Source: <https://java2blog.com/spring-mvc-hibernate-mysql-crud-example/>

We have already seen [@Autowired annotation](https://www.java2blog.com/2012/09/autowired-annotation-in-spring.html) but we have used xml configuration to configure beans and inject it to container but if you use  @Component, @Service, @Repository and @Controller annotations and enable component auto scan, spring will automatically import these bean into container and you don’t have to explicitly define them in xml file.  
So basically all 4 annotations are used to register bean definition with spring’s application context.

**@Component:**

@Component is **generic annotation** for bean definition and registers with application context.

**@Service :**

@Service is **specialised component annotation** which is used to annotate classes which belongs to service layer.

**@Repository :**

@Repository annotation is **specialised component annotation**which is used to annotate classes at DAO layer. It also makes unchecked exception eligible for conversion into Spring DataAccessException.

**@Controller :**

@ Controller annotation is **specialised component annotation** which is used to annotate classes at Presentation  layer. It is widely used in Spring MVC applications.

You should not use @Component annotation unless you are sure that it does not belong to @Service, @Repository and @Controller annotation.

### Enable component Scanning :

All these annotations will work only when you use **context:component-scan** in applicationcontext.xml. It basically scans for above 4 annotated classes and registers bean with Spring application context.

|  |  |
| --- | --- |
| 1  2  3 | <context:component-scan base-package="org.arpit.java2blog" /> |

### Spring annotations example :

In this example,we are going to create country object with help of controller, service and Dao classes by using above annotations.  
Lets first create our bean class  
**Country.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | package org.arpit.java2blog.bean;    public class Country{    String countryName;  long population;    public Country() {    super();  }  public Country(String countryName,long population) {    super();    this.countryName = countryName;    this.population=population;  }    public String getCountryName() {    return countryName;  }  public void setCountryName(String countryName) {    this.countryName = countryName;  }  public long getPopulation() {    return population;  }  public void setPopulation(long population) {    this.population = population;  }    } |

**CountryController,java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | package org.arpit.java2blog.controller;    import org.arpit.java2blog.bean.Country;  import org.arpit.java2blog.service.CountryService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;    @Controller ("countryController")  public class CountryController  {       @Autowired      CountryService countryService;        public Country createNewCountry()      {          return countryService.createNewCountry();      }  } |

**CountryService.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package org.arpit.java2blog.service;    import org.arpit.java2blog.bean.Country;  import org.arpit.java2blog.dao.CountryDAO;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;    @Service("countryService")  public class CountryService {      @Autowired  CountryDAO countryDAO;  public Country createNewCountry() {      return countryDAO.createNewCountry();  }    } |

**CountryDAO.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package org.arpit.java2blog.dao;    import org.arpit.java2blog.bean.Country;  import org.springframework.stereotype.Repository;    @Repository("countryDAO")  public class CountryDAO {    public Country createNewCountry() {    // You should get it from database    Country country = new Country("Ïndia", 40000);    return country;  }    } |

**applicationcontext.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation=" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/context          http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">  <context:component-scan base-package="org.arpit.java2blog" />  </beans> |

**SpringApplicationMain.java**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | package org.arpit.java2blog.main;    import org.arpit.java2blog.bean.Country;  import org.arpit.java2blog.controller.CountryController;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;    public class SpringApplicationMain {  public static void main(String[] args) {      ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");    CountryController controller = (CountryController) context.getBean("countryController");    Country country = controller.createNewCountry();    System.out.println("Country Name : " + country.getCountryName());    System.out.println("Country's Population : " + country.getPopulation());  }  } |

When you run above program, you will get below output:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | Aug 08, 2016 12:00:42 AM org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh  INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@1c2709da: startup date [Mon Aug 08 00:00:42 IST 2016]; root of context hierarchy  Aug 08, 2016 12:00:42 AM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions  INFO: Loading XML bean definitions from class path resource [applicationContext.xml]  Country Name : Ïndia  Country's Population : 40000 |

**pom.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61 | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">    <modelVersion>4.0.0</modelVersion>    <groupId>com.arpit.java2blog</groupId>    <artifactId>SpringRestfulWebServicesWithJSONExample</artifactId>    <packaging>war</packaging>    <version>0.0.1-SNAPSHOT</version>    <name>SpringRestfulWebServicesWithJSONExample Maven Webapp</name>    <url>http://maven.apache.org</url>    <dependencies>      <dependency>        <groupId>junit</groupId>        <artifactId>junit</artifactId>        <version>3.8.1</version>        <scope>test</scope>      </dependency>      <dependency>     <groupId>javax.servlet</groupId>     <artifactId>javax.servlet-api</artifactId>     <version>3.1.0</version>    </dependency>      <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-core</artifactId>     <version>${spring.version}</version>    </dependency>    <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-webmvc</artifactId>     <version>${spring.version}</version>    </dependency>     <dependency>              <groupId>com.fasterxml.jackson.core</groupId>              <artifactId>jackson-databind</artifactId>               <version>2.4.1</version>          </dependency>  </dependencies>  <build>    <finalName>SpringRestfulWebServicesWithJSONExample</finalName>      <plugins>     <plugin>      <groupId>org.apache.maven.plugins</groupId>      <artifactId>maven-compiler-plugin</artifactId>      <version>3.1</version>      <configuration>       <source>${jdk.version}</source>       <target>${jdk.version}</target>      </configuration>     </plugin>    </plugins>    </build>  <properties>    <spring.version>4.2.1.RELEASE</spring.version>    <jdk.version>1.7</jdk.version>  </properties>  </project> |

### Spring application configuration:

**3)**Change web.xml as below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26 | <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  version="3.0">  <display-name>Archetype Created Web Application</display-name>    <welcome-file-list>     <welcome-file>index.jsp</welcome-file>      </welcome-file-list>  <servlet>  <servlet-name>springrest</servlet-name>  <servlet-class>    org.springframework.web.servlet.DispatcherServlet  </servlet-class>  <load-on-startup>1</load-on-startup>  </servlet>    <servlet-mapping>  <servlet-name>springrest</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>    </web-app> |

**4)**create a xml file named springrest-servlet.xml in /WEB-INF/ folder.  
**Please change context:component-scan if you want to use different package for spring to search for controller.Please refer to**[**spring mvc hello world example**](https://www.java2blog.com/2015/09/spring-mvc-hello-world-example.html)**for more understanding.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation=" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/context          http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">    <mvc:annotation-driven/>  <context:component-scan base-package="org.arpit.java2blog.controller" />  <bean    class="org.springframework.web.servlet.view.InternalResourceViewResolver">    <property name="prefix">     <value>/WEB-INF/</value>    </property>    <property name="suffix">     <value>.jsp</value>    </property>  </bean>  <mvc:default-servlet-handler/>  </beans> |

### Create bean class

**5)** Create a bean name “Country.java” in org.arpit.java2blog.bean.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31 | package org.arpit.java2blog.bean;    public class Country{    String countryName;  long population;    public Country() {    super();  }  public Country(String countryName,long population) {    super();    this.countryName = countryName;    this.population=population;  }  public String getCountryName() {    return countryName;  }  public void setCountryName(String countryName) {    this.countryName = countryName;  }  public long getPopulation() {    return population;  }  public void setPopulation(long population) {    this.population = population;  }    } |

### Create Controller

**6)** Create a controller named “CountryController.java” in package **org.arpit.java2blog.controller**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | | package org.arpit.java2blog.controller;    import org.arpit.java2blog.bean.Country;  import org.springframework.stereotype.Controller;  import org.springframework.ui.ModelMap;  import org.springframework.web.bind.annotation.ModelAttribute;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;  import org.springframework.web.bind.annotation.RequestParam;    @Controller  public class CountryController {    @ModelAttribute  public Country getCountry(@RequestParam String countryName, @RequestParam  long population)  {    Country country=new Country();    country.setCountryName(countryName);    country.setPopulation(population);    return country;  }  @RequestMapping(value = "/addCountry", method = RequestMethod.POST)  public String addCountry(@ModelAttribute Country country,ModelMap model) {    model.addAttribute("countryName", country.getCountryName());    model.addAttribute("population", country.getPopulation());    return "countryDetails";  }  } |
| 1  2  3 | Create view | |

