

← NOTE: DO NOT USE SCRIPTLEPTS IN JSP PAGES. USE JSTL AND EL →

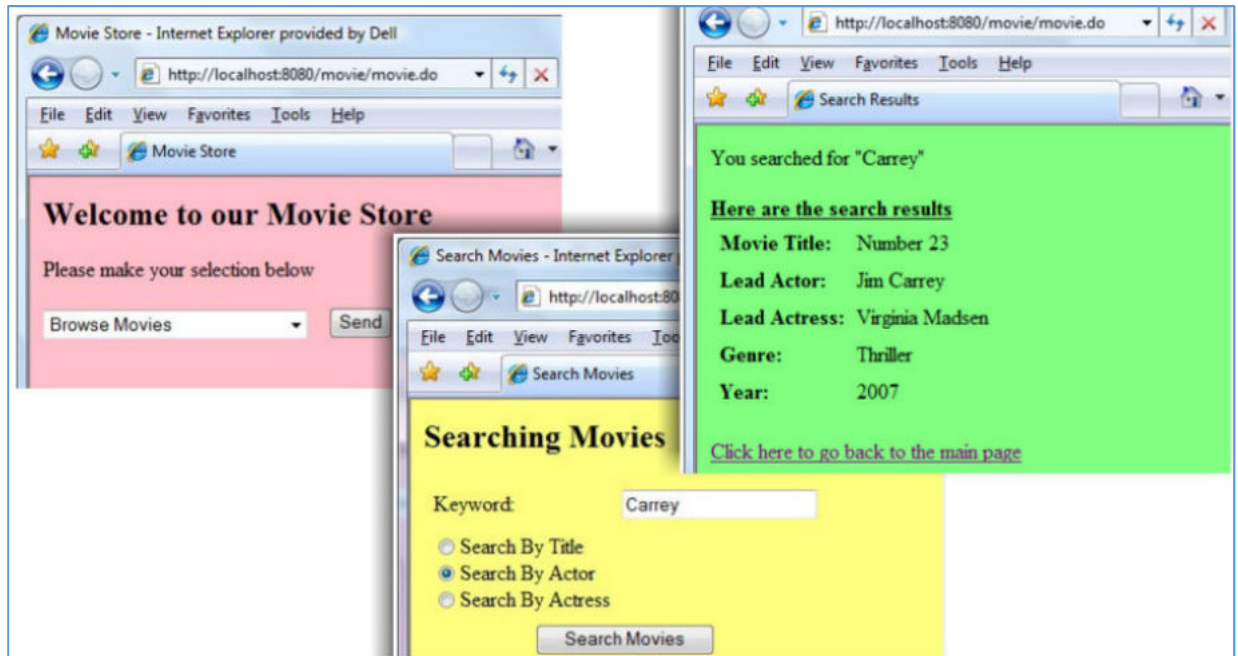
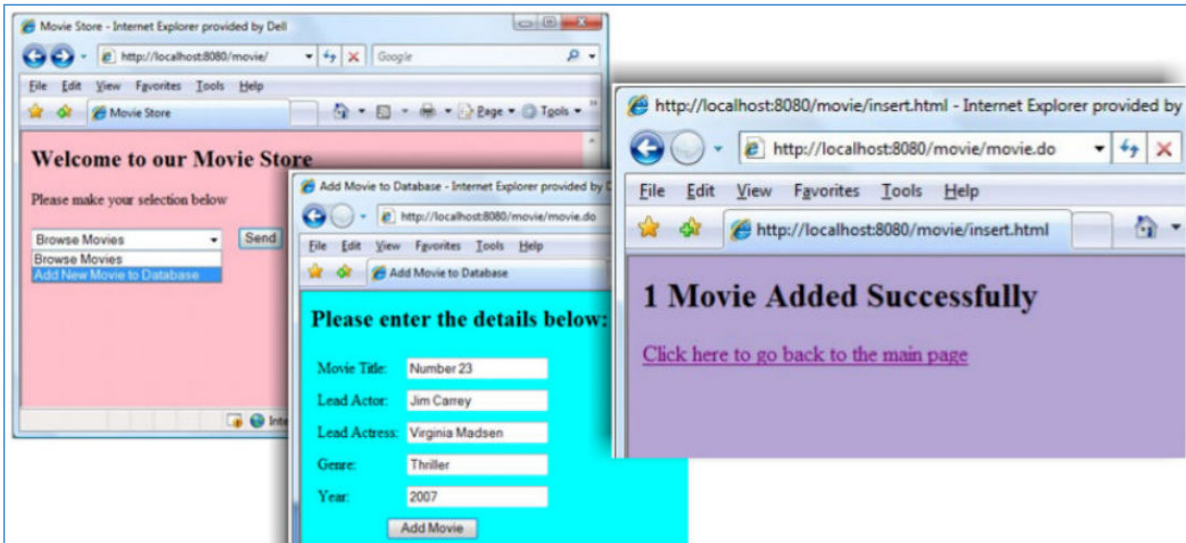
Part 1. JDBC (Reading Assignment)

JDBC provides a standard library for accessing relational databases. By using the JDBC API, you can access a wide variety of SQL databases with exactly the same Java syntax. It is important to note that although the JDBC API standardizes the approach for connecting to databases, the syntax for sending queries and committing transactions, and the data structure representing the result, JDBC does not attempt to standardize the SQL syntax. So, you can use any SQL extensions your database vendor supports. However, since most queries follow standard SQL syntax, using JDBC lets you change database hosts, ports, and even database vendors with minimal changes to your code.

