

# Architettura MVC-2

1

ALBERTO BELUSSI  
ANNO ACCADEMICO 2010/2011

## Verso l'architettura MVC-2

2

Il secondo passo verso l'architettura MVC-2 è quello di separare il controllo dell'esecuzione, che rimane alla servlet, dalla presentazione, che viene delegata ad una o più JSP.



## Approccio Model-View-Controller (MVC)

3

In tale approccio la progettazione di una applicazione web viene divisa in tre livelli:

- “Presentation layer” (VIEW): specifica la modalità e la forma di presentazione dei dati all’utente.
- “Application Logic layer” (MODEL): specifica le procedure di estrazione da DB ed elaborazione dei dati
- “Control layer” (CONTROLLER): specifica il flusso dell’applicazione e controlla l’interazione tra gli altri livelli.

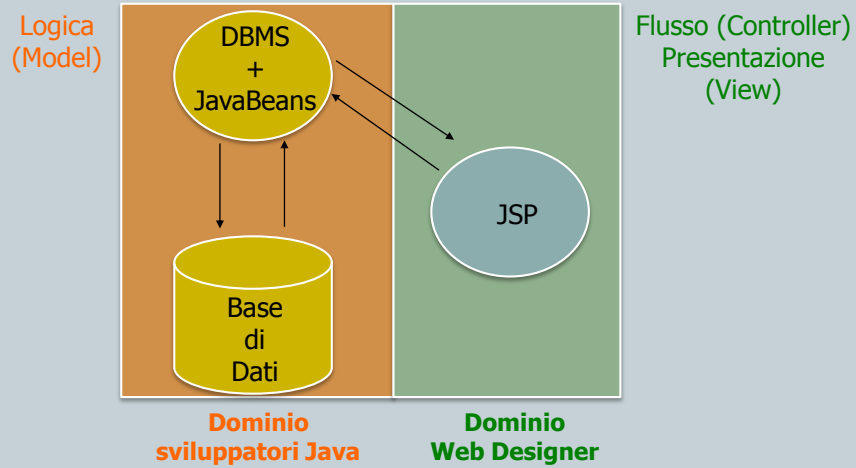
## Approccio Model-View-Controller (MVC)

4

- Adottando l'approccio MVC e la tecnologia Servlet-JSP, un'applicazione web può essere realizzata secondo diversi approcci.
- I due approcci più significativi sono:
  - page-centric
  - **servlet-centric (adottato in questo corso)**

