

PARTE K

USO DI IDE NELLO SVILUPPO

J2EE: ECLIPSE

Motivazioni

- **Eclipse verrà usata come piattaforma per le esercitazioni nell'ambito del corso**
- **Si cerca di non rendere le esercitazioni dipendenti da una IDE**
- **... però siamo consapevoli della comodità di certi strumenti**

Modulo 1

Hands on eclipse

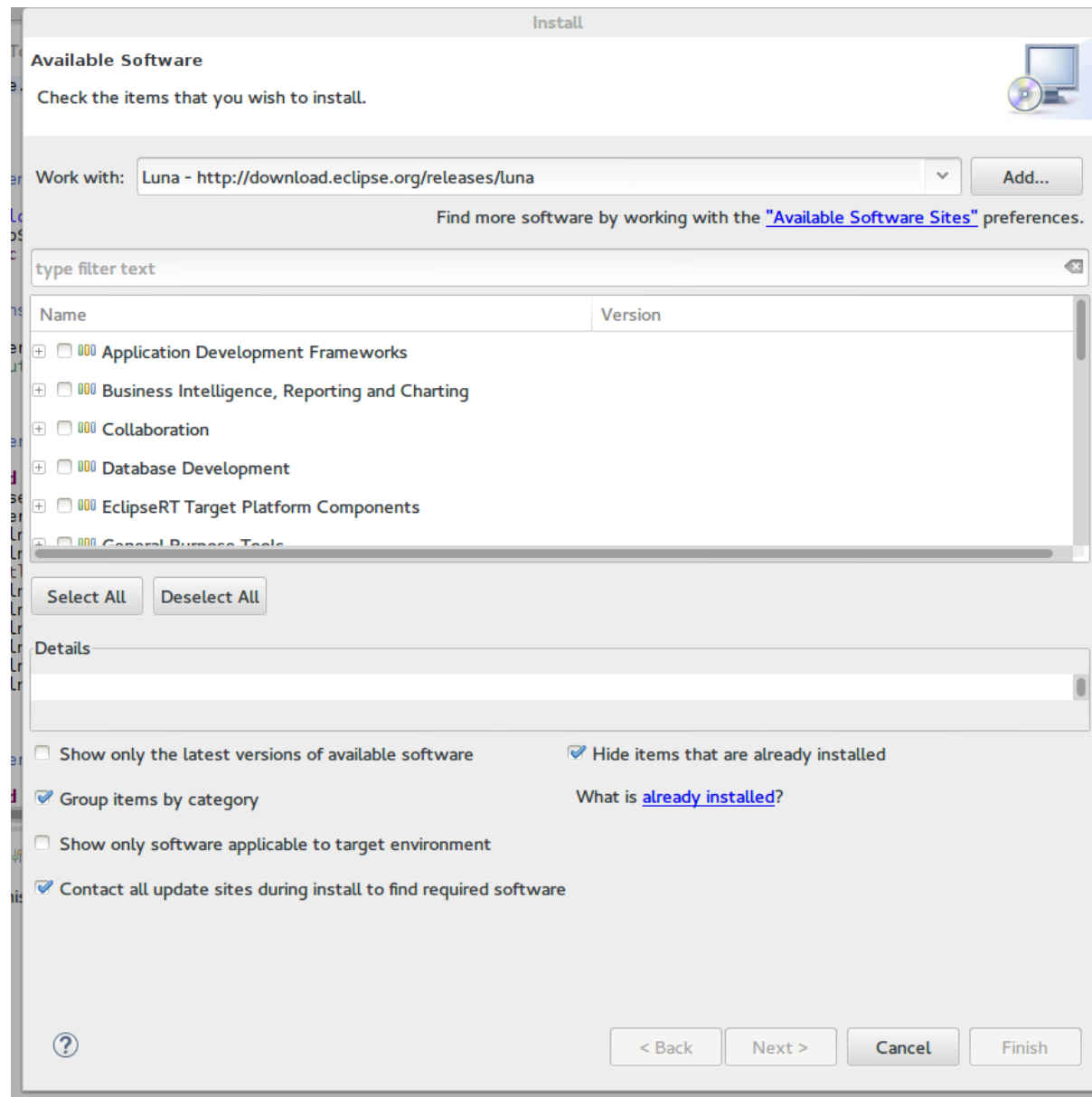
Software necessario

- **Eclipse Web Tool Platform**
- **Tomcat**

Installazione eclipse

- **Installiamo la versione di eclipse per J2EE**
 - eclipse-jee-luna-R-linux-gtk-x86_64.tar.gz
 - Anche la versioni più recenti funzionano correttamente (nei limiti della compatibilità della JVM)
 - Screenshot riferite a versione base (luna)
- **Si verificano che i plugin che servono siano installati**
 - Help → Install New software
 - Si seleziona il repository della versione corrente