Modify**index.jsp a**s below

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | <head>  <style>    .blue-button{  background: #25A6E1;  filter: progid: DXImageTransform.Microsoft.gradient( startColorstr='#25A6E1',endColorstr='#188BC0',GradientType=0);  padding:3px 5px;  color:#fff;  font-family:'Helvetica Neue',sans-serif;  font-size:12px;  border-radius:2px;  -moz-border-radius:2px;  -webkit-border-radius:4px;  border:1px solid #1A87B9  }  table {    font-family: "Helvetica Neue", Helvetica, sans-serif;     width: 50%;  }  th {    background: SteelBlue;    color: white;  }  td,th{                  border: 1px solid gray;                  width: 25%;                  text-align: left;                  padding: 5px 10px;              }  </style>  </head>  <body>  <form action="addCountry" method="post">    <table>    <tr>     <th colspan="2">Add Country</th>    </tr>    <tr>     <td>Country</td>     <td><input type="text" name=countryName></td>    </tr>    <tr>     <td>Population</td>     <td><input type="text" name=population>    </td></tr>    <tr>     <td colspan="2"><input type="submit"      class="blue-button" /></td>    </tr>  </table>    </form>  </body> |

Top of Form

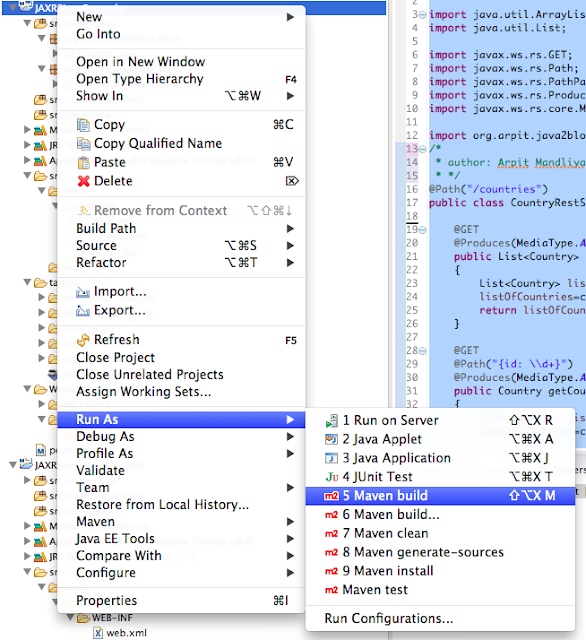
Create **countryDetails.jsp** in /**WEB-INF/**folder

Bottom of Form

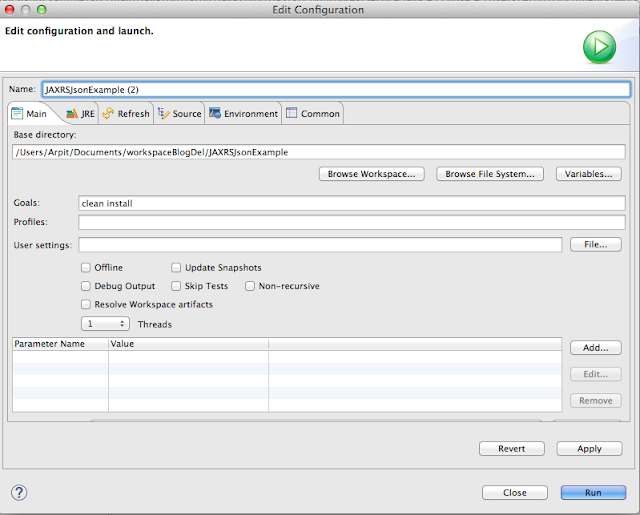
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <%@ page language="java" contentType="text/html; charset=UTF-8"      pageEncoding="UTF-8"%>  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <title>Country Details</title>  </head>  <body>    Added country :  <h3>Country Name: ${countryName}</h3>  <h3>Population : ${population}</h3>  </body>  </html> |

**7)** It ‘s time to do maven build.

Right click on project -> Run as -> Maven build

[](https://www.java2blog.com/wp-content/uploads/2016/08/jerseyMavenBuild-4.png)

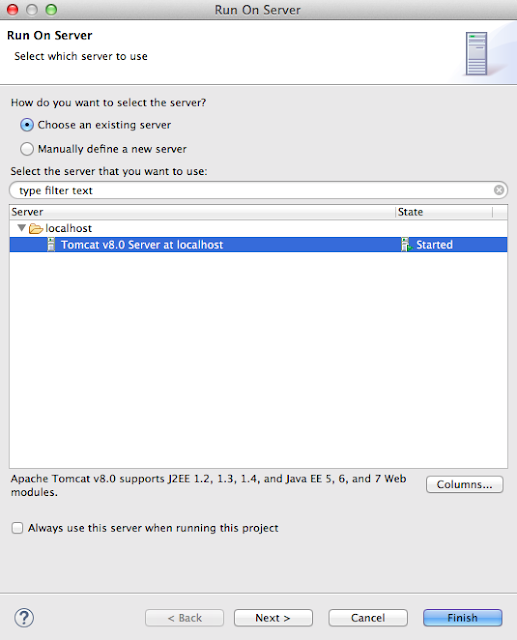
**8)** Provide goals as clean install (given below) and click on run

[](https://www.java2blog.com/wp-content/uploads/2016/08/jerseyMavenCleanInstall-4.png)

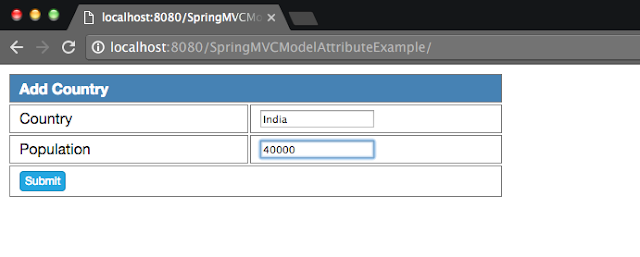
### Run the application

**9)**Right click on project -> run as -> run on server

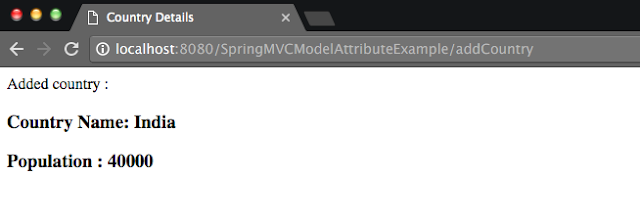
Select apache tomcat and click on finish

[](https://www.java2blog.com/wp-content/uploads/2016/08/RunningProject-4.png)

10) You will see screen as below:

[](https://4.bp.blogspot.com/-cM55I55z71E/V6y1PvA0FLI/AAAAAAAAEnw/9auM-BUNfNE4FgT4KQXsUpjNTKGOVXMGQCLcB/s1600/AddCountry.png)

11) When you click on submit button , you will see below screen.

[](https://4.bp.blogspot.com/-dP2G47MAoTQ/V6y1lv6U_7I/AAAAAAAAEn0/w3NsKbRju4sqAPQTMj9x5zAUaMGwqHoWACLcB/s1600/AddedCountry.png)

[**Previous**](https://java2blog.com/spring-mvc-log4j-example/)

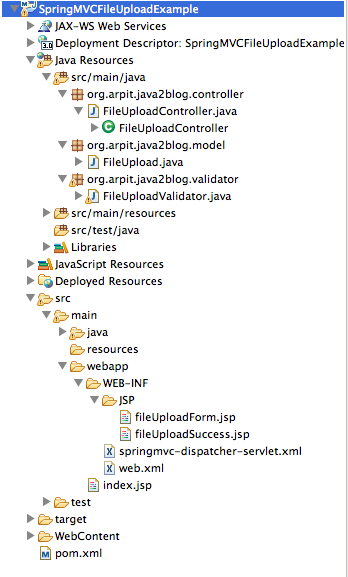
[**Next**](https://java2blog.com/spring-mvc-file-upload-example/)