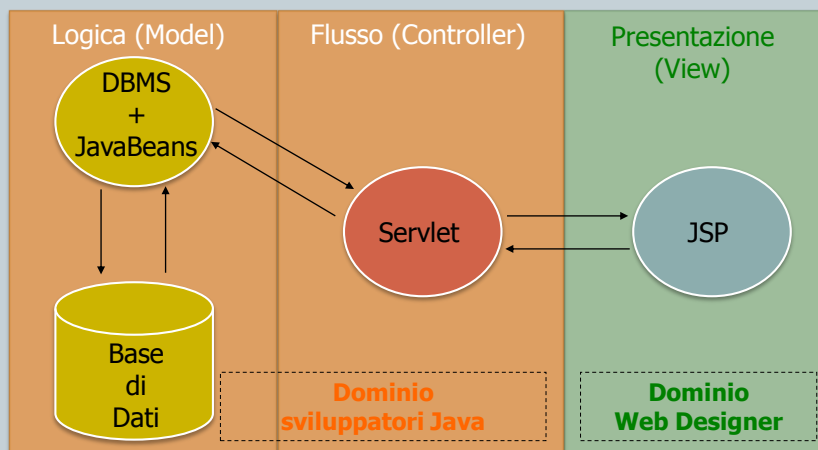
## Approccio Page-centric

5

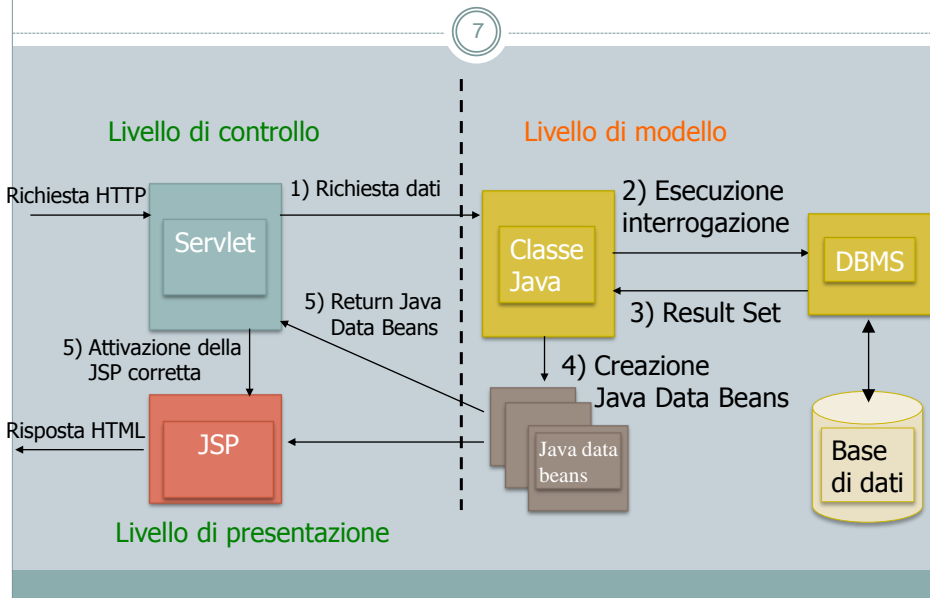


## Approccio Servlet-centric (1)

6



## Approccio Servlet-centric (2)



## Approccio Servlet-centric (3)

8

- L'approccio servlet-centric prevede di utilizzare le pagine JSP solo per la presentazione e delegare il controllo ad una o ad un gruppo di servlet. Le servlet quindi:
  - gestiscono le richieste (vengono cioè invocate tramite URL)
  - elaborano i dati necessari a soddisfare le richieste (interagendo con la classe DBMS.java e utilizzando i Java Data Bean come componenti per rappresentare le informazioni di interesse)
  - trasferiscono il controllo alla JSP designata a presentare i risultati.
- Se il gruppo di servlet che realizzano l'applicazione web viene ristretto a contenere **una sola servlet**, allora l'applicazione ha un solo punto di accesso e questa servlet ha il controllo totale sul flusso dell'applicazione.

## Approccio Servlet-centric (4)

9

### **Passaggio dati fra servlet-JSP:**

I Java Data Bean istanziati dalla servlet devono essere passati alla JSP prima di trasferire ad essa il controllo. A tal fine esiste una coppia di metodi della classe `HttpServletRequest` che permettono di inserire/recuperare in/da `request` (oggetto implicito della JSP) un numero arbitrario di oggetti. Questi metodi sono:

- `setAttribute(String, Object)`
- `getAttribute(String)`

## Approccio Servlet-centric (5)

10

### **Trasferimento del controllo dalla servlet alla JSP**

Quando all'interno di una servlet, dopo aver preparato i dati in JDB e averli inseriti nell'oggetto `request` (parametro del metodo `doGet` o `doPost`), si vuole richiamare una JSP per visualizzare i dati, si dice che si *trasferisce il controllo (forward)* alla JSP.

## Approccio Servlet-centric (6)

11

Per trasferire il controllo è necessario creare un oggetto di tipo `RequestDispatcher` associato alla JSP che si vuole 'invocare'.

Ci sono due modi equivalenti per definire un oggetto `RequestDispatcher` associato ad una JSP all'interno di una servlet:

- `RequestDispatcher rd = request.getRequestDispatcher("PathRelativoJSP")`
- `RequestDispatcher rd = getServletContext.getRequestDispatcher("PathAssolutoJSP")`

## Approccio Servlet-centric (7)

12

Una volta ottenuto l'oggetto `RequestDispatcher rd`, è sufficiente invocare il suo metodo `forward`:

```
forward(HttpServletRequest, HttpServletResponse)
```

per trasferire MOMENTANEAMENTE il controllo alla JSP.

```
rd.forward(request, response)
```

**Attenzione!** Non è un browser redirect e nemmeno una terminazione del metodo `doGet` o `doPost` della servlet... è una semplice chiamata di metodo. Tutto il codice presente DOPO `rd.forward(request, response)`, se esiste, verrà eseguito dopo che la JSP ha finito la sua esecuzione!

## Dalla progettazione logica (page-schema) all'architettura MVC

13

E' possibile indicare alcuni criteri per la traduzione di un progetto logico di un sito web centrato sui dati redatto con il linguaggio proposto (basato sui page-schema) in un insieme di moduli (servlet, JSP e JDB) dell'approccio MVC servlet-centric.

## Dalla progettazione logica (page-schema) all'architettura MVC

14

Assegnato uno schema logico costituito da un insieme di schemi di pagina ( $SP_1, \dots, SP_n$ ) e dalle interrogazioni SQL ( $Q_1, \dots, Q_m$ ) che li alimentano:

- Si genera una servlet **Main.java** per il controllo di flusso. La servlet Main gestisce tutte le richieste HTTP e suppone presente un parametro "ps" nella richiesta HTTP, che indica quale schema di pagina viene richiesto.

## Dalla progettazione logica (page-schema) all'architettura MVC

15

- Per ogni interrogazione  $Q_i$  si genera un JDB per rappresentare le tuple del risultato della sua esecuzione.
- Alcuni bean potrebbero essere molto simili tra loro, in tal caso è opportuno unire tra loro i JDB simili per generare un unico JDB.
- Tutte le interrogazioni SQL vengono gestite dalla classe **DBMS.java** che include:
  - un metodo *getQ<sub>i</sub>* per ogni interrogazione  $Q_i$  e
  - un metodo *makeYYYBean* per ogni tipo YYY di JDB che deve creare.

## Dalla progettazione logica (page-schema) all'architettura MVC

16

- I parametri delle interrogazioni devono essere gestiti come parametri delle richieste HTTP e quindi recuperati dalla servlet Main e passati ai metodi *getXXX*.
- Il contenuto informativo degli schemi di pagina deve essere prodotto dalle JSP. Si genera una JSP per ogni schema di pagina SPi.
- In tali JSP la generazione di link deve essere fatta ponendo attenzione anche ai parametri che vengono richiesti dallo specifico schema di pagina verso il quale si sta generando il link.



## Dalla progettazione logica (page-schema) all'architettura MVC

17

Vedi esempio di architettura MVC-2 servlet-centric  
che presenteremo in laboratorio.

## Riferimenti

18

- Marty Hall. "CORE. Servlets and JavaServer Pages". Sun Microsystems Press.
- Phil Hanna. "JSP. La guida Completa." McGraw-Hill.