# Web-Application

* MVC Design / Architecture
  + Model / View / Controller
* Controller
  + Takes initial control of the type of the operation that is requested by the user
  + Operations
    - Add
      * /add
    - Delete
      * /delete
    - List all the books
      * /list
* Model
  + Model represents data / information
  + Library Application
    - Collection of Books
  + Provide Data
* View
  + Formatting the data for display purposes
  + HTML / JSP / JSF
    - Formatting and rendering the information (that we take from the model)
    - Table
      * Show the collection of books

# Library Project

## Environment Preparation

* Install Eclipse
  + Download “Eclipse IDE 2022‑03” (the Eclipse installer)
  + Run the installer
    - Select the following package during installation
      * “**Eclipse IDE for Enterprise Java & Web Developers**”
  + Proceed with the installation
* Verify JRE in Eclipse
  + Launch Eclipse
    - Ensure that Default’s Eclipse JDK is set
      * Windows -> Preferences
        + Java -> **Installed JREs**
    - Ensure that the java compiler version is ‘Java 11’
      * Windows -> Preferences -> **Java -> Compiler**
* Configure Tomcat in Eclipse
  + Windows -> Preferences -> Server -> Runtime Environments
  + Click Add
  + Choose Apache -> Apache Tomcat v9.0
  + Select details
    - Name - ‘Apache Tomcat v9.0’
    - Click ‘Download & Install’ and as part of this, specify the Tomcat installation direction
      * After specifying the directory, wait for sometime, till the download is done
  + Click Finish

## Project Creation

* Steps
  + Project Creation
    - Type of Project
      * Web -> **Dynamic Web Project**
    - Ensure the following section
      * Target Runtime - ‘apache-tomcat-v9.0’
      * Dynamic Web Module Version
        + 4.0
      * Web Module
        + Context root

Watch out and note the value for the Web-App “**Context Root**”

* + - Go with the default settings for the remainder of the pages
  + Copying Jars
    - * Copy the jars from [here](https://github.com/sr1983-it-gl/GL-Java-FSD-9-LIBRARY/tree/main/Jar%20Files)
      * Paste them in the following project location
        + “\src\main\webapp\WEB-INF\lib”
    - Refresh the project in Eclipse to see the copied jar files in the copied location

## Welcome Page

* File Changes
  + Server
    - Configuration
      * “/WEB-INF/web.xml”
        + Contains entries for

Servlet

servlet-mapping

absolute-ordering

* + - * spring-config.xml
        + Contains entries for

Context-component-scan

View-Resolver

* + - Controller
      * WelcomeController
  + UI
    - welcome.jsp
* Run the project
  + Right Click
    - Run as Server
      * Select ‘Always use this server’
  + Make sure that the project is available on the right-side of ‘Configured’ section
* Access the Library Application
  + Web Browser
    - URL
      * [http://localhost:8080/LibraryManagement/](http://localhost:8080/LibraryManagement/books/listing)
* Common Errors
  + Make sure that the Path references are correctly mentioned
    - Example
      * /WEB-INF
  + Make sure that the component scan package is correct
  + Missing @Transactional annotation
  + Port Conflict Error (Port already used by some other application)
    - Open the command prompt in Admin-mode
    - Use the command to find the process ID
      * “netstat -ao | findstr **8080**”
    - Use the following command to kill the process
      * “taskkill -F -pid <process\_id>”

## Listing Books

* File Changes
  + Database
    - Create the database schema - ‘library-management-v1.0’
  + Server
    - Configuration
      * spring-config.xml
        + Contains entries for

DataSource

Contains the 4 mandatory attributes

Driver Class name

URL

Username

Password

SessionFactoryBean

Contains the following attributes

DataSources

A reference to the ‘dataSource’ bean

Annotated Classes - The list of java classes that are needed to be treated as entities

TransactionManager

Contains the following attributes

A reference to the ‘sessionFactory’

* + - Entity
      * Book
        + ‘@Entity’ annotation

The name of the database table can be customized here

This name to be used in ‘Hibernate Query Language’

* + - * + ‘@Table’ annotation

Should point to the name of the ‘database table

* + - * + ‘@Column’ annotation

Applied on a java field, this should point to the column name

* + - * + ‘@Id’ annotation

To be applied on the java field holding the database’s primary key

* + - * + ‘@GeneratedValue’ annotation

The primary key generator for the primary key attribute

For MySQL, ‘GeneratorType.IDENTITY’ makes use of ‘Auto Incrementation’ mechanism

* + - Service
      * BookService
        + listBooks

The string passed to createQuey (“from Book”) is HQL

* + - Controller
      * BookController
        + listBooks

RequestMapping

Two options

Declaring RequestMapping at the class and also at the method level

Declaring the RequestMapping only at the method level

If there is a requirement, where model needs to be accessed in the JSP pages, then make use of Model object

The method Model.addAttribute() to add model objects

* + UI
    - Book-lister.jsp
      * Make use of TagLibrary (JSTL -> Java Standard Tag Library) to iterate the model contents and to display
        + Note the use of taglib in the declaration

