Lezione 13 - Model View Controler

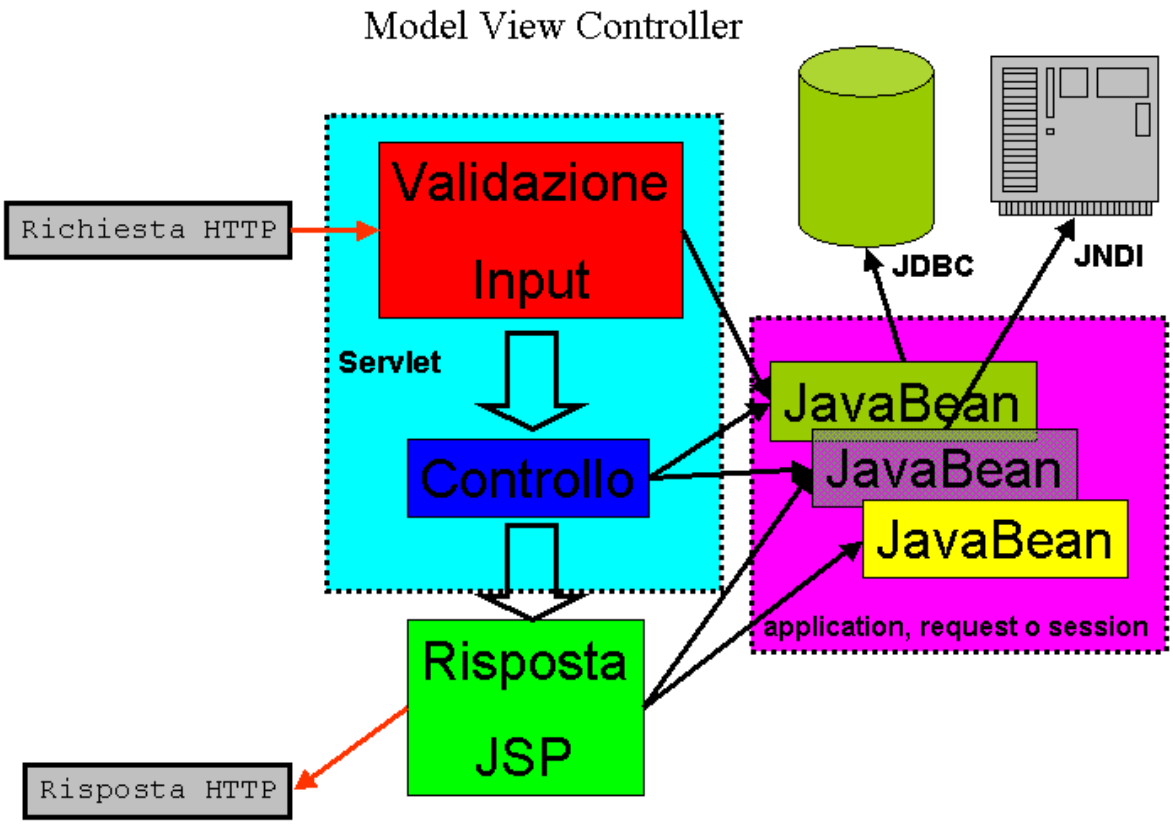
Abbiamo visto che conviene usare le pagine JSP per tutte quelle operazioni in cui la parte di visualizzazione è predominante rispetto a quella di "calcolo"
Le JSP possono essere impiegate per rendere più semplice lo sviluppo e la manutenzione della applicazione nei seguenti scenari:

- Solo JSP: per applicazioni molto semplici, con poco codice Java, che viene gestito con semplici scriptlet e espressioni;
- Utilizzo di Bean e JSP: per applicazioni moderatamente complesse; i bean nascondono la parte di codice più complessa a chi sviluppa la pagina JSP;

Ma questi due semplici scenari non possono essere adottati in tutte le situazioni. Infatti:

- se le applicazioni sono molto complesse, la singola JSP può risultare troppo complicata: troppo codice;
- nonostante sia facile separare il codice in bean, una singola JSP risponde a una singola richiesta: troppe JSP.

Per unire i benefici delle Servlet e JSP e separazione tra controllo (logica) e presentazione (aspetto) si può utilizzare l'archietettura Model-View-Controller schematizzata nella seguente figura:



Integrare Servlet e JSP:

1. La richiesta originale è processata dalla una servlet che esegue la validazione dei dati della richiesta e impartisce gli "ordini" ai bean;
2. I bean conservano le informazioni per il successivo uso da parte della seguente pagina JSP o dell'applicazione stessa;
3. La richiesta è reindirizzata a una pagina JSP per visualizzare il risultato. Inoltre possono essere usate JSP differenti in risposta alla stessa servlet per ottenere presentazioni differenti.

