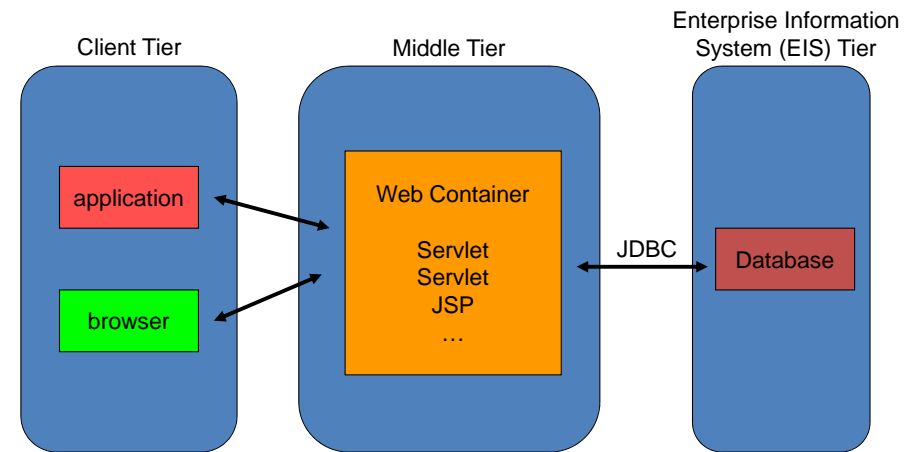


Introduzione alle Servlet

Web Application model



2

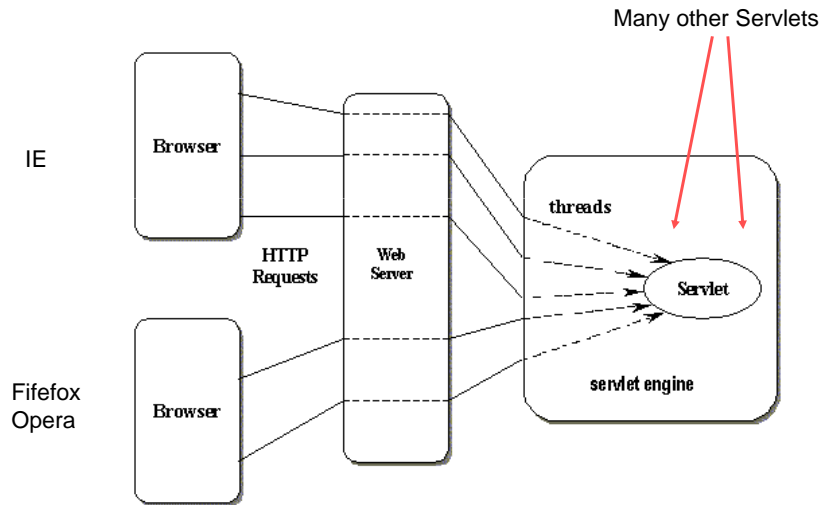
Servlet

- Programma applicativo (in java) che viene eseguito dal server web
 - Accoglie ed elabora richieste (codificate secondo il protocollo in uso, es: http o ftp. Solo http nel caso di server web, al quale arrivano richieste di tipo POST o GET)
 - Non richiede supporto java da parte del client
 - Meccanismo request/response

