

Un esempio “complesso”

- Costruiamo un’applicazione web di una certa complessità.
- Vogliamo realizzare un front-end (interfaccia verso gli utenti) di un sito che vende dei prodotti online.
- Quindi le pagine principali saranno:
 - home page,
 - catalogo dei prodotti disponibili,
 - pagina per aggiungere un prodotto nel carrello,
 - pagina per gestire il carrello (modifica quantità, ecc.),
 - pagina per effettuare un ordine.

Utilizziamo JSP

- Utilizziamo ancora i file per memorizzare i dati in base ai quali costruiremo dinamicamente le pagine.
- Sfruttiamo il supporto per le sessioni in JSP per tener traccia del carrello (i.e., i prodotti selezionati) del client.
- Cerchiamo di non ripetere lo stesso codice (HTML e/o Java) in più pagine, utilizzando la direttiva **include**.

Il codice

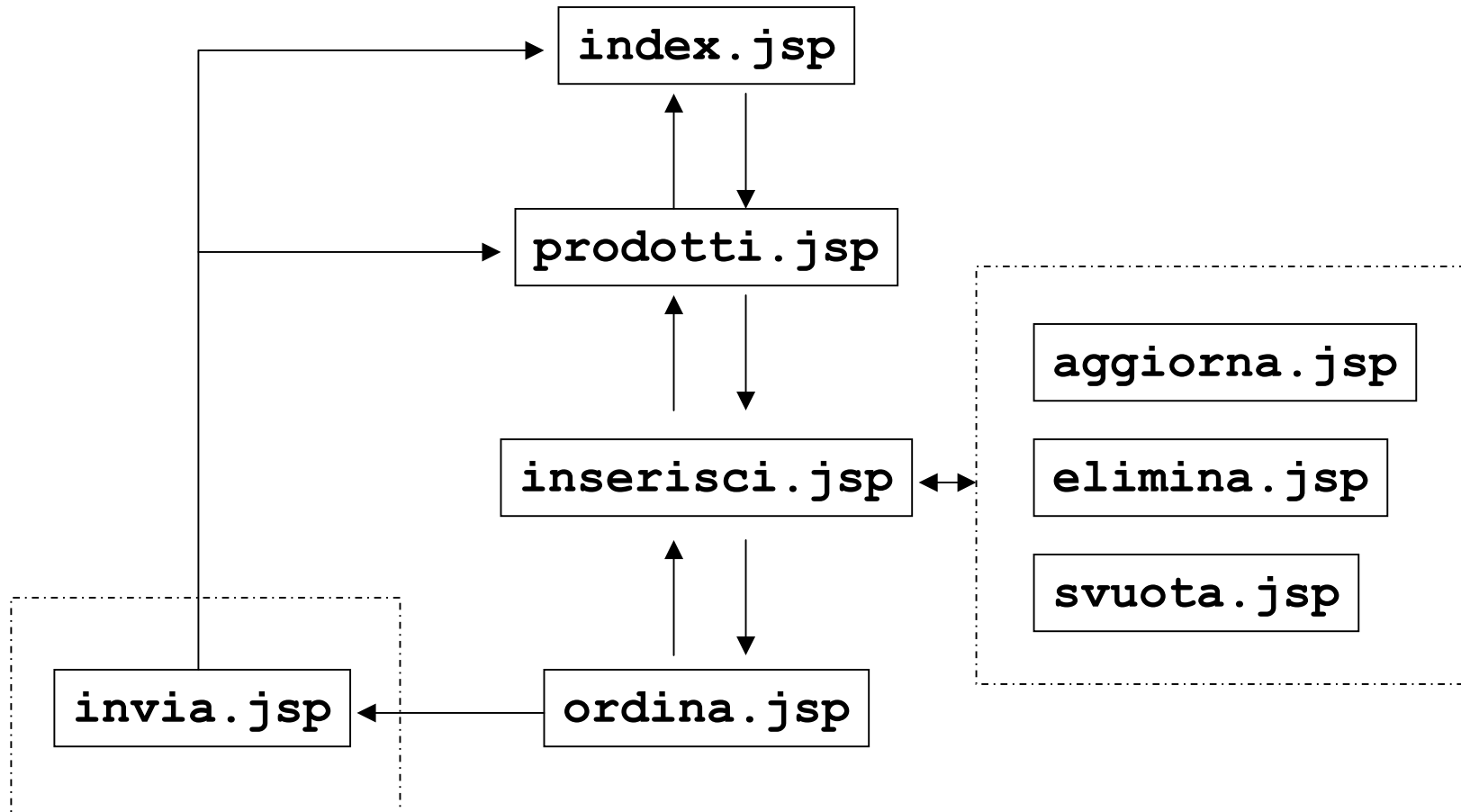
- Il codice è scaricabile da:

`www.dimi.uniud.it/scagnett/LabTecLS3`

`mitel.dimi.uniud.it/med`

- Scaricare il file .zip e decomprimerlo all'interno della directory **servlets** dentro la propria home directory. Assicurarsi che in `/home/<user>/servlets/WEB-INF` ci sia una cartella **data** (dovrebbe essere già presente in quanto utilizzata dalla servlet per gestire il login: lezione 6).

Architettura di MyShop (flusso logico)



Architettura di MyShop: pagine principali

- **index.jsp**: home page del sito (punto di partenza);
- **prodotti.jsp**: pagina con l'elenco dei prodotti ordinabili;
- **inserisci.jsp**: pagina per l'inserimento e la visualizzazione dei prodotti nel carrello;
- **ordina.jsp**: pagina con il modulo d'ordine.

Architettura di MyShop: pagine ausiliarie

- **aggiorna.jsp**: aggiorna le quantità dei prodotti presenti nel carrello;
- **elimina.jsp**: elimina un singolo prodotto dal carrello;
- **svuota.jsp**: svuota interamente il carrello.

Architettura di MyShop: file include

- I file seguenti sono inclusi nelle pagine del sito per mezzo di opportune direttive include:
 - **banner.jsp**: contiene il banner del sito (con logo e data corrente);
 - **footer.jsp**: piè di pagina;
 - **heading.jsp**: intestazione (tag **<HEAD>**);
 - **menu.jsp**: menu di navigazione nel sito.

Osservazioni sui file include

- E' possibile includere con la direttiva **include** sia file dinamici (**.jsp**) che statici (**.html**).
- Anche se un file da includere contiene solo risorse statiche, risulta comunque preferibile utilizzare l'estensione dinamica **jsp**; infatti se in futuro si renderà necessario includere del codice in tale file, non sarà necessario modificare tutte le direttive che lo utilizzano per aggiornarne l'estensione.

Il file `prodotti.txt`

- Il file `prodotti.txt` è un semplice file di testo che contiene le informazioni di un prodotto per ogni linea.
- Le informazioni su ogni linea sono organizzate in campi (separatore: `#`); la prima linea contiene l'intestazione (per la tabella HTML che verrà generata in `prodotti.jsp`):

```
ID#Prodotto#Descrizione#Prezzo (&euro;)#Q.t&agrave;; a magazzino  
1#HD Seagate 80 GB#7200 RPMS, Controller Serial Ata#80,00#10  
2#Philips 107B#Monitor 17", CRT#200,00#5  
3#RAM 512 MB#Modulo SDRAM 512 MB (Kingston)#130,00#100
```

index.jsp (I)

La struttura di **index.jsp** è la stessa delle altre pagine principali:

```
<%@ page import="java.util.Date" %>

<html>
<%@ include file="heading.jsp" %>
<body>
<table width="100%" border="0">
  <tr>
    <td colspan="2" class="blubg">
      <%@ include file="banner.jsp" %>
    </td>
  </tr>
  ...
```

index.jsp (II)

```
<tr>
  <td width="10%" class="blubg" valign="top">
    <%@ include file="menu.jsp" %>
  </td>
  <td align="center" width="90%">
    <div class="intestazione">Benvenuti su MyShop!</div>
  </td>
</tr>
<tr>
  <td colspan="2">
    <%@ include file="footer.jsp" %>
  </td>
</tr>
</table>
</body>
</html>
```

prodotti.jsp (I)

```
...
<%try {
    String filePath=getServletContext().
        getRealPath("/MyShop/prodotti.txt");
    BufferedReader input = new BufferedReader(
        new FileReader(filePath));
%> <div class="intestazione">&nbsp;</div>
<div class="intestazione">Catalogo dei prodotti disponibili:</div>
<div class="intestazione">&nbsp;</div>
<table border="0" cellpadding="5">
<%String linea=input.readLine();
    String[] intestazione=linea.split("#"); %>
<tr>
<% for(int i=1; i<intestazione.length; i++) { %>
    <td class="intestazione_tabella"><%= intestazione[i]%></td>
<% } %>
<td class="intestazione_tabella">Q.t&agrave; da ordinare</td>
<td class="intestazione_tabella">Inserisci nel carrello</td>
</tr>
```

