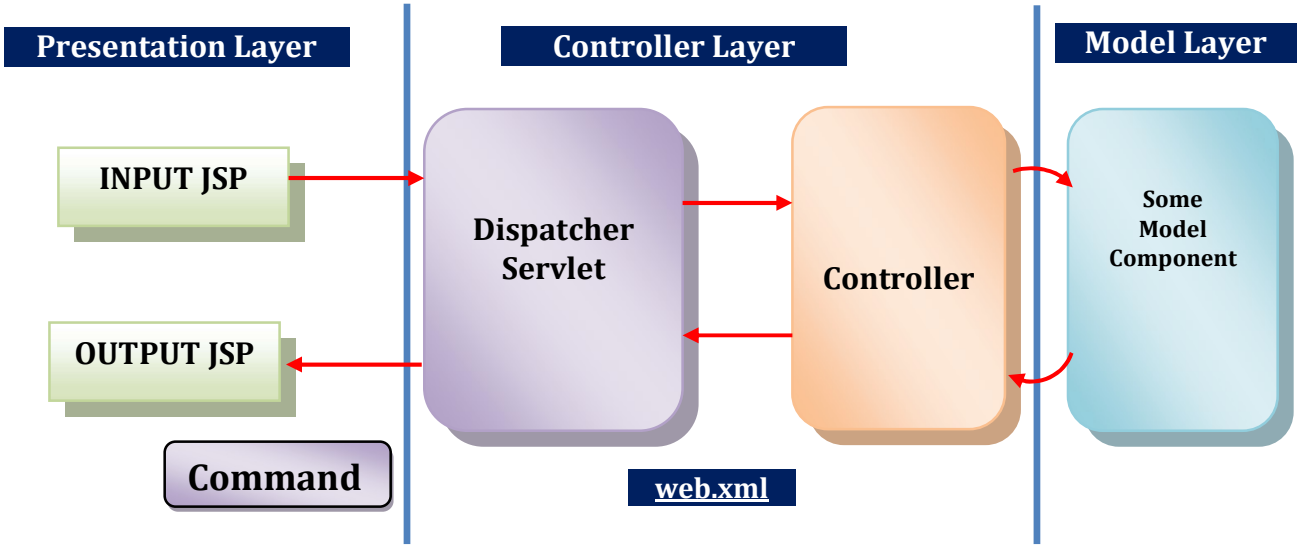
Spring 5.2 MVC

Author
Srinivas Dande

Spring MVC

- Spring MVC is Web Framework which is used to develop web applications easily and quickly with less maintenance.
- Spring MVC covers Web Layer which includes Presentation layer and Controller layer.
- Spring MVC is implemented based on:
  - MVC Architecture.
  - Front Controller Design Pattern.
  - Servlets and JSP.

Spring MVC Basic Architecture



- You need to configure DispatcherServlet to start working with Spring MVC.
- DispatcherServlet can be be configured in XML configuration and Java Configuration differently.
- In the case of XML Configuration, You need to Configure DispatcherServlet in web.xml as follows.

```
<servlet>
        <servlet-name>jlcindia</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>jlcindia</servlet-name>
        <url-pattern>*.jlc</url-pattern>
</servlet-mapping>
```

- In the case of Java Configuration, You need to write Initializer class by extending AbstractAnnotationConfigDispatcherServletInitializer as follows.

```
public class JLCWebAppInitializer extends
AbstractAnnotationConfigDispatcherServletInitializer {

        @Override
        protected Class<?>[] getRootConfigClasses() {
                return new Class[] { JLCConfig.class };
        }

        @Override
        protected Class<?>[] getServletConfigClasses() {
                return new Class[] { JLCConfig.class };
        }

        @Override
        protected String[] getServletMappings() {
                return new String[] { "/" };
        }
}
```

- At web container start up,
  - DispatcherServlet will be loaded, instansiated and initialized by calling init () method.
  - init () of DispatcherServlet will try to identify the Spring Configurations
  - DispatcherServlet creates the ApplicationContext object by reading all the beans specified in the Spring Configurations.