## Spring MVC tutorial:

* [Spring MVC hello world example](https://www.java2blog.com/2015/09/spring-mvc-hello-world-example.html)
* [Spring MVC Hibernate MySQL example](https://www.java2blog.com/2016/08/spring-mvc-hibernate-mysql-crud-example.html)
* [Spring MVC interceptor example](https://www.java2blog.com/2016/07/spring-mvc-interceptor-example.html)
* [Spring MVC angularjs example](https://www.java2blog.com/2016/08/spring-mvc-angularjs-example.html)
* [Spring MVC @RequestMapping example](https://www.java2blog.com/2016/08/spring-mvc-requestmapping-annotation-example.html)
* [Spring Component,Service, Repository and Controller example](https://www.java2blog.com/2016/08/spring-component-service-repository-and-controller-annotations.html)
* [Spring MVC @ModelAttribute annotation example](https://www.java2blog.com/2016/08/spring-mvc-modelattribute-example.html)
* [Spring MVC @RestController annotation example](https://www.java2blog.com/2016/08/spring-restcontroller-example.html)
* [Spring MultiActionController Example](https://www.java2blog.com/2015/09/spring-mvc-multiactioncontroller-example.html)
* [Spring MVC ModelMap](https://www.java2blog.com/2015/09/spring-mvc-modelmap.html)[Spring MVC file upload example](https://www.java2blog.com/2016/02/spring-mvc-file-upload-example.html)
* [Spring restful web service example](https://www.java2blog.com/2015/09/spring-restful-web-services-example.html)
* [Spring restful web service json example](https://www.java2blog.com/2015/09/spring-restful-web-services-json-example.html)
* [Spring Restful web services CRUD example](https://www.java2blog.com/2016/04/spring-restful-web-services-crud-example.html)
* [Spring security hello world example](https://www.java2blog.com/2016/01/spring-security-hello-world-example.html)
* [Spring security custom login form example](https://www.java2blog.com/2016/01/spring-security-custom-login-form.html)

Spring MVC uses Apache common FileUpload API to upload file. It is very easy to configure file upload in Spring MVC.

#### Project structure for below example:

[](https://4.bp.blogspot.com/-Oev6xba-Auc/VsjOACi9yeI/AAAAAAAAEUc/tuRoneq65Og/s1600/SpringMVCFileUploadFolderStructure.png)

Steps for Spring MVC file upload example:

#### 1) Create [dynamic web project using maven](https://www.java2blog.com/2015/09/how-to-create-dynamic-web-project-using.html) named “SpringMVCFileUploadExample”.

#### 2) Maven dependencies



It requires spring dependency as we have put in spring mvc hello world example. It requires apache common io jars too, so we need to put following dependency for it.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <!-- Apache Common io dependencies -->    <!-- Apache Commons FileUpload -->    <dependency>     <groupId>commons-fileupload</groupId>     <artifactId>commons-fileupload</artifactId>     <version>1.3.1</version>    </dependency>      <!-- Apache Commons IO -->    <dependency>     <groupId>commons-io</groupId>     <artifactId>commons-io</artifactId>     <version>2.4</version>    </dependency> |

So pom.xml will be as below

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72 | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>org.arpit.java2blog</groupId>  <artifactId>SpringMVCFileUploadExample</artifactId>  <packaging>war</packaging>  <version>0.0.1-SNAPSHOT</version>  <name>SpringMVCFileUploadExample Maven Webapp</name>  <url>http://maven.apache.org</url>  <dependencies>    <dependency>     <groupId>junit</groupId>     <artifactId>junit</artifactId>     <version>3.8.1</version>     <scope>test</scope>    </dependency>    <dependency>     <groupId>javax.servlet</groupId>     <artifactId>javax.servlet-api</artifactId>     <version>3.1.0</version>    </dependency>    <!-- Spring dependency -->    <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-core</artifactId>     <version>${spring.version}</version>    </dependency>    <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-webmvc</artifactId>     <version>${spring.version}</version>    </dependency>      <!-- Apache Common io dependencies -->    <!-- Apache Commons FileUpload -->    <dependency>     <groupId>commons-fileupload</groupId>     <artifactId>commons-fileupload</artifactId>     <version>1.3.1</version>    </dependency>      <!-- Apache Commons IO -->    <dependency>     <groupId>commons-io</groupId>     <artifactId>commons-io</artifactId>     <version>2.4</version>    </dependency>  </dependencies>  <build>    <finalName>SpringMVCFileUploadExample</finalName>      <plugins>     <plugin>      <groupId>org.apache.maven.plugins</groupId>      <artifactId>maven-compiler-plugin</artifactId>      <version>3.1</version>      <configuration>       <source>${jdk.version}</source>       <target>${jdk.version}</target>      </configuration>     </plugin>    </plugins>    </build>  <properties>    <spring.version>4.2.1.RELEASE</spring.version>    <jdk.version>1.7</jdk.version>  </properties>    </project> |

#### 3) Create model,controller and validator

Create a class model called “FileUpload.java” in packge **org.arpit.java2blog.model**and it will have variable of type MultiFilePart.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | package org.arpit.java2blog.model;  import org.springframework.web.multipart.MultipartFile;    public class FileUpload {    MultipartFile file;    public MultipartFile getFile() {    return file;  }    public void setFile(MultipartFile file) {    this.file = file;  }  } |

Create class FileUploadController.java in package org.arpit.java2blog.controller . It is spring controller and to understand more above controller, please go through [Spring MVC hello world example.](https://www.java2blog.com/2015/09/spring-mvc-hello-world-example.html)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76 | package org.arpit.java2blog.controller;    import java.io.File;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.InputStream;  import java.io.OutputStream;    import org.arpit.java2blog.model.FileUpload;  import org.arpit.java2blog.validator.FileUploadValidator;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.validation.BindingResult;  import org.springframework.web.bind.annotation.ModelAttribute;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.multipart.MultipartFile;  import org.springframework.web.servlet.ModelAndView;    @Controller  public class FileUploadController {    @Autowired  FileUploadValidator fileValidator;    @RequestMapping("/fileUploadForm")  public ModelAndView getUploadForm(     @ModelAttribute("uploadedFile") FileUpload uploadedFile,     BindingResult result) {    return new ModelAndView("fileUploadForm");  }      // When click on submit, below method will get called  @RequestMapping("/submitFileUpload")  public ModelAndView fileUploaded(     @ModelAttribute("uploadedFile") FileUpload uploadedFile,     BindingResult result) {    InputStream inputStream = null;    OutputStream outputStream = null;      // Getting uploaded file    MultipartFile file = uploadedFile.getFile();    fileValidator.validate(uploadedFile, result);      String fileName = file.getOriginalFilename();      // If it has error, redirect it to same page    if (result.hasErrors()) {     return new ModelAndView("fileUploadForm");    }      try {     inputStream = file.getInputStream();       File newFile = new File("/Users/Arpit/Documents/SpringMVCUploadedFiles/" + fileName);     if (!newFile.exists()) {      newFile.createNewFile();     }     outputStream = new FileOutputStream(newFile);     int read = 0;     byte[] bytes = new byte[1024];       while ((read = inputStream.read(bytes)) != -1) {      outputStream.write(bytes, 0, read);     }    } catch (IOException e) {     // TODO Auto-generated catch block     e.printStackTrace();    }      return new ModelAndView("fileUploadSuccess", "fileName", fileName);  }    } |

We need validator for validations such as no files selected. Create validator FileUploadValidator.java in package **org.arpit.java2blog.validator**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | package org.arpit.java2blog.validator;    import org.arpit.java2blog.model.FileUpload;  import org.springframework.validation.Errors;  import org.springframework.validation.Validator;      public class FileUploadValidator implements Validator{    @Override  public boolean supports(Class clazz) {    //just validate the FileUpload instances    return FileUpload.class.isAssignableFrom(clazz);  }    @Override  public void validate(Object target, Errors errors) {      FileUpload file = (FileUpload)target;      if(file.getFile().getSize()==0){     errors.rejectValue("file", "uploadForm.selectFile",           "Please select a file!");    }  }  } |

#### 4)  Spring configuration

It will have simple web.xml for spring as in hello world example.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | <web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee            http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"      version="3.0">      <display-name>Archetype Created Web Application</display-name>       <welcome-file-list>      <welcome-file>index.jsp</welcome-file>    </welcome-file-list>      <servlet>    <servlet-name>springmvc-dispatcher</servlet-name>    <servlet-class>     org.springframework.web.servlet.DispatcherServlet          </servlet-class>    <load-on-startup>1</load-on-startup>  </servlet>    <servlet-mapping>    <servlet-name>springmvc-dispatcher</servlet-name>    <url-pattern>/</url-pattern>  </servlet-mapping>      </web-app> |

We need to change in “springmvc-dispatcher-servlet.xml” as below

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context"  xmlns:mvc="http://www.springframework.org/schema/mvc" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation=" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/context          http://www.springframework.org/schema/context/spring-context-3.0.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">    <context:component-scan base-package="org.arpit.java2blog.controller" />    <bean    class="org.springframework.web.servlet.view.InternalResourceViewResolver">    <property name="prefix">     <value>/WEB-INF/JSP/</value>    </property>    <property name="suffix">     <value>.jsp</value>    </property>  </bean>     <!-- Validator -->     <bean id="fileValidator" class="org.arpit.java2blog.validator.FileUploadValidator" />    <bean id="multipartResolver"    class="org.springframework.web.multipart.commons.CommonsMultipartResolver" />  <mvc:annotation-driven/>    </beans> |

We need to declare multipartResolver bean in above file to enable file upload. We have also put bean entry of validator.