Installazione nuovi plugin



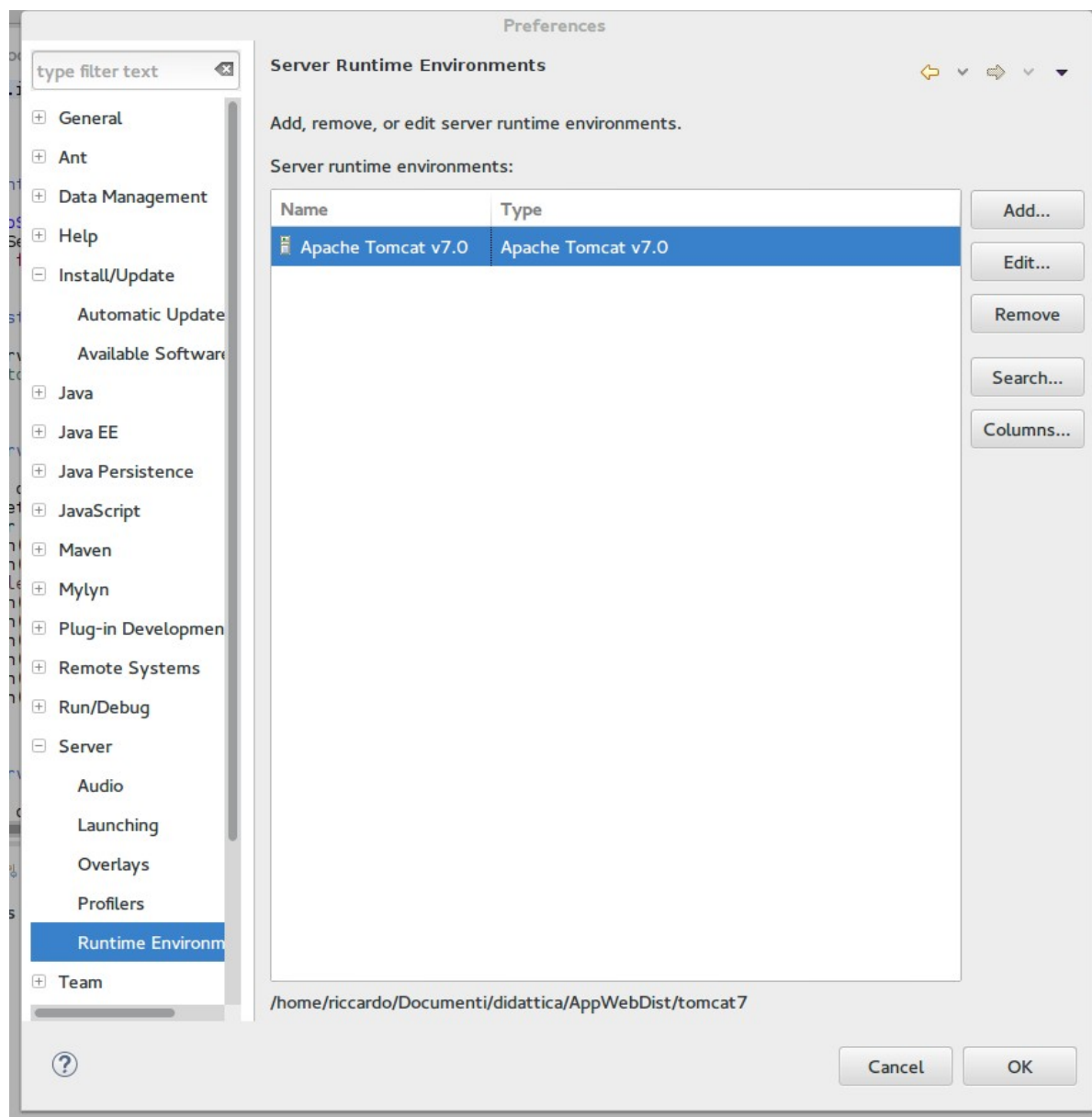
Plugin da installare

- **Pacchetti necessari:**
 - Eclipse Java EE developer tools
 - Eclipse Java Web developer tools
 - Eclipse Web developer tools
- **Nelle ultime versioni di Eclipse dovrebbero essere già inclusi**