Questo modello è chiamato "MVC (Model View Controller" o “Approccio JSP Model 2").
Dato che nella servlet non viene generato l'output, alla fine della fase di controllo della servlet stessa bisogna "inoltrare" la generazione della pagina HTML alla pagina JSP con le seguenti istruzioni:

```
ServletContext sc = getServletContext();

RequestDispatcher rd =sc.getRequestDispatcher(jsp);

rd.forward(req,res);
```

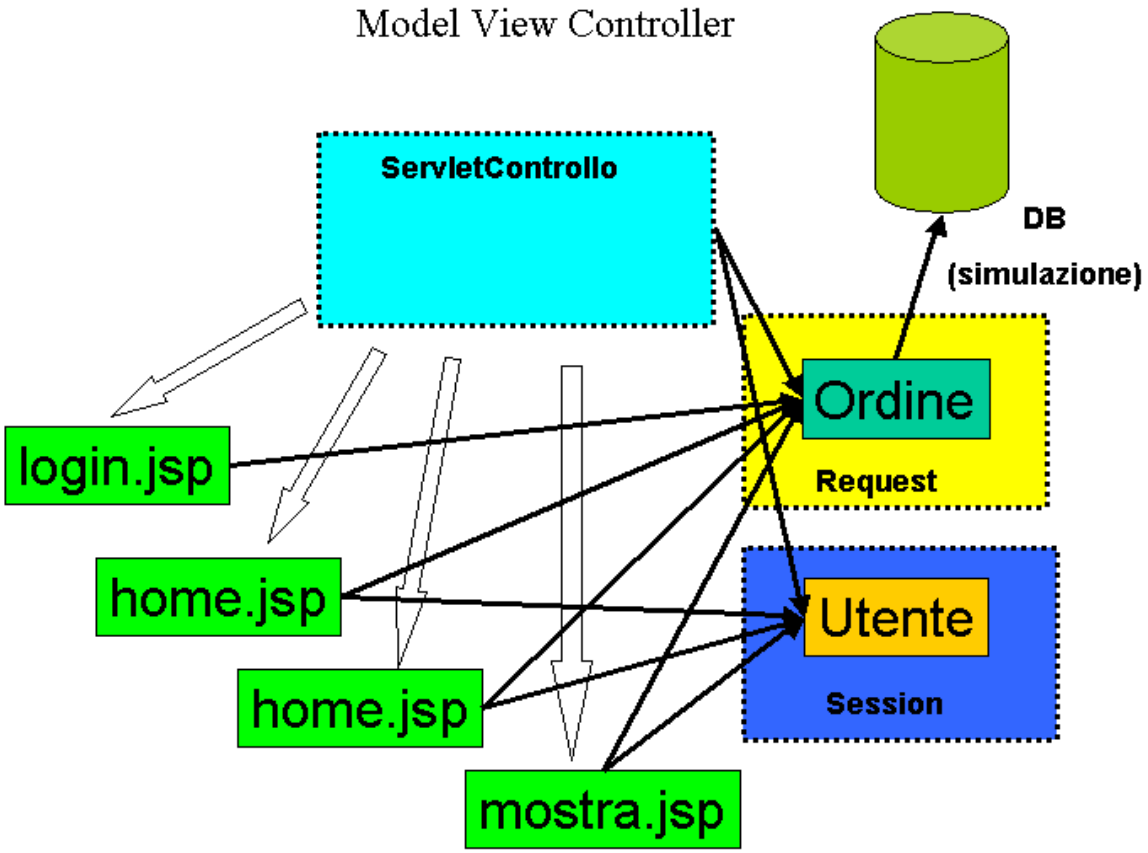
Le istruzioni di sopra possiano essere racchiuse in un metodo della servlet come il seguente:

```
private void forward(HttpServletRequest request, HttpServletResponse
response, String page)
{
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher(page);
rd.forward(request,response);
}
```

Esempio

Nell'esempio vediamo un possibile bean e come viene utilizzato da una servlet di controllo e dalle varie JSP responsabili della visualizzazione delle diverse risposte.

In questo esempio assumiamo che ci sia una sola servlet che effettua tutte le operazioni della nostra applicazione Web. La servlet stessa individua la particolare operazione da fare grazie ad un parametro (nascosto) che si aspetta di trovare nelle richieste. Tale parametro nascosto è chiamato “op”. La servlet esegue l'operazione e a seconda dell'operazione e dell'esito dell'operazione stessa inoltra la visualizzazione alla pagina opportuna.



