

Esercizi svolti - 8

Exercise 8.1

Write a HTTP servlet, named Hello. When it is contacted, it returns “Hello Servlet World” to the browser.

Exercise 8.2

Write a HTTP servlet, named Echo. It receives some parameters (firstname=<nome> & lastname=<cognome>) from a GET request and sends the values to the browser.

Exercise 8.3

Write a HTTP servlet, named Calculate. It receives two operands and an operator (+,-,*,/), by using a form calculator.html, and returns the result of the operation to the browser.

Exercise 8.4

Write a HTTP servlet that returns to the browser the number of times *doGet()* has been invoked.

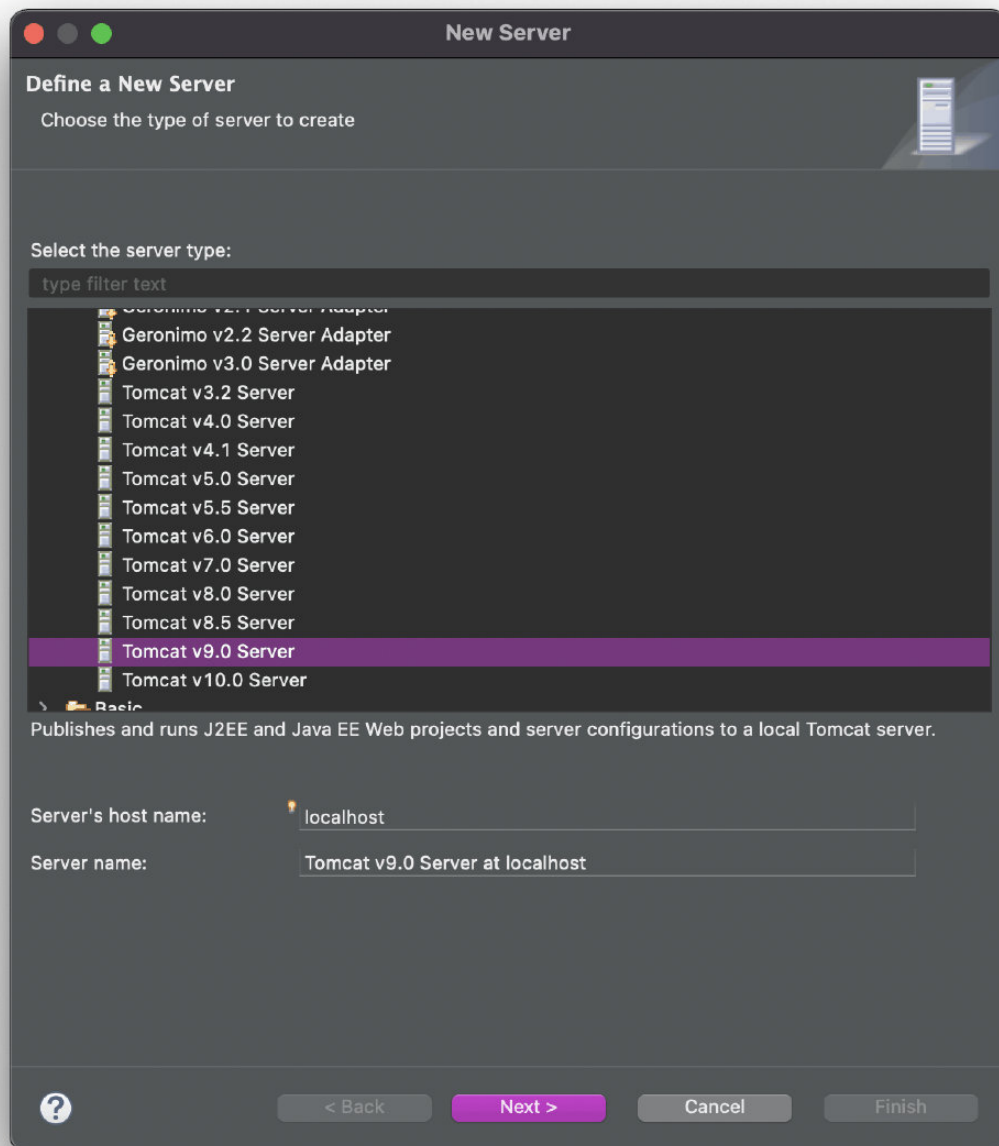
Exercise 8.5

Write a HTTP servlet that writes on a file (a parameter per line) the content of a registration form registration.html and sends a message with the body containing the string “<name>, your registration has been executed correctly”, where <name> is the value of parameter firstname in the form.