Stampa dell'intestazione della tabella



prodotti.jsp (II)

```
<% int riga=0;
while((linea = input.readLine())
    != null) {
    riga++;%>
<tr>
```

```
<form method="post" action="inserisci.jsp">
```

```
<% String[] celle = linea.split("#");
for(int i=1; i<celle.length; i++) { %>
<td class='<%= riga%2==0 ? "riga2_tabella" : "riga1_tabella"%>'>
    <%= celle[i]%></td>
```

```
<% } %>
```

```
<td class='<%= riga%2==0 ? "riga2_tabella" : "riga1_tabella"%>'>
    <input type="text" name="qt" value="1" size="2">
</td>
```

```
<td class='<%= riga%2==0 ? "riga2_tabella" : "riga1_tabella"%>'>
    <input type="submit" name="ordina" value="&gt;&gt;">
    <input type="hidden" name="pid" value="<%= celle[0]%>"></td>
```

```
</form>
```

Nei sorgenti disponibili sulla pagina del corso,
il codice è più complesso per formattare
la valuta in modo appropriato.

prodotti.jsp (III)

...

```
input.close();
```

```
}
```

```
catch (FileNotFoundException e) {
```

```
%>
```

```
<div class="errore">&nbsp;</div>
```

```
<div class="errore">Siamo spiacenti, ma il catalogo dei  
prodotti non &egrave; al momento  
disponibile.<br><br>Riprovate pi&ugrave; tardi.</div>
```

```
<div class="errore">&nbsp;</div>
```

```
<%
```

```
}
```

```
%>
```

...

Nel caso in cui non sia possibile leggere il file prodotti.txt, viene visualizzato un messaggio.

inserisci.jsp

- La pagina **inserisci.jsp** ha una duplice funzionalità:
 - aggiungere un prodotto al carrello,
 - visualizzare il carrello.
- Se richiamata con il parametro **stampa** impostato a **1**, visualizza soltanto il carrello:
inserisci.jsp?stampa=1
- Altrimenti ricerca nei parametri passati il prodotto da aggiungere.


inserisci.jsp (I)

```
...
<%@ page import="java.util.Vector" %>
<%@ page import="java.util.Enumeration" %>
<%@ page import="java.text.NumberFormat" %>
<%

String stampaFlag=request.getParameter("stampa");
if(stampaFlag==null)
    stampaFlag="2";

String prodottiPath=getServletContext().getRealPath("")+
    "/MyShop/prodotti.txt";
```

Caricamento dei prodotti
in un oggetto Vector



```
...
Vector prodotti=new Vector();
try {
    input=new BufferedReader(new FileReader(prodottiPath));
    while((linea = input.readLine()) != null) {
        prodotti.addElement(linea);
    }
    input.close();
} catch (IOException e) { }
```


inserisci.jsp (II)

...

```
String carrello=(String) session.getValue("MyShop.Carrello");
```

```
if(carrello==null) {
```

```
    carrello="carrello"+session.getId()+".txt";
```

```
    session.putValue("MyShop.Carrello",carrello);
```

```
}
```

```
String filePath=getServletContext().getRealPath("")+
```

```
    "/WEB-INF/data/"+carrello;
```

Recupero dalla sessione del nome del file contenente il carrello
(o prima inizializzazione nel caso del primo accesso).

inserisci.jsp (III)

```
if(!stampaFlag.equals("1")) {  
    String idProdotto=request.getParameter("pid");  
    try {  
        qtProdotto =  
            Integer.parseInt(request.getParameter("qt"));  
    } catch (NullPointerException e) {  
        qtProdotto=0;  
    } catch (NumberFormatException e) {  
        qtProdotto=0;  
    }  
}
```

Recupero della quantità dai parametri del form.

```
if(qtProdotto>0) { // se la quantità è un numero valido...  
    boolean prodottoNelCarrello=false;
```

inserisci.jsp (IV)

...

```
try {  
    input=new BufferedReader(new FileReader(filePath));  
    while((linea = input.readLine()) != null) {  
        String[] campi = linea.split("#");  
        carrelloVuoto=false;  
        if(idProdotto.equals(campi[0])) {  
            prodottoNelCarrello=true;  
            break;  
        }  
    }  
    input.close();  
} catch (IOException e) { }
```