Ex:

```
public class DispatcherServlet extends HttpServlet{
ApplicationContext ctx=null;
public void init(ServletConfig cfg){

//Creates the ApplicationContext object
//Reading Beans from Spring Configuration
//Creates the Bean Instanes and Places them in Spring Container.

}
...
}
```

After Deploying and Starting SpringMVC based web Application, Following tasks will happen at Web Container start-up:

1) Web Container loads, creates and initializes DispatcherServlet by calling init () method.
2) DispatcherServlet's init () performs the following:
    a. Identifies Spring Configuration Document file.
    b. Spring Container instance will be created by reading all the beans from Identified Spring configuration Document.
    c. Initializes DispatcherServlet with Spring Container instances.

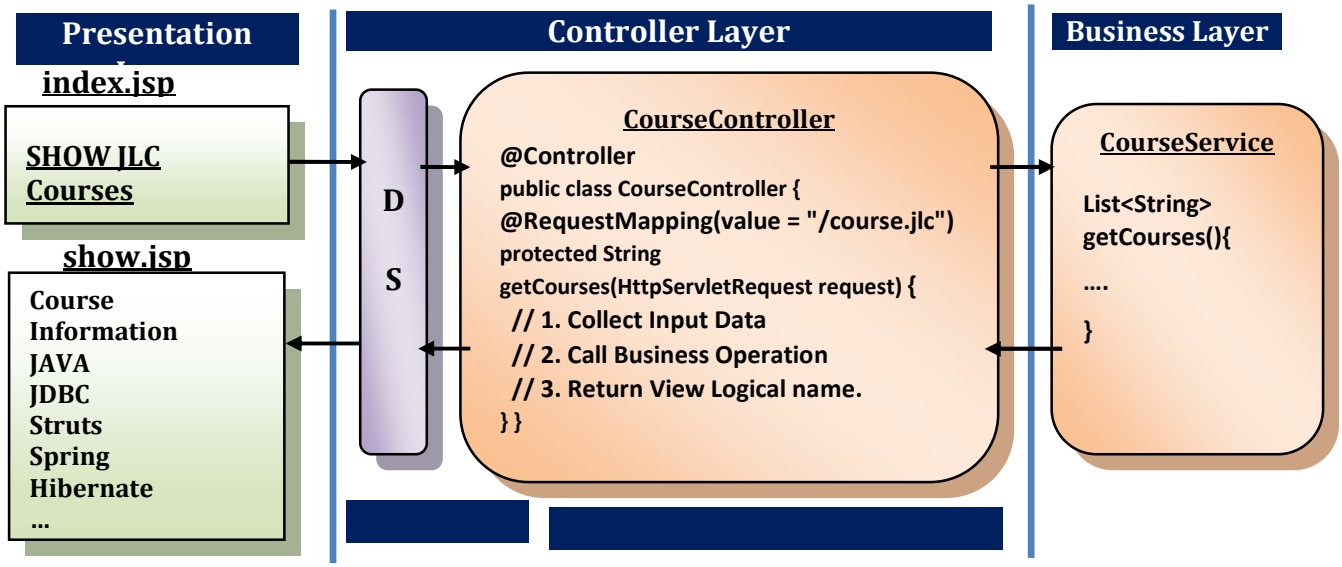
Following tasks will happen at Web Container shutdown-time

1) Web Container calls destroy () method of DispatcherServlet.
2) destroy () method of DispatcherServlet destroys the Spring Container instance.

First Spring MVC Example

1) Create the Web Project as follows:
    a. Select File -> New -> Dynamic Web Project
    b. Provide project name: Lab67
    c. Click on New Runtime Button
        i. Select Apache Tomcat v8.5 and Click on Next.
        ii. Provide Tomcat Installation Directory by clicking on Browse button.
        iii. Select the Required JRE Version.
        iv. Click on Finish Button.
    d. Make sure that Apache Tomcat v8.5 is selected in the Target Runtime.
    e. Select Dynamic Web Module Version as 3.0
    f. Click on Finish button.
2) Copy the Spring 5.2.5 JARs to WEB-INF/lib directory.
3) Copy the following jars to WEB-INF/lib directory.
    a) jstl-1.2.jar
    b) standard.jar
4) Create the Package called com.coursecube.spring
5) Create the Spring Configuration Class called JLCwebConfig under the package com.coursecube.spring.
6) Write DispatcherServlet Initializer class JLCWebAppInitializer.java under the package com.coursecube.spring.
7) Write CourseService.java package com.coursecube.spring.
8) Write CourseController.java package com.coursecube.spring.
9) Configure ViewServler in JLCwebConfig
10) Design the index.jsp and show.jsp and places them in WebContent folder.
11) Depoly and RUN.

