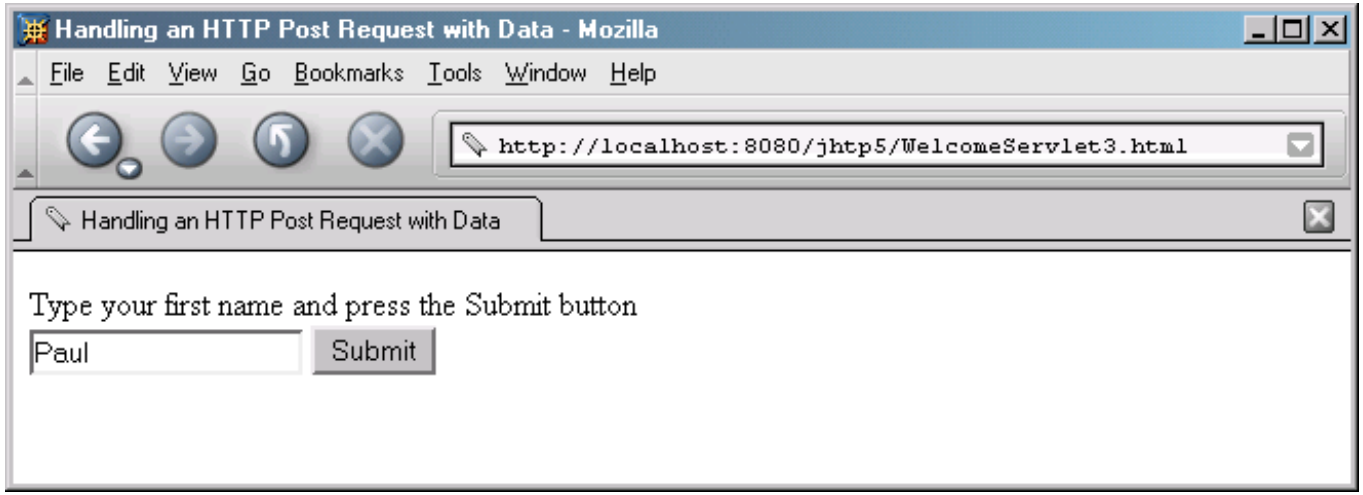
7.1.4 Handling HTTP `get` Requests Containing Data (Cont.)

Descriptor element	Value
`servlet` element	
`servlet-name`	`welcome2`
`description`	`Handling HTTP get requests with data.`
`servlet-class`	`WelcomeServlet2`
`servlet-mapping` element	
`servlet-name`	`welcome2`
`url-pattern`	`/welcome2`

Fig. 24.13 Deployment descriptor information for servlet `WelcomeServlet2`.

7.1.5 Handling HTTP post Requests

- HTTP post request
  - Post data from an HTML form to a server-side form handler
  - Browsers cache Web pages
- Servlet WelcomeServlet3
  - Responds to a **`post`** request that contains data


Handling an HTTP Post Request with Data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/WelcomeServlet3.html
Handling an HTTP Post Request with Data
Type your first name and press the Submit button
Paul
Submit


Processing post requests with data - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://localhost:8080/jhtp5/welcome3
Processing post requests with data
Hello Paul,
Welcome to Servlets!

```
1  // Fig. 7.1.14: WelcomeServlet3.java
2  // Processing post requests containing data.
3
4  import javax.servlet.*;
5  import javax.servlet.http.*;
6  import java.io.*;
7
8  public class WelcomeServlet3 extends HttpServlet {
9
10    // process "post" request from client
11    protected void doPost( HttpServletRequest request,
12       HttpServletResponse response )
13          throws ServletException, IOException
14    {
15       String firstName = request.getParameter( "firstname" );
16
17       response.setContentType( "text/html" );
18       PrintWriter out = response.getWriter();
19
20       // send XHTML page to client
21
22       // start XHTML document
23       out.println( "<?xml version = \"1.0\"?>" );
24
```

