SVILUPPO WEB

- Tomcat Application Server con Eclipse integration
- Servlet, definizione, ciclo di vita
- Test con Postman
- Web Services SOAP e REST
- Servizi esposti in modalità Web Service REST
- Pattern MVC
- JPA, annotations di persistenza

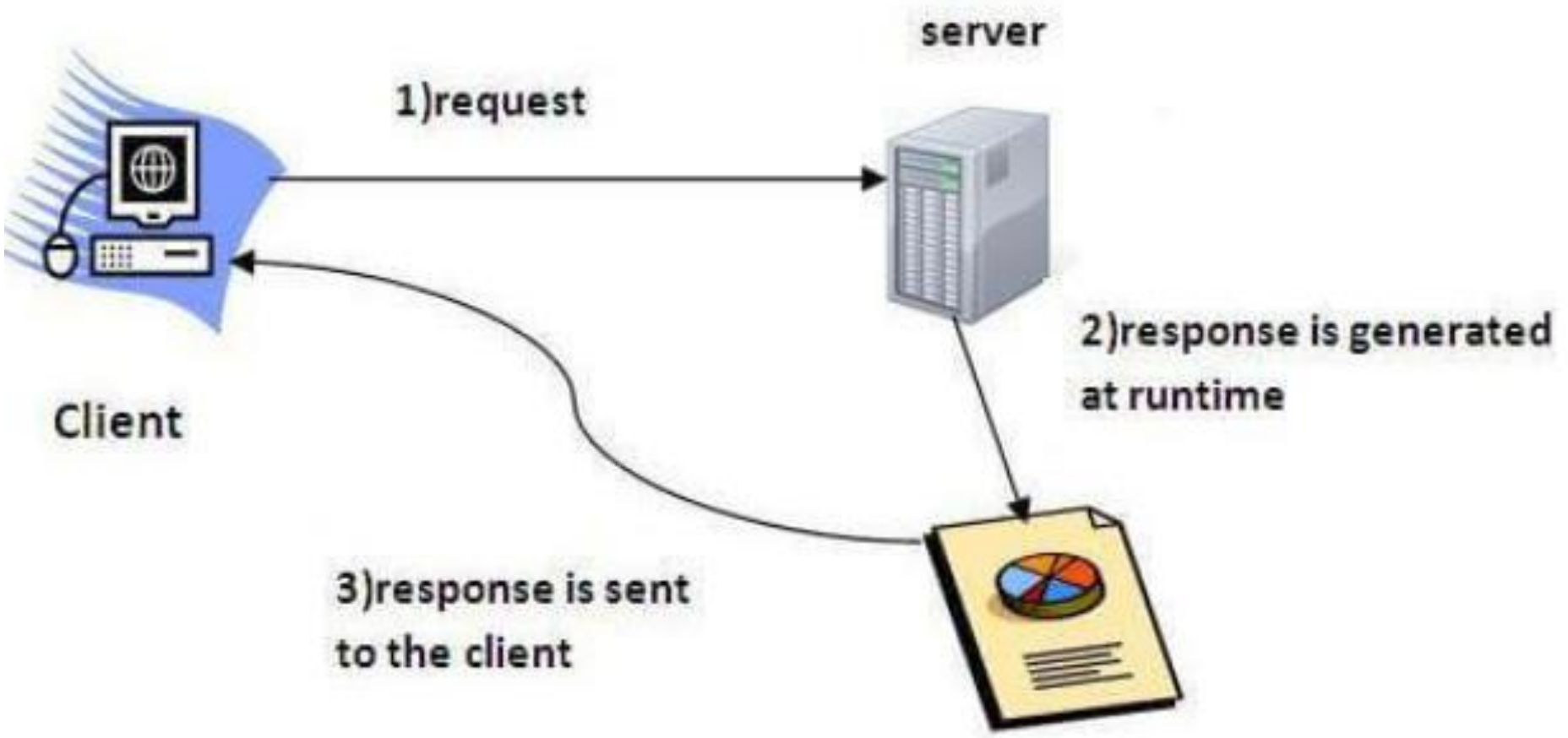
Application Server

- L’application server è il contenitore dove girano le applicazioni che interagiscono con le richieste HTTP e possono essere delle **Servlet** o dei **WebService REST**
- Application server possono essere il server **Apache**, il **Tomcat**, **JBOSS** (IBM), **IIS** (Microsoft),..
- L’application server viene installato sulla macchina ma può essere integrato con eclipse e si può (startare, chiudere, pubblicare,..)
- La pubblicazione consiste nel muovere i .class dal workspace eclipse di sviluppo nelle cartelle (standard) dell’application server

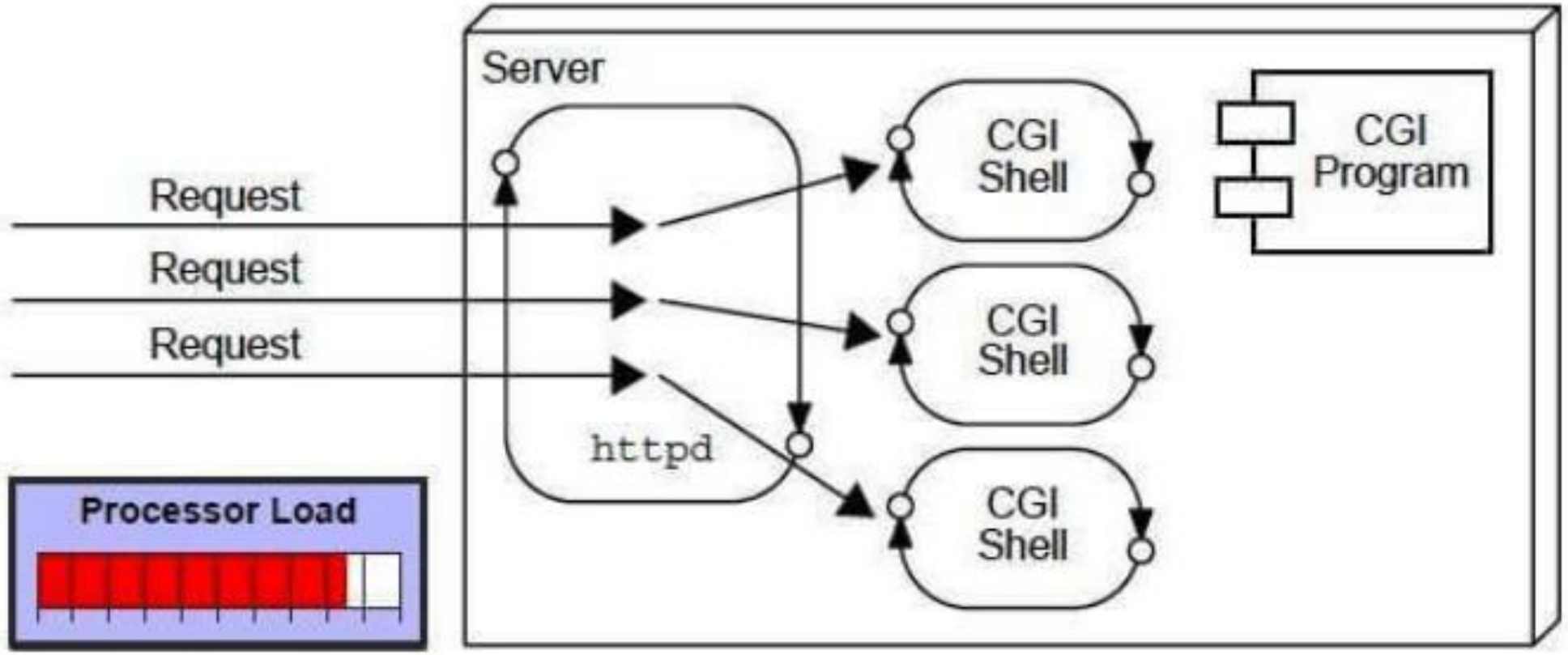
Servlet

- E’ una tecnologia usata per creare una **web application**, risiede sul server e genera una pagina dinamica
- Servlet è una **API** che fornisce molte interfacce e classi
- Servlet è una **interface** che deve essere **implementata** per creare qualsiasi Servelet
- Servlet è una **classe** che estende le capacità del server e risponde alle richieste in arrivo. La servelet può rispondere a qualsiasi richiesta.
- Servlet è una **componente web** che è ‘deployata’ sul server per creare una pagina web dinamica

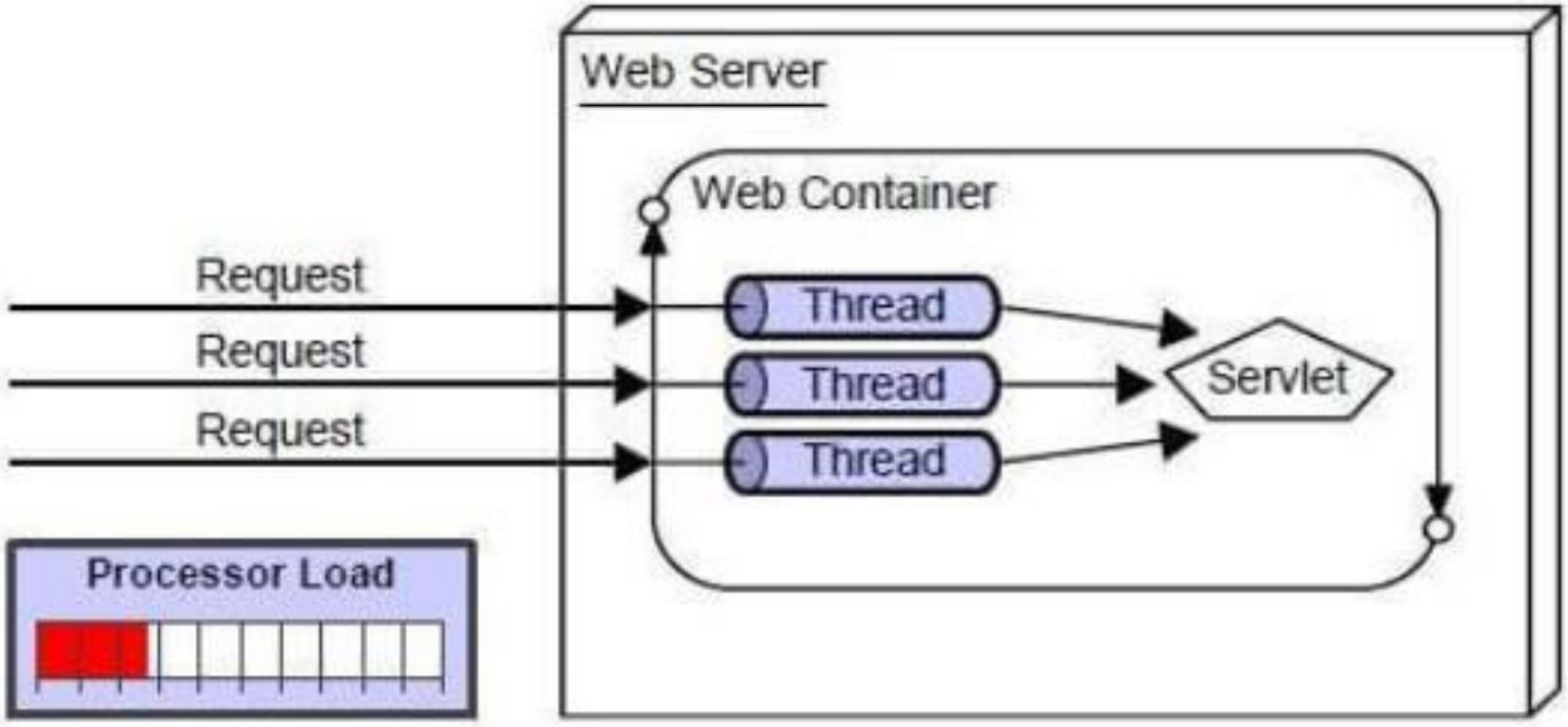
Servelet Schema Generale



CGI (Common Gateway Interface)



Vantaggi della Servlet



Servlet API

- **javax.servlet** e **javax.servlet.http** packages rappresentano interfacce e classi per le servelet api
- **javax.servlet** package contiene molte interfacce e classi usate dalla servlet o dal web container. Queste non sono specifiche di nessun protocollo.
- **javax.servlet.http** package contiene interfacce e classi responsabili delle sole richieste **HTTP**

Interfaces in javax.servlet package

1.Servlet
2.ServletRequest
3.ServletResponse
4.RequestDispatcher
5.ServletConfig
6.ServletContext
7.SingleThreadModel
8.Filter
9.FilterConfig
10.FilterChain
11.ServletRequestListener
12.ServletRequestAttributeListener
13.ServletContextListener
14.ServletContextAttributeListener

Classes in javax.servlet package

- GenericServlet
- ServletInputStream
- ServletOutputStream
- ServletRequestWrapper
- ServletResponseWrapper
- ServletRequestEvent
- ServletContextEvent
- ServletRequestAttributeEvent
- ServletContextAttributeEvent
- ServletException
- UnavailableException

Interfaces in javax.servlet.http package

1. HttpServletRequest
2. HttpServletResponse
3. HttpSession
4. HttpSessionListener
5. HttpSessionAttributeListener
6. HttpSessionBindingListener
7. HttpSessionActivationListener
8. HttpSessionContext (deprecated now)

Classes in javax.servlet.http package

- HttpServlet
- Cookie
- HttpServletRequestWrapper
- HttpServletResponseWrapper
- HttpSessionEvent
- HttpSessionBindingEvent
- HttpUtils (deprecated now)

Servlet Interface

- **Servlet interface fornisce** I comportamenti comuni alle servlets. Servlet interface definisce i metodi che tuttele servlet devono implementare.
- Servlet interface **deve essere implementata** per creare qualsiasi servlet (**direttamente** o **indirettamente**). Fornisce 3 metodi del ciclo di vita di una servelet, usati per inizializzare la servlet, servire le requests, e per distruggere la servlet.
- Sevelet interface definisce anche 2 metodi che non fanno parte del ciclo di vita della servelet.

Metodi di Servlet interface

Method	Description
public void init(ServletConfig config)	initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service(ServletRequest request,ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig()	returns the object of ServletConfig.
public String getServletInfo()	returns information about servlet such as writer, copyright, version etc.