Bean

Nota: nell'esempio di sotto l'accesso al DB non viene implementato. Gli oggetti risiedono solo in memoria RAM e vengono memorizzati un una HashMap. La chiave primaria degli oggetti si suppone venga fornita dall'utente dell'applicazione Web (ipotesi non realistica).

```
public class Ordine
{
   static java.util.Map memory = new java.util.HashMap();
   int progressivo=-1;
   String descrizione=null;

   public int getProgressivo()
   {
      return progressivo;
   }

   public void setProgressivo(int progressivo)
   {
      this.progressivo=progressivo;
   }

   public String getDescrizione()
   {
      return descrizione;
   }

   public void setDescrizione(String descrizione)
   {
      this.descrizione=descrizione;
   }
   public void insert()
   {
     //... sostituire con accesso DB
     memory.put(new Integer(progressivo),this);
   }
   public void update()
   {
     //... sostituire con accesso DB
     memory.put(new Integer(progressivo),this);
   }
   static public Ordine getOrdine(int id)
   {
     //... sostituire con accesso DB
     return (Ordine) memory.get(new Integer(progressivo));
   }
   static public java.util.Set getOrdini()
   {
     //... sostituire con accesso DB
      return memory.entrySet();
   }
}
```

File Ordine.java

Servlet

```
public Servlet extends HttpServlet{

  public void init(...){
    super.init(...);
    // creazione e inizializzazione oggetti con scope application
  }
  private void forward(... request, ... response, ... page).../vedi sopra
  public void doGet/doPost(... request, ... response){
    ...

    String op = request.getParameter("op");
    HttpSession session = request.getSession(true);
    Utente u = (Utente) session.getAttribute("user");
    if ((u==null || op==null) && !"login".equals(op)){
       forward(request,response,"/login.jsp");
       return;
    }
    if ("login".equals(op) )
    {
       u = new Utente(request.getParameter("account"));
       if (u==null || !u.checkPassword(request.getParameter("password"))
         forward(request,response,"/login.jsp");
       else
       {
         session.setAttribute("user",u);
         forward(request,response,"/home.jsp");
       }
       return;
     }

     if ("inserimento".equals(op))
     {
        Ordine nuovo = new Ordine();
        try
        {
          nuovo.setProgressivo(Integer.parse(request.getParameter("progressivo")));
        }
        catch (Exception e)
        {
          forward(request,response,"/inputError.jsp");
          return;
        }
        nuovo.setDescrione(request.getParameter("descrizione"));
        nuovo.insert();
        forward(request,response,"/home.jsp");
        return;
     }
    else if ("moduloInserimento".equals(op))
     {
        forward(request,response,"/moduloInserimento.jsp");
        return;
     }
     else if ("mostra".equals(op))
     {
        try
        {
          ordine=Ordine.load(Integer.parse(request.getParameter("progressivo")));
        }
        catch (Exception e)
        {
          forward(request,response,"/inputError.jsp");
          return;
        }
        request.setAttribute("ordine",ordine);
        forward(request,response,"/mostra.jsp");
        return;
```

```
    }
    else ...
  }
}
```

file Servlet.java

Pagine JSP

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
   "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Ordini</title>
</head>
<body>
<h2>Login</h2>
<form action="<%=application.getContextPath()%>/Servlet" method="POST">
<input type=hidden name="op" value="login">
<table>
<tr><td>User:</td><td><input name="account" type="text"></td></tr>
<tr><td>Password:</td><td><input name="password" type="password"></td></tr>
<tr><td> </td><td><input type="submit" value="OK"></td></tr>
</table>
</form>
</body>
</html>
```

file login.jsp

```
...
<% Utente user = (Utente) session.getAttribute("user");%>
   if (user==null){%>
<jsp:forward page="/login.jsp" />
<%}%>
<html>
<head>
<title>Home page</title>
</head>
<body>
<h2>Home</h2>
<ul>
<li><a href="<%=application.getContextPath()+"/Servlet"%>
?op=moduloInserimento">Inserisci nuovo ordine</a>
<li><a href="<%=application.getContextPath()+"/Servlet"%>
?op=visualizza">Visualizza Ordini</a>
<li>...
</ul>
</body>
</html>
```

file home.jsp

Invece di controllare in ogni servlet se l'utente e' presente in sessione, basta salvare le pagine jsp nella cartella WEB-INF. Tutto il contenuto di tale cartella non e' accessibile all'esterno del motore delle servlet. L'unico modi di raggiungerle e' tramite forward da una Servlet o da un Jsp visibile. Nelle rimanenti pagine non riportiamo piu' tale controlli.