Part 2. Create an MVC application to browse movies and add new movies to the DB. (You can change/improve the views). You could use AbstractControllers, but the AddMoviePage should be SimpleFormController. For those of you who don't know how to create a database, we could create one on the server, and send you the Connection information.

Database Name: moviedb
Table Name: movies

Field Name	Data Type
title	varchar(80)
actor	varchar(30)
actress	varchar(30)
genre	varchar(20)
year	integer



Part 3. Prepared Statement and Spring MVC

Use PreparedStatement to enter the books to the database whose details shown below. Implement using Spring MVC. (You may change/improve the views.)

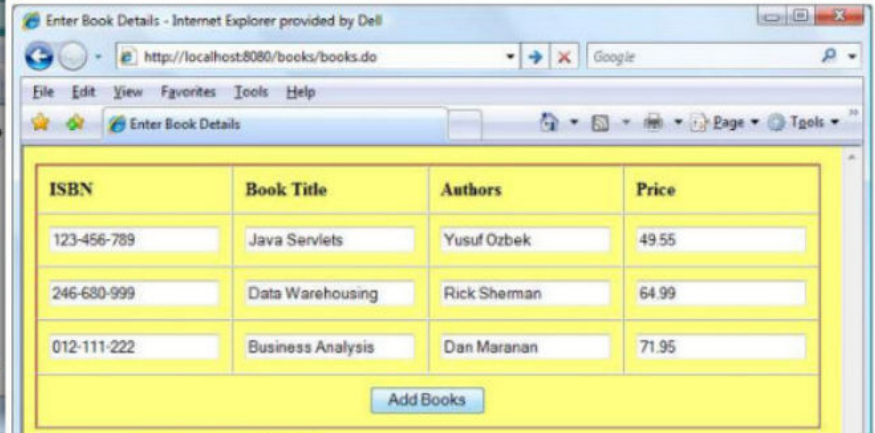
Database Name: booksdb
Table Name: books

FieldName	DataType
isbn	varchar(12)
title	varchar(60)
authors	varchar(60)
price	float

1. Ask the user to enter number of books to be added to the booksdb

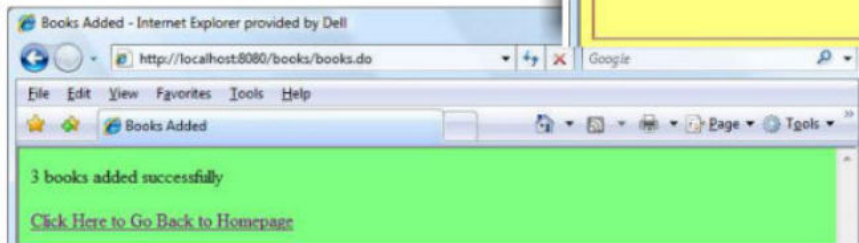


2. Based on the number user entered in step 1, create the following form to allow multiple entries.



ISBN	Book Title	Authors	Price
<input type="text" value="123-456-789"/>	<input type="text" value="Java Servlets"/>	<input type="text" value="Yusuf Ozbek"/>	<input type="text" value="49.55"/>
<input type="text" value="246-680-999"/>	<input type="text" value="Data Warehousing"/>	<input type="text" value="Rick Sherman"/>	<input type="text" value="64.99"/>
<input type="text" value="012-111-222"/>	<input type="text" value="Business Analysis"/>	<input type="text" value="Dan Maranan"/>	<input type="text" value="71.95"/>

3. Now, insert the books to the booksdb using PreparedStatement



PART 4. READING ASSIGNMENT - Custom Tags

- Chapter07 – Custom Tags Basics
- Chapter08 – Custom Tags Advanced Features

PART 5. READING ASSIGNMENT - Spring IoC

- Chapter07 - The IoC container (7.1, 7.2, 7.3, 7.4, 7.5)

<https://docs.spring.io/spring/docs/4.3.14.RELEASE/spring-framework-reference/htmlsingle/#beans>

PART 6. PROGRAMMING ASSIGNMENT (Use ONLY 1 CONTROLLER and 1 JSP page)

Read the attached CSV file using CsvJdbc Driver - <http://csvjdbc.sourceforge.net>

JSP page initially displays a FORM having a textbox for the user to enter the name of the CSV file. This form will be submitted to a Spring MVC controller that will read the name of the file, and connect to the CSV file using CsvJdbc.

Once the connection is established, the Controller will get the data from the CSV file, and place into a Model, and then call the same JSP page again. This Model to be displayed on the same JSP page using JSTL and EL a Tabular format (Textboxes in HTML table).

This form to be submitted to the same Spring MVC Controller which will use a PreparedStatement and Batch queries to insert into a MySQL database. Once this is successful, number of rows inserted to be passed to the same JSP page to be displayed.

PART 7. PROGRAMMING ASSIGNMENT

Redo Part-6 of this assignment by utilizing a CustomTag in the JSP page to display the data retrieved from CSV. The name of the CSV file is to be passed to CustomTagHandler class either as an attribute or body content (you decide which one to use). In the doTag method, you simply read the name of the CSV file using CsvJdbc, and retrieve all the data and display on the JSP page. You may create the tabular format in the doTag method. Once the table is displayed on the JSP page, use PreparedStatement and Batch queries to insert into a MySQL database as you did in Part 6.