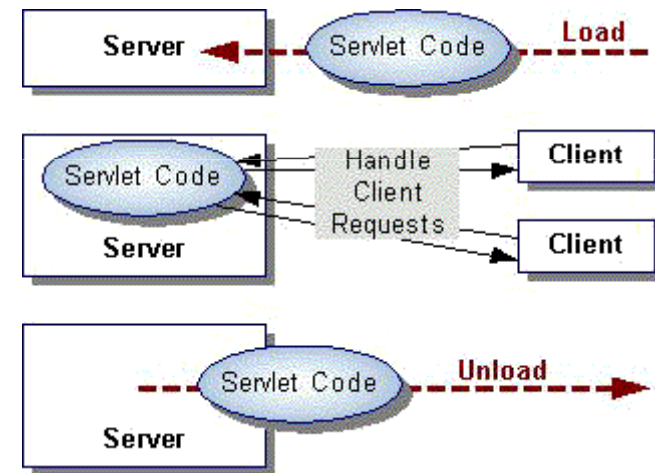
Servlet e Servlet Container

- Una servlet è una classe Java (che implementa l'interfaccia **Servlet**).
- Un servlet container può ospitare più servlet (con relativi alias).
- Quando una servlet viene invocata per la prima volta, il servlet engine genera un **thread** Java che inizializza l'oggetto Servlet.
 - Questo *persiste* per tutta la durata del processo relativo al servlet container (salvo esplicita de-allocazione).
- Ogni servlet è un thread all'interno del Servlet Container (vs CGI dove viene eseguito un processo esterno)

Servlet Engine (container)



The Servlet Life Cycle

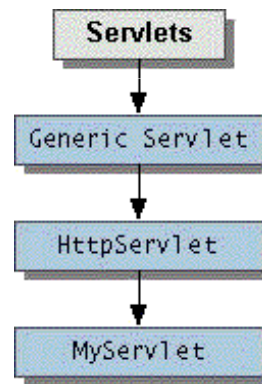


6

Basic Servlet Structure

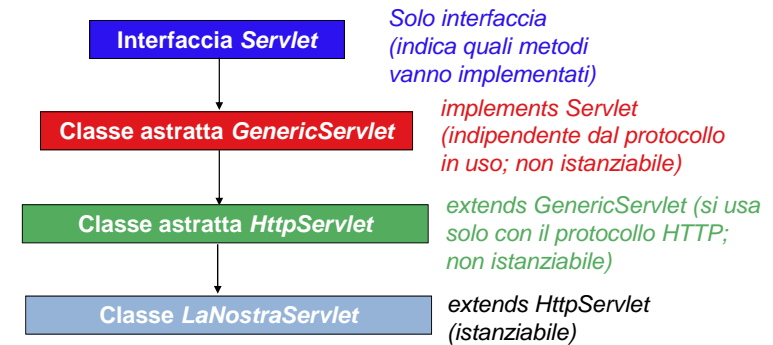
- The skeleton of a common servlet.

```
public class MyServlet extends HttpServlet
{
    public void init()
    {
        // Initialization here
    }
    public void service()
    {
        // Your work happens here
    }
    public void destroy()
    {
        // release resources here
    }
}
```



7

Servlet API



Servlet

- Le servlet possono essere utilizzate qualunque sia il servizio espletato dal server, ovvero qualunque sia il protocollo di interazione client/server: es HTTP, FTP
- Nella sua forma più generale, una servlet è un'estensione di una **classe `javax.servlet.GenericServlet`** che implementa l'**interfaccia `javax.servlet.Servlet`**
- Le servlet usate nel web sono estensioni della **classe `javax.servlet.http.HttpServlet`** che implementa l'**interfaccia `javax.servlet.Servlet`**

Interfaccia **Servlet**

- Interfaccia `Servlet` (package `javax.servlet`)
 - Tutte le servlet devono implementare questa interfaccia
 - Tutti i metodi dell'interfaccia `Servlet` vengono invocati dal servlet container secondo un prefissato **ciclo di vita**

Ciclo di vita di una servlet

- Caricamento della servlet in memoria
- Il container delle servlet invoca il metodo **`init`**
 - Solo in questo momento la servlet è in grado di rispondere alla prima richiesta
 - Inizializzazione delle variabili globali
 - Invocato una sola volta
- Il metodo **`service`** gestisce le richieste
 - Riceve la richiesta
 - Elabora la richiesta
 - Confeziona un oggetto risposta
 - Invocato ad ogni richiesta del client
- Il metodo **`destroy`** rilascia le risorse allocate dalla servlet quando il container la termina