GenericServlet class

- Implementa **Servlet**, **ServletConfig** e **Serializable** interfaces. Fornisce l'implementazione di tutti I metodi di quelle interface eccetto il service method.
- Può trattare qualsiasi tipo di request e quindi è **protocol-independent.**
- Si puo creare una servlet generica **ereditando** da GenericServlet and fornendo l'implementazione del metodo **service**.

Methods of GenericServlet class

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6. **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)
7. **public ServletContext getServletContext()** returns the object of ServletContext.
8. **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
9. **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.
10. **public String getServletName()** returns the name of the servlet object.
11. **public void log(String msg)** writes the given message in the servlet log file.
12. **public void log(String msg,Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

HttpServlet class

- HttpServlet class extends **GenericServlet** class e implementa la Serializable interface.
- Fornisce metodi http specifici methods come **doGet**, **doPost**, **doHead**, **doTrace** etc

Methods of HttpServlet class 1

1. **public void service(ServletRequest req,ServletResponse res)** dispatches the request to the protected service method by converting the request and response object into http type.
2. **protected void service(HttpServletRequest req, HttpServletResponse res)** receives the request from the service method, and dispatches the request to the doXXX() method depending on the incoming http request type.
3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
5. **protected void doHead(HttpServletRequest req, HttpServletResponse res)**
   handles the HEAD request. It is invoked by the web container.

Methods of HttpServlet class 2

1. **protected void doOptions(HttpServletRequest req, HttpServletResponse res)** handles the OPTIONS request. It is invoked by the web container.
2. **protected void doPut(HttpServletRequest req, HttpServletResponse res)** handles the PUT request. It is invoked by the web container.
3. **protected void doTrace(HttpServletRequest req, HttpServletResponse res)** handles the TRACE request. It is invoked by the web container.
4. **protected void doDelete(HttpServletRequest req, HttpServletResponse res)** handles the DELETE request. It is invoked by the web container.
5. **protected long getLastModified(HttpServletRequest req)** returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

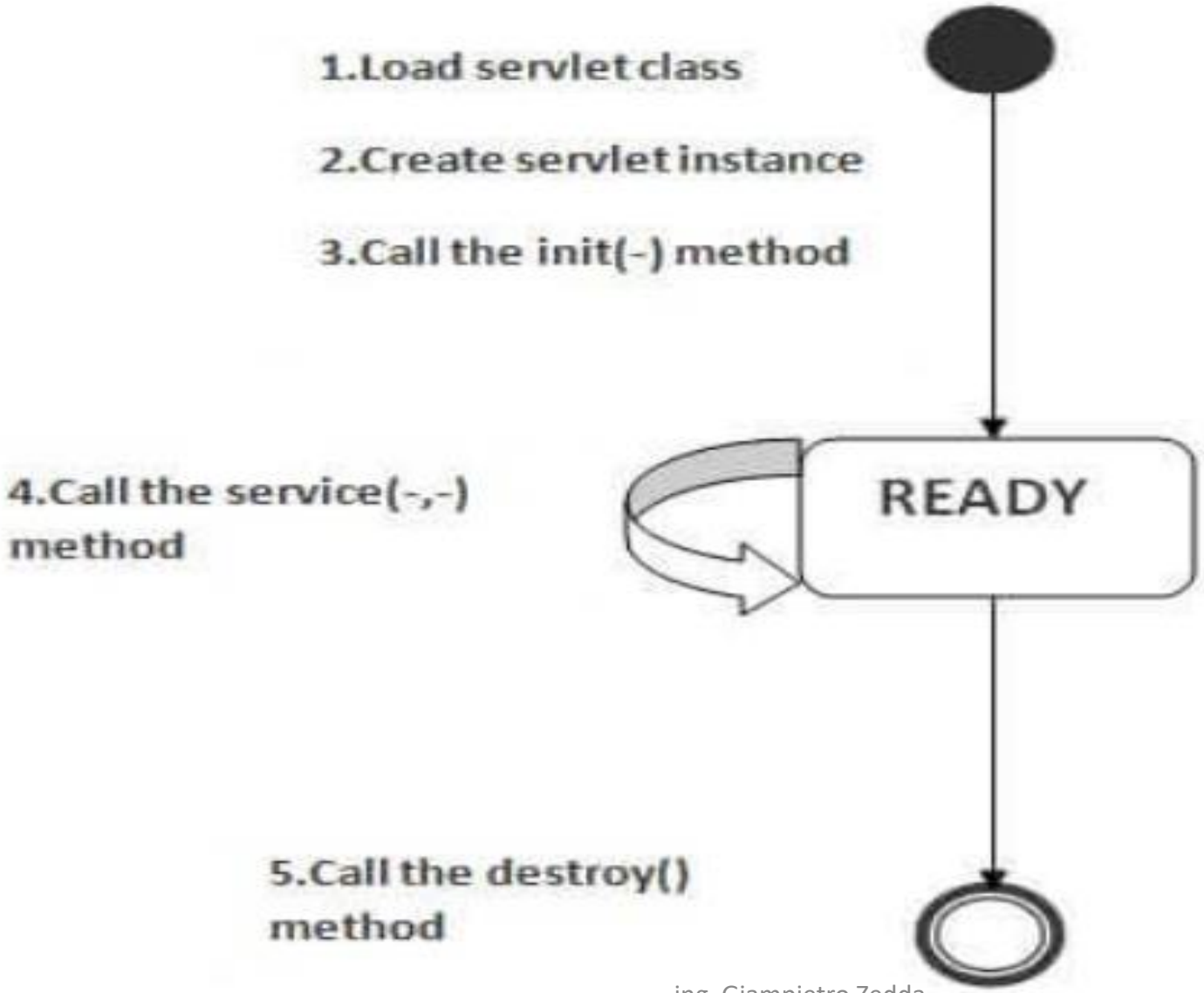
Servlet Life Cycle 1

Il web container **mantiene** il ciclo di vita dell'istanza di una servlet.

Sotto il ciclo di vita di una servlet:

1. Servlet class è caricata.
2. Servlet instance è creata.
3. init method è invocato.
4. service method è invocato.
5. destroy method è invocato.

Servlet Life Cycle 2



Servlet Life Cycle 3

- Come indicato dallo schema, ci sono 3 stati di una servelet: **new**, **ready** and **end**.
- La servlet è nello stato **new** quando l'istanza della servlet è creata.
- Dopo l'invocazione del metodo **init()**, Servlet va nello stato **ready**.
- Nello stato ready, servlet ouò eseguire tutti i task.
- Quando il web container invoca il metodo **destroy()**, la servlet va nello stato **end**.

Creazione di una servlet 1

Si può creare una servlet in **3** modi

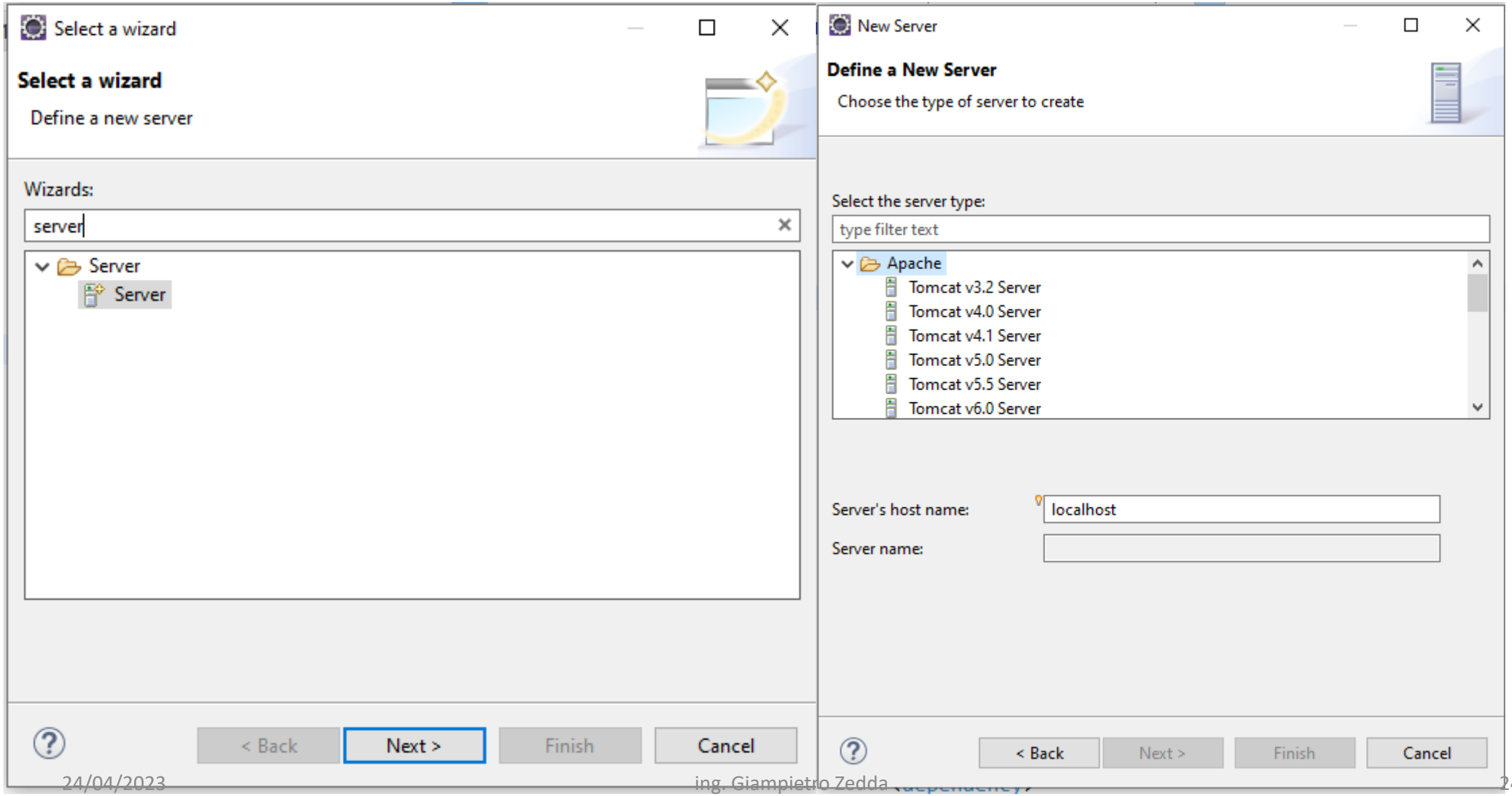
- Implementando la **Servlet interface**
- Ereditando da **GenericClass**
- Ereditando da **HTTPServlet**

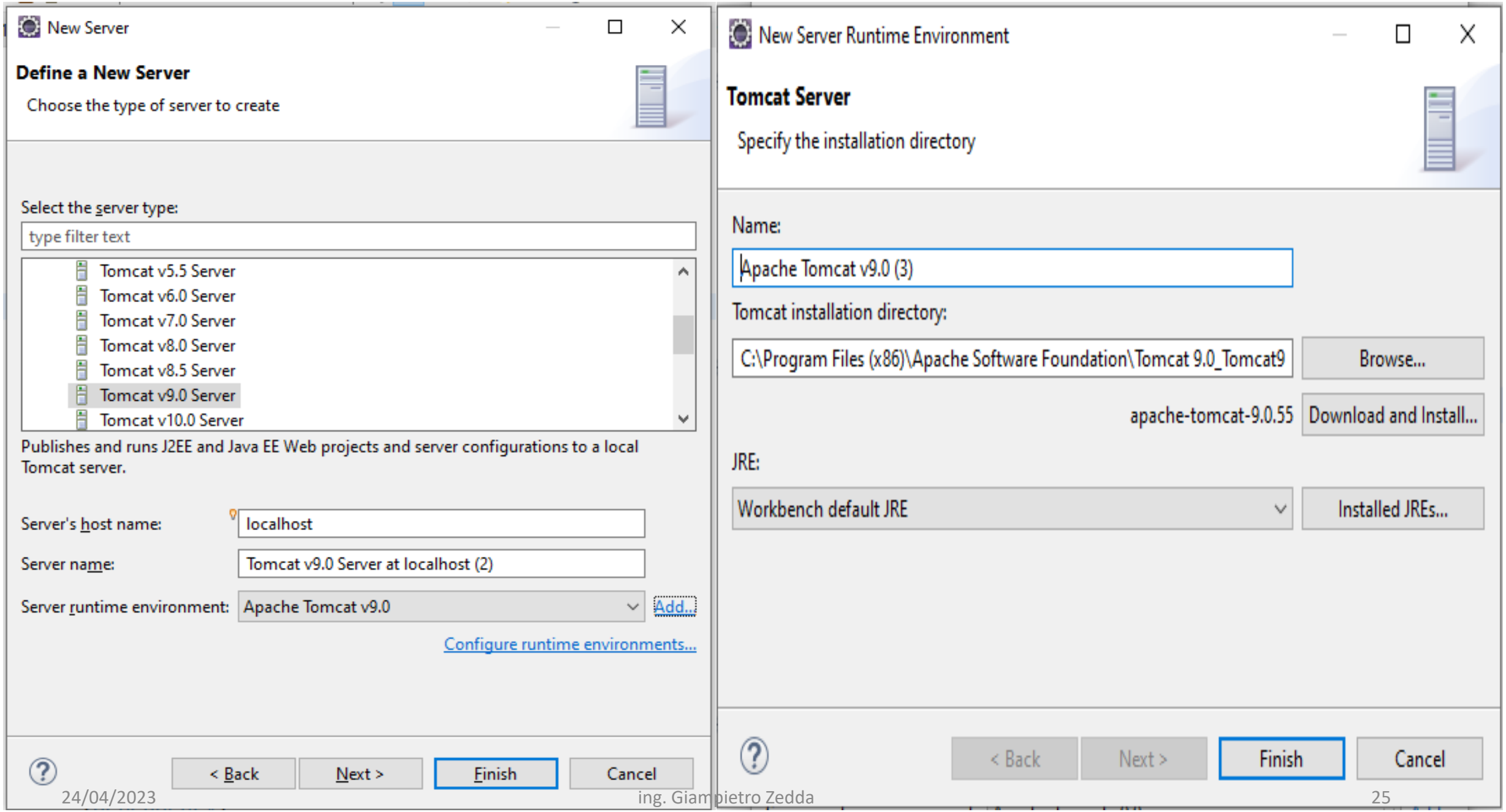
Il modo più comune è quello di ereditare da HTTPServlet, perché vengono forniti i metodi specifici quali **doGet()**, **doPost()**, **doHead()** etc.

