

Laboratorio 6. MVC (23/11/2007)

In questo laboratorio svilupperemo una Web Application costituita sia da pagine JSP che da Servlet sfruttando il design pattern MVC. Questo impone la suddivisione dell'applicazione in tre parti distinte: **Model, View, Controller**. La parte View si occupa esclusivamente dell'interfaccia verso l'utente, la parte Controller dell'esecuzione di operazioni business e della preparazione dei contenuti per la parte View, la parte Model è costituita dai servizi di business (servizi di accesso a databases, EJB ecc).

La Web Application che andremo ad implementare permetterà di aggiungere e modificare i prodotti di un ipotetico catalogo. Ogni prodotto è caratterizzato dal suo codice (unico), dalla descrizione e dal prezzo.

Al primo accesso sarà presentata una pagina con la tabella dei prodotti presenti attualmente in catalogo. La tabella avrà quattro colonne, tre per le informazioni del prodotto e una contenente un link per la modifica del prodotto corrispondente. Nella medesima pagina sarà presente anche un link per l'inserimento di un nuovo prodotto in catalogo.

In caso di modifica/creazione di un prodotto sarà visualizzata una pagina contenente un form per la modifica/inserimento dei relativi dati. La form conterrà due campi di testo (descrizione e prezzo) poiché non sarà possibile modificare/inserire il codice del prodotto.

Al termine della modifica/creazione verrà di nuovo presentata la lista dei prodotti aggiornata.

Le operazioni di recupero del catalogo, inserimento e modifica di un prodotto saranno effettuate accedendo ad un database tramite la libreria di supporto *dbmanager*. Quella che utilizzeremo in questo laboratorio è una versione modificata di quella utilizzata nel laboratorio dedicato alle Servlet rispetto alla quale è stata aggiornata la classe *DBManager*.

In particolare sono stati aggiunti i metodi seguenti:

- `CatalogItem getItem(int cod);`

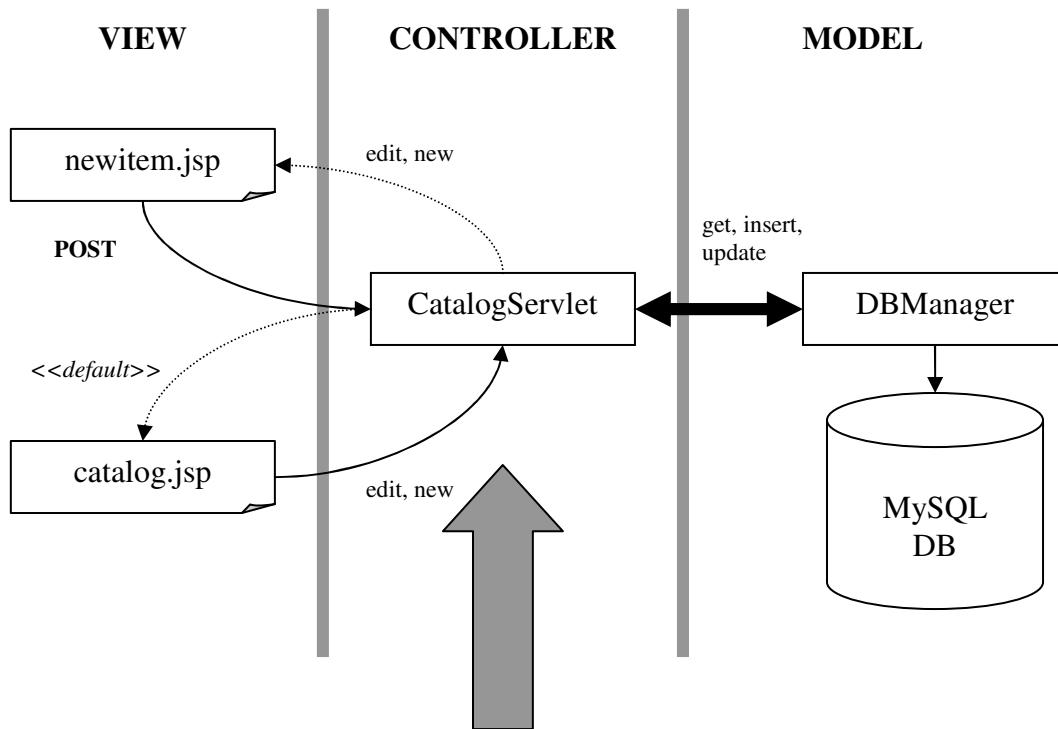
Restituisce un singolo prodotto dato il suo codice oppure null se non esiste.

- `updateCatalogItem(CatalogItem newItem);`

Aggiorna il prodotto specificato (identificato dal codice prodotto contenuto in *newItem*)

- `insertCatalogItem(CatalogItem newItem);`

Inserisce un nuovo prodotto nel catalogo.



Suggerimenti

1. I parametri di connessione al DB possono essere inseriti come context parameters in `web.xml`. In NetBeans, questa azione può essere effettuata direttamente dalla scheda General dell'interfaccia grafica del file di configurazione.
2. La libreria di supporto `dbmanager.jar` e la libreria dei driver JDBC (MySQL Connector-J) devono essere poste nella directory `WEB-INF/lib` in modo da essere visibili dalla Web Application. In NetBeans, vanno aggiunte nella libreria attraverso il pannello Projects, cartella Libraries, tasto dx, "Add JAR/folder...".
3. Per rispettare l'atomicità delle connessioni al DB, le servlet devono implementare l'interfaccia `SingleThreadModel`.
4. Attenzione all'URL pattern che usate per la `CatalogServlet`: deve essere riutilizzato per i link e i form nelle JSP.
5. In `newitem.jsp` il codice del prodotto non è modificabile ma deve essere comunque messo in richiesta al momento del post verso `CatalogServlet`. La scelta consigliata è di metterlo in un campo hidden del form di inserimento/modifica.
6. Il `CatalogItem` che viene passato a `newitem.jsp` in caso di modifica è un bean (valgono le tecniche viste nell'ultimo laboratorio). Il catalogo intero viene invece passato a `catalog.jsp` come array di `CatalogItem` e quindi non possono essere usati i tag JSP per accederlo. Ricordate quindi di importare la classe `CatalogItem` in `catalog.jsp` con l'apposito tag JSP e accedete all'array attraverso codice Java standard.