Classi astratte per definire le Servlet

- Esistono due classi astratte che implementano l'interfaccia `Servlet`
 - **`GenericServlet`** (package `javax.servlet`)
 - **`HttpServlet`** (package `javax.servlet.http`)
(quest'ultima implementa l'interfaccia `Servlet` indirettamente, estendendo `GenericServlet`)
- Queste classi forniscono l'implementazione di default di tutti i metodi dell'interfaccia `Servlet`
- Il metodo chiave è **`service`**:
riceve gli oggetti **`ServletRequest`** e **`ServletResponse`** che forniscono accesso agli stream di i/o permettendo la ricezione e l'invio di informazioni al client

Metodi dell'interfaccia Servlet

Method	Description
<code>void init(ServletConfig config)</code>	The servlet container calls this method once during a servlet's execution cycle to initialize the servlet. The <code>ServletConfig</code> argument is supplied by the servlet container that executes the servlet.
<code>ServletConfig getServletConfig()</code>	This method returns a reference to an object that implements interface <code>ServletConfig</code> . This object provides access to the servlet's configuration information such as servlet initialization parameters and the servlet's <code>ServletContext</code> , which provides the servlet with access to its environment (i.e., the servlet container in which the servlet executes).
<code>String getServletInfo()</code>	This method is defined by a servlet programmer to return a string containing servlet information such as the servlet's author and version.
<code>void service(ServletRequest request, ServletResponse response)</code>	The servlet container calls this method to respond to a client request to the servlet.
<code>void destroy()</code>	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.

La classe HttpServlet

- Sovrascrive il metodo `service` della classe `GenericServlet`
- Prevede i metodi per rispondere alle richieste HTTP
 - GET
 - POST (ma anche HEAD, PUT, OPTIONS ecc.)
- Il metodo `service()` invoca i metodi `doGet` e `doPost`
 - Il metodo `doGet()` risponde alle richieste GET
 - Il metodo `doPost()` risponde alle richieste POST
 - Ricevono come parametri gli oggetti `HttpServletRequest` e `HttpServletResponse`
- Non implementare il metodo `service` se si usa questo tipo di servlet

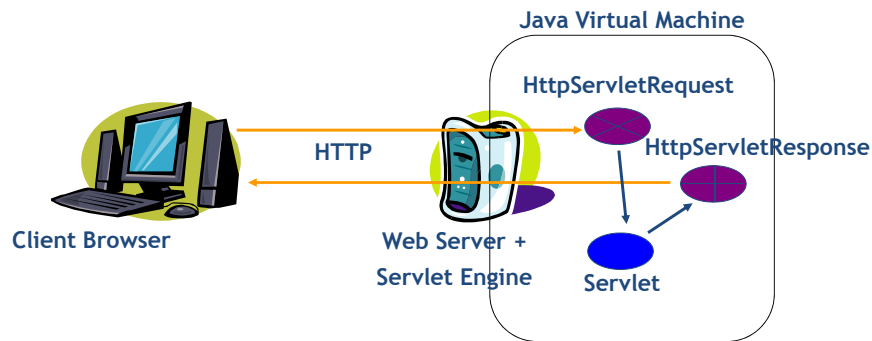
Altri metodi della classe HttpServlet

Method	Description
<code>doDelete</code>	Called in response to an HTTP <i>delete</i> 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).
<code>doHead</code>	Called in response to an HTTP <i>head</i> 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.
<code>doOptions</code>	Called in response to an HTTP <i>options</i> 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.
<code>doPut</code>	Called in response to an HTTP <i>put</i> 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).
<code>doTrace</code>	Called in response to an HTTP <i>trace</i> 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).

Funzionamento delle servlet HTTP

- Il server web riceve dal browser del client una richiesta HTTP GET o POST
- Il server web direziona la richiesta HTTP al motore servlet o `servlet engine (pr. è engine)` o `servlet container`
- Se la servlet non è ancora in memoria viene caricata e inizializzata dal servlet engine (esecuzione del metodo `init`)
- Il servlet engine incapsula la richiesta HTTP in una classe `HttpServletRequest` e la passa al metodo `doPost` o `doGet` della servlet
- La servlet risponde scrivendo il codice HTML nella `HttpServletResponse` che viene rimandata al web server e poi riconsegnata al client via HTTP

Servlet e Web Server



Interfaccia `HttpServletRequest`

- Il servlet engine che esegue la servlet
 - Crea un oggetto `HttpServletRequest`
 - Lo passa al metodo `service` della servlet `http`
- L'oggetto `HttpServletRequest` contiene la richiesta del client
- Esistono molti metodi per estrarre informazioni dalla richiesta del client. Ne vediamo alcuni →

Interfaccia `HttpServletRequest` (Cont.)