Installare tomcat

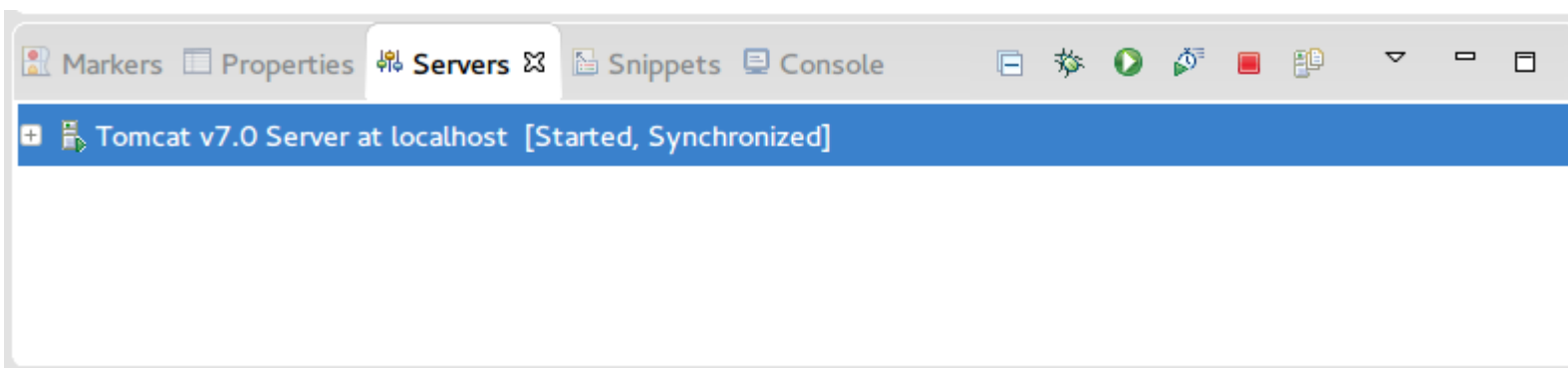
- **Accedere alla configurazione dei runtime environment**
 - Window → Preferences → Server → Runtime environments
- **Creare un nuovo server di tipo Tomcat 7.0**
 - Se si usa la versioni 8, 8.5 o 9 di tomcat specificare il tipo corretto

Installare Tomcat



Installare tomcat

- **Ora il server si può controllare dalla lista dei server nella IDE**



Ultima nota

- **Tomcat funziona meglio se si prende possesso dell'installazione remota del server**
- **Per questo bisogna aprire il server nelle finestre di editing**

Ultima nota

Overview

General Information

Specify the host name and other common settings.

Server name:

Host name:


Runtime Environment:

Configuration path:

[Open launch configuration](#)

Server Locations

Specify the server path (i.e. catalina.base) and deploy path. Server must be published with no modules present to make changes.

- Use workspace metadata (does not modify Tomcat installation)
- Use Tomcat installation (takes control of Tomcat installation) 
- Use custom location (does not modify Tomcat installation)

Server path:

[Set deploy path to the default value \(currently set\)](#)

Deploy path:

Publishing

Timeouts

Ports

Modify the server ports.