Creazione di Servlet con Eclipse e Maven

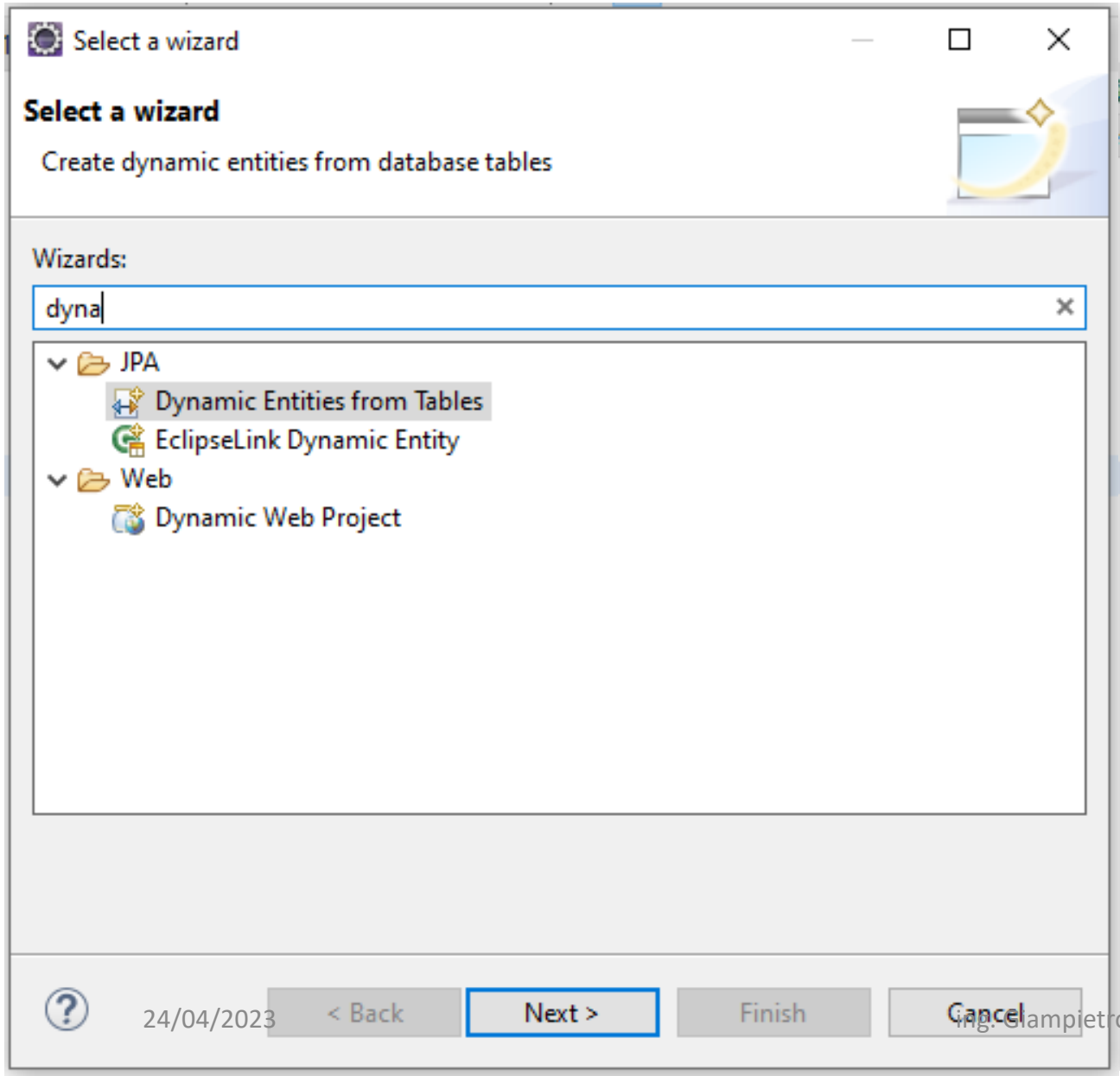
1. Installazione application server (Tomcat)
2. Configurazione Eclipse per il server Tomcat
3. Creazione di un dynamic web project
4. Convertirlo in un progetto Maven
5. Inserire nel .pom almeno la dipendenza per javax.servlet
6. Fare file/new di una servlet, lasciando i default
7. Nella finestra di Eclipse server, aggiungere il progetto e pubblicarlo
8. Sempre nella finestra server avviare il tomcat se non già avviato
9. Andare in un browser qualsiasi e digitare per esempio
   http://localhost:8080/TestServelet3/servelet3
   Dove TestServelet3 è il progetto e servelet3 la servlet definita

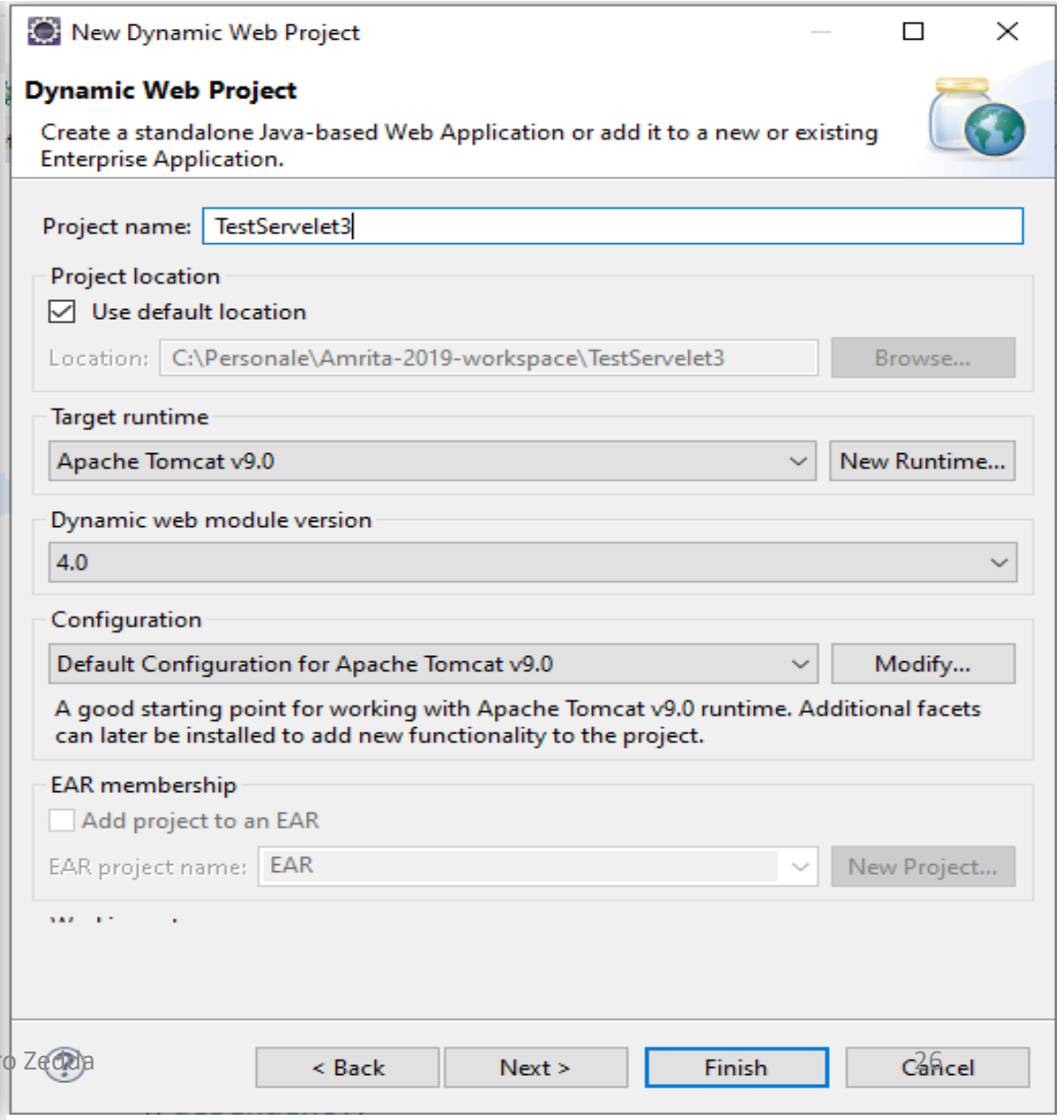
2) Configurazione Eclipse per il server Tomcat

