**What this project does??**

**This is Full Spring MVC and Hibernate application with XML configuration.**

Note that this is fourth project in the series of Spring MVC which explains functioning of controller layer and service layer and DAO layer and how database operation takes place in Spring Hibernate project. **It carries out CRUD (i.e. Create, Read, Update and Delete) operations on database tables.** Before moving further please go through projects 'SpringMVCApplicationFlow' which is first project in Spring MVC Series. It does not contains any web related operations, it explains functionality of 'spring-servlet.xml' in Spring MVC context, ' SpringMVCHelloWorld' which explains how controller layer functions in Spring MVC and 'SpringMVCArtihmeticOperationsDemo' which explains functioning of controller and service layer in Spring Framework

Thus

' SpringMVCApplicationFlow': explains functionality of 'spring-servlet.xml' in Spring MVC context

' SpringMVCHelloWorld': explains how controller layer functions in Spring MVC

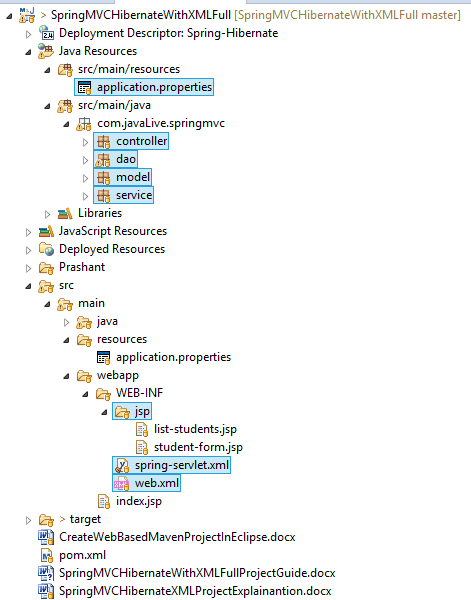
'SprinArithematicOperation': explains how controller layer and service layer functions in Spring MVC

'SpringMVCHibernateWithXMLFull' : explains full Spring+Hibernate database table CRUD operation **with full XML configuration** which includes all three layers of Spring MVC viz. Controller, Service and DAO.

This project explains basic functioning of Spring MVC. How controller acts and handle request, makes call to service layer which contains business logic, collects the result of business logic from service layer and render the specified jsp accordingly.

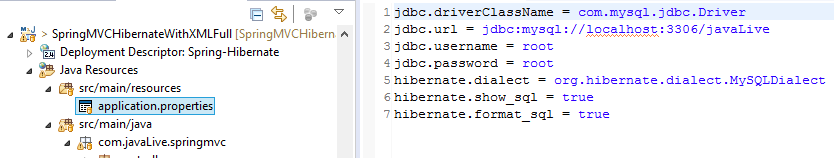
This is web based maven project.

**Note that this is xml based spring MVC project. We have web config file web.xml and spring configuration file spring-servlet.xml. (Location of these files are shown in screen shot of project structure below.)**

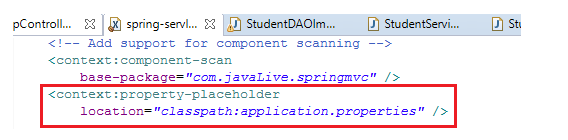
****

**In above project please go through files, packages and folders as follows:**

application.properties which contains database related properties as below:

****

We have mentioned path of above file in spring-ervlet.xml file as below:



Packages controller which contains controllers.

Package service: which contains service classes.

Package DAO: which contains DAO classes.

Package model: which contains Entity classes.

Folder WEB-INF/jsp which contains various JSP files.

Files WEB-INF/spring-servlet.xml and WEB-INF/web.xml.

File index.jsp which is welcome file of this project.

**Steps to create project:-**

1. This is web based maven project. Create webapp maven project. Please refer the file **'CreateWebBasedMavenProjectInEclipse.docx'** in this project to know more about creation of web based maven project.

2. Add require dependencies for spring as shown in pom.xml file.

3. Create required packages and add the files.

**Functioning of the project**

As mentioned in the opening statement, this is Spring MVC project responsible for carrying out CRUD operations on given database tables using Hibernate.

Detailed explanation of this project is given in file 'SpringMVCHibernateXMLProjectExplainantion.docx'.