Lab 67: First Spring MVC Example



Lab67: Files required

1. index.jsp	2. show.jsp
3. CourseController.java	4. CourseService.java
5. JLCWebConfig.java	6. JLCWebAppInitializer.java

1. index.jsp

```
<!DOCTYPE html>
<html>  <body>
<h2> Welcome to JLC</h2>
<h2> <a href="myjava.jlc"> Show Java Courses</a></h2>
<h2> <a href="myspring.jlc"> Show Spring Courses</a></h2>
<h2> <a href="myweb.jlc"> Show Web Courses</a></h2>
</body>  </html>
```

2. show.jsp

```
<%@ taglib prefix="c"  uri="http://java.sun.com/jsp/jstl/core" %>

<!DOCTYPE html>
<html>   <body>
<h2> List of ${CourseName} Courses </h2>
<ul>
<c:forEach var="cou"  items="${MyCourses}">
<li> ${cou}</li>
</c:forEach>
</ul>
</body>  </html>
```

3. CourseController.java

```
package com.coursecube.spring;

import java.util.List;
import javax.servlet.http.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
public class CourseController {

@Autowired
CourseService courseService;

public CourseController(){
System.out.println("CourseController - D.C");
}

@GetMapping(value="myjava.jlc")
public String getJavaCourses(HttpServletRequest request){
System.out.println("CC - getJavaCourses()");
List<String> mylist= courseService.getJavaCourses();

request.setAttribute("MyCourses", mylist);
request.setAttribute("CourseName", "Java");

return "show";
}

@GetMapping(value="myspring.jlc")
public String getSpringCourses(HttpSession session){
System.out.println("CC - getSpringCourses()");
List<String> mylist= courseService.getSpringCourses();

session.setAttribute("MyCourses", mylist);
session.setAttribute("CourseName", "Spring");

return "show";
}
```

```
@RequestMapping(value="myweb.jlc",method=RequestMethod.GET)
public String getWebCourses(Model model){
System.out.println("CC - getWebCourses()");
List<String> mylist= courseService.getWebCourses();

model.addAttribute("MyCourses", mylist);
model.addAttribute("CourseName", "Web");

return "show";
}
}
```

4. CourseService.java

```
package com.coursecube.spring;

import java.util.*;
import org.springframework.stereotype.Service;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Service
public class CourseService {
	public CourseService(){
		System.out.println("CourseService - D.C");
	}
	public List<String> getJavaCourses(){
		System.out.println("CS - getJavaCourses()");
		List<String> mylist=new ArrayList<>();
		mylist.add("Java8");		mylist.add("JDBC");
		mylist.add("Servlets");		mylist.add("JSP");
		return mylist;
	}
	public List<String> getSpringCourses(){
		System.out.println("CS - getSpringCourses()");
		List<String> mylist=new ArrayList<>();
		mylist.add("Spring5");		mylist.add("Spring Rest");
		mylist.add("Srping MVC");	mylist.add("Spring Boot");
		return mylist;
	}
	public List<String> getWebCourses(){
		System.out.println("CS - getWebCourses()");
		List<String> mylist=new ArrayList<>();
		mylist.add("Java Script");	mylist.add("Angular");
		mylist.add("React JS");		mylist.add("Vue JS");
		return mylist;
	}
}
```