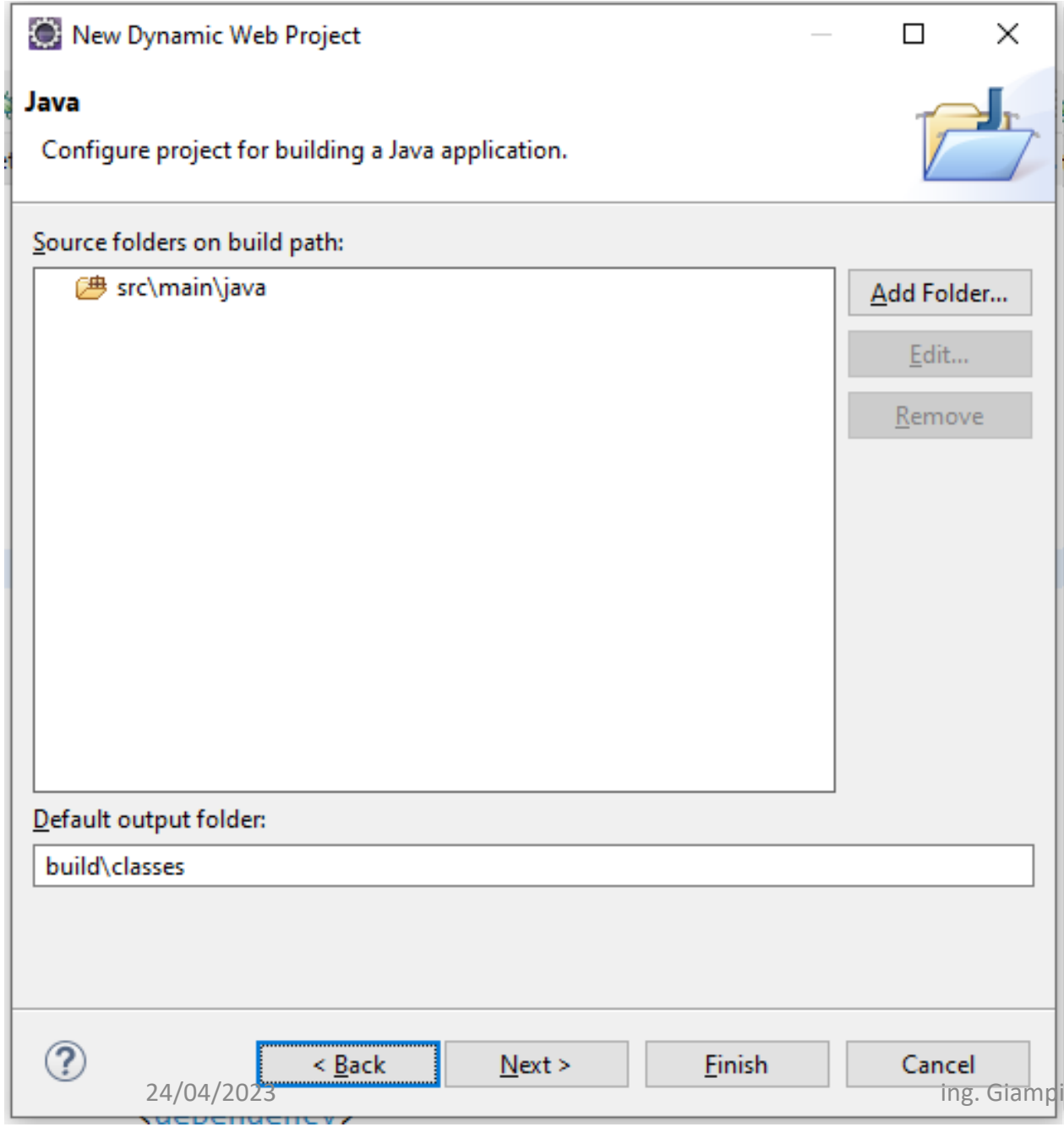


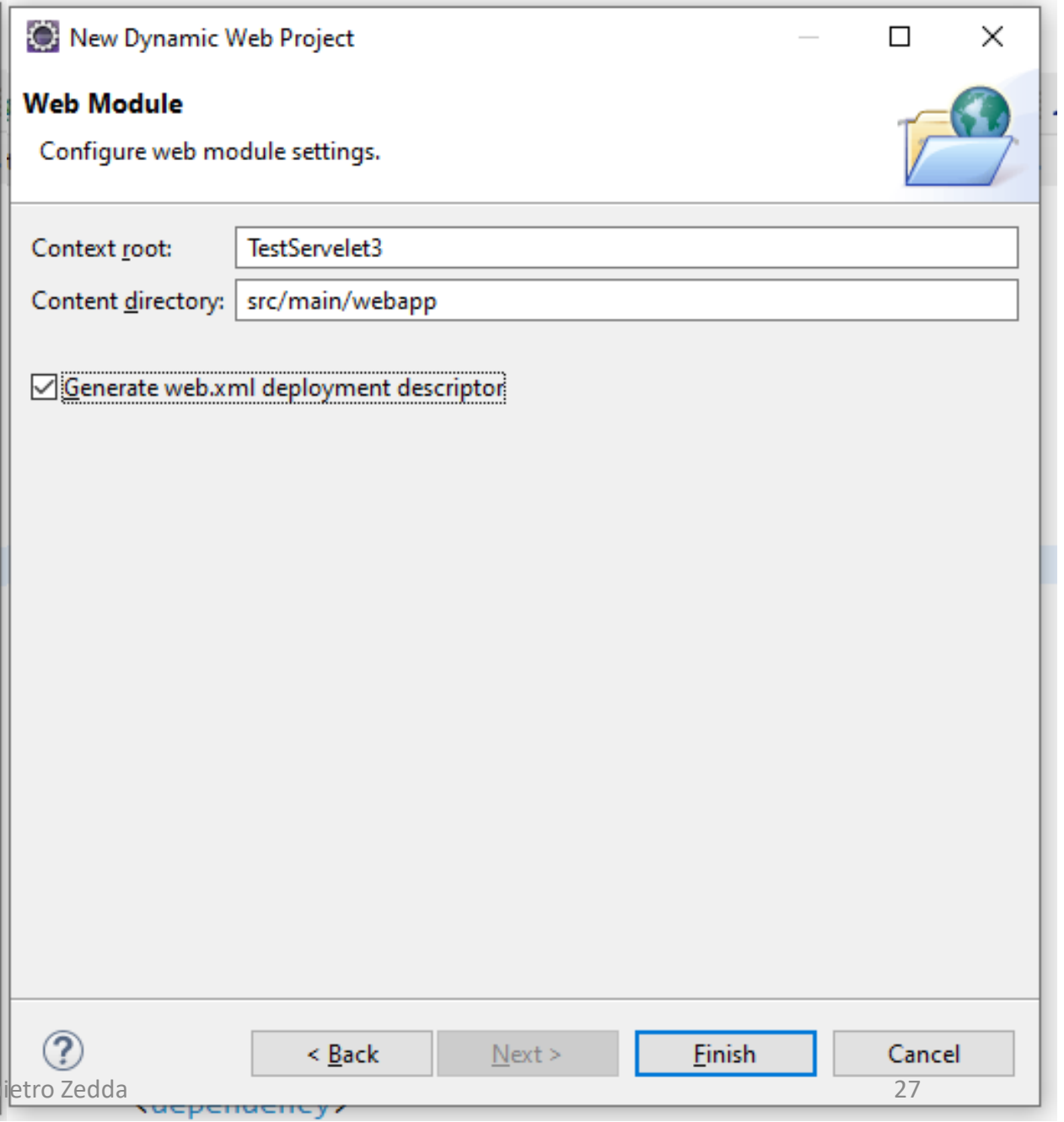

New Server
Define a New Server
Choose the type of server to create
Select the server type:
type filter text
Tomcat v5.5 Server
Tomcat v6.0 Server
Tomcat v7.0 Server
Tomcat v8.0 Server
Tomcat v8.5 Server
Tomcat v9.0 Server
Tomcat v10.0 Server
Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.
Server's host name:
localhost
Server name:
Tomcat v9.0 Server at localhost (2)
Server runtime environment:
Apache Tomcat v9.0
Add...
Configure runtime environments...
< Back
Next >
Finish
Cancel
New Server Runtime Environment
Tomcat Server
Specify the installation directory
Name:
Apache Tomcat v9.0 (3)
Tomcat installation directory:
C:\Program Files (x86)\Apache Software Foundation\Tomcat 9.0_Tomcat9
Browse...
apache-tomcat-9.0.55
Download and Install...
JRE:
Workbench default JRE
Installed JREs...
< Back
Next >
Finish
Cancel

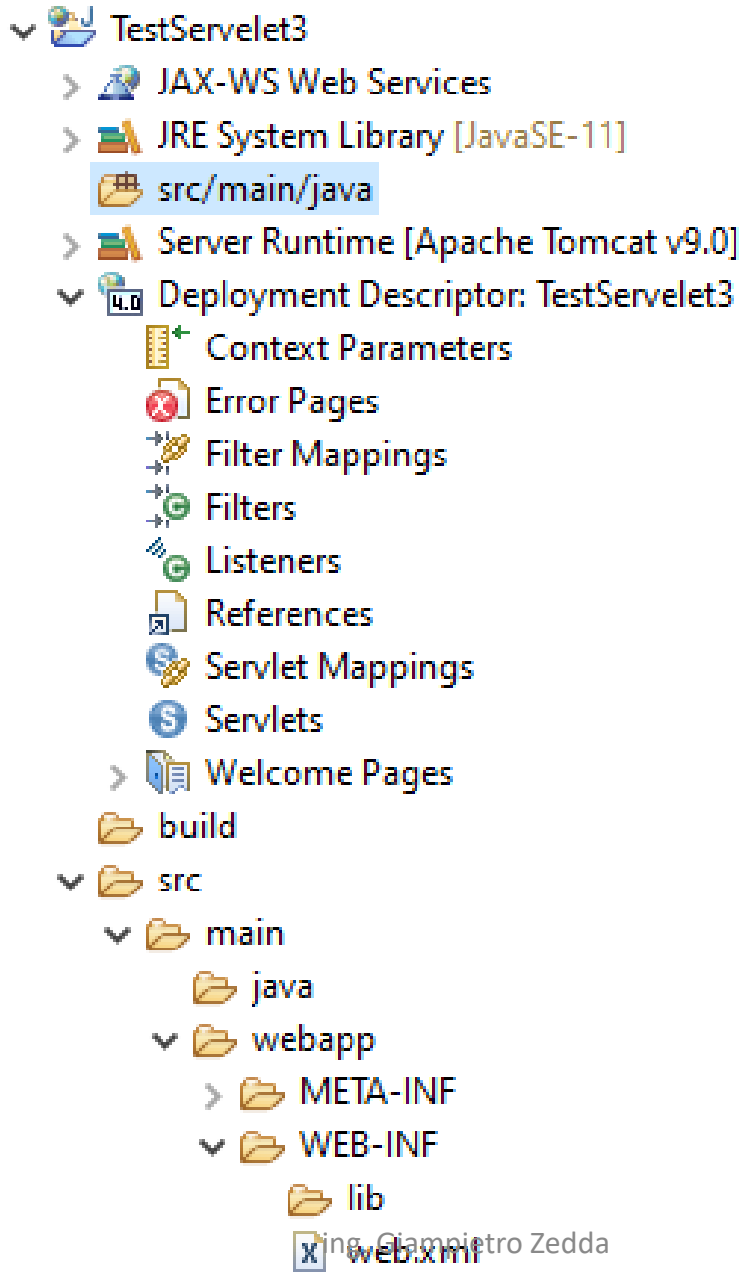
3) Creazione di un Dynamic Web Project



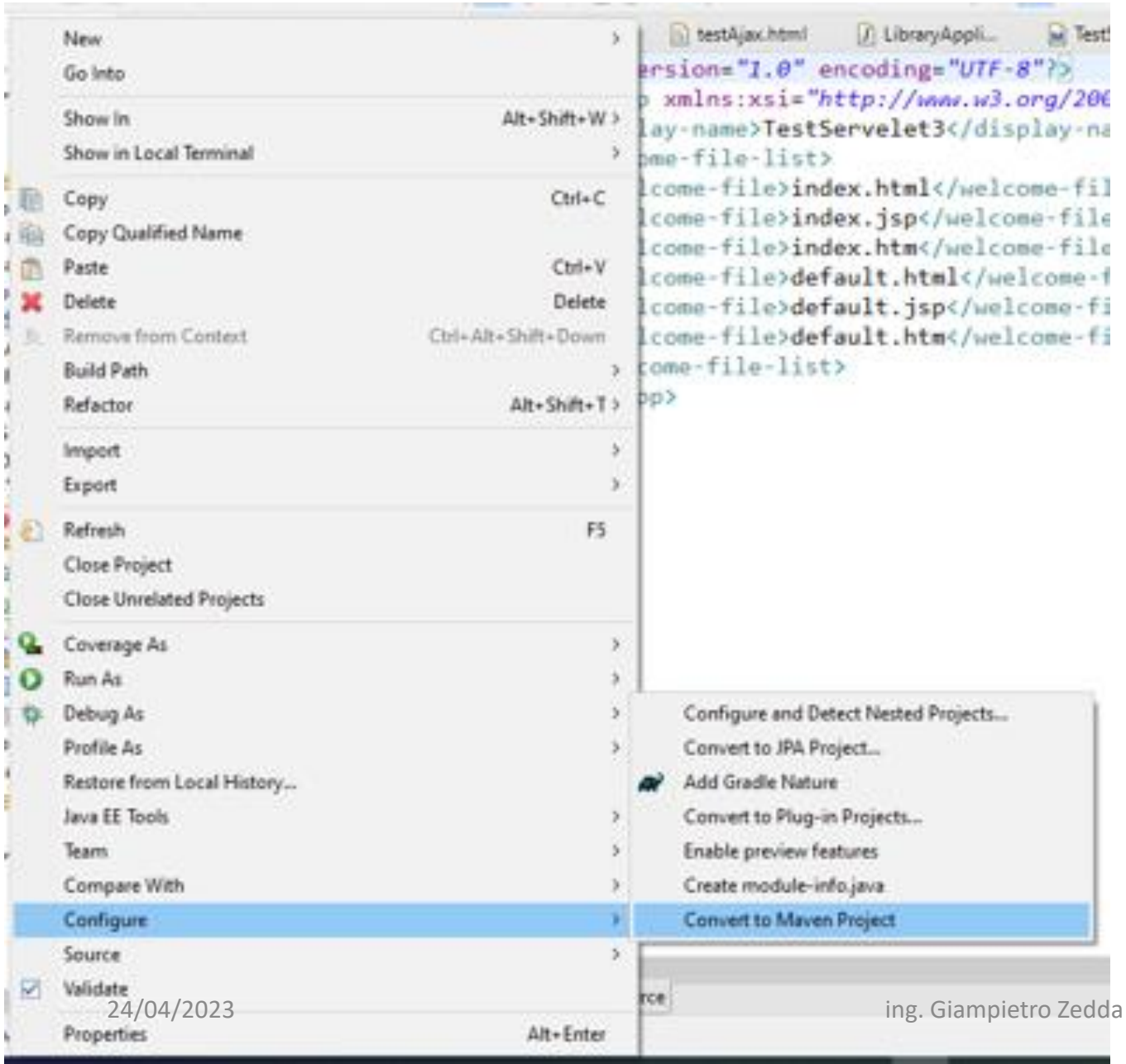


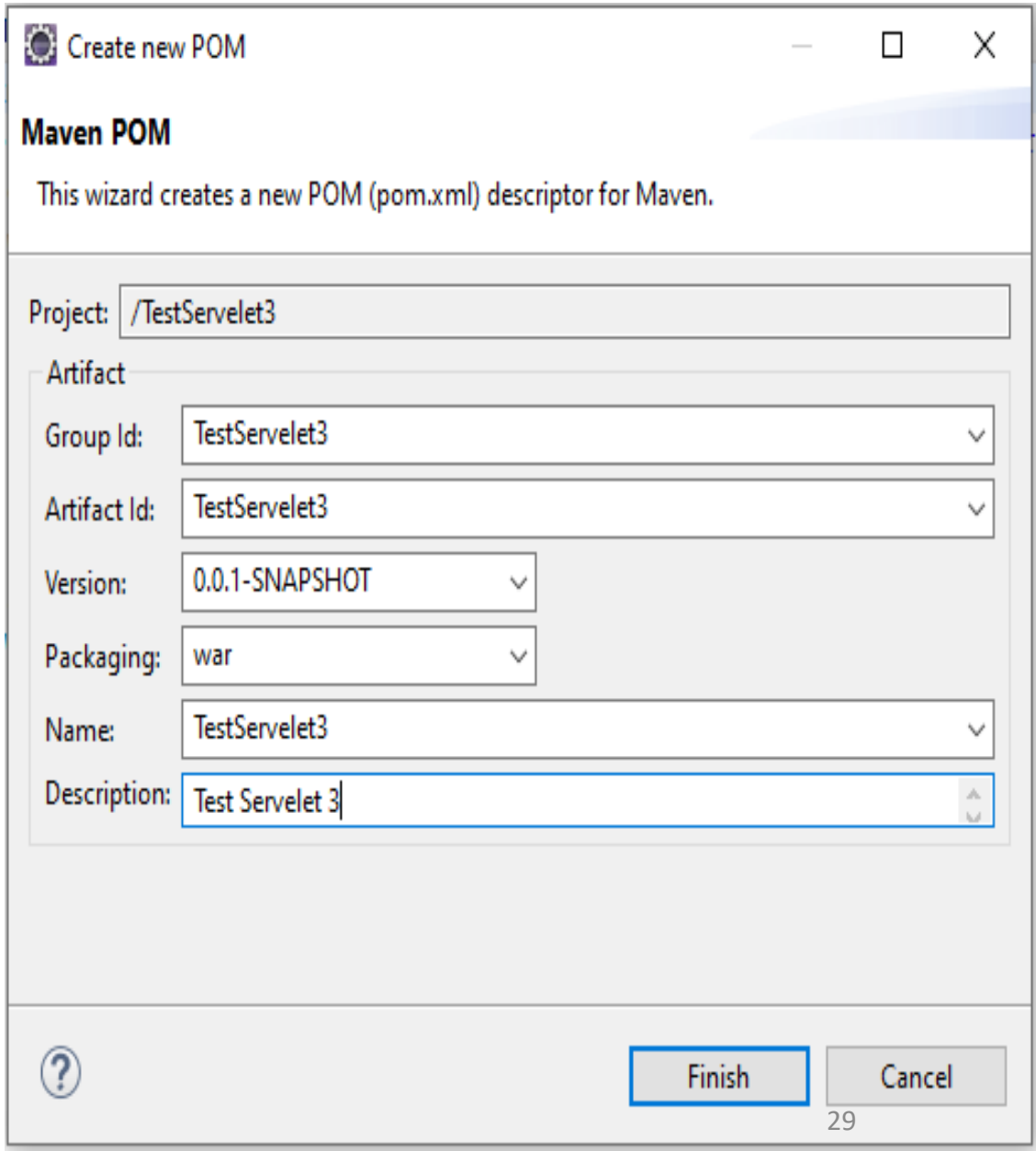

New Dynamic Web Project
Java
Configure project for building a Java application.
Source folders on build path:
src\main\java
Add Folder...
Edit...
Remove
Default output folder:
build\classes
< Back
Next >
Finish
Cancel