#### 5) Views

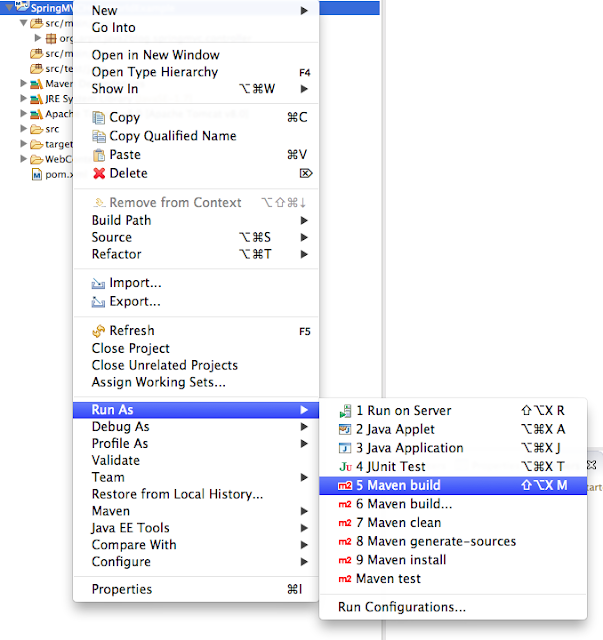
Finally create views for file upload form and file upload success page. Create fileUploadForm.jsp in /WEB-INF/JSP/ folder

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form"%>  <html>  <head>  <script type="text/javascript" src="jquery-1.2.6.min.js"></script>  <title>Spring MVC file upload</title>  </head>  <body>  <center>    <h2>Spring MVC file upload</h2>    <h3>Please select a file to upload !</h3>      <form:form method="post" enctype="multipart/form-data"     modelAttribute="uploadedFile" action="submitFileUpload.htm">     <table>      <tr>       <td>Upload File: </td>       <td><input type="file" name="file" />       </td>       <td style="color: red; font-style: italic;"><form:errors         path="file" />       </td>      </tr>      <tr>       <td> </td>       <td><input type="submit" value="Upload" />       </td>       <td> </td>      </tr>     </table>    </form:form>  </center>  </body>  </html> |

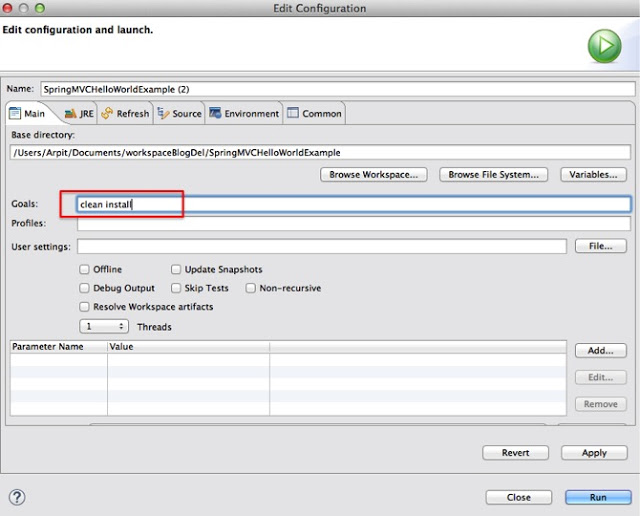
create filUploadSuccess.jsp /WEB-INF/JSP/ folder

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>    <html>  <body>  <h2>Spring MVC file upload example</h2>    FileName : "  <strong> ${fileName} </strong>" - Uploaded Successful.    </body>  </html> |

#### 6) Its time for maven build

[](https://www.java2blog.com/wp-content/uploads/2016/02/mavenBuild.png)

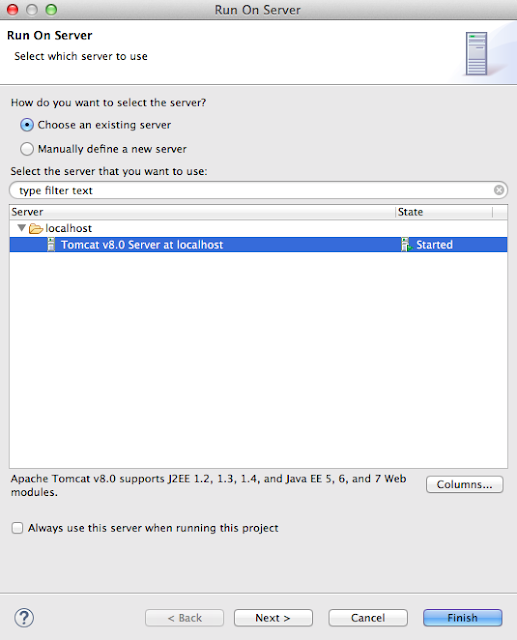
Provide goals as clean install (given below) and click on run

[](https://www.java2blog.com/wp-content/uploads/2016/02/springhelloworldCleanBuild.jpg)

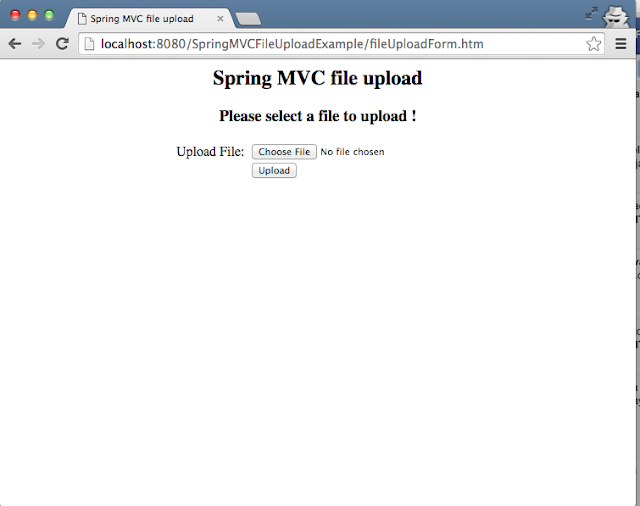
#### 7) Run the application

Right click on project -> run as -> run on server

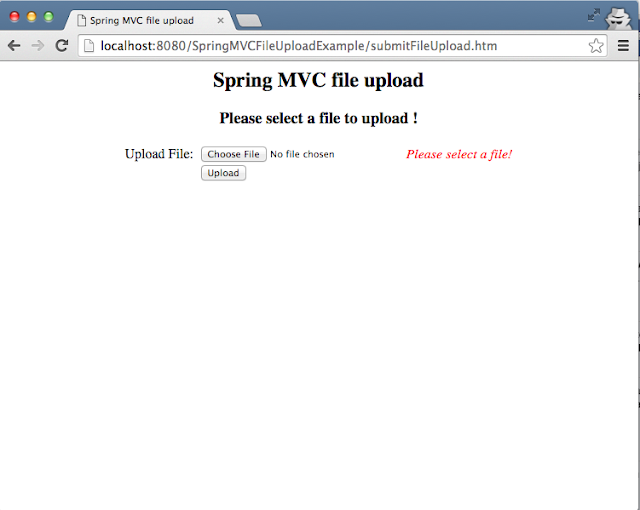
Select apache tomcat and click on finish

[](https://www.java2blog.com/wp-content/uploads/2016/02/RunningProject-2.png)

