Spring MVC Request Processing Flow

A) When you request for resource(JSP), following tasks will happen:

1) DS takes the incomming request.
2) DS contacts the Handler mapping with incomming request URI **(/showHello or /showLogin)**
3) Handler mapping returns the Controller (**LoginController**) and corresponding methods of the Controller with the specified for the request URI **(/showHello or /showLogin**)
4) DS prepares/create parameter object required for controller method

   - request – available with DS

   - Model - create the Model

   etc
5) DS calls the controller method by passing required parameters

   String Vname= lc.showHello(request,Model)

   String Vname= lc.showLogin(Model,request)
6) DS gets the View Logical Name **(hello or login)** from Controller method.
7) DS contacts the ViewResolver with the view logical name **(hello or login)**
8) DS gets the actual JSP **(/hello.jsp or /login.jsp)** from ViewResolver.
9) DS checks whether Identified JSP contains <form:form> tag or not.
10) If Identified View **(/hello.jsp)** does not contain <form:form> tag then that view will be forwarded to Client.
11) If Identified View **(/login.jsp)** contains <form:form> tag then DS will do the following
    a. DS Collects the Model Attribute from Request Scope.
    b. DS gets Command Object from Model Attribute
    c. DS gets Data from command object by calling getter methods on command object
    d. DS populates Data that into the correnponding fields of the jsp form.
    e. DS finally forwards the identified view **(/login.jsp)** to the Client.

B) <u>When you send the request by submiting form, following tasks will happen:</u>

1) DS takes the incomming request.
2) DS contacts the Handler mapping with incomming request URI **(/verifyUser)**
3) Handler mapping returns the Controller (**LoginController**) and corresponding methods of the Controller with the specified for the request URI **(/verifyUser**)
4) DS Creates or Retrives the Model Attribute (which is the Command Object) from the Request or Session Scopes.
5) DS Collects client submitted data and popuplates client submitted data into command object by calling setter methods.
6) DS prepares/creates parameter object required for controller method
   - BindingResult – creates this
   - request – available with DS
   - Model - create the Model

   etc
7) DS calls the controller method by passing required parameters
   a. String Vname= lc.verifyUser (command, results);
8) DS gets the View Logical Name **(home or login)** from Controller method.
9) DS contacts the ViewResolver with the view logical name **(home or login)**
10) DS gets the actual JSP **(/home.jsp or /login.jsp)** from ViewResolver.
11) DispatcherServlet forwards the identified view (/home.jsp or /login.jsp) to the Client.

Lab70: Files required

1. index.jsp	**2. search.jsp**
3. show.jsp	**4. Student.java**
5. StudentController.java	**6. JLCWebConfig.java**
7. JLCWebAppInitializer.java	

1. index.jsp

```
<!DOCTYPE html>
<html>
<body>
<h2> Welcome to JLC</h2>
<h2> <a href="showSearchForm"> Search Student</a></h2>
<h2> <a href="showCookies"> Show Cookies </a></h2>
<h2> <a href="showHeaders"> Show Headers </a></h2>
</body>
</html>
```

2. search.jsp

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

<!DOCTYPE html>
<html>
<body>
<h2> Search Form</h2>

<form:form action="searchStudent"  method="post"   modelAttribute="mystudent">
<table>
<tr>
<td> Email </td>
<td> <form:input path="email"/> </td>
<td> <form:errors path="email"/> </td>
</tr>
<tr>
<td> Phone </td>
<td> <form:input path="phone"/> </td>
<td>  <form:errors path="phone"/>  </td>
</tr>
<tr>
<td> <input type="submit" value="Search Now"/> </td>
</tr>
</table>
</form:form>
</body>
</html>
```

3. show.jsp

```
<!DOCTYPE html>
<html>    <body>
<h2>This is show.jsp </h2>
<h2>This is show.jsp </h2>
<h2>This is show.jsp </h2>
<h2> </h2>
</body>   </html>
```

4. Student.java

```
package com.coursecube.spring;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public class Student {
private String email;
private String phone;

//Setters and Getters

}
```

5. StudentController.java

```
package com.coursecube.spring;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
@SessionAttributes("mystudent")
public class StudentController {