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

Interfaccia `HttpServletResponse`

- Il servlet engine
 - Crea un oggetto `HttpServletResponse`
 - Lo passa al metodo `service` della servlet (attraverso i metodi `doGet` e `doPost`)
- Ogni chiamata ai metodi `doGet` e `doPost` riceve un oggetto che implementa l'interfaccia `HttpServletResponse`
- Esistono molti metodi che consentono la scrittura della risposta da inviare al client. Ne vediamo alcuni →

Interfaccia HttpServletResponse

Method	Description
<code>void addCookie(Cookie cookie)</code>	
	Used to add a <code>Cookie</code> to the header of the response to the client. The <code>Cookie</code> 's maximum age and whether Cookies are enabled on the client determine if <code>Cookies</code> are stored on the client.
<code>ServletOutputStream getOutputStream()</code>	
	Obtains a byte-based output stream for sending binary data to the client.
<code>PrintWriter getWriter()</code>	
	Obtains a character-based output stream for sending text data to the client.
<code>void setContentType(String type)</code>	
	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 " <code>text/html</code> " indicates that the response is an HTML document, so the browser displays the HTML page.

Jakarta project

- Progetto Jakarta (Apache Software Foundation)
- Implementazione di riferimento degli standard per la realizzazione di servlet e di Java Server Pages
- Obiettivo: *"[...] to provide commercial-quality server solutions based on the Java Platform that are developed in an open and cooperative fashion"*

Architetture per l'esecuzione di servlet

- Servlet container (o servlet engine)
 - E' il server che esegue una servlet
- Servlet e JSP sono supportate direttamente o attraverso plugin dalla maggior parte dei Web servers e application server
 - Apache TomCat
 - Sun ONE Application Server
 - Microsoft's Internet Information Server (IIS)
 - Apache HTTP Server + modulo JServ
 - BEA's WebLogic Application Server
 - IBM's WebSphere Application Server
 - World Wide Web Consortium's Jigsaw Web Server

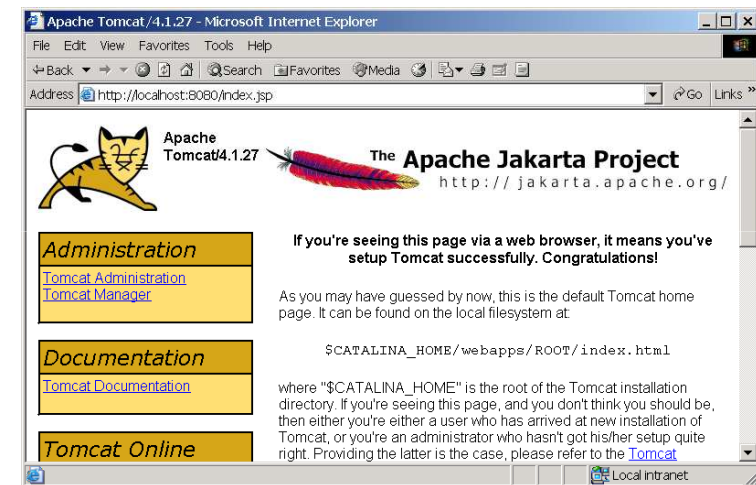
Configurare il server Apache Tomcat (1)

1. Installare j2sdk
2. Definire la variabile `PATH=c:\j2sdk---\bin` (per poter trovare il compilatore)
3. Definire la variabile `JAVAHOME=c:\j2sdk---` (perché tomcat trovi la jvm)
4. Installare il server TomCat

Configurare il server Apache Tomcat (2)