Declare a `doPost` method to responds to post requests.

```
25          out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD " +
26             "XHTML 1.0 Strict//EN\" \"http://www.w3.org" +
27             "/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
28
29          out.println( "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
30
31          // head section of document
32          out.println( "<head>" );
33          out.println(
34             "<title>Processing post requests with data</title>" );
35          out.println( "</head>" );
36
37          // body section of document
38          out.println( "<body>" );
39          out.println( "<h1>Hello " + firstName + ",<br />" );
40          out.println( "Welcome to Servlets!</h1>" );
41          out.println( "</body>" );
42
43          // end XHTML document
44          out.println( "</html>" );
45          out.close();  // close stream to complete the page
46       }
47    }
```

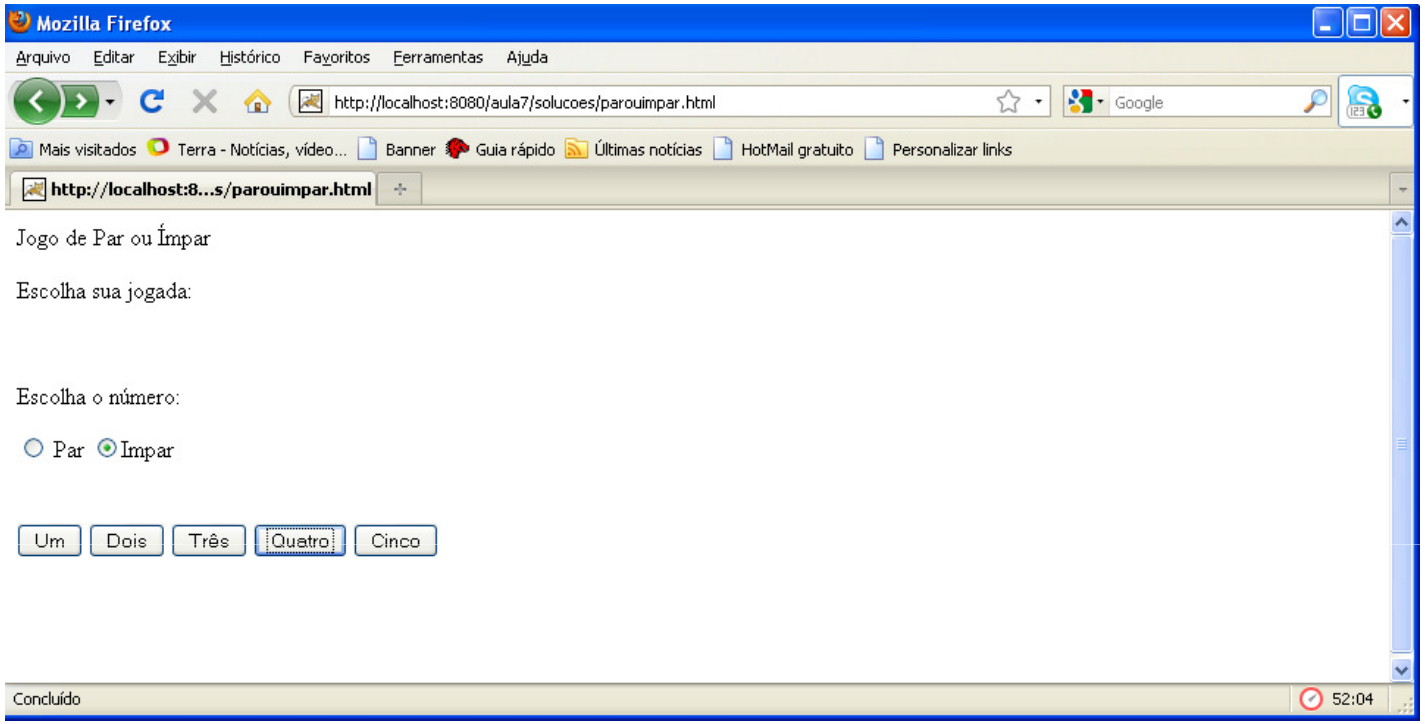
```
1  <?xml version = "1.0"?>
2  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
3     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
4
5  <!-- Fig. 7.1.15: WelcomeServlet3.html -->
6
7  <html xmlns = "http://www.w3.org/1999/xhtml">
8  <head>
9     <title>Handling an HTTP Post Request with Data</title>
10 </head>
11
12 <body>
13    <form action = "/aula7/welcome3" method = "post">
14
15       <p><label>
16          Type your first name and press the Submit button
17          <br /><input type = "text" name = "firstname" />
18          <input type = "submit" value = "Submit" />
19       </label></p>
20
21    </form>
22 </body>
23 </html>
```