Here we will just go through high level flow of this project with screen shots as follows:

**Before move on please study functioning of spring-servlet.xml, how beans get loaded at runtime and how component scan works.(@Controller in class com.javaLive.springmvc.controller.AppController, @Service in class com.javaLive.springmvc.service .StudentServiceImpl and @Repository in class com.javaLive.springmvc.dao .StudentDAOImpl are subtype of @Component)**

**A .** Right click on the project-> RunAS-> Run on Server.

Now behind the scene following steps takes place

a) server searches for web.xml

b) control goes to spring-servlet.xml as we have mentioned dispatcher servlet in web.xml.

c) Now Spring Framework comes to action, it loads various beans mentioned in spring-servlet.xml viz. InternalResourceViewResolver, Datasource, HibernateSessionFactory and TransactionManager etc and under component scan process, it notices and marks controller, service and DAO layers.

Note that we have used various properties defined in application.properties file in resources folder to define DataSource and SessionFactory beans here.

**B.** Server renders welcome file page index.jsp which redirects the request to link 'student/list'. Now controller viz. ' com.javaLive.springmvc.controller. AppController' get called as it contains the link mentioned in the redirect method.

Now we will see how actually Student records get fetched from database step by step:



**Step 1:** Please note that this is partial link. Part of the link that is common to all the links given in the controller is mentioned at the top of class definition.(Please refer the above screen shot highlighted as Step 1.

**Step 2:** This is specific link part which ultimately call the method given below it.

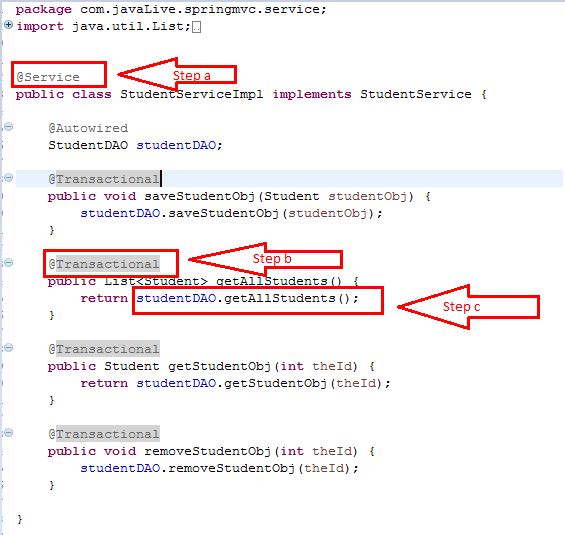
Please go through annotation @RequstMapping and parameter Model very carefully. For this purpose refer following link:

<https://dzone.com/articles/using-the-spring-requestmapping-annotation>.

**Step 3:** Model class responsible for holding result from service layer and delivering the result to specified JSP page.(Note that this Model class does not have any relationship with model package. It is only coincidence, don't confuse.) Also refer above link to know more about this.

We can also use more sophisticated class viz. ModelMap which have same functionality as of Model with more advance features. We are going to use it in next project viz. SpringHibernateExample.

**Step 4:** In this step call is made to particular method of Service layer class (viz. getAllStudents() method of com.javaLive.springmvc.service. StudentServiceImpl.java.

****

This step has nesting of step as Step a, Step b and Step c

**Step a:** Note that this class have @Service annotation which tell Spring that this is service layer class.

**Step b : This is most important part of this file. Please go through this annotation and study why it is used as Service layer and not at DAO layer.(This is very popular interview question.)**

**Please refer below link for more detail**

[https://www.marcobehler.com/guides/spring-transaction-management-transactional-in-depth#spring-section](https://www.marcobehler.com/guides/spring-transaction-management-transactional-in-depth%23spring-section)

**Step C:** In this step call is made to DAO layer i.e. class StudentDAOImpl in this case.

This step again have nesting of steps as Step i, Step ii, Step iii, Step iv



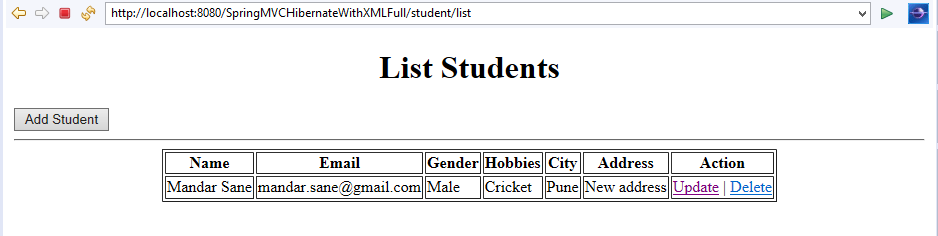
Step i: Note that this class have @Repository annotation which tell Spring that this is DAO layer class.

Step ii: This is the interface having method declarations to deal with Student data in database.

Step iii: Note that this is dependency injection of SessionFactory bean defined in spring-servlet.xml file. Pay attention to the way in @Autowired annotation is used in real world.