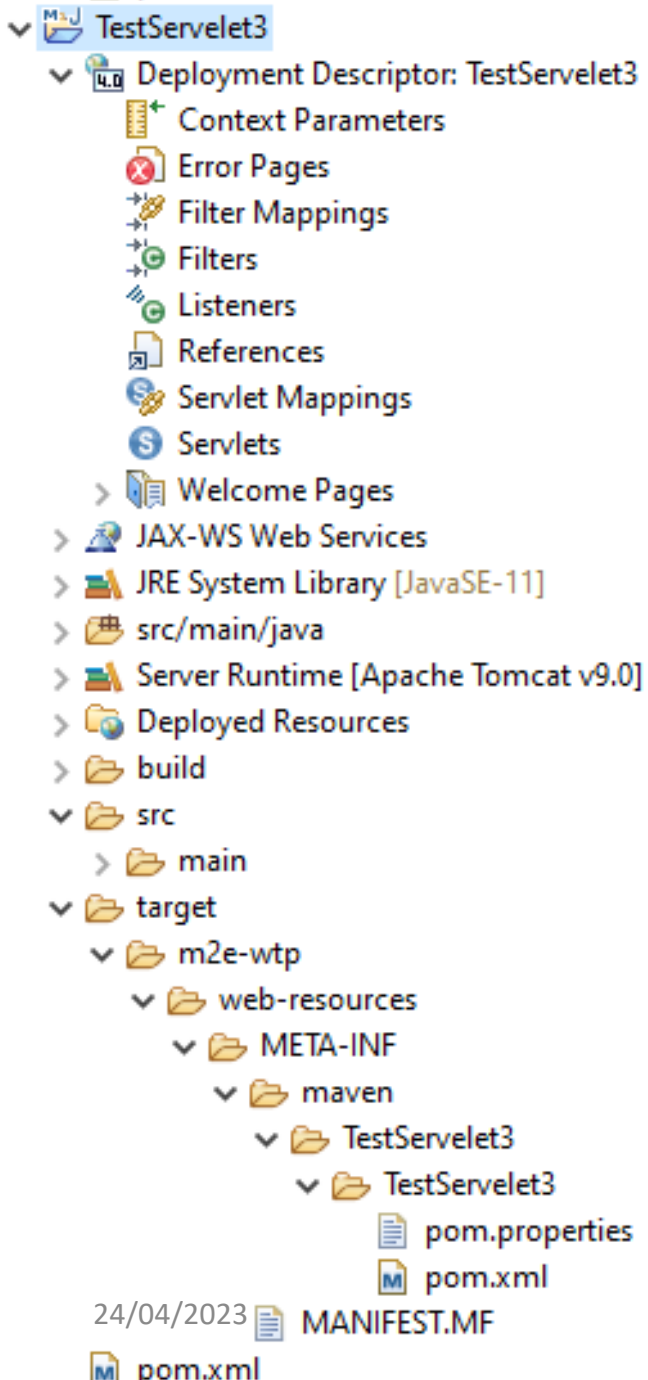

New Dynamic Web Project
Web Module
Configure web module settings.
Context root:
TestServelet3
Content directory:
src/main/webapp
Generate web.xml deployment descriptor
< Back
Next >
Finish
Cancel


TestServelet3
JAX-WS Web Services
JRE System Library [JavaSE-11]
src/main/java
Server Runtime [Apache Tomcat v9.0]
Deployment Descriptor: TestServelet3
Context Parameters
Error Pages
Filter Mappings
Filters
Listeners
References
Servlet Mappings
Servlets
Welcome Pages
build
src
main
java
webapp
META-INF
WEB-INF
lib
web.xml

4) Conversione in un progetto Maven





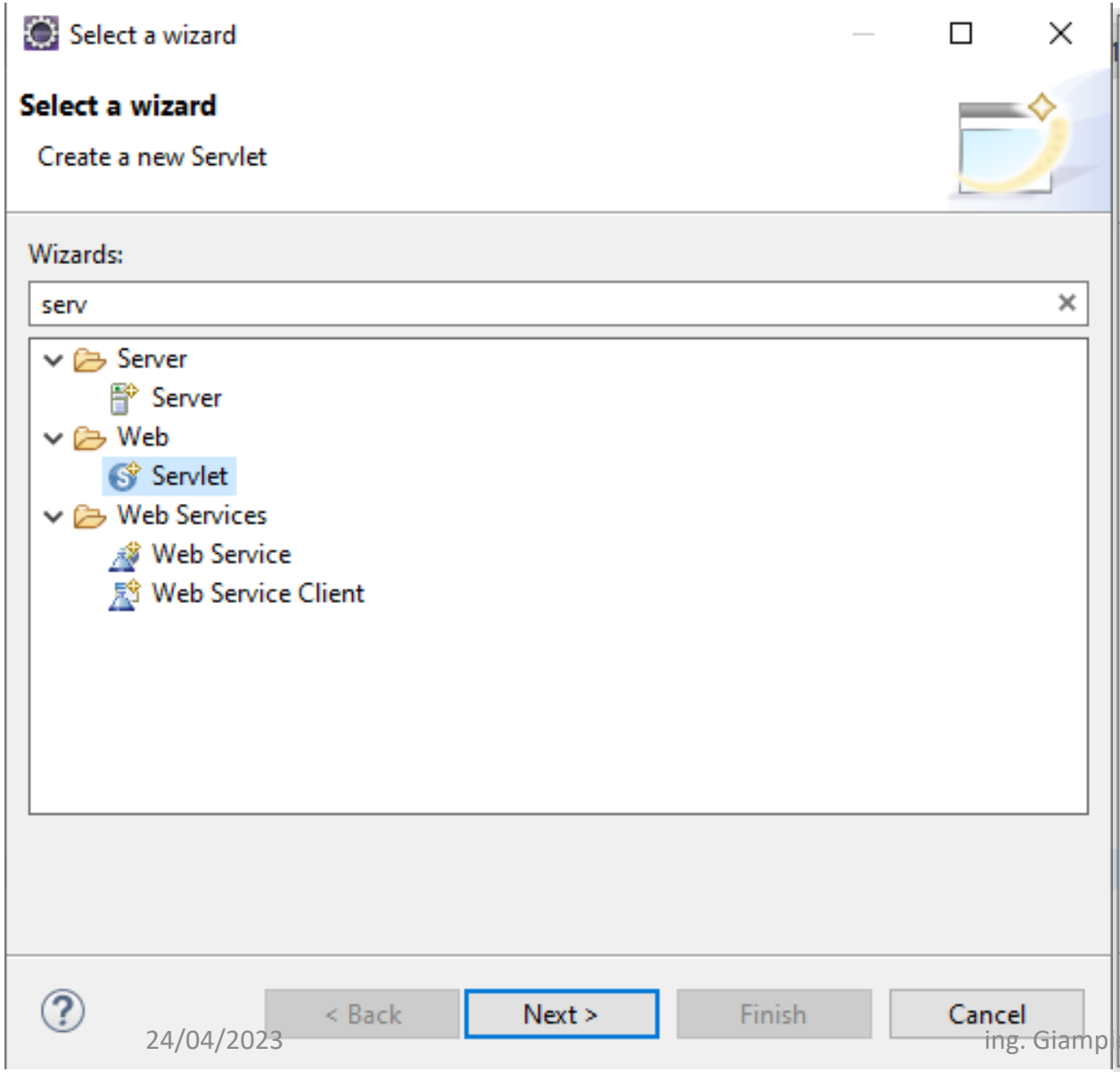


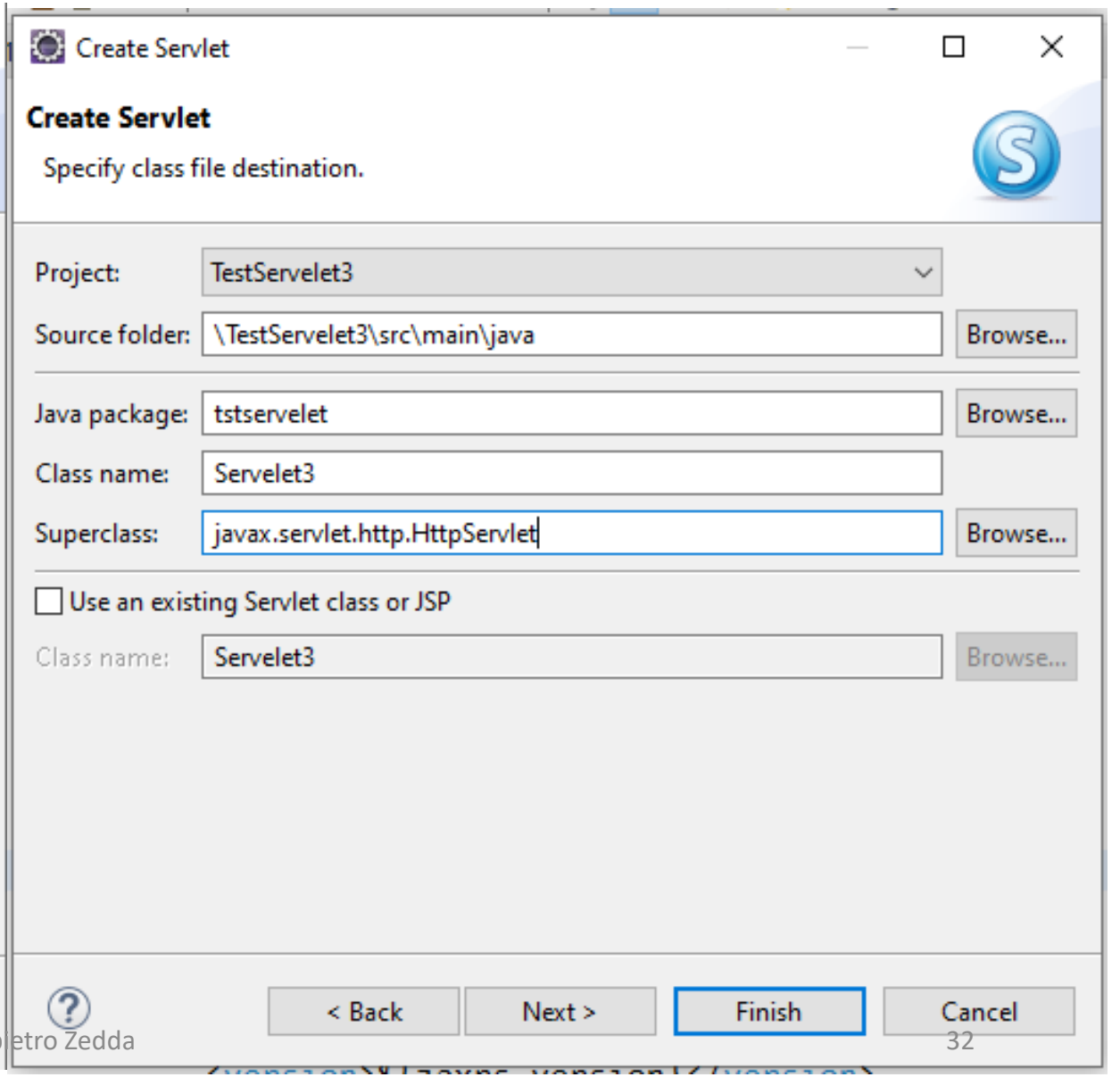

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xs
  <modelVersion>4.0.0</modelVersion>
  <groupId>TestServelet3</groupId>
  <artifactId>TestServelet3</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
  <name>TestServelet3</name>
  <description>Test Servelet 3</description>