Port Name	Port Number
Tomcat admin port	8005
HTTP/1.1	8080
AJP/1.3	8009

MIME Mappings

Overview **Modules**

Modulo 2

Una servlet di esempio

Creazione di un nuovo progetto

- **Dal menu:**

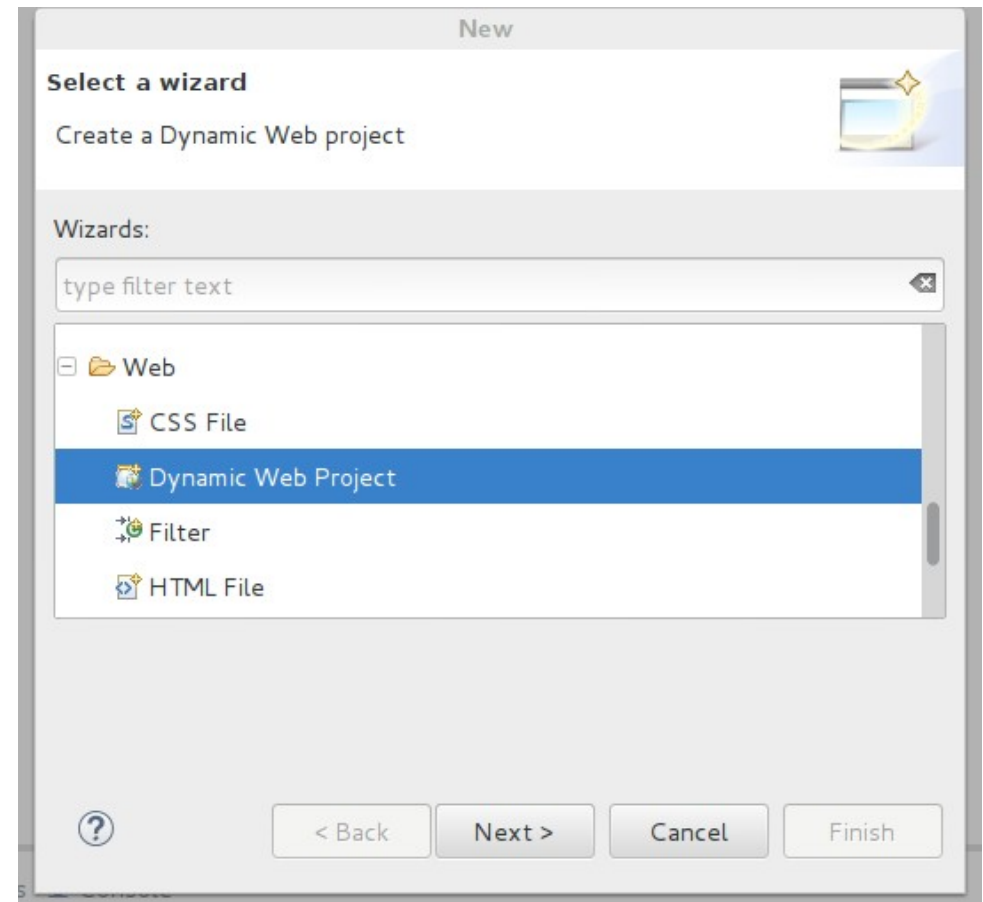
File →

New →

Other...→

Web →

Dynamic Web Project



Creazione di un nuovo progetto

New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: FirstServlet

Project location

Use default location

Location: /home/riccardo/Documents/didattica/AppWebDist/

Target runtime

Apache Tomcat v7.0

Dynamic web module version

3.0

Configuration

Default Configuration for Apache Tomcat v7.0

A good starting point for working with Apache Tomcat v7.0 runtime. Additional facets can later be installed to add new functionality to the project.

EAR membership

Add project to an EAR

EAR project name: EARExample

Working sets

Add project to working sets

Working sets:

New Dynamic Web Project

Web Module

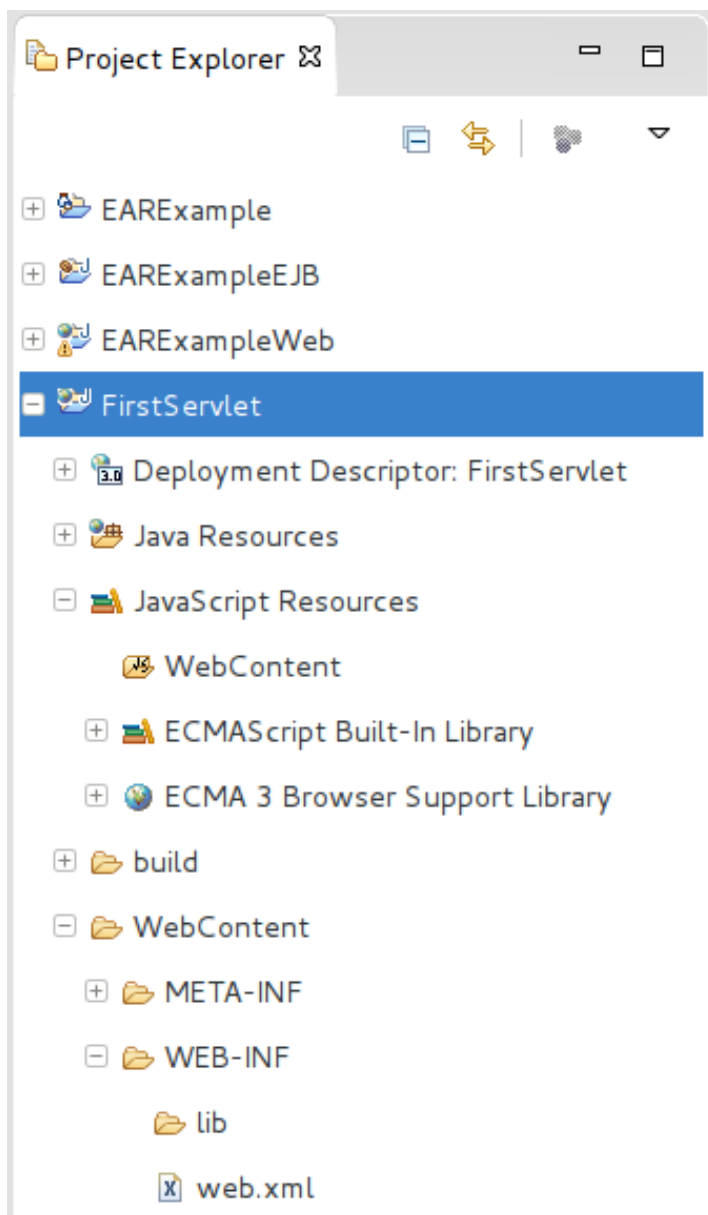
Configure web module settings.