```
...
<html>
<head>
<title>Ordini</title>
</head>
<body>
<h2>Inserimento Ordine</h2>
<form action="<%=application.getContextPath()%>/Servlet" method="POST">
<input type=hidden name="op" value="inserimento">
<table>
<tr><td>Progressivo:</td><td><input name="progressivo" type="text"></td></tr>
<tr><td>Descrizione:</td><td><input name="descrizione" type="text"></td></tr>
<tr><td> </td><td><input type="submit" value="OK"></td></tr>
</table>
</form>
</body>
</html>
```

file moduloInserimento.jsp

```
...
<%@ page import="package.Ordine"%>
<% Ordine ordine = (Ordine) request.getAttribute("ordine");%>
<html>
<head>
<title>Ordini</title>
</head>
<body>
<h2>Visualizza Ordine</h2>
<table>
<tr><td>Progressivo:</td><td><%=""+ordine.getProgressivo()%></td></tr>
<tr><td>Descrizione:</td><td><%=ordine.getDescrizione()></td></tr>
</table>
<a href="<%=application.getContextPath()%>/Servlet"?op=modifica&progressivo=<%=""+ordine.g
</body>
</html>
```

file mostra.jsp

```
...
<jsp:useBean id="ordine" class="package.Ordine" scope="request" />
<html>
<head>
<title>Ordini</title>
</head>
<body>
<h2>Visualizza Ordine</h2>
<table>
<tr><td>Progressivo:</td><td><jsp:getProperty name="ordine" property="progressivo" /></td>
</tr>
<tr><td>Descrizione:</td><td><jsp:getProperty name="ordine" property="descrizione" /></td>
</tr>
</table>
<a href="<%=application.getContextPath()%>/Servlet?op=modifica&progressivo=
<jsp:getProperty name="ordine" property="progressivo"/>">Modifica questo ordine</a>
</body>
</html>
```

file alternativo mostra.jsp

```
...
<%@ page import="package.Ordine"%>
<% java.util.Iterator iterator = Ordine.getOrdini().iterator();
   Ordine ordine =null; %>
<html>
<head>
<title>Ordini</title>
</head>
<body>
<h2>Visualizza Ordini</h2>
<table>
<tr><td>Progressivo</td><td>Descrizione</td></tr>
<% while(iterator.hasNext())
{
  ordine = (Ordine) iterator.next();%>
<tr><td><%=""+ordine.getProgressivo()%></td><td><%=ordine.getDescrizione()></td></tr>
<%}%>
</table>
<a href="<%=application.getContextPath()%>/Servlet?op=moduloInserimento">
Inserisci ordine</a>
</body>
</html>
```

file mostraOrdini.jsp

Regole generali:

- le pagine JSP dovrebbero ridurre al minimo l'utilizzo dei metodi che modificano gli stati del bean;
- la servlet dovrebbe prima controllare l'input, gli utenti e gli accessi e poi effettuare gli accessi in lettura e scrittura ai bean;

- Se le pagine JSP sono contenute entro la direttori “WEB-INF” dell'applicazione, il motore delle servlet non le rende disponibili all'utente tramite accesso diretto (ovvero digitando l'url sul browser). Saranno accessibili solo indirettamente tramite il forward della servlet o di altra pagina JSP. Quindi di norma vanno messe sotto la directory “WEB-INF” tutte le pagine JSP che richiedono autenticazione.

Vantaggi:

1. JSP molto semplici, con le librerie di tag possono essere scritte senza codice Java (quindi da un esperto di codice HTML senza esperienza di programmazione);
2. la possibilità che una stessa servlet o pagina JSP esegua il forward su differenti pagine JSP semplifica enormemente le JSP che in questo modo possono non contenere costrutti di controllo;
3. il controllo è centralizzato nella servlet, posso riutilizzare il codice Java. Per la stessa ragione è preferibile ridurre al minimo il codice Java presente nella JSP o spostandolo nel controllo oppure definenfo un nuovo tag;
4. riuso del modello in contesti applicativi diversi. I Bean di per se stessi non sono legati ne al protocollo HTTP, ne al codice HTML. Quindi possono essere riusati in applicazioni differenti (ad esempio un applicazione non Web);

Esercizi:

1. Completare l'esempio visto sopra implementando la persistenza nel DB Derby;
2. Definire un tag che permetta di iterare sugli elementi di un insieme tramite un iteratore.