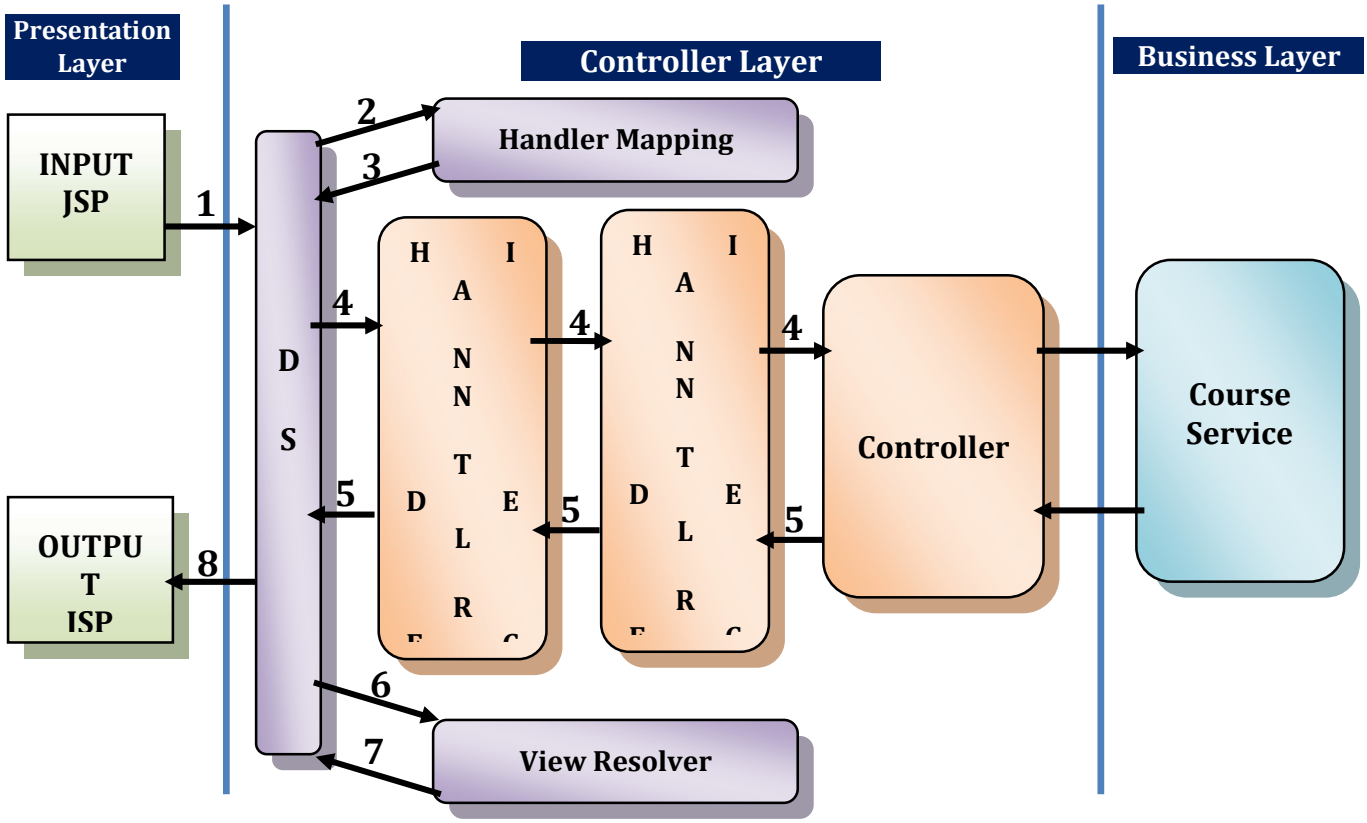
5. JLCWebConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Configuration
@ComponentScan({ "com.coursecube.spring" })
public class JLCConfig {
	@Bean
	public InternalResourceViewResolver viewResolver() {
		InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
		viewResolver.setViewClass(JstlView.class);
		viewResolver.setPrefix("/");
		viewResolver.setSuffix(".jsp");
		return viewResolver;
	}
}
```

6. JLCWebAppInitializer.java

```
package com.coursecube.spring;