5. Definire la variabile
CATALINA_HOME=c:\Programmi\Apache...\Tomcat---\
6. Definire la variabile
CLASSPATH=c:\Programmi\Apache...\Tomcat---\common\lib\servlet-api.jar;c:\Programmi\Apache...\Tomcat---\common\lib\jsp-api.jar
7. TEST: Avviare il server Tomcat (startup),
invocare il server da un browser all'indirizzo
http://localhost:8080/

Configurare il server Apache Tomcat



Se Tomcat è installato correttamente dovreste vedere questa pagina all'indirizzo <http://127.0.0.1:8080/>

```
1
2 // A simple servlet to process get requests.
3
4 import javax.servlet.*; ← Importa i package javax.servlet
5 import javax.servlet.http.*; ← e javax.servlet.http.
6 import java.io.*;
7
8 public class ServletDiSaluto extends HttpServlet ← Estende HttpServlet in
9                                     modo che gestisca le
10                                     richieste HTTP get e
11                                     post.
12     // process "get" requests from clients
13     protected void doGet( HttpServletRequest request, ← Sovrascrive il metodo
14                         HttpServletResponse response ) ← doGet per fornire
15                                     funzionalità personalizzate.
16     {
17         throws ServletException, IOException
18         response.setContentType( "text/html" ); ← Utilizza l'oggetto response e il
19         PrintWriter out = response.getWriter(); ← relativo metodo setContentType
20                                     per specificare il tipo di contenuto dei
21                                     dati che devono essere spediti in
22                                     risposta al client.
23
24         // send XHTML page to client
25         // start XHTML document
26         out.println( "<?xml version = \"1.0\"?>" ); ← Utilizza il metodo getWriter
27                                     dell'oggetto response per ottenere un
28                                     riferimento all'oggetto PrintWriter
29                                     che abilita la servlet a spedire dati al
30                                     client.
31
32         out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\" \"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd\">" ); ← Crea il documento XHTML
33                                     scrivendo stringhe attraverso il
34                                     metodo println dell'oggetto out.
35     }
36 }
```

ServletDiSaluto.java

```
26
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 ← Chiude l'output stream,
42                                     esegue il flush del buffer di
43                                     output e spedisce le
44                                     informazioni al client.
45 }
```

File index.html

```
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 <!--File index.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 = "/DirectoryDisaluto/AliasServletDisaluto" 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>
```

Struttura della directory di una web application

- Context root – top level directory di una applicazione web

Si crea una sottodirectory nella directory webapps di Tomcat

Struttura della directory di una web application

Directory	Description
context root	This is the root directory for the Web application. The name of this directory is chosen by the Web application developer. 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 image subdirectory in this directory. The examples that follow use directorydisaluto 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.

Deployment descriptor (web.xml)

- Dopo avere configurato l'albero delle directory, dobbiamo configurare l'applicazione web in modo che possa gestire le richieste

→ Si usa un file web.xml che chiamiamo
deployment descriptor

Specifica diversi parametri come:

- il nome della classe che definisce la servlet,
- i percorsi che causano l'invocazione della servlet da parte del container


```

1 <!DOCTYPE web-app PUBLIC
2 "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
3 "http://java.sun.com/j2ee/tds/web-app_2_2.dtd">
4
5 <web-app>
6
7 <!-- General description of your V
8 <display-name>
9 NomeDisplayDellaWebApplication
10 Viene visualizzato dall'admin
11 </display-name>
12
13 <description>
14 Si tratta di un'applicazione dove facciamo
15 Esempi di servlet.
16 </description>
17
18 <!-- Servlet definitions -->
19 <servlet>
20
21 <servlet-name>Paperino</servlet-name>
22
23 <description>
24 Una servlet che gestisce le richieste di GET
25 </description>
26
27 <servlet-class>
28 ServletDiSaluto
29 </servlet-class>
30 </servlet>

```

L'elemento **<web-app>** definisce la configurazione di tutte le servlet della applicazione web

L'elemento **<display-name>** specifica un nome che viene utilizzato dall'amministratore del server (e visualizzato dal pannello del manager)