Step iv: This is actual call to database with the help of Hibernate, which get list of all students in the table and hold the result in a List object and return that list to Service layer class.

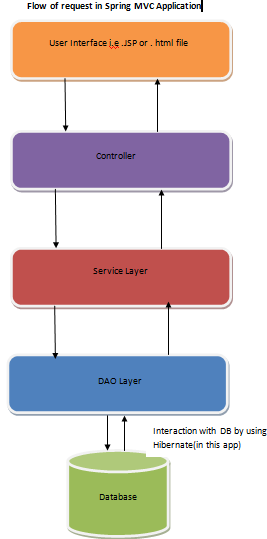
**Step 5**(of the sceen shot on Page 4): Now above list is added to model attribute and list-student JSP page is called as below which displays the list specified in Model(please go through the code of this jsp page).



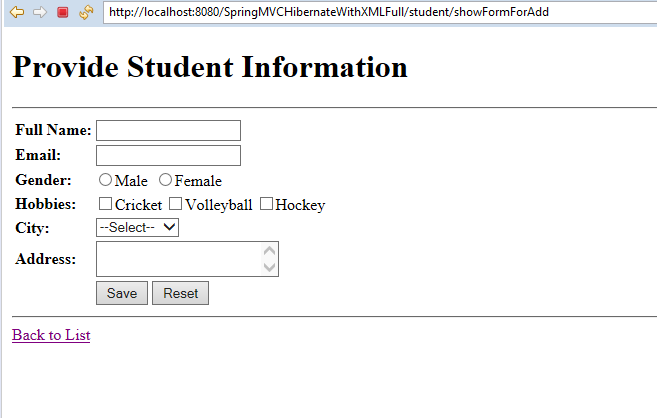
Please note that it fetches the data that is already present in the given database table which completes one of our CRUD operation i.e. Read.

**\*\*Note that thes step sequence is followed for insert, update and delete operations with respective pages and methods.Hence we have not mentioned them separetly for those operations given below. Try to understand then with above sequence.**

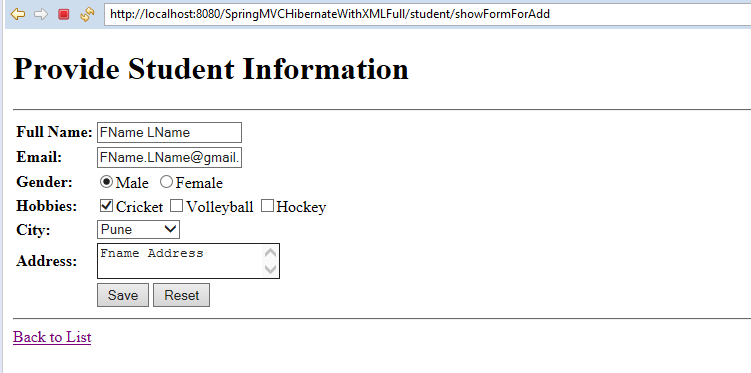
**Below diagram illustrates overview of flow of request in Spring MVC application:**

****

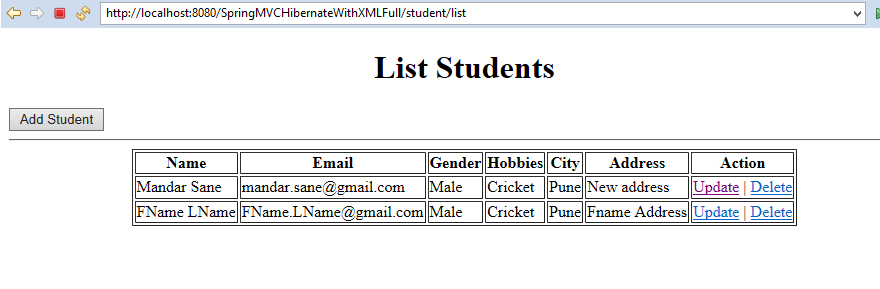
**C.** Now click on button 'Add Student', it leads to rendering following page i.e. WEB-INF/jsp/ student-form.jsp