import org.springframework.web.servlet.support.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public class JLCWebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
@Override
protected Class<?>[] getRootConfigClasses() {
System.out.println(" ** getRootConfigClasses **");
return new Class[] { JLCWebConfig.class };
}
@Override
protected Class<?>[] getServletConfigClasses() {
System.out.println(" ** getServletConfigClasses **");
return new Class[] { JLCWebConfig.class };
}
@Override
protected String[] getServletMappings() {
System.out.println(" ** getServletMappings **");
return new String[] { "*.jlc" };
}
}
```

Spring MVC Baisc Flow



1) DispatcherServlet takes the incomming request.
2) DispatcherServlet contacts the Handler mapping with incomming request URI **(/myspring.jlc)**
3) Handler Mapping identifies and returns the Controller (CourseController) specified for the request URI **(/myspring.jlc).**
4) DispatcherServlet Identifies and invokes one or more Handler Interceptors registered with Spring Container if any and then DispatcherServlet invokes the Controller
    a. **getSpringCourses () method of CourseController**
5) After finishing Controller method exection, all the registered Handler Interceptors will be called in the reverse order one by one.
    a. **Finally DispatcherServlet gets the view logical name.**
6) DispatcherServlet contacts the ViewResolver with the view logical name **(show).**
7) DispatcherServlet gets the view **(/show.jsp)** from ViewResolver.
8) DispatcherServlet forwards the identified view **(/show.jsp)** to the Client.

Lab68: Login Example using Spring MVC

Lab68: Files required

1. index.jsp	2. login.jsp
3. home.jsp	4. User.java
5. UserValidator.java	6. LoginController.java
7. JLCWebConfig.java	8. JLCWebAppInitializer.java

1. index.jsp

```
<html>  <body>
<br><h1>Java Learning Center  </h1>
<h2>
 <a href="showLogin.jlc">User Login</a>
 </h2>
</body></html>
```

2. login.jsp

```
<%@ taglib  uri="http://www.springframework.org/tags/form"  prefix="form" %>

<html>  <body>
<div align="center">
<h1>User Account Login</h1>
<form:form action="verifyUser.jlc" method="post" modelAttribute="myuser">
<table>
<tr><td>Username</td>
<td><form:input path="username" /></td>
<td><font color=red size=5> <form:errors path="username" /></font></td>
</tr>
<tr><td>Password</td>
<td><form:password path="password" /></td>
<td><font color=red size=5> <form:errors  path="password" /></font></td>
</tr>
<tr>
<td colspan="3">  <input type="submit" value="Account Login"> </td>
</tr>
</table>
</form:form>
</div>  </body></html>
```

3. home.jsp

```
<html>  <body>
<h1>Hello ${myuser.username} ! Your Login Successful</h1>
<h2>This is your Home Page</h2>
<h2>This is your Home Page</h2>
<h2>This is your Home Page</h2>
</body>  </html>
```

4. User.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public class User {
        private String username;
        private String password;
         //Setters and Getters
}
```

5. UserValidator.java

```
package com.coursecube.spring;

import org.springframework.stereotype.Component;
import org.springframework.validation.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Component
public class UserValidator implements Validator {

public boolean supports(Class clazz) {
return User.class.equals(clazz);
}

public void validate(Object command, Errors errors) {
User user = (User) command;

if (user.getUsername() == null || user.getUsername().length() == 0) {
errors.rejectValue("username", "errors.required", new Object[] {"Username"}, "Username
Mandatory.");
}

if (user.getPassword() == null || user.getPassword().length() == 0) {
errors.rejectValue("password", "errors.required",  new Object[] {"Password"}, "Password
Mandatory.");
}

}
}
```

6. LoginController.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import javax.servlet.ServletException;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
public class LoginController {

@Autowired
private UserValidator userValidator;

@GetMapping("/showLogin.jlc")
public String showLoginForm(Model model) throws ServletException {

System.out.println("showLoginForm");
User user = new User();
user.setUsername("Srinivas Dande");
model.addAttribute("myuser", user);
return "login";
}

@PostMapping("/verifyUser.jlc")
public String verifyUser(@ModelAttribute("myuser") User user, BindingResult result) {
System.out.println("verifyUser()");
userValidator.validate(user, result);

if (result.hasErrors()) {
System.out.println(result.getErrorCount());
return "login";
}

String un = user.getUsername();
String pw = user.getPassword();
if (un.equals(pw))
return "home";

return "login";
}
}
```

7. JLCWebConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Configuration
@ComponentScan({ "com.coursecube.spring" })
public class JLCConfig {
	@Bean
	public InternalResourceViewResolver viewResolver() {
		InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
		viewResolver.setViewClass(JstlView.class);
		viewResolver.setPrefix("/");
		viewResolver.setSuffix(".jsp");
		return viewResolver;
	}
}
```

8. JLCWebAppInitializer.java

```
package com.coursecube.spring;

import org.springframework.web.servlet.support.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public class JLCWebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
@Override
protected Class<?>[] getRootConfigClasses() {
System.out.println(" ** getRootConfigClasses **");
return new Class[] { JLCWebConfig.class };
}
@Override
protected Class<?>[] getServletConfigClasses() {
System.out.println(" ** getServletConfigClasses **");
return new Class[] { JLCWebConfig.class };
}
@Override
protected String[] getServletMappings() {
System.out.println(" ** getServletMappings **");
return new String[] { "*.jlc" };
}
}
```

Lab 69: Registration Example using Spring MVC with Annotations.