The URL where the taglib is declared

A prefix for access purposes

* + - * + To call a certain operation, the syntax is ‘prefix:operationName’

Examples

<c:out>

<c:forEach>

* + - Welcome.jsp
      * Reference to ‘list-books.jsp’ through anchor tag
* Common Errors
  + In the dispatcher-configuration file
    - Data Source
      * Watch out for the database name
      * Watch out for username/password credentials
    - Session Factory
      * Watch out for the fully qualified name for Book
  + Usage of @Repository on BookServiceImpl

## Update Book

* File Changes
  + UI
    - book-lister.jsp
      * Through the anchor tag, add a link to ‘begin-update’
      * Make sure to pass the ‘book-id’
      * To pass request parameters to the Server, the syntax is
        + “/ab/b/c?var1=val1&var2=val2”
        + Example

“/book/update?bookId=1”

* + - Book-details-page.jsp
      * Create a form to list down the fields (taken from the model object)
        + Name / Author / Category
      * Make use of the hidden attribute to send the ‘book-id’ information back to the server
      * Make use of the POST method during form submission
  + Server
    - Service
      * BookService
        + findById

Finds the book object based on id

* + - * + update

Updates the existing book object with the new book details

* + - Controller
      * BookController
        + beginUpdate

Makes use of the RequestMapping (“/begin-update”)

Gathers the ‘bookId’ input through ‘RequestParam’

Fetches the ‘book’ object and stores in the model

* + - * + handleSaveOrUpdate

Retrieves the existing book object

Calls the service method to update the book object with the new details

Calls to ‘book-list’ page so that user will be able to see the updated book information

## Delete Book

* File Changes
  + Server
    - Service
      * BookService
        + deleteById

Retrive the book object based on the book-id

Delete the book object

* + - Controller
      * BookController
        + Delegate the control to service-implementation
        + Redirect the UI control to ‘book-lister’ so that user will be able to see the deleted-book updated in the UI
  + UI
    - Book-lister.jsp
      * Add a javascript confirmation
      * Based on the response, invoke ‘/delete’
      * Pass the ‘book-id’ to the server
      * console.log statements
        + On the page -> Right click -> Inspect -> Console tab (next to the ‘Elements’ tab)
    - Book-form.jsp

## Adding new Book

* File Changes
  + Server
    - Service
      * BookService
        + add

Saves the book object to the database

* + - Controller
      * BookController
        + handleBeginAdd

Create a new book object and populate it in the model

Redirect to ‘book-details-form’

The same page is reused here

* + - * + handleAddOrUpdate

Check for the ‘id’ value and based on that decide whether its ‘update’ or ‘save’

* + - UI
      * book-lister.jsp
        + Addition of add button
        + Invocation of the button leads to ‘begin-add’ URL
      * Book-form.jsp

## Searching Books

* File Changes
  + Server
    - Service
      * BookService
        + searchBy

Construction of query using ‘like’ operator

Empty check considerations before query construction

* + - Controller
      * BookController
        + handleSearch

If both the fields are empty, then redirect the control directly to ‘book-lister’

Search for books and populate in the model object

* + - UI
      * Book-lister.jsp
        + Makes use of form for containing ‘name’ and ‘author’ fields
        + Make use of ‘search’ text-boxes

# References Table

| **Item** | **Value** |
| --- | --- |
| Web-app Schema | * + - xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"     - xmlns="http://xmlns.jcp.org/xml/ns/javaee"     - xsi:schemaLocation="<http://xmlns.jcp.org/xml/ns/javaee> http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"     - id="WebApp\_ID" version="3.1" |
| Dispatcher Servlet - Class name | * + - org.springframework.web.servlet.DispatcherServlet |
| Beans schema | xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc"  xsi:schemaLocation="  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  http://www.springframework.org/schema/context/spring-context.xsd  http://www.springframework.org/schema/mvc  http://www.springframework.org/schema/mvc/spring-mvc.xsd" |
| View Resolver - Class Name | * org.springframework.web.servlet.view.InternalResourceViewResolver |
| JSTL View - Class name | * org.springframework.web.servlet.view.JstlView |
| Welcome JSP - Directive | * <%@ page language="java" contentType="text/html; charset=ISO-8859-1" * pageEncoding="ISO-8859-1"%> |
| Datasource Classname | * org.springframework.jdbc.datasource.DriverManagerDataSource |
| Driver classname | * com.mysql.jdbc.Driver |
| JDBC URL | * jdbc:mysql://localhost:3306/library\_database |
| Session Factory Bean class name | * org.springframework.orm.hibernate5.LocalSessionFactoryBean |
| Hibernate Dialect | * org.hibernate.dialect.MySQL8Dialect |
| Transaction Manager - Class name | * org.springframework.orm.hibernate5.HibernateTransactionManager |
| Stylesheet Addition | <link rel="stylesheet"  href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css"  integrity="sha384-GJzZqFGwb1QTTN6wy59ffF1BuGJpLSa9DkKMp0DgiMDm4iYMj70gZWKYbI706tWS"  crossorigin="anonymous"> |