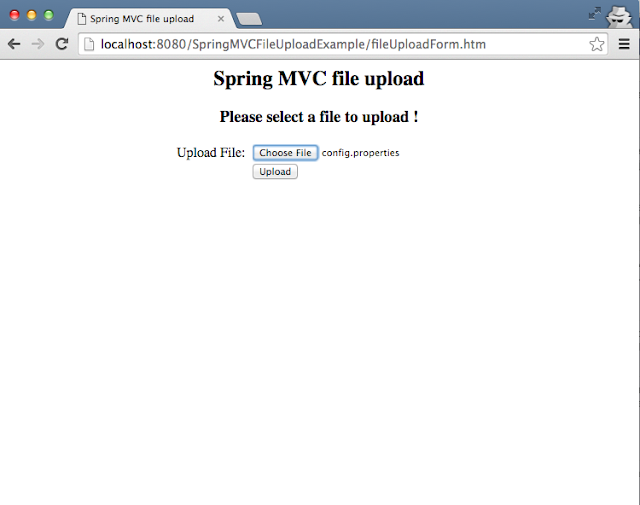
**You will see below screen:**  
**URL : http://localhost:8080/SpringMVCFileUploadExample/fileUploadForm.htm**

[](https://1.bp.blogspot.com/-_jteUoytxN4/VsjIXsk5EKI/AAAAAAAAET8/-zQqn7zylbU/s1600/SpringMVCFileUpload.png)

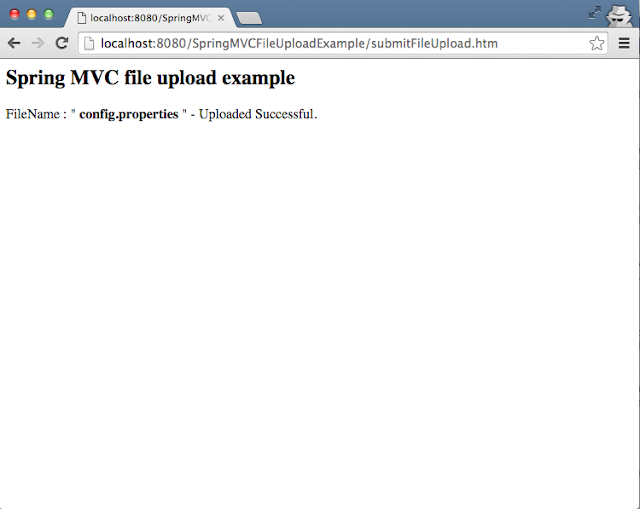
**If you directly click on upload button without selecting any file, it will give you error as “Please select a file”**

[](https://4.bp.blogspot.com/-yKG0KgZ_XE0/VsjI50vXLQI/AAAAAAAAEUA/wXs9VH_tKgk/s1600/SpringMVCFileUploadValidation.png)

**Lets upload “config.properties”**

[](https://3.bp.blogspot.com/-2HgbBYnzwpc/VsjJEeAsUPI/AAAAAAAAEUE/56sMfDN_cyg/s1600/SpringMVCSelectFile.png)

**If your file is successfully uploaded, you will see below screen.**

[](https://3.bp.blogspot.com/-yGoiHvcDDyo/VsjJO1MBcVI/AAAAAAAAEUI/mezcu6oomOg/s1600/SpringMVCFileUploadSuccess.png)

**You can check in the folder whose path you have given in above application.**

[https://1.bp.blogspot.com/-jSI0eMOmGCQ/VsjJZDHfaBI/AAAAAAAAEUM/eETUqQZ055M/s1600/FolderAfterUpload.png](https://1.bp.blogspot.com/-jSI0eMOmGCQ/VsjJZDHfaBI/AAAAAAAAEUM/eETUqQZ055M/s1600/FolderAfterUpload.png)

#### 8) Source code

[Download](https://github.com/arpitmandliya/SpringMVCFileUploadExample/archive/master.zip)

Last updated : June 26th, 2017

[**NO COMMENTS**](https://java2blog.com/spring-mvc-spring-data-hibernate-mysql-example/#respond)

# Spring MVC + Spring Data + Hibernate + MySQL example

[**Previous**](https://java2blog.com/spring-data-jpa-example/)

[**Next**](https://java2blog.com/spring-boot-hibernate-example/)

In this post, we are going to see integration of Spring MVC,Spring Data,hibernate and mysql CRUD example.  
We have already seen [Spring MVC, hibernate and mysql example](https://www.java2blog.com/2016/08/spring-mvc-hibernate-mysql-crud-example.html) in previous tutorial.  
[Spring Data JPA](https://www.java2blog.com/2016/09/spring-data-jpa-example.html) provides CRUD API, so you don’t have to write boiler plate code. You just need to create repository interface and spring will provide implementation automatically.

## Spring MVC tutorial:

* [Spring MVC hello world example](https://www.java2blog.com/2015/09/spring-mvc-hello-world-example.html)
* [Spring MVC Hibernate MySQL example](https://www.java2blog.com/2016/08/spring-mvc-hibernate-mysql-crud-example.html)
* [Spring MVC interceptor example](https://www.java2blog.com/2016/07/spring-mvc-interceptor-example.html)
* [Spring MVC angularjs example](https://www.java2blog.com/2016/08/spring-mvc-angularjs-example.html)
* [Spring MVC @RequestMapping example](https://www.java2blog.com/2016/08/spring-mvc-requestmapping-annotation-example.html)
* [Spring Component,Service, Repository and Controller example](https://www.java2blog.com/2016/08/spring-component-service-repository-and-controller-annotations.html)
* [Spring MVC @ModelAttribute annotation example](https://www.java2blog.com/2016/08/spring-mvc-modelattribute-example.html)
* [Spring MVC @RestController annotation example](https://www.java2blog.com/2016/08/spring-restcontroller-example.html)
* [Spring MultiActionController Example](https://www.java2blog.com/2015/09/spring-mvc-multiactioncontroller-example.html)
* [Spring MVC ModelMap](https://www.java2blog.com/2015/09/spring-mvc-modelmap.html)
* [Spring MVC file upload example](https://www.java2blog.com/2016/02/spring-mvc-file-upload-example.html)
* [Spring restful web service example](https://www.java2blog.com/2015/09/spring-restful-web-services-example.html)
* [Spring restful web service json example](https://www.java2blog.com/2015/09/spring-restful-web-services-json-example.html)
* [Spring Restful web services CRUD example](https://www.java2blog.com/2016/04/spring-restful-web-services-crud-example.html)
* [Spring security hello world example](https://www.java2blog.com/2016/01/spring-security-hello-world-example.html)
* [Spring security custom login form example](https://www.java2blog.com/2016/01/spring-security-custom-login-form.html)

Here are steps to create a project with Spring MVC , Spring data, hibernate and mySQL crud example.

### Source code:

[Download](https://github.com/arpitmandliya/SpringMVCSpringDataHibernateExample/archive/master.zip)

**1)**Create a [dynamic web project using maven in eclipse](https://www.java2blog.com/2015/09/how-to-create-dynamic-web-project-using.html) named “SpringMVCSpringDataHibernateExample”