@ModelAttribute("mystudent")
public Student createStudent() {
System.out.println("-------SC -createStudent()-------");
Student stu=new Student();
//stu.setEmail("sd@jlc");
//stu.setPhone("12345");
return stu;
}
@GetMapping(value = "showSearchForm")
public String showSearchForm(Model model,HttpServletRequest request,HttpSession session) {

System.out.println("-------SC -showSearchForm()-------");
System.out.println(model.containsAttribute("mystudent"));
Object obj1=request.getAttribute("mystudent");
System.out.println(obj1);
Object obj2=session.getAttribute("mystudent");
System.out.println(obj2);
return "search";
}
@PostMapping(value = "searchStudent")
public String searchStudent(
@RequestParam(name = "email",required = true) String email,
@RequestParam(name = "phone",required = false)String phone,
Model model,
HttpServletRequest request,
HttpSession session  ) {

System.out.println("-------SC -searchStudent()-------");
System.out.println(email);
System.out.println(phone);
System.out.println(model.containsAttribute("mystudent"));
Object obj1=request.getAttribute("mystudent");
System.out.println(obj1);
Object obj2=session.getAttribute("mystudent");
System.out.println(obj2);
return "show";
}
@GetMapping(value = "showCookies")
public String showCookies(
@CookieValue(name = "JSESSIONID",required = false) String sid) {

System.out.println("-------SC -showCookies()-------");
System.out.println(sid);
return "show";
}
@GetMapping(value = "showHeaders")
public String showHeaders(
@RequestHeader(name="Accept-Encoding",required = true) String encoding,
@RequestHeader(name="Accept-Language",required = true) String lang    ) {
System.out.println("-------SC -showHeaders()-------");
System.out.println(encoding);
System.out.println(lang);
return "show";
}
}
```

6. 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;
        }
}
```

7. 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() {
return new Class[] { JLCWebConfig.class };
}
@Override
protected Class<?>[] getServletConfigClasses() {
return new Class[] { JLCWebConfig.class };
}
@Override
protected String[] getServletMappings() {
return new String[] { "/" };
}
}
```

Handling Exceptions

- Spring Provides two ways to handle the Exceptions
  1) Local Exception handler
  2) Global Exception handler

Lab71: Files required

1. index.jsp	**2. sidsearch.jsp**
3. sidresults.jsp	**4. bidsearch.jsp**
5. bidresults.jsp	**6. Student.java**
7. SidValidator.java	**8. BidValidator.java**
9. messages.properties	**10. SidSearchController.java**
11. BidSearchController.java	**12. StudentService.java**
13. StudentServiceImpl.java	**14. StudentNotFoundException.java**
15. InvalidBatchIdException.java	**16. JLCWebConfig.java**
17. JLCWebAppInitializer.java Same as Lab70	

1. index.jsp

```
<!DOCTYPE html>
<html>   <body>
<h2> Welcome to JLC</h2>
<hr/>  <br/>
<h2> <a href="sidsearch"> Search  By Sid </a></h2>
<h2> <a href="bidsearch"> Search  By Bid  </a></h2>
</body>  </html>
```

2. sidsearch.jsp

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

<!DOCTYPE html>
<html>  <body>
<h2>Sid Search Form</h2>
<form:form action="searchBySid" method="post"   modelAttribute="mystudent">
<table>
<tr> <td>Student ID</td> </tr>
<tr> <td><form:input path="sid" /></td>  </tr>
<tr> <td><form:errors path="sid" style="color:red;" />
 <h3 style="color:red"> ${ErrMsg} </h3>   </td> </tr>
<tr>     <td><input type="submit" value="Search Now" /></td>        </tr>
</table>
</form:form>
</body>  </html>
```

3. sidresults.jsp

```
<html>
<body>
<br>
<h1>   Java Learning Center  </h1>
<h2>Student Search Results</h2>
<table>
<tr>
<td>Student ID</td>
<td>${STU.sid }</td>
<tr>
<td>Batch ID</td>
<td>${STU.bid }</td>
<tr>
<td>Student Name</td>
<td>${STU.sname }</td>
<tr>
<td>Email ID</td>
<td>${STU.email}</td>
<tr>
<td>Phone No</td>
<td>${STU.phone }</td>
</table>
</body>
</html>
```

