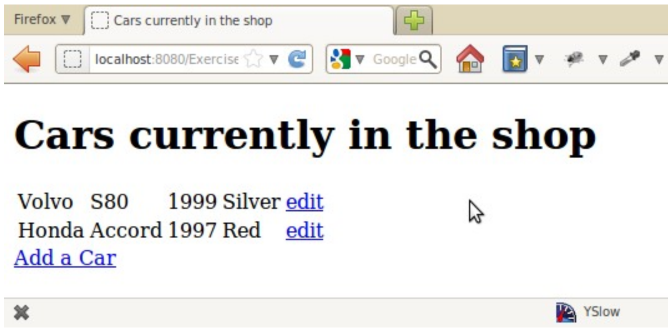
Lesson 9 Web MVC and Validation
Exercise 8.1 Spring MVC

The Setup:

In this exercise we will create a simple CRUD (Create, Retrieve, Update, Delete) application with Spring MVC. Start by opening **exercise8.1** and add the following dependencies to the project's pom.xml:

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>4.1.6.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>4.1.6.RELEASE</version>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

Once everything is setup running the project on the tomcat server should open the following page in your browser:

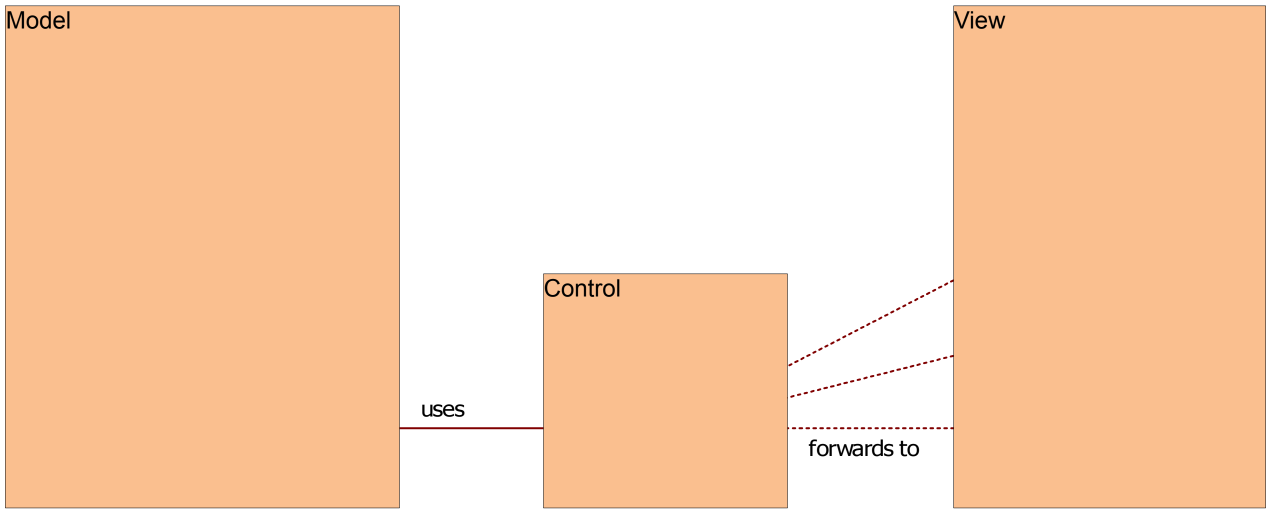


Cars currently in the shop

```
Volvo   S80    1999 Silver edit
Honda Accord 1997 Red    edit
Add a Car
```

The Application:

The provided code is reasonably simple, **CarController** uses **CarDao** to create, retrieve, update and delete **Car** objects, after which it forwards to one of the views.



The Exercise:

The goal of this exercise is to make a Book store CRUD application similar to the Car Shop curd application. We've provided the basic **Book** and **BookDao** classes along with the exercise skeleton code.

The core items that you will need to make are the **BookController** class, and a new set of views related to the book store e.g. BookList, BookDetail, AddBook.

You can either copy paste many of the files from the car shop application and update them, or for a greater (although potentially somewhat more frustrating) learning experience you can start from scratch.

Note: Please be aware that **CarController.java** has a mapping for "/", if you directly copy paste CarController.java to **BookController.java** you will end up with a "/" mapping in BookController.java as well. Spring does not like having the same path mapped to two different methods.. In other words, remember to remove one of the two, they cannot exist simultaneously

Exercise 8.2 – Spring Open Session in View

The Setup:

The goal of this exercise is to test Spring open session in view support. Start by importing the exercise8_2 project, and add the spring context, tx and orm dependencies (the same Spring dependencies as in Exercise 8.1). Also add the following spring web dependency:

```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-webmvc</artifactId>
<version>4.1.6.RELEASE</version>
</dependency>
```

In essence the exercise will add Spring onto this Hibernate web application. If you want you can try it on your own, or follow the steps in the exercise section below.

The Exercise:

Update the web.xml file to use Spring's open session in view filter, and be sure to also add the spring ContextLoaderListener. You can delete our own OpenSessionInViewFilter class.

Create a springconfig.xml file inside the WEB-INF directory, and create beans for the studentService and studentDao classes (setup dependency injection between them).

Also configure Hibernate inside the springconfig.xml file, and inject the sessionFactory into the StudentDao. You can remove the hibernate.cfg.xml file and the HibernateUtil class.

Update the StudentCourseServlet class to retrieve the studentService instance from Spring instead of creating a new instance.

Annotate the appropriate methods on the StudentService and StudentDAO classes as @Transactional, and remove the programmatic transactional demarcation.

Exercise 8.3 Spring Validation

The Setup

Create a copy of your solution to exercise8.2 and call it exercise8.3. Remember to also change the project name inside the pom.xml's `<artifactId>` and `<name>` tags.

In addition to the dependencies for exercise 8.3 add the following for hibernate validator:

```
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>5.1.2.Final</version>
        </dependency>
```

The Application

The application will be the book application you've created as a solution to exercise 8.2.

The Exercise

Add validation to the book class. Update the book controller to do the validation. and change / update your addBook.html to addBook.jsp so that you can use the spring form tags to properly show the errors.

You want to add validation annotations onto the book class to make sure that the title and the author are not blank, the price is greater than zero, and the ISBN matches the following pattern:

\d{3}-\d{10}

Optional Additional Exercise:

- Add a published date to the book class, and validate that it's in the @Past