### Maven dependencies

**2)**We will use Spring 4 and hibernate 4 for this project.

**pom.xml**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103  104  105  106  107  108  109  110  111  112  113  114  115  116  117  118  119  120  121  122  123  124  125  126  127  128 | <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.arpit.java2blog</groupId>  <artifactId>SpringMVCHibernateCRUDExample</artifactId>  <packaging>war</packaging>  <version>0.0.1-SNAPSHOT</version>  <name>SpringMVCHibernateCRUDExample Maven Webapp</name>  <url>http://maven.apache.org</url>  <dependencies>    <dependency>     <groupId>junit</groupId>     <artifactId>junit</artifactId>     <version>3.8.1</version>     <scope>test</scope>    </dependency>    <dependency>     <groupId>javax.servlet</groupId>     <artifactId>javax.servlet-api</artifactId>     <version>3.1.0</version>    </dependency>      <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-core</artifactId>     <version>${spring.version}</version>    </dependency>    <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-webmvc</artifactId>     <version>${spring.version}</version>    </dependency>    <dependency>     <groupId>com.fasterxml.jackson.core</groupId>     <artifactId>jackson-databind</artifactId>     <version>2.4.1</version>    </dependency>    <!-- Hibernate -->    <dependency>     <groupId>org.hibernate</groupId>     <artifactId>hibernate-core</artifactId>     <version>${hibernate.version}</version>    </dependency>    <dependency>     <groupId>org.hibernate</groupId>     <artifactId>hibernate-entitymanager</artifactId>     <version>${hibernate.version}</version>    </dependency>      <!-- Apache Commons DBCP -->    <dependency>     <groupId>commons-dbcp</groupId>     <artifactId>commons-dbcp</artifactId>     <version>1.4</version>    </dependency>    <!-- Spring ORM -->    <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-orm</artifactId>     <version>${spring.version}</version>    </dependency>    <dependency>     <groupId>org.springframework.data</groupId>     <artifactId>spring-data-jpa</artifactId>     <version>1.8.0.RELEASE</version>    </dependency>    <!-- AspectJ -->    <dependency>     <groupId>org.aspectj</groupId>     <artifactId>aspectjrt</artifactId>     <version>${org.aspectj-version}</version>    </dependency>    <dependency>     <groupId>mysql</groupId>     <artifactId>mysql-connector-java</artifactId>     <version>5.1.6</version>    </dependency>    <!-- https://mvnrepository.com/artifact/jstl/jstl -->    <dependency>     <groupId>jstl</groupId>     <artifactId>jstl</artifactId>     <version>1.2</version>    </dependency>    <dependency>     <groupId>org.springframework</groupId>     <artifactId>spring-aop</artifactId>     <version>4.2.4.RELEASE</version>    </dependency>    <dependency>     <groupId>javax.servlet</groupId>     <artifactId>jsp-api</artifactId>     <version>2.0</version>     <scope>provided</scope>    </dependency>  </dependencies>  <build>    <finalName>SpringMVCHibernateCRUDExample</finalName>      <plugins>     <plugin>      <groupId>org.apache.maven.plugins</groupId>      <artifactId>maven-compiler-plugin</artifactId>      <version>3.1</version>      <configuration>       <source>${jdk.version}</source>       <target>${jdk.version}</target>      </configuration>     </plugin>     <plugin>      <groupId>org.apache.maven.plugins</groupId>      <artifactId>maven-war-plugin</artifactId>      <configuration>       <failOnMissingWebXml>false</failOnMissingWebXml>      </configuration>     </plugin>    </plugins>    </build>  <properties>    <spring.version>4.2.1.RELEASE</spring.version>    <security.version>4.0.3.RELEASE</security.version>    <jdk.version>1.7</jdk.version>    <hibernate.version>4.3.5.Final</hibernate.version>    <org.aspectj-version>1.7.4</org.aspectj-version>  </properties>  </project> |

#### Spring application configuration:

**3)**Change web.xml as below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns="http://java.sun.com/xml/ns/javaee"  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  version="3.0">  <display-name>Archetype Created Web Application</display-name>  <context-param>    <param-name>contextConfigLocation</param-name>    <param-value>/WEB-INF/applicationcontext.xml</param-value>  </context-param>  <listener>      <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <servlet>  <servlet-name>spring</servlet-name>  <servlet-class>    org.springframework.web.servlet.DispatcherServlet  </servlet-class>  <load-on-startup>1</load-on-startup>  </servlet>    <servlet-mapping>  <servlet-name>spring</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>    </web-app> |

**4)**create a xml file named spring-servlet.xml in /WEB-INF/ folder.  
**Please change context:component-scan if you want to use different package for spring to search for controller.Please refer to**[**spring mvc hello world example**](https://www.java2blog.com/2015/09/spring-mvc-hello-world-example.html)**for more understanding.**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48 | <?xml version="1.0" encoding="UTF-8"?>  <beans:beans xmlns="http://www.springframework.org/schema/mvc"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:beans="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx"  xmlns:jpa="http://www.springframework.org/schema/data/jpa"  xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd    http://www.springframework.org/schema/data/jpa http://www.springframework.org/schema/data/jpa/spring-jpa.xsd    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">    <annotation-driven />  <context:component-scan base-package="org.arpit.java2blog" />  <resources mapping="/resources/\*\*" location="/resources/" />    <beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">    <beans:property name="prefix" value="/WEB-INF/views/" />    <beans:property name="suffix" value=".jsp" />  </beans:bean>    <beans:bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">    <beans:property name="entityManagerFactory" ref="entityManagerFactory" />  </beans:bean>  <jpa:repositories base-package="org.arpit.java2blog.repository" />    <beans:bean id="entityManagerFactory"    class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">    <beans:property name="dataSource" ref="dataSource" />    <beans:property name="packagesToScan" value="org.arpit.java2blog" />    <beans:property name="jpaVendorAdapter">     <beans:bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter">      <beans:property name="generateDdl" value="true" />     </beans:bean>    </beans:property>  </beans:bean>    <beans:bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"    destroy-method="close">    <beans:property name="driverClassName" value="com.mysql.jdbc.Driver" />    <beans:property name="url"     value="jdbc:mysql://localhost:3306/CountryData" />    <beans:property name="username" value="root" />    <beans:property name="password" value="arpit123" />  </beans:bean >    </beans:beans> |

In Spring-servlet.xml, we have done hibernate and spring data configuration.  
**dataSource** bean is used to specify java data source. We need to provide driver, URL , Username and Password.  
**transactionManager** bean is used to configure hibernate transaction manager.

Create a applicationcontext.xml in WEB-INF folder, this file is used for bean configuration as we are using spring-servlet.xml for bean configuration , we will keep this file empty.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | <!--?xml version="1.0" encoding="UTF-8"?-->  <beans:beans xmlns="http://www.springframework.org/schema/mvc"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:beans="http://www.springframework.org/schema/beans"  xmlns:context="http://www.springframework.org/schema/context" xmlns:tx="http://www.springframework.org/schema/tx"  xsi:schemaLocation="http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd    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/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd">  </beans:beans> |

### Create bean class

**4)** Create a bean name “Country.java” in org.arpit.java2blog.bean.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | package org.arpit.java2blog.model;    import javax.persistence.Column;  import javax.persistence.Entity;  import javax.persistence.GeneratedValue;  import javax.persistence.GenerationType;  import javax.persistence.Id;  import javax.persistence.Table;    /\*  \* This is our model class and it corresponds to Country table in database  \*/  @Entity  @Table(name="COUNTRY")  public class Country{    @Id  @Column(name="id")  @GeneratedValue(strategy=GenerationType.IDENTITY)  int id;    @Column(name="countryName")  String countryName;    @Column(name="population")  long population;    public Country() {    super();  }  public Country(int i, String countryName,long population) {    super();    this.id = i;    this.countryName = countryName;    this.population=population;  }  public int getId() {    return id;  }  public void setId(int id) {    this.id = id;  }  public String getCountryName() {    return countryName;  }  public void setCountryName(String countryName) {    this.countryName = countryName;  }  public long getPopulation() {    return population;  }  public void setPopulation(long population) {    this.population = population;  }    } |

**@Entity** is used for making a persistent pojo class.For this java class,you will have corresponding table in database. @Column is used to map annotated attribute to corresponding column in table. So Create Country table in mysql database with following code:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE TABLE COUNTRY  (     id int PRIMARY KEY NOT NULL AUTO\_INCREMENT,     countryName varchar(100) NOT NULL,     population int NOT NULL  )  ; |