Exercise 8.6

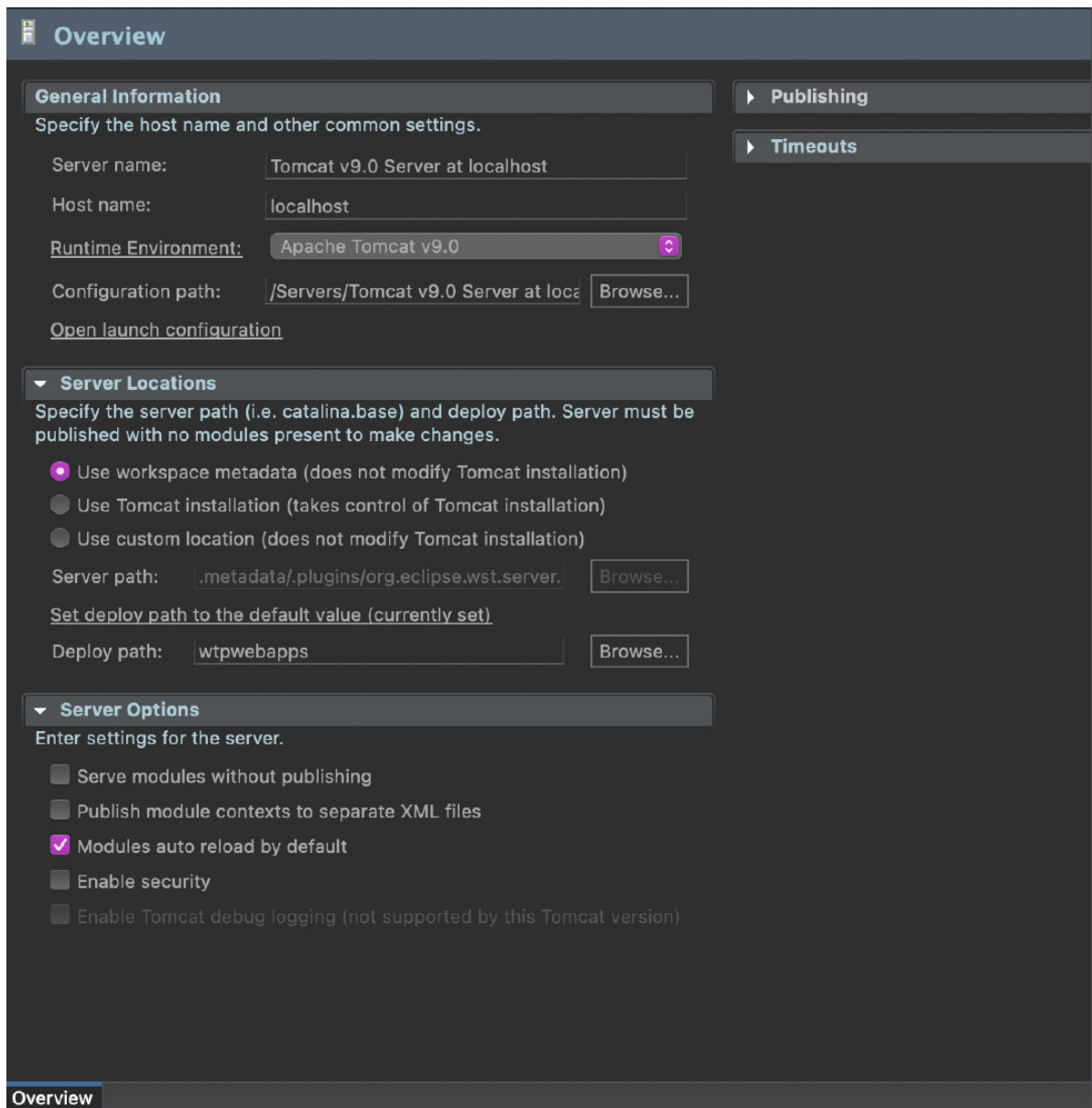
Write a HTTP servlet to compare the users credentials submitted through a form (login.html) with the registration information saved with exercise 8.5 and returns a different message according to the authentication result.