  <build>
    <plugins>
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <release>11</release>
        </configuration>
      </plugin>
      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.2.3</version>
      </plugin>
    </plugins>
  </build>
</project>
```

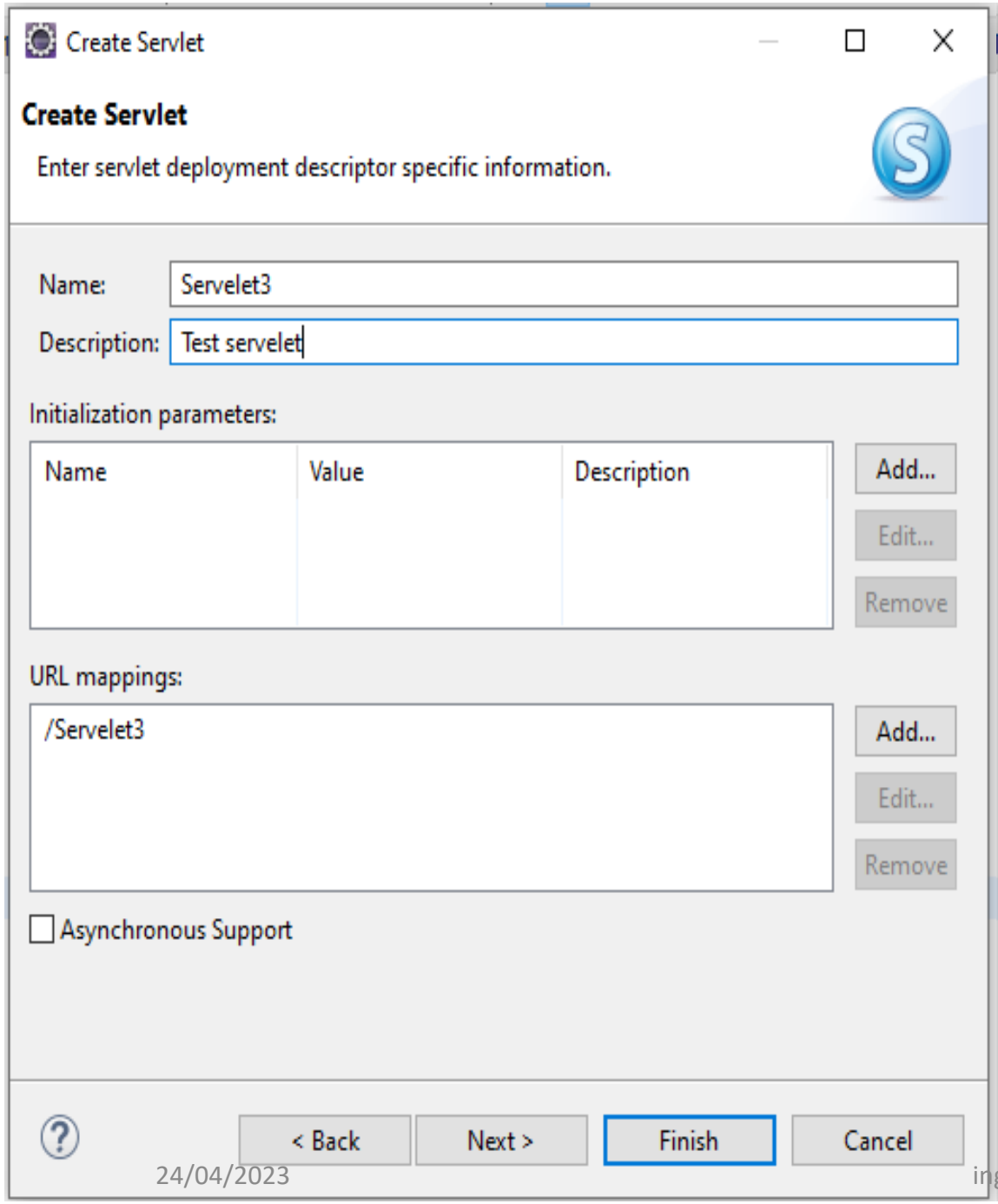
5) Inserimento nel Pom della dipendenza javax.servlet

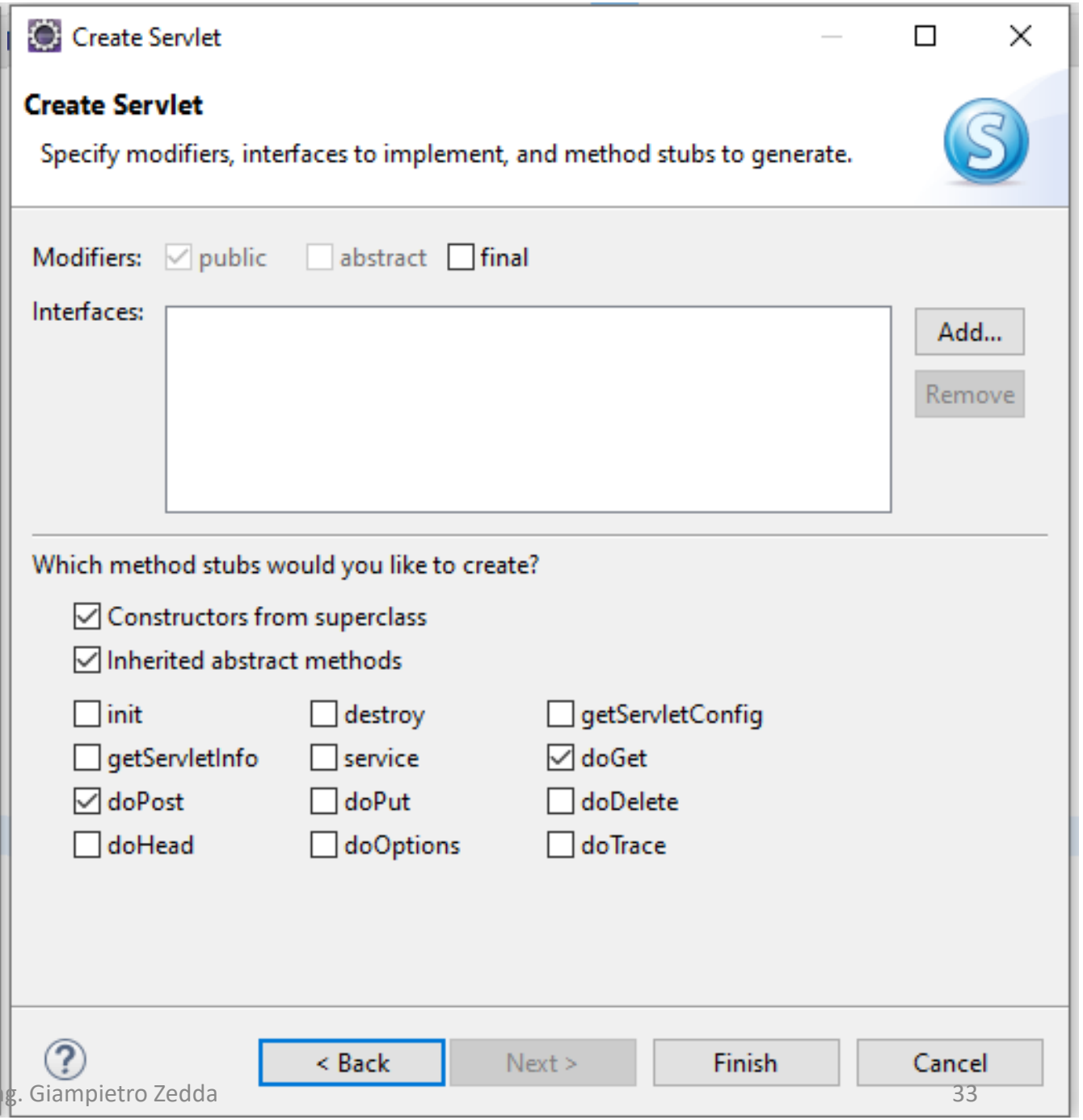
```
<repositories>
    <repository>
        <id>maven2-repository.java.net</id>
        <name>Java.net Repository for Maven</name>
        <url>http://download.java.net/maven/2/</url>
        <layout>default</layout>
    </repository>
</repositories>

<dependencies>
    <!-- Servelet -->
    <dependency>
        <groupId>javax.servlet</groupId>
        <artifactId>servlet-api</artifactId>
        <version>2.3</version>
        <scope>provided</scope>
    </dependency>
```

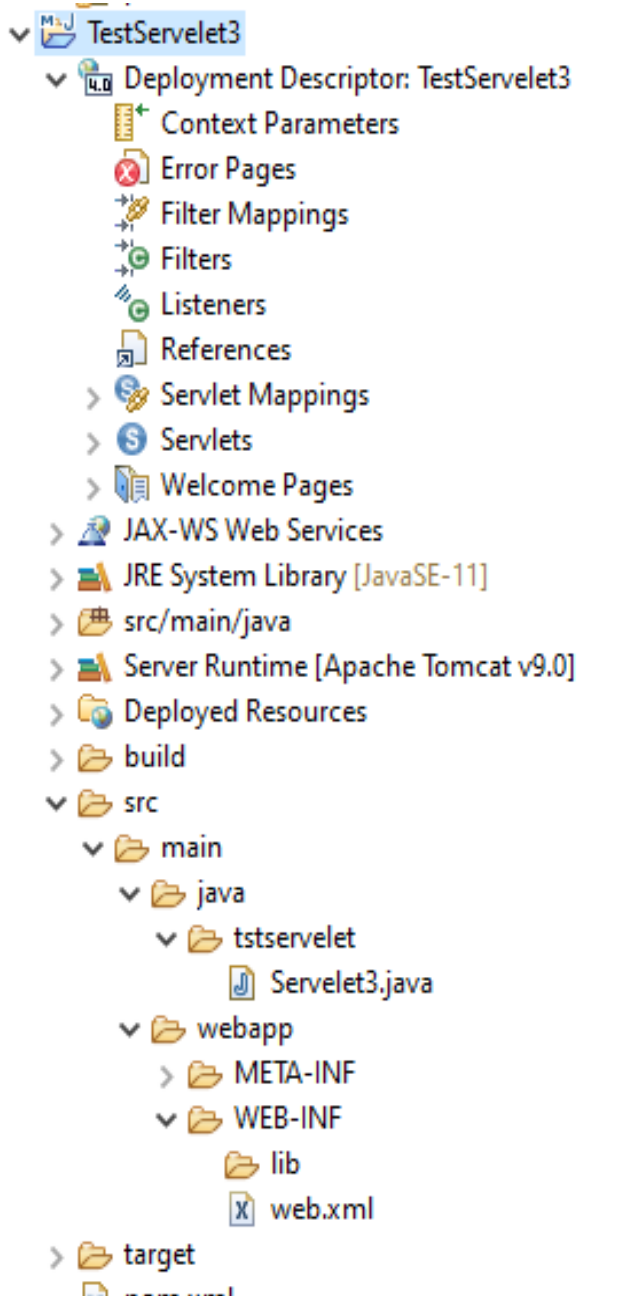
6) Creazione servelet





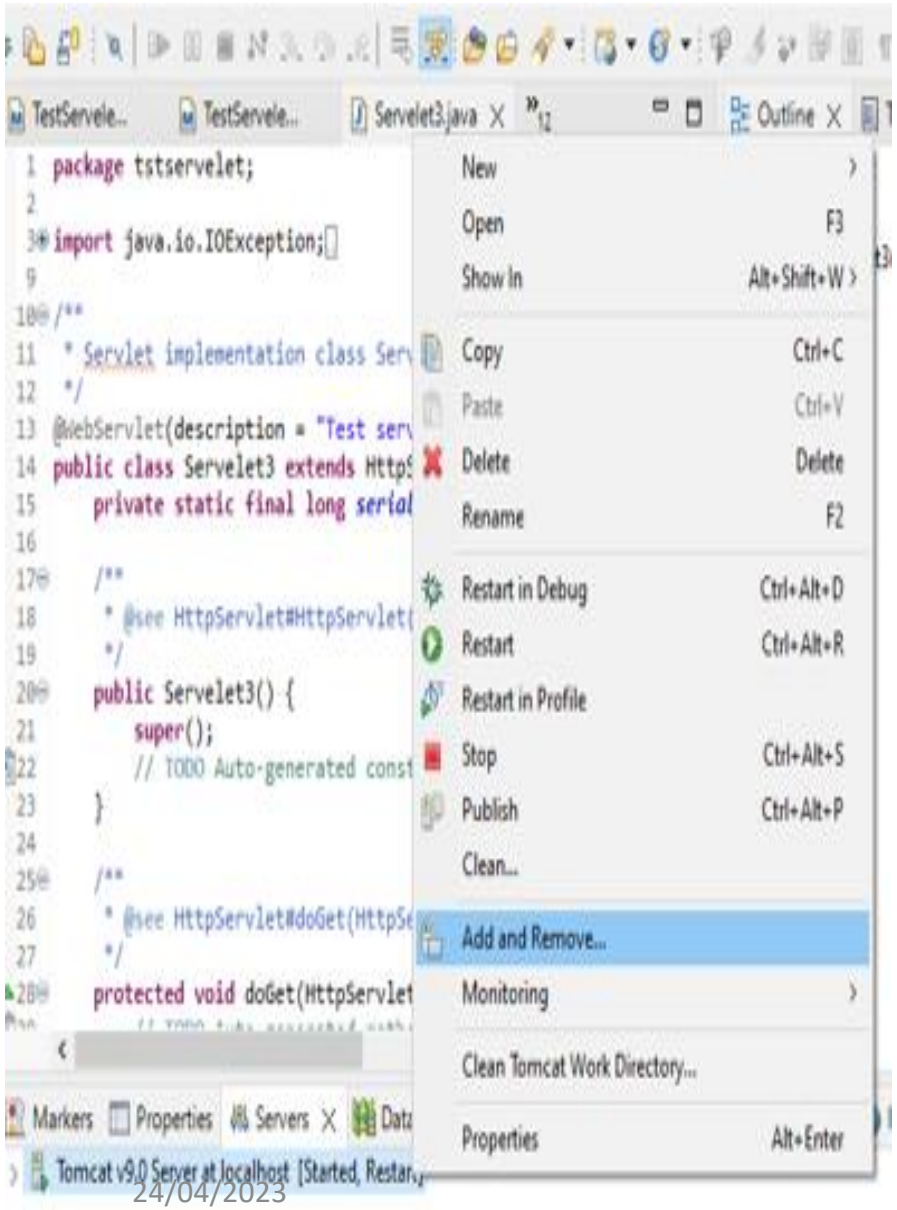

Create Servlet
Create Servlet
Enter servlet deployment descriptor specific information.
Name:
Servelet3
Description:
Test servelet
Initialization parameters:
Name
Value
Description
Add...
Edit...
Remove
URL mappings:
/Servelet3
Add...
Edit...
Remove
Asynchronous Support
< Back
Next >
Finish
Cancel

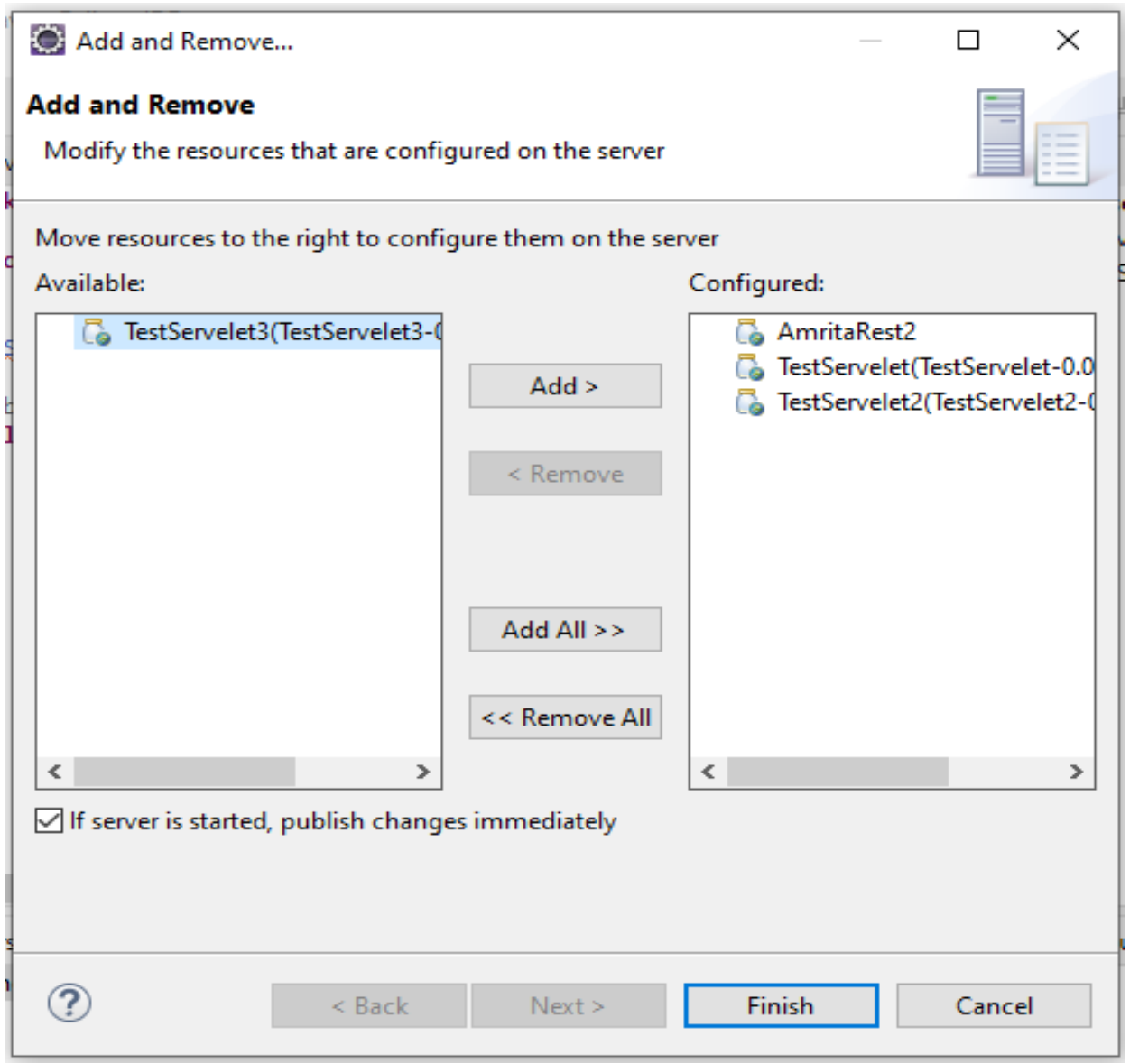

Create Servlet
Create Servlet
Specify modifiers, interfaces to implement, and method stubs to generate.
Modifiers:
public
abstract
final
Interfaces:
Add...
Remove
Which method stubs would you like to create?
Constructors from superclass
Inherited abstract methods
init
destroy
getServletConfig
getServletInfo
service
doGet
doPost
doPut
doDelete
doHead
doOptions
doTrace
< Back
Next >
Finish
Cancel