### Create Controller

**5)** Create a controller named “CountryController.java” in package **org.arpit.java2blog.controller**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66 | package org.arpit.java2blog.controller;      import java.util.List;    import org.arpit.java2blog.model.Country;  import org.arpit.java2blog.service.CountryService;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.ui.Model;  import org.springframework.web.bind.annotation.ModelAttribute;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestBody;  import org.springframework.web.bind.annotation.RequestMapping;  import org.springframework.web.bind.annotation.RequestMethod;    @Controller  public class CountryController {    @Autowired  CountryService countryService;    @RequestMapping(value = "/getAllCountries", method = RequestMethod.GET, headers = "Accept=application/json")  public String getCountries(Model model) {      List listOfCountries = countryService.getAllCountries();    model.addAttribute("country", new Country());    model.addAttribute("listOfCountries", listOfCountries);    return "countryDetails";  }    @RequestMapping(value = "/getCountry/{id}", method = RequestMethod.GET, headers = "Accept=application/json")  public Country getCountryById(@PathVariable int id) {    return countryService.getCountry(id);  }    @RequestMapping(value = "/addCountry", method = RequestMethod.POST, headers = "Accept=application/json")  public String addCountry(@ModelAttribute("country") Country country) {    if(country.getId()==0)    {     countryService.addCountry(country);    }    else    {     countryService.updateCountry(country);    }      return "redirect:/getAllCountries";  }    @RequestMapping(value = "/updateCountry/{id}", method = RequestMethod.GET, headers = "Accept=application/json")  public String updateCountry(@PathVariable("id") int id,Model model) {     model.addAttribute("country", this.countryService.getCountry(id));           model.addAttribute("listOfCountries", this.countryService.getAllCountries());           return "countryDetails";  }    @RequestMapping(value = "/deleteCountry/{id}", method = RequestMethod.GET, headers = "Accept=application/json")  public String deleteCountry(@PathVariable("id") int id) {    countryService.deleteCountry(id);     return "redirect:/getAllCountries";    }  } |

### Create Repository interface

Create a interface called CountryRepository**.java**in package **org.arpit.java2blog.repository.  You don’t have to provide implementation of this interface.**Spring automatically provides implementation of common methods such as save, delete , findOne once you extends to CrudRepository Interface.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | package org.arpit.java2blog.repository;    import org.arpit.java2blog.model.Country;  import org.springframework.data.repository.CrudRepository;    public interface CountryRepository extends CrudRepository<Country,Integer> {    } |

CrudRepository has two generic argument.  
**Country:**It is for what type of entity you are going to save in database.  
**Integer :** It is datatype of identifier for entity object.

### Create Service class

**6)** Create a class CountryService.java in package **org.arpit.java2blog.service**  
It is service level class. It will call DAO layer class.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53 | package org.arpit.java2blog.service;    import java.util.ArrayList;  import java.util.Iterator;  import java.util.List;    import org.arpit.java2blog.model.Country;  import org.arpit.java2blog.repository.CountryRepository;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;    @Service("countryService")  public class CountryService {    @Autowired  CountryRepository countryRepository;    @Transactional  public List getAllCountries() {    List countries=new ArrayList();    Iterable countriesIterable=countryRepository.findAll();    Iterator countriesIterator=countriesIterable.iterator();    while(countriesIterator.hasNext())    {     countries.add(countriesIterator.next());    }    return countries;  }    @Transactional  public Country getCountry(int id) {    return countryRepository.findOne(id);  }    @Transactional  public void addCountry(Country country) {    countryRepository.save(country);  }    @Transactional  public void updateCountry(Country country) {    countryRepository.save(country);    }    @Transactional  public void deleteCountry(int id) {    countryRepository.delete(id);  }  } |

@Service is [specialised component annotation](https://www.java2blog.com/2016/08/spring-component-service-repository-and-controller-annotations.html) which is used to create bean at Service layer.

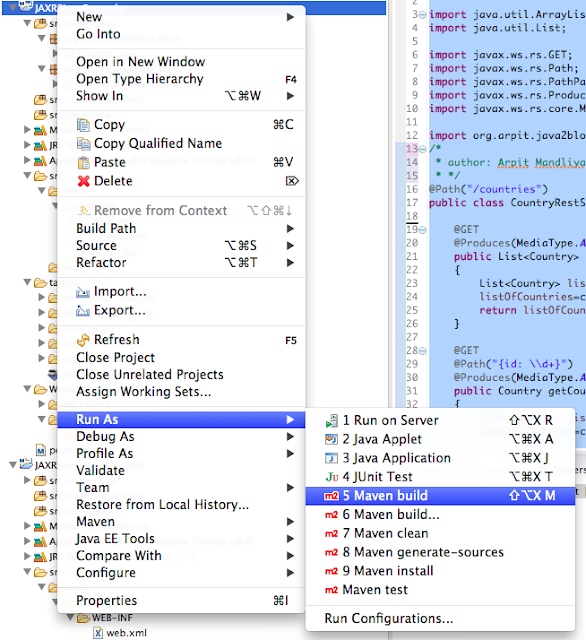
### Create view

Create view called **countryDetails.jsp in WEB-INF/view/ folder.**

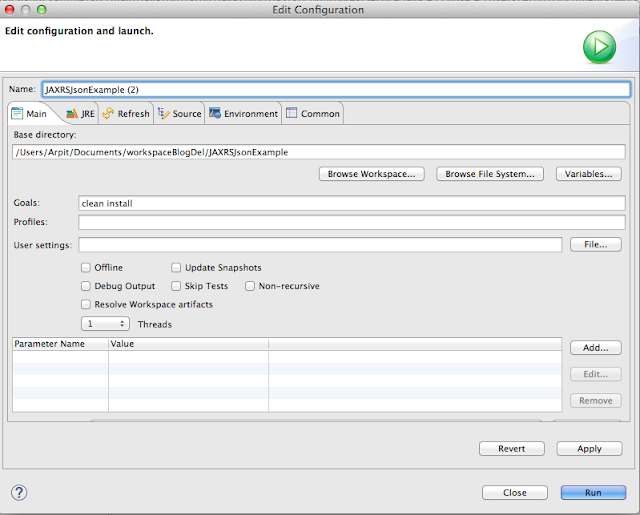
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>  <%@ taglib uri="http://www.springframework.org/tags" prefix="spring" %>  <%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>  <html>  <head>  <style>  .blue-button{  background: #25A6E1;  filter: progid: DXImageTransform.Microsoft.gradient( startColorstr='#25A6E1',endColorstr='#188BC0',GradientType=0);  padding:3px 5px;  color:#fff;  font-family:'Helvetica Neue',sans-serif;  font-size:12px;  border-radius:2px;  -moz-border-radius:2px;  -webkit-border-radius:4px;  border:1px solid #1A87B9  }  table {    font-family: "Helvetica Neue", Helvetica, sans-serif;     width: 50%;  }  th {    background: SteelBlue;    color: white;  }  td,th{                  border: 1px solid gray;                  width: 25%;                  text-align: left;                  padding: 5px 10px;              }  </style>  </head>  <body>  <form:form method="post" modelAttribute="country" action="${pageContext.request.contextPath}/addCountry">  <table>    <tr>     <th colspan="2">Add Country</th>    </tr>    <tr>    <form:hidden path="id" />            <td><form:label path="countryName">Country Name:</form:label></td>            <td><form:input path="countryName" size="30" maxlength="30"></form:input></td>          </tr>    <tr>         <td><form:label path="population">Population:</form:label></td>            <td><form:input path="population" size="30" maxlength="30"></form:input></td>    </tr>    <tr>     <td colspan="2"><input type="submit"      class="blue-button" /></td>    </tr>  </table>  </form:form>  </br>  <h3>Country List</h3>  <c:if test="${!empty listOfCountries}">  <table class="tg">  <tr>    <th width="80">Id</th>    <th width="120">Country Name</th>    <th width="120">Population</th>    <th width="60">Edit</th>    <th width="60">Delete</th>  </tr>  <c:forEach items="${listOfCountries}" var="country">    <tr>     <td>${country.id}</td>     <td>${country.countryName}</td>     <td>${country.population}</td>     <td><a href="<c:url value='/updateCountry/${country.id}' />" >Edit</a></td>     <td><a href="<c:url value='/deleteCountry/${country.id}' />" >Delete</a></td>    </tr>  </c:forEach>  </table>  </c:if>  </body>  </html> |

**8)** It ‘s time to do maven build.

Right click on project -> Run as -> Maven build

[](https://www.java2blog.com/wp-content/uploads/2016/09/jerseyMavenBuild.png)