Context root: FirstServlet

Content directory: WebContent

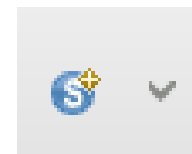
Generate web.xml deployment descriptor

Risultato finale



- **Tutto è pronto per creare una nuova servlet nel progetto appena creato**

- **Per creare la nuova servlet:**



Creazione della servlet

Create Servlet

Specify class file destination.

Project: FirstServlet

Source folder: /FirstServlet/src

Java package: it.unimore.ing.weblab.firstServlet

Class name: FirstServlet

Superclass: javax.servlet.http.HttpServlet

Use an existing Servlet class or JSP

Class name: FirstServlet

Create Servlet

Enter servlet deployment descriptor specific information.

Name: FirstServlet

Description:

Initialization parameters:

Name	Value	Description
------	-------	-------------

URL mappings:

/FirstServlet

Asynchronous Support

Creazione della servlet

Create Servlet

Specify modifiers, interfaces to implement, and method stubs to generate.

Modifiers: public abstract final

Interfaces:

Which method stubs would you like to create?

Constructors from superclass
 Inherited abstract methods

<input type="checkbox"/> init	<input type="checkbox"/> destroy	<input type="checkbox"/> getServletConfig
<input type="checkbox"/> getServletInfo	<input type="checkbox"/> service	<input checked="" type="checkbox"/> doGet
<input checked="" type="checkbox"/> doPost	<input type="checkbox"/> doPut	<input type="checkbox"/> doDelete
<input type="checkbox"/> doHead	<input type="checkbox"/> doOptions	<input type="checkbox"/> doTrace

- **A questo punto la classe è stata creata**
- **E' pronto lo scheletro della classe**
- **Bisogna solo riempire i metodi con il codice**

Esempio di servlet

```
package it.unimore.ing.weblab.firstServlet;  
  
import java.io.IOException;  
import java.io.PrintWriter;  
  
import javax.servlet.ServletException;  
import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;
```

Esempio di servlet

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

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

Esempio di servlet

```
/**
 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
 */
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
    ServletException, IOException {
// TODO Auto-generated method stub
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head>");
    String title = "Hello world";
    out.println("<title>" + title + "</title>");
    out.println("</head>");
    out.println("<body bgcolor=\"white\">");
    out.println("<h1>" + title + "</h1>");
    out.println("</body>");
    out.println("</html>");
}
```