```java
package tstservelet;

import java.io.IOException;

/**
 * Servlet implementation class Servelet3
 */
@WebServlet(description = "Test servelet", urlPatterns = { "/Servelet3" })
public class Servelet3 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Servelet3() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        response.getWriter().append("Served at: ").append(request.getContextPath());
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}
```

7) Pubblicare il progetto nel Tomcat

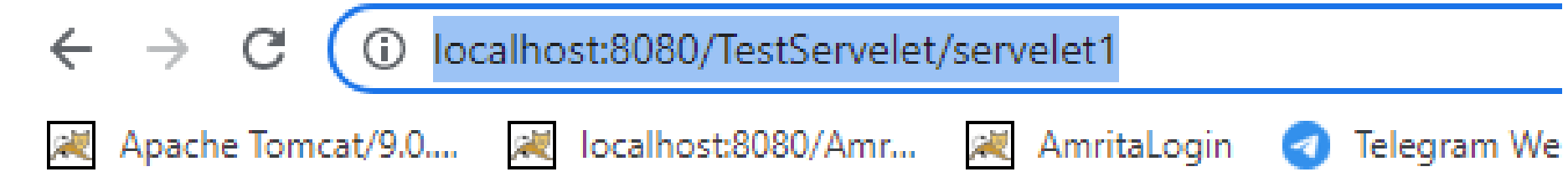




Esercizio 1 - http://localhost:8080/TestServelet/servelet1

```java
/**
 * Servlet implementation class TestServelet
 */
@WebServlet("/servelet1")
public class Servelet1 extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public Servelet1() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.print("<html><body>");
        out.print("<h2>" + "Hello Servelet Served at: " + request.getContextPath() + "</h2>");
        out.print("</body></html>");
        out.print("Served at: " + request.getContextPath());
    }

```

```
37
38      /**
39       * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
40       */
41      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
42          // TODO Auto-generated method stub
43          doGet(request, response);
44      }
45
46  }
```



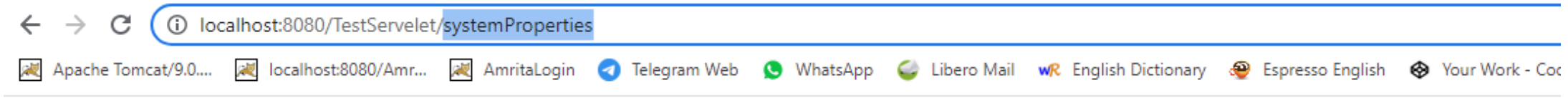
Hello Servelet Served at: /TestServelet

Served at: /TestServelet

Esercizio 2 - http://localhost:8080/TestServelet/systemProperties

```
12 import javax.servlet.http.HttpServletRequest;
13 import javax.servlet.http.HttpServletResponse;
14
15 /**
16  * Servlet implementation class TestServelet
17  */
18 @WebServlet("/systemProperties")
19 public class SystemProperties extends HttpServlet {
20     private static final long serialVersionUID = 1L;
21
22     /**
23      * @see HttpServlet#HttpServlet()
24      */
25     public SystemProperties() {
26         super();
27         // TODO Auto-generated constructor stub
28     }
29
30     /**
31      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
32      */
33     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
34         String html = "";
35         Properties prp = System.getProperties();
36         Set<Entry<Object, Object>> prpEntrySet = prp.entrySet();
37         response.setContentType("text/html");
38         PrintWriter out = response.getWriter();
39         html = "<h1>" + "System properties" + request.getContextPath() + "</h1>";
40         html+= "<table>";
41         html+= "<tr><th align='left'>Key</th><th align='left'>Valore</th></tr>";
```

```
42          for (Entry<Object, Object> prpEntry : prpEntrySet) {
43              html+="<tr><td>" + prpEntry.getKey() + "</td>";
44              html+="<td>" + prpEntry.getValue() + "</td></tr>";
45          }
46          html+="</table>";
47          html+="</body></html>";
48          out.append(html);
49      }
50
51      /**
52       * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
53       */
54      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
55          // TODO Auto-generated method stub
56          doGet(request, response);
57      }
58
59  }
60
```



System properties/TestServelet

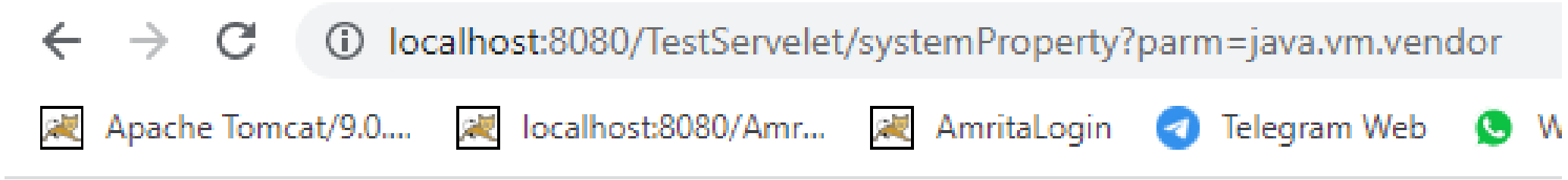
Key	Valore
sun.desktop	windows
awt.toolkit	sun.awt.windows.WToolkit
java.specification.version	11
sun.cpu.isalist	amd64
sun.jnu.encoding	Cp1252
java.class.path	C:\Program Files (x86)\Apache Software Foundation\Tomcat 9.0_Tomcat9\bin\bootst
java.vm.vendor	Oracle Corporation
sun.arch.data.model	64
user.variant	
java.vendor.url	http://java.oracle.com/
catalina.useNaming	true
user.timezone	Europe/Berlin
os.name	Windows 10
java.vm.specification.version	11
sun.java.launcher	SUN_STANDARD
user.country	IT
sun.boot.library.path	C:\Program Files\Java\jdk-11.0.2\bin
sun.java.command	org.apache.catalina.startup.Bootstrap start

Esercizio 3 - http://localhost:8080/TestServelet/systemProperty?parm=java.vm.vendor

```
19 @WebServlet("/systemProperty")
20 public class SystemProperty extends HttpServlet {
21     private static final long serialVersionUID = 1L;
22 
23     /**
24      * @see HttpServlet#HttpServlet()
25      */
26     public SystemProperty() {
27         super();
28         // TODO Auto-generated constructor stub
29     }
30 
31     /**
32      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
33      */
34     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
35         String html = "";
36         String parmName = "";
37         String parmValue = "";
38 
39         response.setContentType("text/html");
40         PrintWriter out = response.getWriter();
41 
42         Properties prp = System.getProperties();
43         Set<Entry<Object, Object>> prpEntrySet = prp.entrySet();
44 
```

```
44
45          // Recupero parametro (il primo)
46          Enumeration<String> parmNames = request.getParameterNames();
47          while (parmNames.hasMoreElements()) {
48             parmName = (String) parmNames.nextElement();
49             break;
50          }
51
52          // Deve avere nome "parm"
53          if (parmName.equals("")) {
54              out.append("<h2>Richiesta effettuata con parametro errato</h2>");
55              return;
56          }
57          // Recupero valore parametro (key in System.properties)
58          parmValue = request.getParameter("parm");
59          if (parmValue == null) {
60              out.append("<h2> Valore parametromalformato</h2>");
61              return;
62          }
63
64          // Header html tabella
65          html = "<h1>" + "System property" + request.getContextPath() + "</h1>";
66          html+= "<table>";
67          html+= "<tr><th align='left'>Key</th><th align='left'>Valore</th></tr>";
68
```

```
68
69          // Verifica se valore parametro definito in System.properties come key
70          // Se si inserisco la riga in tabella
71          for (Entry<Object, Object> prpEntry : prpEntrySet) {
72              if (prpEntry.getKey().equals(parmValue)) {
73                  html+="<tr><td>" + parmValue + "</td>";
74                  html+="<td>" + prpEntry.getValue() + "</td></tr>";
75                  break;
76              }
77          }
78
79          // Footer html tabella
80          html+="</table>";
81          html+="</body></html>";
82          out.append(html);
83      }
84
85      /**
86       * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
87       */
88      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
89          // TODO Auto-generated method stub
90          doGet(request, response);
91      }
92
93  }
```



System property/TestServelet

Key	Valore
java.vm.vendor	Oracle Corporation

Esercizio 4 - http://localhost:8080/TestServelet/testForm.html

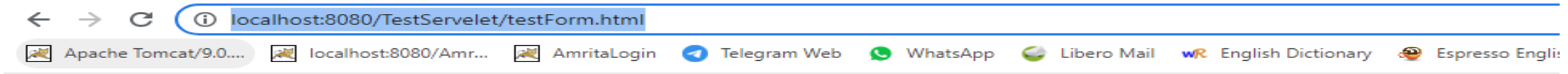
```
1  <html>
2  <title>Test Form e doPOST</title>
3  <head>
4  <meta charset="ISO-8859-1">
5  <link rel="stylesheet" type="text/css" href="testForm.css"/>
6  <style>
7  </style>
8  </head>
9  <body onload=init()>
10 <h1>Test Servlet doPOST da Form</h1>
11 <form   name="TestForm"
12         method="post"
13         action="http://localhost:8080/TestServelet/testForm">
14
15     <br>
16       <table id='anagrafica' class='anagrafica'>
17         <tr>
18           <th>Attributo</th><th>Valore</th>
19         </tr>
20         <tr>
21           <td> <b>Id Anagrafica</b> </td>
22           <td class='id'>
23                 <input type="text" placeholder="Enter Id Anagrafica" id="id" name="id" required>
24           </td>
25         <tr>
26           <td> <b>Nome</b> </td>
27           <td class='data'><input type="text" placeholder="Enter nome" id="inomed" name="nome" required></td>
28         </tr>
29         <tr>
30           <td> <b>Cognome</b> </td>
31           <td class='data'><input type="text" placeholder="Enter Cognome" id="cognome" name="cognome" required></td>
```

```
        </tr>
        <tr>
          <td> <b>Istruzione</b> </td>
          <td class='data'><input type="text" placeholder="Enter istruzione" id="istruzione" name="istruzione" required>
          </td>
         </tr>
      </table>
     <br>
     <br>
      <input type="submit" value="Submit">
     <br>
     <br>
     <div id="error"></div>
</form>
<script>
"use strict";

</script>

</body>
</html>

```

```java
import java.io.IOException;

/**
 * Servlet implementation class TestServelet
 */
@WebServlet("/testForm")
public class TestForm extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public TestForm() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        String html = "";
        String parmName = "";
        String parmValue = "";
        String idValue = "";
        String nomeValue = "";
        String cognomeValue = "";
        String istruzioneValue = "";

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
```

```
45
46          Properties prp = System.getProperties();
47          Set<Entry<Object, Object>> prpEntrySet = prp.entrySet();
48
49          // Recupero valori parametri
50          Enumeration<String> parmNames = request.getParameterNames();
51          while (parmNames.hasMoreElements()) {
52              parmName = (String) parmNames.nextElement();
53              switch (parmName) {
54              case "id":
55                  idValue = request.getParameter(parmName);
56                  break;
57              case "nome":
58                  nomeValue = request.getParameter(parmName);
59                  break;
60              case "cognome":
61                  cognomeValue = request.getParameter(parmName);
62                  break;
63              case "istruzione":
64                  istruzioneValue = request.getParameter(parmName);
65                  break;
66              default:
67                  break;
68              }
69          }
70
71          // Header html tabella
72          html = "<br>";
73          html+= "<h1>" + "Valori recuperati dal form a fronte di POST" + "</h1>";
74          html+= "<table>";
75          html+= "<tr><th align='left'>Key</th><th align='left'>Valore</th></tr>";
```

```
76
77          // Verifica se valore parametro definito in System.properties come key
78          // Se si inserisco la riga in tabella
79          html+="<tr><td>" + "Id" + "</td>";
80          html+="<td>" + idValue + "</td></tr>";
81          html+="<tr><td>" + "Nome" + "</td>";
82          html+="<td>" + nomeValue + "</td></tr>";
83          html+="<tr><td>" + "Cognome" + "</td>";
84          html+="<td>" + cognomeValue + "</td></tr>";
85          html+="<tr><td>" + "Cognome" + "</td>";
86          html+="<td>" + istruzioneValue + "</td></tr>";
87
88          // Footer html tabella
89          html+="</table>";
90          html+="</body></html>";
91          out.append(html);
92      }
93
94      /**
95       * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
96       */
97      protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
98          // TODO Auto-generated method stub
99          doGet(request, response);
100     }
101
102 }
103
```



Test Servlet doPOST da Form

Attributo	Valore
Id Anagrafica	Enter Id Anagrafica
Nome	Enter nome
Cognome	Enter Cognome
Istruzione	Enter istruzione

Submit

Test Servlet doPOST da Form

Attributo	Valore
Id Anagrafica	ID01
Nome	CARLO
Cognome	ROSSI
Istruzione	LAUREA

Submit

Valori recuperati dal form a fronte di POST

Key	Valore
Id	ID01
Nome	CARLO
Cognome	ROSSI
Cognome	LAUREA

Esercizio 5 - http://localhost:8080/TestServelet/counter

```java
@WebServlet("/counter")
public class CounterAccess extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private volatile int counter = 0;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public CounterAccess() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h2>" + "Hello Servelet Served at: " + request.getContextPath() + "</h2>");
        out.println("Contatore Accessi: " + counter++);
        out.println("<br>");
        out.println("<br>");
        out.println("Served at: " + request.getContextPath());
        out.println("</body></html>");
    }
```