Configurazione Eclipse

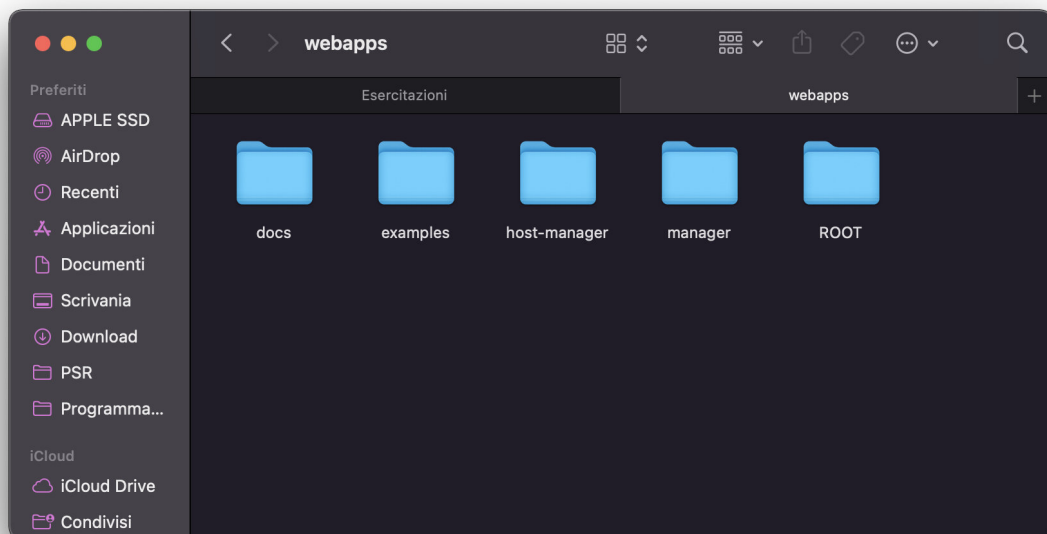


Selezioniamo il server tomcat 9.0 come server.

Una volta aggiunto il server possiamo iniziare a configurare il server cliccando due volte sul server stesso:

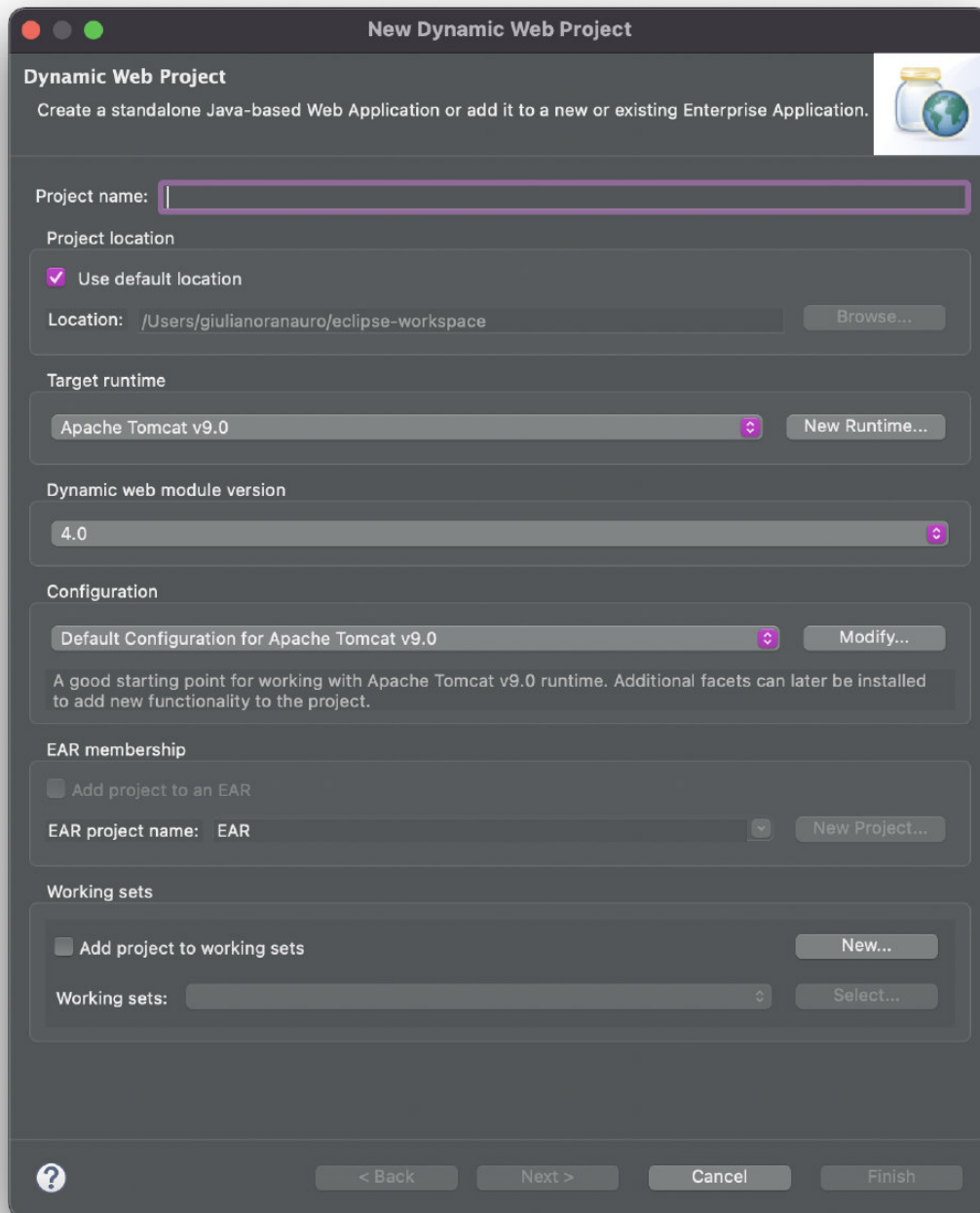


Ogni volta che dispiegheremo un'applicazione web, verrà creata una nuova directory nella directory **webapp**:



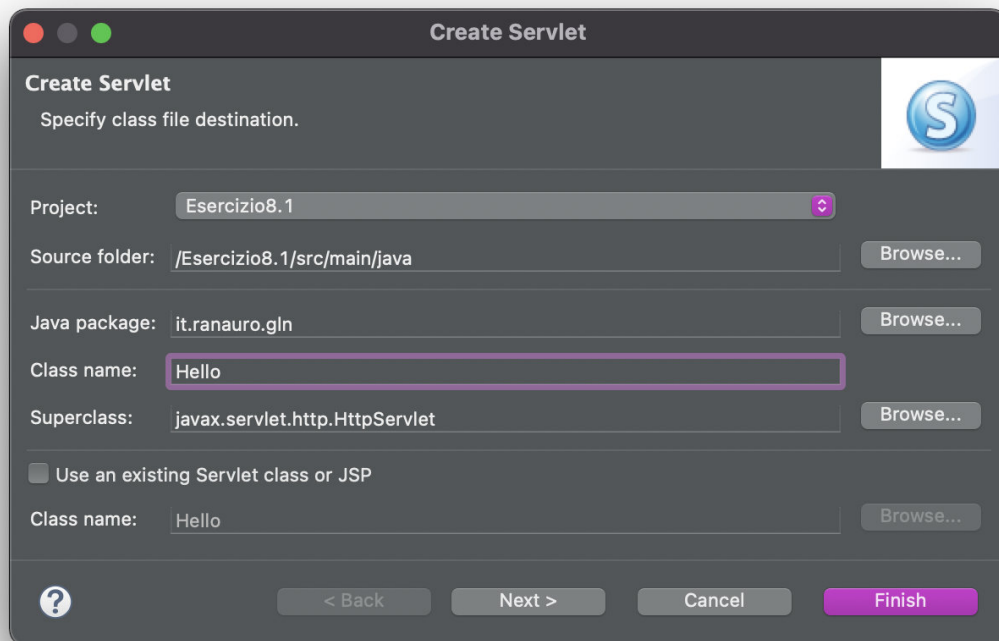
Creazione di progetto con eclipse

Creiamo un nuovo progetto dinamico:

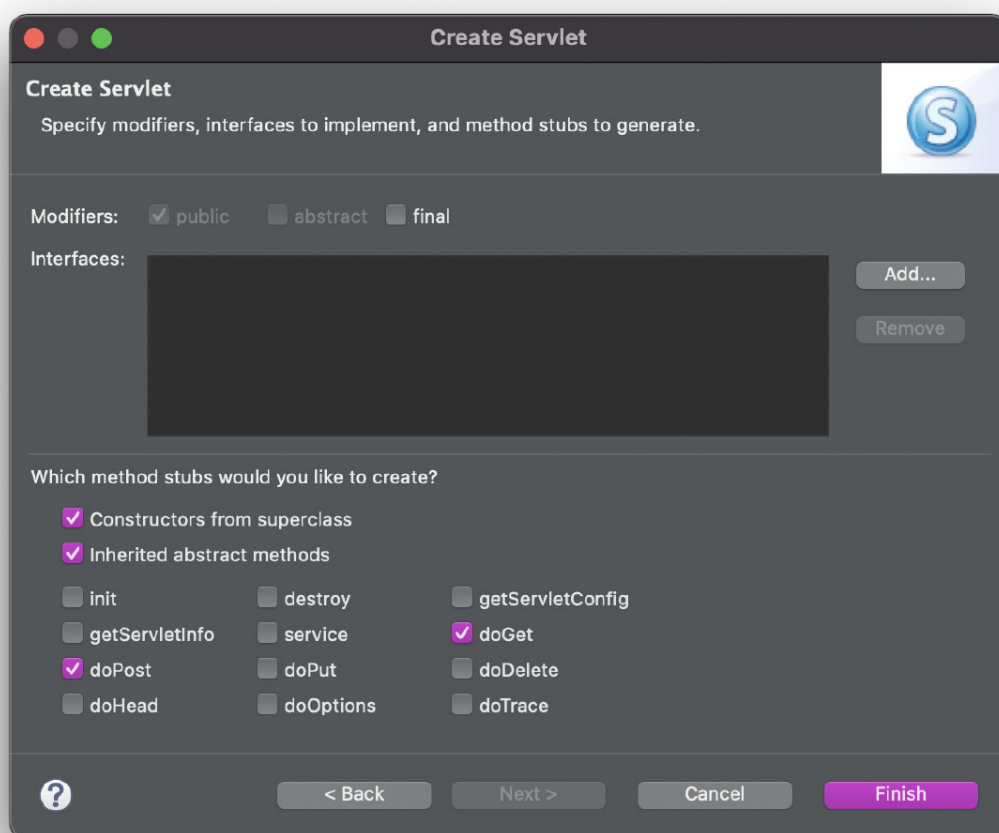


The screenshot shows the 'New Dynamic Web Project' dialog box in Eclipse. The title bar reads 'New Dynamic Web Project'. Below the title, there's a subtitle 'Dynamic Web Project' and a description: 'Create a standalone Java-based Web Application or add it to a new or existing Enterprise Application.' with a small icon of a jar and a globe. The dialog is divided into several sections: 'Project name:' with a text input field; 'Project location' with a checked option 'Use default location' and a 'Location:' field showing '/Users/giulianoranauro/eclipse-workspace' and a 'Browse...' button; 'Target runtime' with a dropdown menu set to 'Apache Tomcat v9.0' and a 'New Runtime...' button; 'Dynamic web module version' with a dropdown menu set to '4.0'; 'Configuration' with a dropdown menu set to 'Default Configuration for Apache Tomcat v9.0' and a 'Modify...' button, followed by a descriptive text; 'EAR membership' with an unchecked option 'Add project to an EAR', an 'EAR project name:' field set to 'EAR', and a 'New Project...' button; and 'Working sets' with an unchecked option 'Add project to working sets', a 'New...' button, and a 'Working sets:' field with a 'Select...' button. At the bottom, there's a help icon, a '< Back' button, a 'Next >' button, a 'Cancel' button, and a 'Finish' button.