L'elemento **<description>** specifica una descrizione della applicazione che viene visualizzata nel pannello del manager

L'elemento **<servlet>** descrive una specifica servlet: se ci sono più servlet ci saranno più elementi **<servlet>**

L'elemento **<servlet-name>** definisce un nome per la servlet all'interno di questo file

L'elemento **<description>** fornisce una descrizione di questa servlet specifica

L'elemento **<servlet-class>** specifica il nome completo della classe servlet compilata

```

31 <!-- Servlet mappings -->
32 <servlet-mapping>
33 <servlet-name>Paperino</servlet-name>
34 <url-pattern>/AliasServletDiSaluto</url-pattern>
35 </servlet-mapping>
36
37 </web-app>

```

L'elemento **<servlet-mapping>** specifica l'associazione tra il nome dato alla servlet nell'interno del file web.xml (**<servlet-name>**) e l'**<url-pattern>** con cui la servlet potrà essere invocata

n.b.: non si specifica la context root nell'url-pattern!

Ricordiamoci che all'interno del server la tecnologia è **java**, mentre il colloquio con il client avviene nel protocollo **HTTP**.

L'associazione tra l'URL usata nelle richieste HTTP e la classe java che deve essere eseguita dal server è lo scopo principale del deployment descriptor.

Tale associazione è creata nel file web.xml attraverso l'elemento **<servlet-name>** che viene associato prima con il nome della classe (**<servlet-class>**) e poi con l'URL (**<url-pattern>**).

n.b. se la servlet fosse stata parte di un package, avremmo dovuto includere nella directory **classes** l'intera struttura del package

Gestione di una richiesta HTTP get Contenente Dati

- Servlet **WelcomeServlet2**
 - Rispondere ad una richiesta di tipo **get** contenente dati

```

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
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 = "/DirectoryDiSaluto/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.

```

2 // Processing HTTP get requests containing data.
3
4
5 import javax.servlet.*;
6 import javax.servlet.http.*;
7 import java.io.*;
8
9 public class WelcomeServlet2 extends HttpServlet {
10
11 // process "get" request from client
12 protected void doGet( HttpServletRequest request,
13     HttpServletResponse response )
14     throws ServletException, IOException
15 {
16     String firstName = request.getParameter( "firstname" );
17
18     response.setContentType( "text/html" );
19     PrintWriter out = response.getWriter();
20
21 // send XHTML document to client
22
23 // start XHTML document
24 out.println( "<?xml version = \"1.0\"?>" );
25
26 out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD \" +
27     \"XHTML 1.0 Strict//EN\" \"http://www.w3.org\" +
28     \"/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
29
30 out.println(
31     "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
32

```

onde
get

Il metodo
getParameter
dell'oggetto request
riceve come argomento
il nome del parametro e
restituisce il
corrispondente valore
sotto forma di una
stringa (tipo String)

```

33 // head section of document
34 out.println( "<head>" );
35 out.println(
36     "<title>Processing get requests with data</title>" );
37 out.println( "</head>" );
38
39 // body section of document
40 out.println( "<body>" );
41 out.println( "<h1>Hello " + firstName + "<br />" );
42 out.println( "Welcome to Servlets!</h1>" );
43 out.println( "</body>" );
44
45 // end XHTML document
46 out.println( "</html>" );
47 out.close(); // close stream to complete the page
48 }
49 }

```

La stringa ottenuta alla
linea 16 viene usata
per costruire la risposta
al client.

Modifica del file web.xml per includere la nuova servlet

Descriptor element	Value
<i>servlet element</i>	
servlet-name	Welcome2
description	Handling HTTP get requests with data.
servlet-class	WelcomeServlet2
<i>servlet-mapping element</i>	
servlet-name	Welcome2
url-pattern	/Welcome2

Modifica del file web.xml per includere la nuova servlet

- Si **aggiungono** le seguenti direttive:

```

<servlet>
  <servlet-name>Welcome2</servlet-name>
  <display-name>
    NomeDisplayDiSalutoPersonalizzato</display-name>
  <description>Facciamo un test con personalizzazione</description>
  <servlet-class>WelcomeServlet2</servlet-class>
</servlet>

```