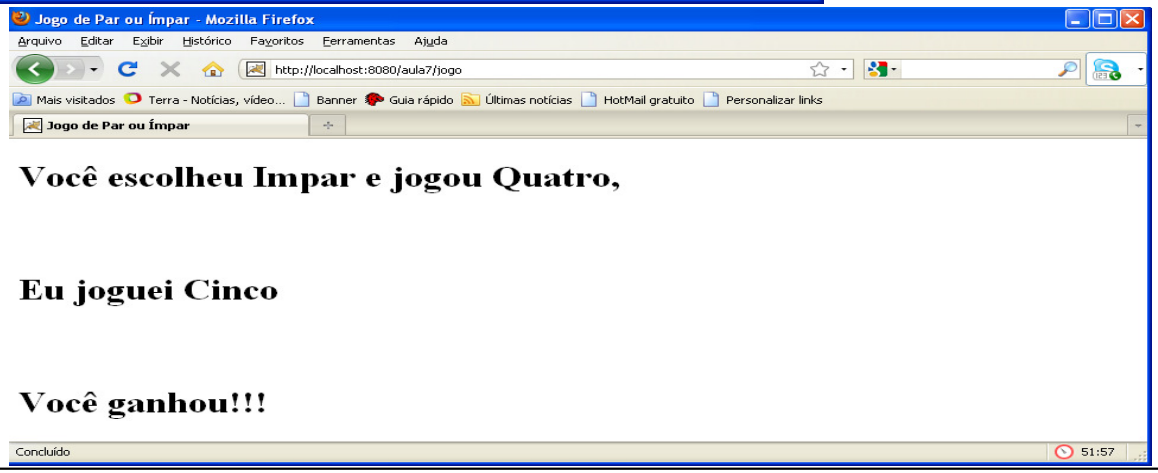
Provide a `form` in which the user can input a name in the `text` input element `firstname`, then click the `Submit` button to invoke `WelcomeServlet3`.

Exercícios

- Como garantir a equivalência entre Requisições GET e POST em um Servlet?
- Crie um novo servlet que implemente um jogo de par ou ímpar com o usuário. Passos
  - Crie uma página html usando radio buttons e botões para selecionar números de 1 a 5
    - <input type="radio" name="parImpar" value="Par"> Par
    - <input type="radio" name="parImpar" value="Impar" checked>Impar
  - Crie uma classe Servlet que escolha um número aleatório de 1 a 5
    - Random random=**new Random();**
    - **int minhaJogada=random.nextInt(5);**
  - E implemente a lógica do jogo
  - Crie o mapeamento do servlet em web.xml
  - Faça o deploy e reinicie o Tomcat

Exercício


Mozilla Firefox
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
http://localhost:8080/aula7/solucoes/parouimpar.html
Google
Mais visitados Terra - Notícias, vídeo... Banner Guia rápido Últimas notícias HotMail gratuito Personalizar links
http://localhost:8...s/parouimpar.html
Jogo de Par ou Ímpar
Escolha sua jogada:
Escolha o número:
Par Impar
Um Dois Três Quatro Cinco
Concluído
52:04


Jogo de Par ou Ímpar - Mozilla Firefox
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
http://localhost:8080/aula7/jogo
Mais visitados Terra - Notícias, vídeo... Banner Guia rápido Últimas notícias HotMail gratuito Personalizar links
Jogo de Par ou Ímpar
Você escolheu Impar e jogou Quatro,
Eu joguei Cinco
Você ganhou!!!
Concluído
51:57