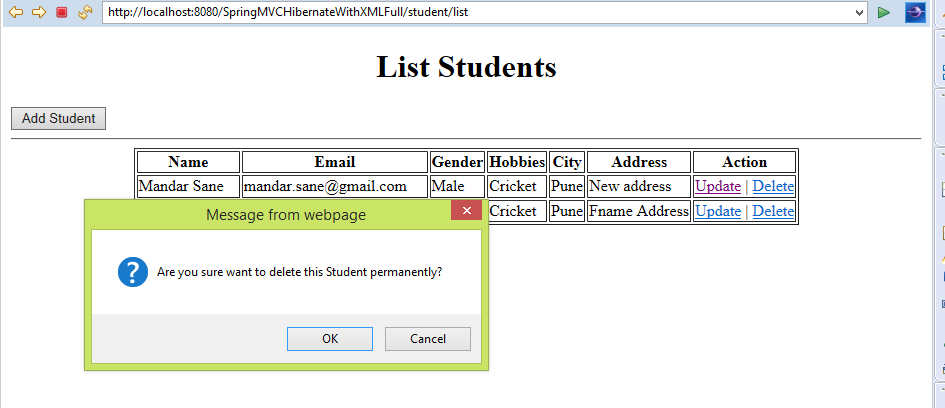
4. Now fill the details as below and click 'Save'.



5. This shows up student list with recent record added as below:

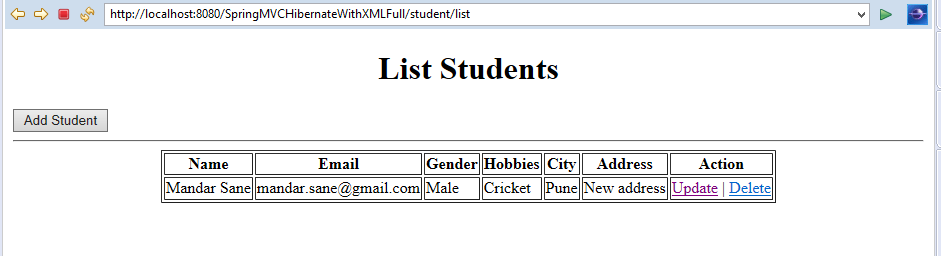
This is **Create** operation of the database table record.

6. Now click on 'Delete' link of second record. It leads to pop up of confirmation box as below:

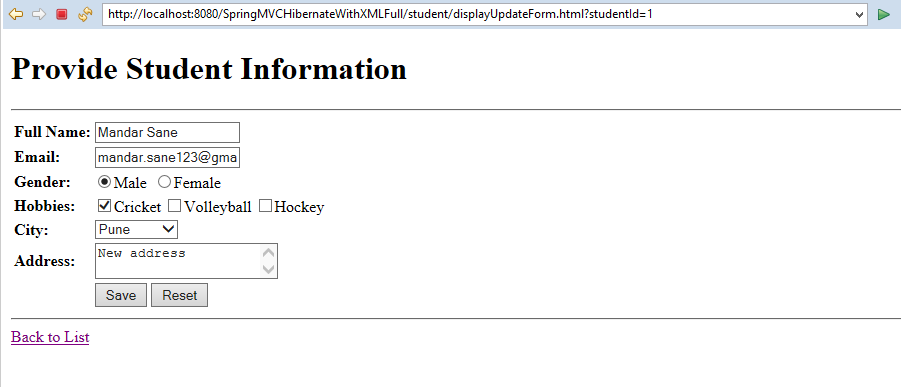


7. Now click OK. It again displays updated list as below.

This is **Delete** operation of the database table record.

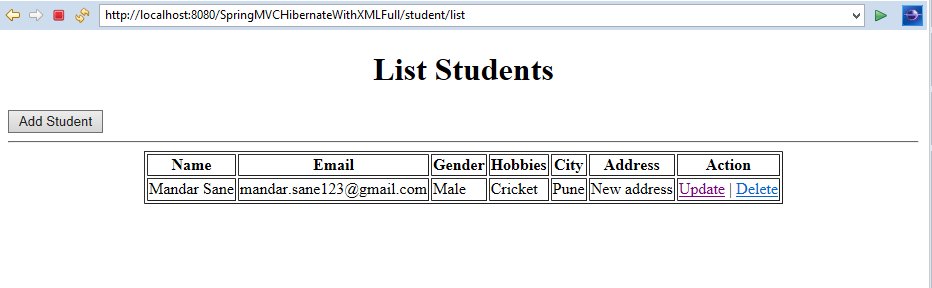


8. Click on 'Update' link of the given record



9. Update email address and click 'Save' button.

Now list with updated record get displayed as below:



This is **Update** operation of the database table record.

**\*\*\*Most important thing is please move to further projects only after complete understanding of this project because next syllabus is largely based on process flow and concepts of this project.**

Thus this project does CRUD operations on given database table by using Spring Framework and Hibernate.

Detailed explanation of this project is given in file 'SpringMVCHibernateXMLProjectExplainantion.docx' in this project.