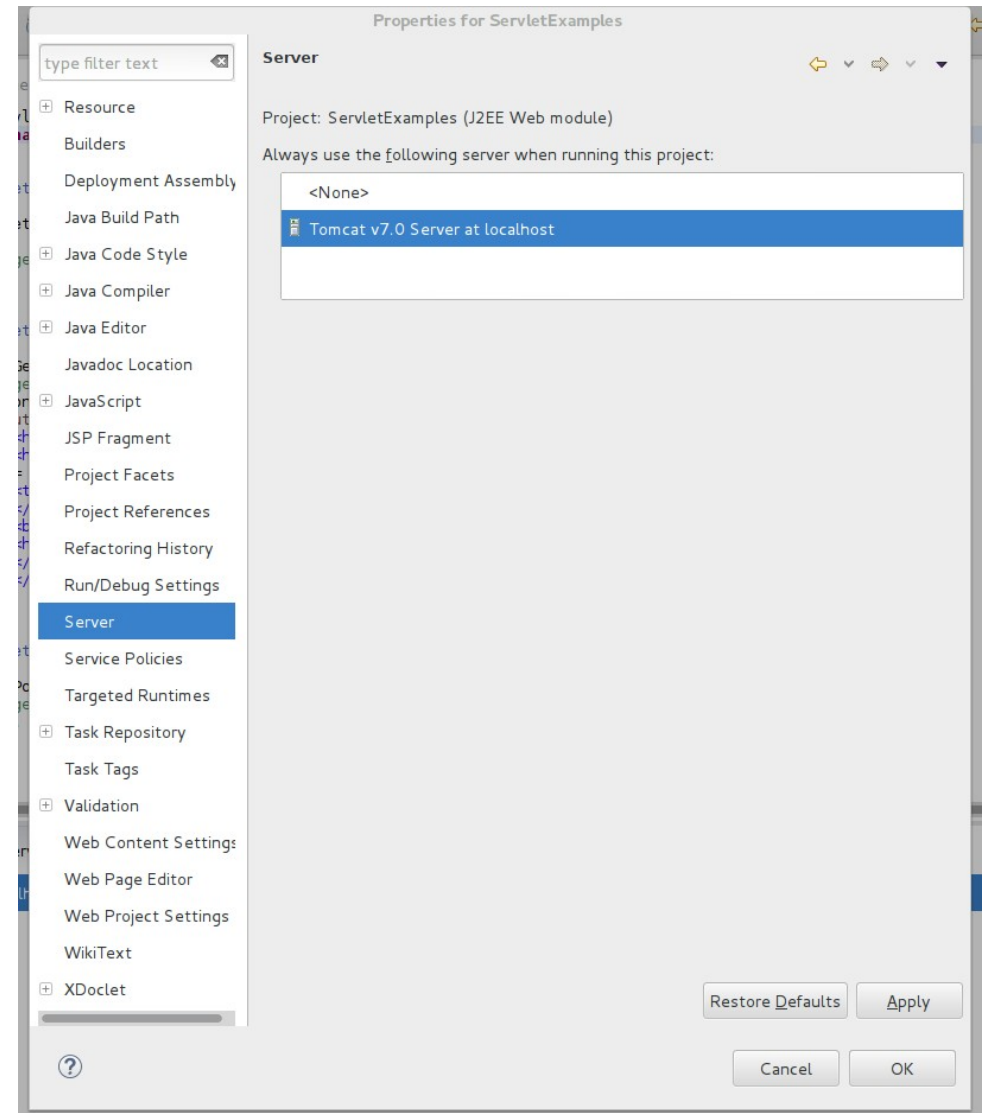
Esempio di servlet

```
/**
 * @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);
}

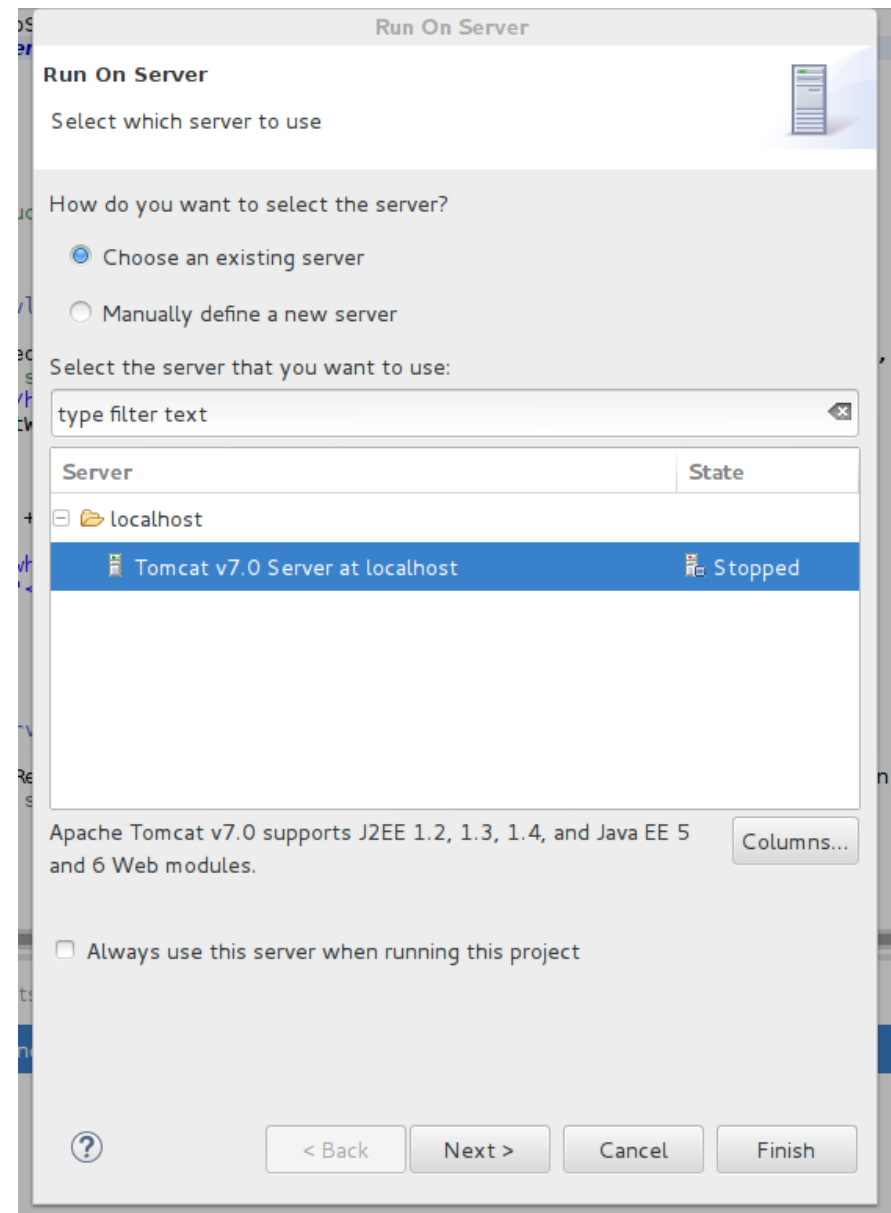
}
```