A questo punto dobbiamo scrivere il codice del servlet che vogliamo dispiegare in tomcat per avere il comportamento richiesto dalla traccia 8.1; Creiamo quindi una servlet:



Possiamo inoltre selezionare diversi metodi di servizio che vogliamo implementare nella specifica servlet che stiamo realizzando:



Viene generata automaticamente la nostra classe:

```

1 package it.ranauro.gln;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 /**
11  * Servlet implementation class Hello
12  */
13 @WebServlet("/Hello")
14 public class Hello extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public Hello() {
21         super();
22         // TODO Auto-generated constructor stub
23     }
24
25     /**
26      * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
27      */
28     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
29         // TODO Auto-generated method stub
30         response.getWriter().append("Served at: ").append(request.getContextPath());
31     }
32 }
33
34

```

Notiamo che viene generata anche un'**annotazione**; le annotazioni sono in generale degli input forniti agli ambienti che utilizzano il codice che andiamo a scrivere. Attraverso queste annotazioni forniamo informazioni aggiuntive agli ambienti di programmazione.

Stiamo dicendo che si tratta di una web servlet tramite **extends HttpServlet**.

Per fare sì che il server risponda con una stringa Hello World cosa dobbiamo fare?

Inviare una risposta

Per inviare una risposta per prima cosa impostiamo il contenuto tramite MIME, dicendo

```
response.setContentType("text/plain");
```

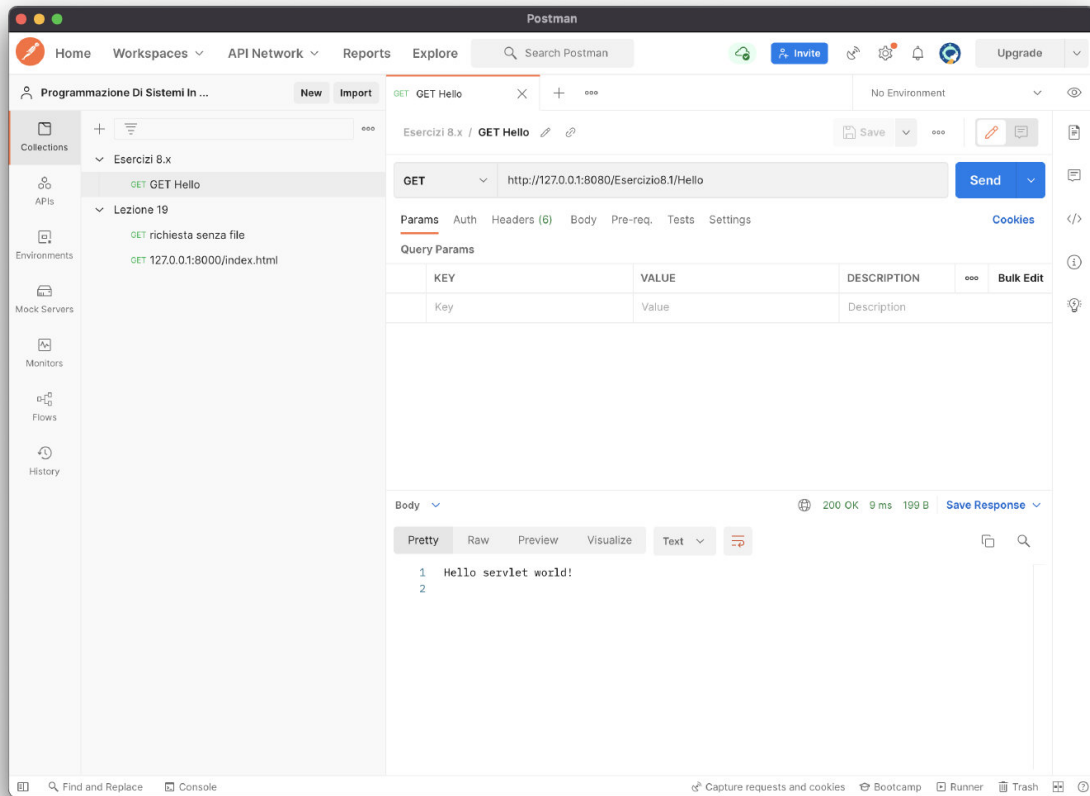
Dopodiché recuperiamo un writer con `PrintWriter pw = response.getWriter();`.