```

<servlet-mapping>
  <servlet-name>Welcome2</servlet-name>
  <url-pattern>/Welcome2</url-pattern>
</servlet-mapping>

```



Servlet che gestisce richieste di POST

- HTTP post request
 - Spedire dati da un form HTML ad un handler di form localizzato sul server
 - I browser mettono in cache le pagine Web visitate (non fanno caching delle risposte post)
- Servlet WelcomeServlet3
 - Risponde ad una richiesta di tipo **post** contenente dati

```

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 <!-- 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 = "/DirectoryDiSaluto/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>

```

Viene fornito un **form** in cui l'utente può scrivere il proprio nome nell'elemento di input **firstname** di tipo **text**. Quando viene cliccato il bottone **Submit** viene invocata **WelcomeServlet3**.

```

2 // Processing post requests containing data.
3
4
5 import javax.servlet.*;
6 import javax.servlet.http.*;
7 import java.io.*;
8
9 public class WelcomeServlet3 extends HttpServlet {
10
11 // process "post" request from client
12 protected void doPost( HttpServletRequest request,
13 HttpServletResponse response )
14 throws ServletException, IOException
15 {
16 String firstName = request.getParameter( "firstname" );
17
18 response.setContentType( "text/html" );
19 PrintWriter out = response.getWriter();
20
21 // send XHTML page to client
22
23 // start XHTML document
24 out.println( "<?xml version = \"1.0\"?>" );
25
26 out.println( "<!DOCTYPE html PUBLIC \"-//W3C//DTD \" +
27 \"XHTML 1.0 Strict//EN\" \"http://www.w3.org\" +
28 \"/TR/xhtml1/DTD/xhtml1-strict.dtd\">" );
29
30 out.println(
31 "<html xmlns = \"http://www.w3.org/1999/xhtml\">" );
32

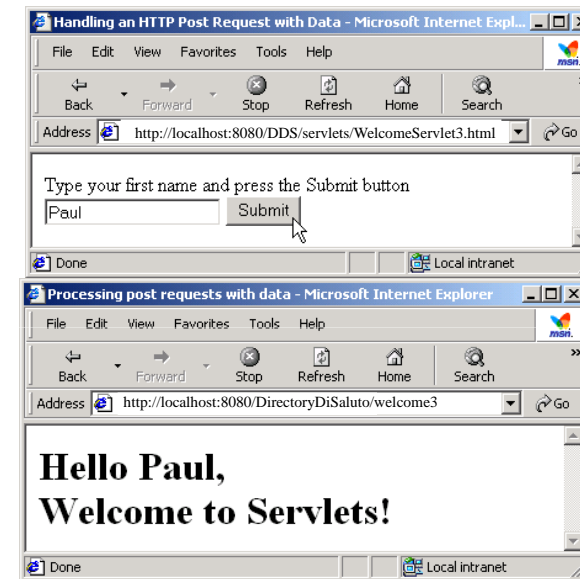
```

Define a **doPost** method to responds to post requests.

```

33 // head section of document
34 out.println( "<head>" );
35 out.println(
36     "<title>Processing post requests with data</title>" );
37 out.println( "</head>" );
38
39 // body section of document
40 out.println( "<body>" );
41 out.println( "<h1>Hello " + firstName + ",<br />" );
42 out.println( "Welcome to Servlets!</h1>" );
43 out.println( "</body>" );
44
45 // end XHTML document
46 out.println( "</html>" );
47 out.close(); // close stream to complete the page
48 }
49 }

```



Modifiche al file web.xml

Descriptor element	Value
<i>servlet element</i>	
servlet-name	welcome3
description	Handling HTTP post requests with data.
servlet-class	WelcomeServlet3
<i>servlet-mapping element</i>	
servlet-name	welcome3
url-pattern	/welcome3

Non dimenticate di ricaricare l'applicazione dopo avere ultimato e salvato le modifiche al file web.xml