Lab69: Files required

1. index.jsp	2. register.jsp
3. home.jsp	4. Student.java
5. StudentValidator.java	6. RegisterController.java
7. JLCWebConfig.java	8. JLCWebAppInitializer.java

1. index.jsp

```
<html>
<body>
 <br><h1>Java Learning Center<br/>
<a href="showRegister.jlc">New Student Registration</a></h1>
</body>
</html>
```

2. register.jsp

```
<<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form"%>
<html>
<body>
<form:form action="registerStudent.jlc" method="post"
modelAttribute="student">
<table>
<tr>   <td align="center" colspan="3">Student Registration Form</td>     </tr>
<tr>
<td><p>Student ID:</td>
<td><form:input path="sid" /></td>
<td><form:errors path="sid" /></td>
</tr>
<tr>
<td>Name:</td>
<td><form:input path="sname" /></td>
<td><font color=red size=4><form:errors path="sname" /></font></td>
</tr>
<tr>
<td>Email Id:</td>
<td><form:input path="email" /></td>
<td><font color=red size=4><form:errors path="email" /></font></td>
</tr>
<tr>
<td>Phone No:</td>
<td><form:input path="phone" /></td>
<td><font color=red size=4><form:errors path="phone" /></font></td>
</tr>
```

```
<tr>
<td>Suitable Timings</td>
<td><form:checkbox path="timings" value="07.30A.M - 09.30A.M" />
07.30A.M - 09.30A.M <br> <form:checkbox path="timings"
value="10.30A.M - 02.30P.M" /> 10.30A.M - 02.30P.M <br>
<form:checkbox path="timings" value="04.00A.M - 06.00P.M" />
04.00A.M - 06.00P.M <br> <form:checkbox path="timings"
value="06.30A.M - 08.30P.M" /> 06.30A.M - 08.30P.M <br>
<form:checkbox path="timings" value="Weekends" /> Weekends (Only
Advance) <br></td>
<td><font color=red size=4><form:errors path="timings" /></font></td>
</tr>
<tr>
<td>Gender</td>
<td><form:radiobutton path="gender" value="Male" />Male <br />
<form:radiobutton path="gender" value="Female" />Female <br /></td>
<td><font color=red size=4><form:errors path="gender" /></font></td>
</tr>
<tr>
<td>Qualification</td>
<td><form:select path="qualification">
<form:option value="-----Select option-----" />
<form:option value="M.Sc" />
<form:option value="B.Sc" />
<form:option value="M.C.A" />
<form:option value="B.C.A" />
<form:option value="M.Tech" />
<form:option value="B.Tech" />
</form:select></td>
<td><font color=red size=4><form:errors
path="qualification" /></font></td>
</tr>
<tr>
<td>Remarks</td>
<td><form:textarea path="remarks" rows="5" cols="40" /></td>
<td><font color=red size=4><form:errors path="remarks" /></font></td>
</tr>
<tr><td align="center" colspan="3">
<input type="submit" value="Register Now" /></td>
</tr>
</table>
</form:form>
</body>
</html>
```

3. home.jsp

```
<html><body>
<h1>Hi ${student.sname};Your Registration  Successful</h1>
<h1>This is your Home Page</h1>
<h1>This is your Home Page</h1>
<h1>This is your Home Page</h1>
</body></html>
```

4. Student.java

```
package com.coursecube.spring;
/*
 * @Author : Srinivas Dande
 * @company : Java Learning Center
 * */
public class Student {
	private String sid;
	private String sname;
	private String phone;
	private String email;
	private String[] timings;
	private String qualification;
	private String gender;
	private String remarks;

	// Setters and Getters
}
```