Deployment della nuova servlet

- **Ci assicuriamo che il server di default per il nostro progetto sia Tomcat**
- **Project → properties → server**



- **Lanciamo al servlet
Run →
run (Ctrl + F11)**

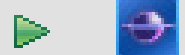


FirstServlet.java

Hello world



http://localhost:8080/FirstServlet/FirstServlet



Hello world

Modulo 3

Esercizi aggiuntivi

Servlet Data

- **La servlet deve mostrare data e ora attuali**
 - Si veda `java.util.Date`
- **Il valore dell'output deve essere aggiornato ogni secondo**
 - Bisogna forzare un reload della pagina
- **Suggerimento:**
 - Cosa fa il seguente tag nella sezione `<head>` della pagina?
`<meta http-equiv="refresh" content="1">`

Servlet Calcolatrice

- **La servlet deve eseguire le 4 operazioni di base (+, -, *, /) su due operandi**
- **Gli operandi devono essere inviati in modo interattivo dall'utente mediante un form**

Servlet Calcolatrice

- **Leggere informazioni dalla richiesta:**
 - `String url = request.getRequestURL().toString();`
 - `String a=request.getParameter("a");`
 - `String b=request.getParameter("b");`
 - `String op=request.getParameter("operazione");`
- **Per la creazione dei form vogliamo che i valori siano memorizzati tra un'invocazione e l'altra della servlet**
 - `out.println("<form method=GET
action=\""+url+"\">");`
 - `out.print("<input type=text name=a");`
 - `if (a != null){out.print(" value=\""+a+"\"");}`
 - `out.println(" />");`

Servlet Sessione utente

- **Creare una servlet che gestisce un contatore usando la sessione utente**
 - Il contatore deve essere conservato come variabile di sessione
 - Si deve mostrare il valore del contatore e fornire informazioni all'utente sullo stato della sessione
 - Si deve consentire all'utente di incrementare e decrementare il contatore

Servlet Session utente

- **Per recuperare un riferimento alla sessione:**
 - `HttpSession session = request.getSession(true);`
- **Per recuperare un valore dalla sessione:**
 - `int c;`
`c=(int)session.getAttribute("Counter");`
- **Per salvare un valore nella sessione:**
 - `session.setAttribute("Counter", c);`