APPLICAZIONE WEB

La web application ha una struttura standard (non posso decidere dove mettere i file come facevo con i progetti java).
Ciò che metto in "WebContent" viene visto da tomcat (alla fine, faccio lo zip del contenuto di quella cartella che potrò spostare in altri progetti).
Dentro ci sono 2 cartelle: META-INF che contiene informazioni aggiuntive (praticamente non lo useremo), WEB-INF contiene informazioni sul server (web.xml), “lib” conterrà le librerie, “class” conterrà le classi. Il tutto in uno zip prende il nome di web archive (wrar,zip) il quale tomcat sà aprire in automatico. (genero lo zip, lo sposto sulla macchina del webserver, lo metto nella cartella giusta di tomcat "webapps", ed esso parte).

HTML nasce da zero; mette a disposizione un insieme finito e determinato di tag (l'HTML potrebbe essere generalizzato rendendo l'insieme dei tag sceglibile e ciò si chiama XML che è estendibile perché permette di sostituire/aggiungere i tag. Però ci sono cose che l'xml non riesce a riprodurre dell'html quindi si è creato un sottogruppo chiamato xhtml che è una variante/estensione dell'html 4 MA non mette vincoli; l'html 5 non si è adattato al xml e quindi noi useremo html 5 e basta).
Con html 5 posso controllare bene l'allineamento dei componenti grafici.

Le pagine html sono come alberi e alcuni nodi possono avere attributi.
Dall'inizio alla fine di un nodo, si hanno i figli o attributi.

```
<head>
  <meta ...attributi...>
  ...figli di head...
</head>
```

Clicco col tasto destro sul progetto, faccio "run as" dal server (eventualmente riavviando il server) e si apre una pagina del browser.
- il file web.xml dà informazioni utili sul progetto/applicazione.
- faccio New → Servlet, le diamo un package (unipr.informatica.esercizio4.web), scegliamo il nome della classe (ContatoreServlet) e si collega ad http (o anche https non cambia).
Cambio URL mapping con “contatore”.
Tomcat decide a suo modo quante servlet creare e quindi se in servlet ho una variabile (stato) che cambia, ciò è un problema.
L'ID sessione c'è sempre durante la sessione.
Con il logout cancello forzatamente la sessione.

Creo contatore.jsp che contiene java unito ad html5:
- tutto quello che scrivo tra parentesi <%@ ... @%> è java altrimenti è html5

Se si imposta il content type, gli si dice come andranno interpretati i byte inviati.
- La servlet è tutto java con pezzi di html.
- Jsp è tutto html con pezzi di java.

Applicazione ibrida: scarico una "app" che apre un browser e si collega ad un server il quale mi fa usare la app (in questo modo posso aggiornare la app lato client).

In un file .jsp:

```
out.println(“testo”);  // Tutto quello che scrivo qui, finisce nello string di risposta
out.flush();  // Invia i dati svuotando il buffer
```

Ordine di creazione file:
- index.html: pagina html di inizio che tramite “submit” e method “post”, invia i dati alla “ContatoreServlet.java”.
- ContatoreServlet.java: (è una “Servlet” e non una “Class”) implementa i metodi “doGet” e “doPost”.
Es: nella index ho una form che usa come method “post”; in tal caso svolgerò il codice implementato in “doPost” della “ContatoreServlet”.
Alla fine del codice, si va in “contatore.jsp”.
- contatore.jsp: riceve i parametri invitati con la form della “index.html” inoltre con “getAttribute” riceve il valore del contatore dalla “ContatoreServlet”.

ECLIPSE

Esercizio4

src
- it.unipr.informatica.esercizio4.web
  - ContatoreServlet.java

WebContent
- WEB-INF
  - web.xml
- contatore.jsp
- index.html

index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1"> // Indica come i successivi caratteri verranno analizzati; se scrivo un charset che il browser non conosce, esso fa quello che vuole.
// Lo stesso charset con cui scrivo in eclipse dovrebbe essere uguale al charset che uso nel meta.
// Apro le proprietà del progetto impostando UTF8 se voglio usarlo.
<title>Esercizio 4</title>
</head>
<body>

Pagina di prova.
<i>Testo in italico.</i>
<br>
<a href="contatore?par1=valore1&par2=valore2">Vai alla servlet</a>
<br>
<form action="contatore" method="post"> // So a chi mandare il parametro e cosa usare
<input type="hidden" name="par1" value="valore1" />
<br>
<input type="text" name="par2" value="valore2" /> // type = “hidden” è nascosto; = “text” si vede
<br>
<input type="submit" value="Invia i dati" /> // Con questo metodo di invio dati non si vedono i dati nella URL inoltre la lunghezza è arbitraria (256 caratteri se li invio tramite URL)
// Con l'html posso inviare i dati SOLO con il bottone submit e questo graficamente potrebbe non piacere.
</form>

</body>
</html>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>  // Nel “web.xml” non c'è niente da modificare.
// “web.xml” va a prendere un file di welcome nel caso non ci fosse scritto niente nel body.
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>Esercizio4</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <servlet>  // Inizio dati della servlet
    <description></description>
    <display-name>ContatoreServlet</display-name>
    <servlet-name>ContatoreServlet</servlet-name>
    <servlet-class>it.unipr.informatica.esercizio4.web.ContatoreServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ContatoreServlet</servlet-name>
    <url-pattern>/contatore</url-pattern>  // Da notare che uso come mapping della servlet “contatore”
  </servlet-mapping>
</web-app>
```

ContatoreServlet.java

```
package it.unipr.informatica.esercizio4.web;

@SuppressWarnings("serial")  // Evito di vedere errori nella pagina (danno fastidio agli utenti)
public class ContatoreServlet extends HttpServlet {
	public static final String CONTATORE = "contatore_servlet.contatore";

	@Override
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		HttpSession session = request.getSession();  // Contiene tutte le informazioni di una connessione;
getSession ritorna la sessione
		if(session.getAttribute(CONTATORE) == null)  // Prende l'attributo ed eventualmente lo imposta
con un valore
			session.setAttribute(CONTATORE, 0);

		int contatore = (int)session.getAttribute(CONTATORE);  // Se incremento questo, incremento
una copia
		contatore++;
// Ciò che ho fatto con il contatore potrei farlo anche per gli altri parametri ma meglio evitare
		session.setAttribute(CONTATORE, contatore);
		request.getRequestDispatcher("contatore.jsp").forward(request, response);  // Vado nella
“contatore.jsp”
	}
```

```
	@Override
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
		doGet(request, response);  // Finirò sempre in doGet in cui genererò una pagina html
// doGet e doPost sovrascrivono i metodi della classe da cui derivano cioè HttpServlet
	}
}
```

contatore.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
   pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Pagina Contatore</title>
</head>
<body>
Testo della pagina dinamica.
<br>
Il valore del parametro 1 e':
<%
	out.print(request.getParameter("par1"));  // par1 è definito implicitamente nell'url della pagina html
quindi non gho bisogno di fare una get per prenderlo
%>
<br>
Il valore del parametro 2 e':
<%
	out.print(request.getParameter("par2"));
%>
<br>
Il valore del contatore e':
<%
	Integer contatore = (Integer)session.getAttribute("contatore_servlet.contatore");
	out.print(contatore);
%>
<br>
Il valore del session ID e':
<%
	out.print(session.getId());
%>

<%
	if(contatore > 5)
		session.invalidate();
%>
</body>
</html>
```