5. StudentValidator.java

```
package com.coursecube.spring;

import org.springframework.stereotype.Component;
import org.springframework.validation.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Component
public class StudentValidator implements Validator {
public boolean supports(Class clazz) {
return Student.class.equals(clazz);
}
public void validate(Object obj, Errors errors) {
Student stu = (Student) obj;

if (stu.getSname() == null || stu.getSname().length() == 0) {
errors.rejectValue("sname", "errors.sname.required", new Object[] {}, " Name is Required.");
}
```

```
if (stu.getEmail() == null || stu.getEmail().length() == 0) {
errors.rejectValue("email", "errors.email.required", new Object[] {}, " Email is Required.");
} else if (!((stu.getEmail().contains("@")) && (stu.getEmail().endsWith(".com")
|| stu.getEmail().endsWith(".co.in") || stu.getEmail().endsWith(".in")))) {
errors.rejectValue("email", "errors.email.invalid", new Object[] {}, " Invalid Email .");
}

if (stu.getPhone() == null || stu.getPhone().length() == 0) {
errors.rejectValue("phone", "errors.phone.required", new Object[] {}, " Phone is Required.");
} else if (stu.getPhone().length() != 10) {
errors.rejectValue("phone", "errors.phone.invalid", new Object[] {}, " Phone contains 10 digits.");
} else if (stu.getPhone().length() == 10) {
try {
Integer.parseInt(stu.getPhone());
} catch (Exception e) {
errors.rejectValue("phone", "errors.phone.invalid", new Object[] {}, " Phone contains only digits.");
}
}

if (stu.getTimings().length < 1) {
errors.rejectValue("timings", "errors.timings", new Object[] {}, " Select Suitable Timings .");
}

if (stu.getGender() == null || stu.getGender().length() == 0) {
errors.rejectValue("gender", "errors.gender", new Object[] {}, " Gender is Required.");
}

if (stu.getQualification() == null || stu.getQualification().length() == 0
|| stu.getQualification().equals("-----Select option-----")) {
errors.rejectValue("qualification", "errors.qualification", new Object[] {}, "Select Qualification.");
}
}
}
```

6. RegisterController.java

```
package com.coursecube.spring;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import javax.servlet.ServletException;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
```

```
* */
@Controller
public class RegisterController {

@Autowired
private StudentValidator studentValidator;

@RequestMapping(value = "/showRegister.jlc")
public String showRegisterForm(Model model) throws ServletException {
System.out.println("showRegisterForm");
Student stu = new Student();
stu.setSid("JLC-99");
model.addAttribute("student", stu);
return "register";
}

@PostMapping(value = "/registerStudent.jlc")
public String registerStudent(@ModelAttribute("student") Student stu, BindingResult result)
throws ServletException {
System.out.println("registerStudent");
studentValidator.validate(stu, result);

if (result.hasErrors()) {
System.out.println(result.getErrorCount());
return "register";
}

System.out.println(stu.getSid());
System.out.println(stu.getSname());
System.out.println(stu.getEmail());
System.out.println(stu.getPhone());
System.out.println(stu.getGender());
System.out.println(stu.getQualification());
String tim[] = stu.getTimings();
for (int i = 0; i < tim.length; i++) {
System.out.println(tim[i]);
}
System.out.println(stu.getRemarks());
return "home";
}
}
```

7. JLCWebConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.*;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Configuration
@ComponentScan({ "com.coursecube.spring" })
public class JLCConfig {
	@Bean
	public InternalResourceViewResolver viewResolver() {
		InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
		viewResolver.setViewClass(JstlView.class);
		viewResolver.setPrefix("/");
		viewResolver.setSuffix(".jsp");
		return viewResolver;
	}
}
```

8. JLCWebAppInitializer.java

```
package com.coursecube.spring;

import org.springframework.web.servlet.support.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public class JLCWebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {
@Override
protected Class<?>[] getRootConfigClasses() {
System.out.println(" ** getRootConfigClasses **");
return new Class[] { JLCWebConfig.class };
}
@Override
protected Class<?>[] getServletConfigClasses() {
System.out.println(" ** getServletConfigClasses **");
return new Class[] { JLCWebConfig.class };
}
@Override
protected String[] getServletMappings() {
System.out.println(" ** getServletMappings **");
return new String[] { "*.jlc" };
}  }
```