```
    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }

}
```

- La classe servlet CounterAccess viene istanziata dal container (tomcat) alla **partenza**
- Ad ogni richiesta HTTP del client il container **crea un thread** sull'oggetto counterAccess
- Il codice e i dati di CounterAccess sono **condivisi** da tutti i thread
- **counter** è stato definito volatile per garantire che il suo valore sia corretto fra un thread e un altro

Introduzione ai Web Services

Obbiettivo è quello di sfruttare la Rete per far colloquiare software e sistemi diversi tra loro grazie alla standardizzazione di:

URI – meccanismo per indirizzare risorse di rete
HTTP – Protocollo semplice e leggero per richiedere una risorsa ad una macchina
HTML – Linguaggio per la rappresentazione dei contenuti

Per evolversi nella direzione di una piattaforma applicativa distribuita.

Su questi concetti intorno all'anno 2000 nascono i web service:
un sistema software progettato a supporto l'interazione trasparente tra applicazioni, utilizzando le tecnologie standard del web.

Applicazioni sviluppate con linguaggi diversi, che girano su sistemi operativi eterogenei, potenzialmente incompatibili.

Web Services SOAP e REST

Esistono due approcci alla creazione di un web service

- Un primo approccio è basato sul protocollo standard **SOAP** (**Simple Object Access Protocol**), per lo scambio di messaggi per l'invocazione di servizi remoti, si prefigge di riprodurre in ambito Web un approccio a chiamate remote, Remote Procedure Call.
- Un secondo approccio è ispirato ai principi architetturali tipici del Web basato sulla descrizione di risorse, sul modo di individuarle nel Web e sul modo di trasferirle da una macchina all'altra.
  Questo è l'approccio più utilizzato e prende il nome di **REST** (**REpresentational State Transfer**) per consumare e implementare servizi **RESTful**.

Web Services REST – Principi architetturali

- **REST** definisce uno stile architetturale per la progettazione di un sistema, non si riferisce a un sistema concreto e ben definito e non è uno standard stabilito da un organismo internazionale.
- I principi di **REST** non sono necessariamente legati al Web, nel senso che si tratta di principi astratti di cui il World Wide Web ne risulta essere un esempio concreto.
- Il Web ha tutto quello che serve per essere considerata una piattaforma di elaborazione distribuita secondo i principi REST, non sono necessarie altre sovrastrutture per realizzare quello che è il Web programmabile.
- A differenza di **SOAP** che necessita di una sovrastruttura di definizione del servizio (**WSDL**)

Web Services REST – Principi architetturali 2

L'approccio REST può tradursi in 5 principi, del resto insiti nella struttura stessa del web:

1. Identificazione delle risorse (in SOAP ci sono le chiamate remote)
2. Utilizzo esplicito dei metodi HTTP
3. Risorse autodescrittive
4. Collegamenti tra risorse
5. Comunicazioni senza stato

Web Services REST – Principi architetturali 3

Identificazione delle risorse :

- Per risorsa si intende un qualsiasi elemento oggetto di elaborazione. Per esempio concreto una risorsa può essere un cliente, un libro, un articolo, un qualsiasi oggetto su cui è possibile effettuare operazioni.
- Ogni risorsa deve essere identificata univocamente

Esempi di possibili identificatori di risorse

  - http://www.myapp.com/clienti/1234
  - http://www.myapp.com/ordini/2011/98765
  - http://www.myapp.com/prodotti/7654
  - http://www.myapp.com/ordini/2011
  - http://www.myapp.com/prodotti?colore=rosso

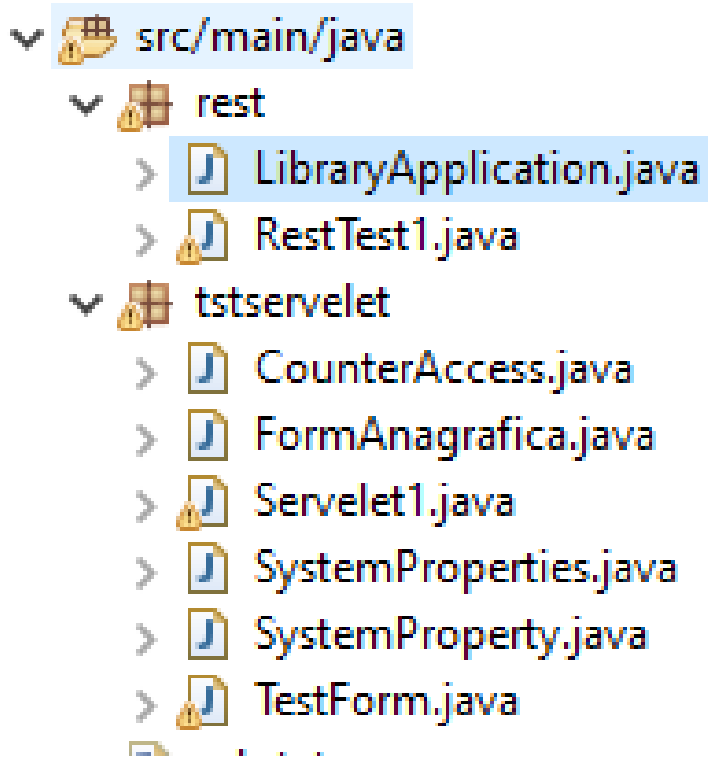
- Nel progettare un Web Service in modalità REST è utile evitare l'uso di verbi negli URI ma limitarsi ad utilizzare nomi, ricordandosi che un URI identifica una risorsa.

Web Services REST – Mappare operazioni CRUD

Metodo HTTP	Operazione CRUD	Descrizione
POST	Create	Crea una nuova risorsa
GET	Read	Ottiene una risorsa esistente
PUT	Update	Aggiorna una risorsa o ne modifica lo stato
DELETE	Delete	Elimina una risorsa

Web Services REST con Eclipse e Tomcat

Definire un package specifico che contiene le classi che implementano i web service (norma di buona programmazione)



Web Services REST con Eclipse e Tomcat 2

- Dichiarare il path del package con i web services relativo all'applicazione attraverso l'annotation `@ApplicationPath('rest')`
- L'annotation può essere inserita in una classe con un nome qualsiasi

```
package rest;

import javax.ws.rs.ApplicationPath;

//import org.glassfish.jersey.server.ResourceConfig;

@ApplicationPath("rest")
public class LibraryApplication extends Application {
}
```

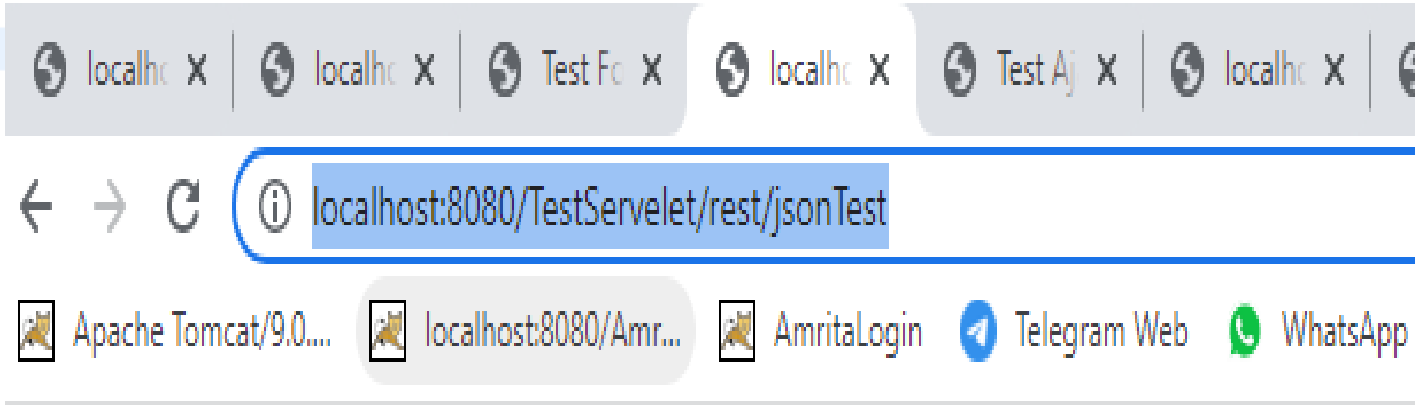
Web Services REST con Eclipse e Tomcat 3

Il path del package può essere configurato anche nel file web.xml

```
<servlet>
    <servlet-name>AmritaRest2</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>rest</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>AmritaRest2</servlet-name>
    <url-pattern>/rest/*</url-pattern>
  </servlet-mapping>
```

Esercizio 6 - http://localhost:8080/TestServelet/rest/jsonTest

```java
@Path("/")
public class RestTest1 {

    @GET
    @Produces("application/json")
    @Path("jsonTest")
    public Response getJson() {

        String resultJson="";

        JSONObject jsonObject = null;
        JSONArray jsonArray = new JSONArray();

        jsonObject = new JSONObject();
        jsonObject.put("id11", "Value11");
        jsonObject.put("id12", "Value12");
        jsonArray.put(jsonObject);
        jsonObject = new JSONObject();
        jsonObject.append("id2X", "Value21");
        jsonObject.append("id2X", "Value22");
        jsonArray.put(jsonObject);
        resultJson = jsonArray.toString();

        return Response
                .status(200)
                .entity(resultJson)
                .build();

    }
```



[{"id11":"Value11","id12":"Value12"},{"id2X":["Value21","Value22"]}]

Esercizio 7 - http://localhost:8080/TestServelet/testAjax.html

testAjax.html | TestServele... | SystemPrope... | RestTest1.java | urls.txt | ViewerRelati... | Inspector.html | RestImpactS... | Imp

```
1  <html>
2  <title>Test Ajax 1</title>
3  <head>
4  <meta charset="ISO-8859-1">
5  <link rel="stylesheet" type="text/css" href="testAjax.css"/>
6  <style>
7  </style>
8  </head>
9  <body onload=init()>
10
11 <h1>Test Ajax 1</h1>
12 <br>
13 <label for="id"><b>Id Anagrafica</b></label>
14 <input type="text" placeholder="Enter Id Anagrafica" id="id" name="id" tabindex="1" required>
15 <button type="button" onclick="onclick_callWebService()">Get info via webService</button>
16
17 <br>
18 <br>
19
20 <div id="anagrafica">
21
22   <table id='anagrafica' class='anagrafica'>
23     <tr>
24       <th>Id</th><th>Nome</th><th>Cognome</th><th>Istruzione</th>
25     </tr>
26     <tr>
```

```
28          <td class='data'id='nome'> </td>
29          <td class='data' id='cognome'>.</td>
30          <td class='data' id='istruzione'>.</td>
31        </tr>
32    </table>
33
34 </div>
35
36 <br>
37 <br>
38 <div id="error"></div>
39
40 <script>
41 "use strict";
42
43 var hostName = "";
44 var urlAnagraficaGET = 'http://localhost:8080/TestServelet/rest/anagrafica';
45 var url="";
46 var userInp="";
47
48
49 function init() {
50     if(document.readyState === 'loading') {
51         document.addEventListener('DOMContentLoaded', afterLoaded);
52     } else {
53         //The DOMContentLoaded event has already fired. Just run the code.
54         afterLoaded();
```

```
55      }
56
57  }
58
59  function afterLoaded() {
60      // document.getElementById('id01').style.display='block';
61      var hostName = window.location.href;
62      var arSplit=hostName.split("/AmritaLogin.html");
63      hostName=arSplit[0];
64      var baseUrl=hostName + "/rest";
65  }
66
67
68  function createRequest() {
69      var xmlhttp;
70      try {
71          xmlhttp = new XMLHttpRequest();
72      } catch (trymicrosoft) {
73          try {
74              xmlhttp = new ActiveXObject("MsXML2.XMLHTTP");
75          } catch (othermicrosoft) {
76              try {
77                  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
78              } catch (failed) {
79                  xmlhttp = null;
80              }
81          }
```

```
82      }
83      if (xmlhttp == null)
84          alert("Error creating Ajax request object!");
85      return xmlhttp;
86  }
87
88  function onclick_callWebService() {
89      var idInp = false;
90      idInp=document.getElementById("id").value;
91
92      document.getElementById("error").innerHTML = "";
93      if (idInp == "" ) {
94          document.getElementById("error").innerHTML = "Id Anagrafica non fornita";
95          document.getElementById("id").focus();
96          return;
97      }
98
99      // Ajax server request to get Objects
100     var xmlhttp;
101     xmlhttp=createRequest();
102     url = urlAnagraficaGET + "/"+ idInp
103     xmlhttp.open("GET", url, true);
104     xmlhttp.onprogress = function () {
105         if (xmlhttp.status!=200){
106            alert("Error Loading page " + url + "\n" + xmlhttp.status + "\n"
107                    + xmlhttp.responseText);
108         }
```

```
109     };
110     xmlhttp.send(null);
111     xmlhttp.onreadystatechange = function() {handleJsonResponseAnagrafica(xmlhttp);};
112 }
113
114
115 function handleJsonResponseAnagrafica(xmlhttp) {
116     var obj;
117
118     if (xmlhttp.readyState!=4 || xmlhttp.status!=200){return;}
119
120     // Jsson from webservice available
121     obj=JSON.parse(xmlhttp.responseText);
122
123     document.getElementById("id").innerHTML = obj.id;
124     document.getElementById("nome").innerHTML = obj.nome;
125     document.getElementById("cognome").innerHTML = obj.cognome;
126     document.getElementById("istruzione").innerHTML = obj.istruzione;
127
128     document.getElementById("nome").focus();
129 }
130
131
132 </script>
133
134 </body>
135 </html>
```

Esercizio 7 - http://localhost:8080/TestServelet/testAjax.html

```
@GET
@Produces("application/json")
@Path("anagrafica/{id}")
public Response getAnagrafica(@PathParam("id") String id) {

    String resultJson="";

    JSONObject jsonObject = new JSONObject();
    jsonObject.put("id", "ID01");
    jsonObject.put("nome", "Alberto");
    jsonObject.put("cognome", "Rossi");
    jsonObject.put("istruzione", "DIPLOMA");
    resultJson = jsonObject.toString();

    return Response
            .status(200)
            .entity(resultJson)
            .build();

}
```