Successivamente non ci resta altro che inviare la nostra stringa con `pw.println("Hello servlet world! ");`

Come dispiogliamo il codice?

La prima cosa da fare è selezionare con **add/remove** il nostro Package. Dopo aver aggiunto il package al server tomcat dobbiamo effettuare un **publish**;

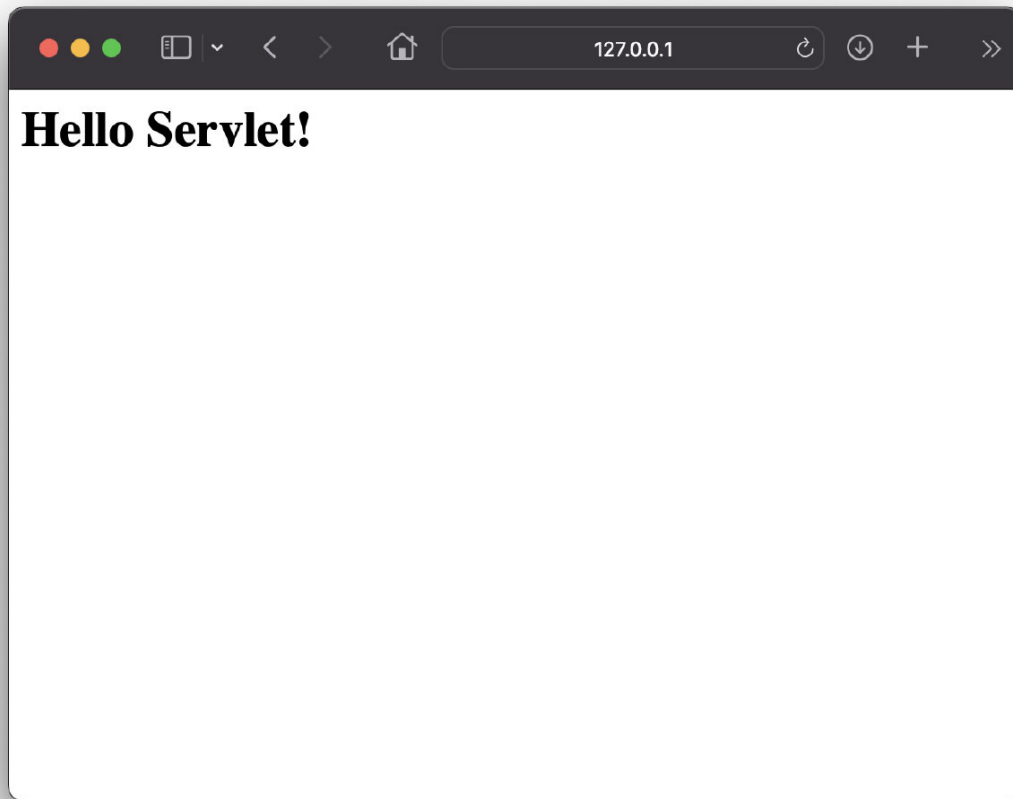
Per testare il funzionamento ci basta avviare il server tomcat ed effettuare una richiesta di tipo GET sull'URI: `http://127.0.0.1:8080/Esercizio8.1/Hello`



Ci basta cambiare leggermente la struttura della gestione del metodo GET per ottenere una visualizzazione non più text/plain ma text/html :

```
response.setContentType("text/html"); // diciamo al browser che  
vogliamo inviare del testo semplice  
PrintWriter pw = response.getWriter();  
  
pw.println("<H1> Hello servlet! </H1>");
```

Se effettuiamo la stessa richiesta fatta prima con Postman, con un browser ci viene renderizzato il messaggio di risposta:



Esercizio 8.2

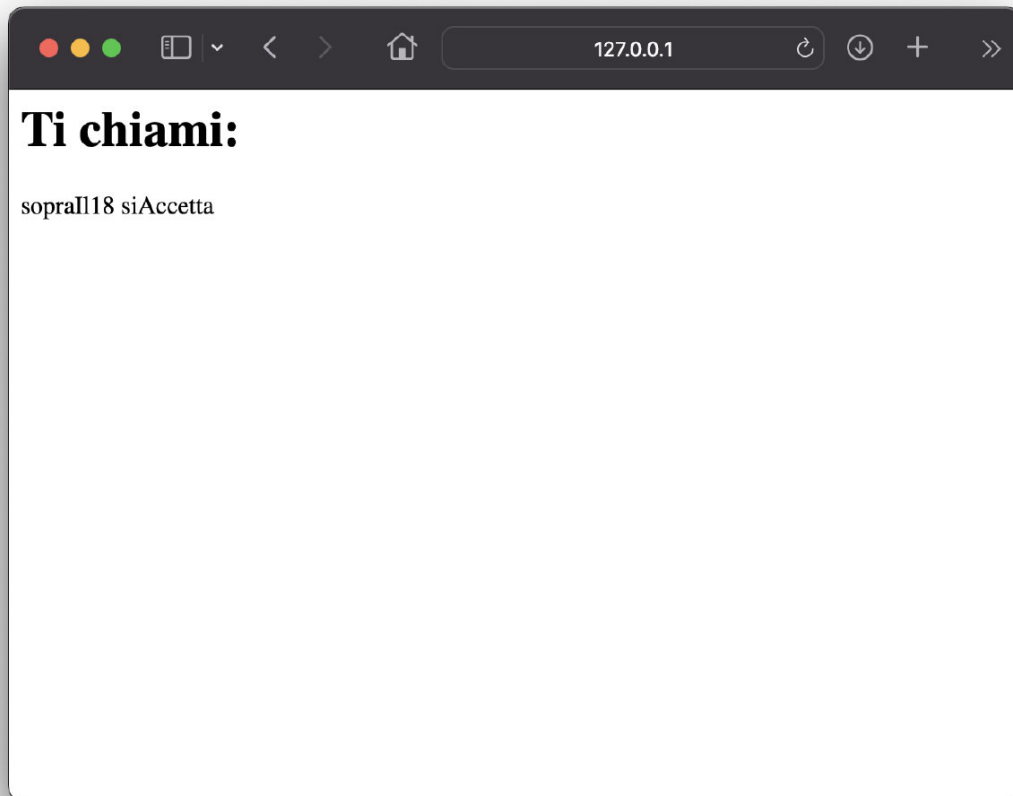
Il server deve rispondere con dei parametri passati nella richiesta GET dove `firstname=<nome>` e `lastname=<cognome>`.

Per recuperare i due parametri abbiamo bisogno del metodo `request.getParameter()`; come parametro del metodo dobbiamo passare il nome del parametro che vogliamo recuperare, nel nostro caso "firstname" e "lastname".

Facciamo un semplice "documento" HTML dove prevediamo la stampa delle due variabili lette con:

```
String responseHTML = "<H1>"
    + "Ti chiami:"
    + "</H1>"
    + "" + firstname + " " + lastname;
```

Quindi ci basta visitare l'indirizzo `http://127.0.0.1:8080/Esercizio8.2/Echo?firstname=sopraIl18&lastname=siAccetta` per ottenere:



Esercizio 8.3

Dobbiamo scrivere una servlet chiamata `calculate`: riceve due operandi ed un operatore (+, -, *, /), usando un form `calculator.html`, e che ritorna il risultato all'interno del browser.

Per realizzare una calcolatrice possiamo avvalerci di file HTML.

Ci basta aprire il file `calculator.html` per trovare il form, che invierà una richiesta GET: `http://127.0.0.1:8080/Esercizio8.3/Calculate?op1=2&op2=3&operation=%2b`; in questo caso stiamo facendo l'operazione 3+2.

Esercizio 8.4

Vogliamo definire una servlet che quando viene contattata dal client restituisce il numero di contatti.

Ci basta creare una variabile di istanza `contacts` che tiene conto delle volte che il server è stato contattato. Ogni volta che entriamo nel metodo **doGet** ci basta incrementare `contacts` e stampare sul writer.