Controllo della presenza del prodotto nel carrello
per evitare l'inserimento di duplicati.

inserisci.jsp (V)

...

```
try {  
    if(!prodottoNelCarrello) {  
        BufferedWriter output=new BufferedWriter(new  
            FileWriter(filePath, true) );  
        output.write(idProdotto+"#"+  
            (new Integer(qtProdotto)).toString()+"\n");  
        output.close();  
        carrelloVuoto=false;  
    }  
} catch (IOException e) { }  
}  
}
```

Apertura del file in "accodamento".

Aggiunta del prodotto al carrello.

inserisci.jsp (VI)

```
else {  
    try {  
        input=new BufferedReader(new FileReader(filePath));  
        carrelloVuoto=input.readLine() == null;  
        input.close();  
    } catch (IOException e) { }  
}  
%>  
...
```

Se inserisci.jsp è stata richiamata con il parametro stampa impostato a 1, si controlla se il carrello è vuoto.

Il resto della pagina serve solo a visualizzare il contenuto del carrello tramite una tabella HTML (il codice è analogo a quello della pagina **prodotti.jsp**).

ordina.jsp

- Il codice di **ordina.jsp** non presenta nuove peculiarità rispetto a quanto già visto.
- Viene controllato che il carrello non sia vuoto (condizione necessaria per poter effettuare un ordine).
- Viene presentato un form con i campi (dati anagrafici, modalità di pagamento e di consegna) che verranno processati dalla pagina **invia.jsp** (che registrerà l'ordine nel file **ordini.txt** in **/home/servlets/WEB-INF/data**).

Localizzazione della valuta (I)

- Per stampare le valute nel formato più appropriato per la località corrente (separatore decimale corretto ecc.), si può utilizzare il codice seguente:

```
NumberFormat fmt = NumberFormat.  
    getInstance (Locale . ITALIAN) ;  
fmt . setMinimumFractionDigits (2) ;  
fmt . setMaximumFractionDigits (2) ;
```

- In questo modo gli importi in Euro verranno stampati con due cifre dopo il separatore decimale (la virgola).

Localizzazione della valuta (II)

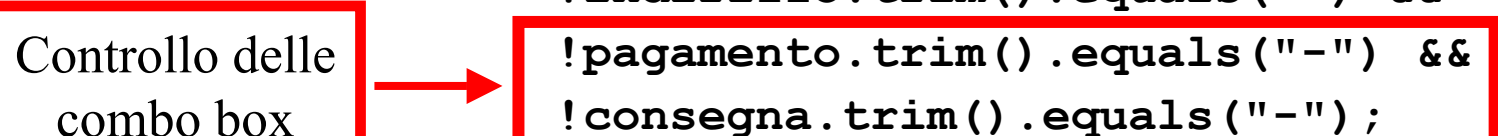
- Grazie alle classi **NumberFormat** e **Locale**, utilizzate come esposto nel lucido precedente, è poi possibile formattare valori che rappresentano valuta nel modo seguente:

```
output.write("Totale: "+  
fmt.format(new Double(totale)) +  
" Euro\n\n\n\n");
```


Controllare i campi di un form (I)

```
String nome = request.getParameter("nome");
String cognome = request.getParameter("cognome");
String indirizzo = request.getParameter("indirizzo");
String pagamento = request.getParameter("pagamento");
String consegna = request.getParameter("consegna");

// variabile booleana: true -> valori corretti,
//                      false -> errore di compilazione.
boolean formValido=!nome.trim().equals("") &&
                  !cognome.trim().equals("") &&
                  !indirizzo.trim().equals("") &&
                  !pagamento.trim().equals("-") &&
                  !consegna.trim().equals("-");
```



Controllare i campi di un form (II)

```
<%  
    if(formValido) {  
%>  
        messaggio e/o codice per gestire i valori  
        inviati (ad esempio registrando l'ordine)  
%>  
    }  
    else {  
%>  
        messaggio d'errore  
%>  
    }  
%>
```

Esercizio

- Provare ad aggiungere una funzionalità di registrazione degli utenti in modo che:
 - al primo ordine venga richiesto anche di scegliere un nome utente ed una password,
 - al momento degli ordini successivi l'utente possa scegliere di fare il login senza dover reinserire i propri dati.