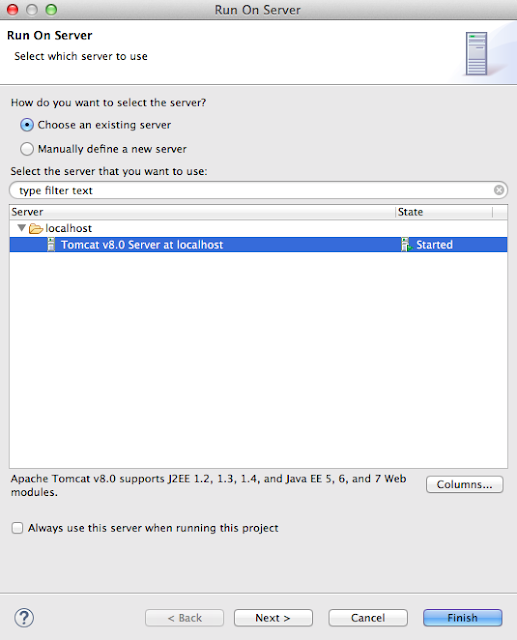
**9)** Provide goals as clean install (given below) and click on run

[](https://www.java2blog.com/wp-content/uploads/2016/09/jerseyMavenCleanInstall.png)

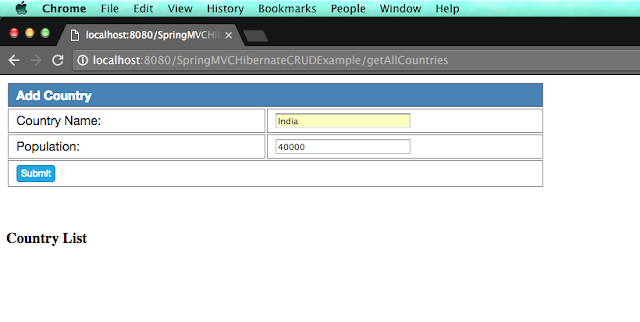
### Run the application

**10)**Right click on project -> run as -> run on server

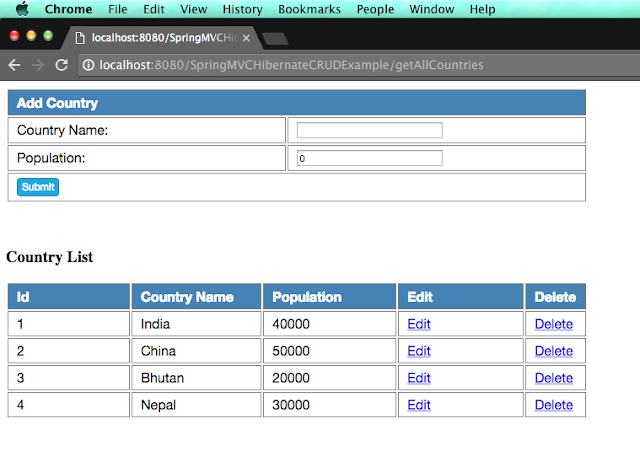
Select apache tomcat and click on finish

[](https://www.java2blog.com/wp-content/uploads/2016/09/RunningProject-1.png)

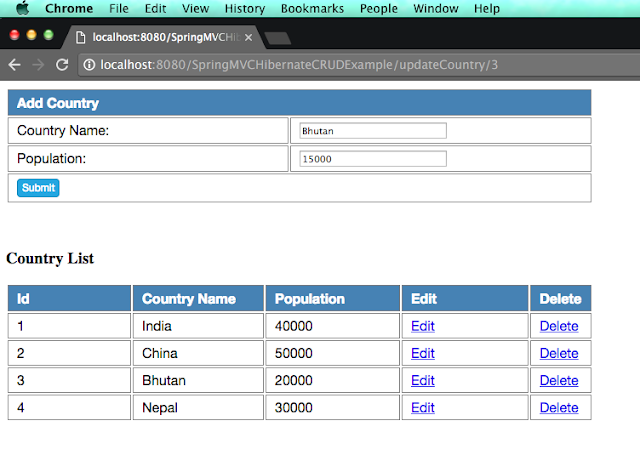
**11)**Now lets hit below URL to getAllCountries.  
**http://localhost:8080/SpringMVCSpringDataHibernateExample/getAllCountries**  
You will get below screen:

[](https://1.bp.blogspot.com/-yF054QAD098/V78pcPNEIXI/AAAAAAAAEqU/VMyCoMhA7aAyiID0VZtftUssAGtA24e2QCLcB/s1600/emptyCountryList.png)

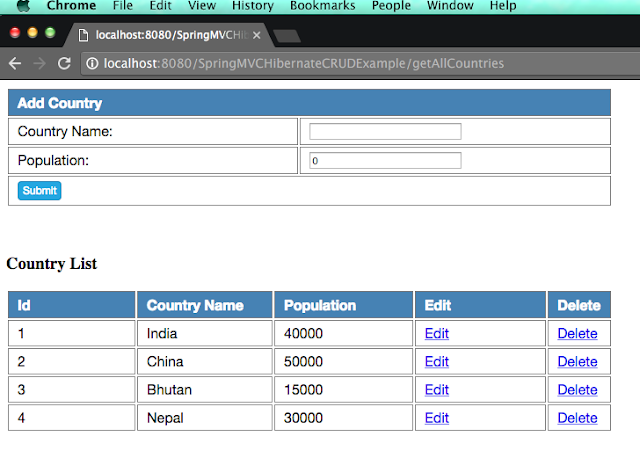
As you can see, we did not add any country to the list, so it is empty.  
Lets add Country named India to country list and click submit.  
Similarly we will add China , Bhutan and Nepal respectively and you will see below screen.

[](https://2.bp.blogspot.com/-h3ZBypVfan8/V78qJjOorZI/AAAAAAAAEqY/YY1nI-_6l78QXRhsZJJUIW6FHgVaEsSSQCLcB/s1600/AllCountryList.png)

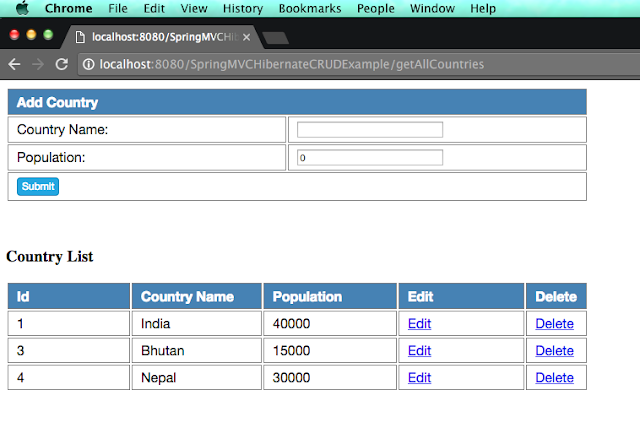
Lets edit population of Bhutan to 15000. Click on edit button corresponds to Bhutan.

[](https://4.bp.blogspot.com/-1C8bGRZZWkQ/V78qgzj3bnI/AAAAAAAAEqc/GeUWQOP8pqAxnvK9GoBPgGFMJ3xBT1akwCLcB/s1600/editBhutanPopulation.png)

**When you click on submit, you will get below screen.**

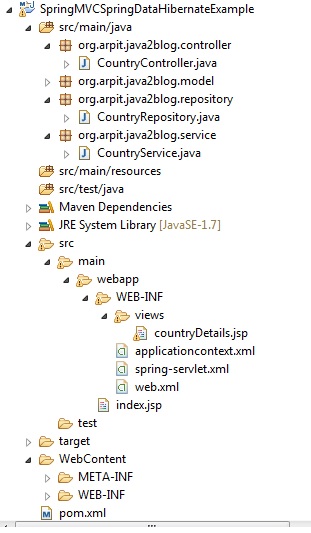
[](https://2.bp.blogspot.com/-V7MlMaKHAq8/V78qrSed6RI/AAAAAAAAEqg/Fs4vncFRwUcZZqdfUZblIYc8ZQbwt9CEwCLcB/s1600/editBhutanDone.png)

Lets delete country China from above list, click on delete button corresponds to 2nd row.  
You will see below screen.

[](https://4.bp.blogspot.com/-jG6QCwQBZXE/V78rCP_nOJI/AAAAAAAAEqk/-Y7vEid1wpw3D_l1LnuVl_Sw7_YJ9MJBwCLcB/s1600/deleteCountryChina.png)

**As you can see, china got deleted from above list.**

#### Project structure

[](https://4.bp.blogspot.com/-25vk7SebZoE/V9fLnp0a9GI/AAAAAAAAEuQ/ITS4Kui9u6UEBGIqVACKQx_B6WbjYcN1wCLcB/s1600/SpringMVCSpringData.jpg)