4. bidsearch.jsp

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

<!DOCTYPE html>
<html>
<body>
<h2> Search Form</h2>
<form:form action="searchByBid" method="post"    modelAttribute="mystudent">
<table>
<tr> <td> Batch ID </td> </tr>
<tr> <td> <form:input path="bid"/> </td> </tr>
<tr> <td colspan="2"> <form:errors path="bid" style="color:red; "/>
  <h3 style="color:red"> ${ErrMsg} </h3>   </td> </tr>
<tr><td colspan="2"> <input type="submit" value="Search Now"/> </td> </tr>
</table>
</form:form>

</body>
</html>
```

5. bidresults.jsp

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

<html>
<body>
<br>
<h1>                Java Learning Center  </h1>
<h2>Student Search Results</h2>
<table>
<tr>
<td>Student ID</td>
<td>Batch ID</td>
<td>Student Name</td>
<td>Email ID</td>
<td>Phone No</td>
</tr>
<c:forEach var="STU" items="${MyStuLIST}">
<tr>
<td>${STU.sid }</td>
<td>${STU.bid }</td>
<td>${STU.sname }</td>
<td>${STU.email}</td>
<td>${STU.phone }</td>
</tr>
</c:forEach>
</table>
</body>
</html>
```

6. Student.java

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

//Setters and Getters

}
```

7. SidValidator.java

```
package com.coursecube.spring;

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

@Override
public void validate(Object command, Errors errors) {
Student stu=(Student)command;
String sid=stu.getSid();

if(sid==null || sid.length()==0) {
errors.rejectValue("sid","errors.sid.required");
} else if(!sid.startsWith("JLC-")) {
errors.rejectValue("sid","errors.sid.format");
}else {
//JLC-asd   -- Not Valid
//JLC-12345  - Not Valid
//JLC-555  - Valid

String str=sid.substring(4);
try {
int x=Integer.parseInt(str);
if(x<=99 || x>999) {
errors.rejectValue("sid","errors.sid.range");
}
}catch(Exception ex) {
errors.rejectValue("sid","errors.sid.integer");
}
}

}

public boolean supports(Class<?> cls) {
return Student.class.equals(cls);
}

}
```

8. BidValidator.java

```
package com.coursecube.spring;

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

@Override
public void validate(Object command, Errors errors) {
Student stu=(Student)command;
String bid=stu.getBid();

if(bid==null || bid.length()==0) {
errors.rejectValue("bid","errors.bid.required");
} else if(!bid.startsWith("B-")) {
errors.rejectValue("bid","errors.bid.format");
}else {
//B-asd
//B-12345
String str=bid.substring(2);
try {
int x=Integer.parseInt(str);
if(x<=9 || x>99) {
errors.rejectValue("bid","errors.bid.range");
}
}catch(Exception ex) {
errors.rejectValue("bid","errors.bid.integer");
}
}
}

public boolean supports(Class<?> cls) {
return Student.class.equals(cls);
}

}
```

9. messages.properties

```
errors.sid.required=Student ID is Required.
errors.sid.format=Student ID must starts with JLC-
errors.sid.integer=Only Digits allowed after JLC-
errors.sid.range=3 Digits allowed after JLC-

errors.bid.required=Batch ID is Required.
errors.bid.format=Batch ID must starts with B-
errors.bid.integer=Only Digits allowed after B-
errors.bid.range=2 Digits allowed after B-
```

10. SidSearchController.java

```
package com.coursecube.spring;

import javax.servlet.http.HttpServletRequest;
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.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
@SessionAttributes("mystudent")
public class SidSearchController {

@Autowired
SidValidator sidValidator;

@Autowired
StudentService studentService;

//Defining Command Object
@ModelAttribute("mystudent")
public Student createStudent() {
System.out.println("--------createStudent()-------");
Student stu= new Student();
return stu;
}

@GetMapping(value = "sidsearch")
public String showSearchPage() {
System.out.println("--------showSearchPage()-------");
return "sidsearch";
}
@PostMapping(value = "searchBySid")
public String searchStudentBySid(@ModelAttribute("mystudent") Student stu,BindingResult
results,Model model) {
System.out.println("--------searchStudentBySid()-------");

sidValidator.validate(stu, results);
if(results.hasErrors()) {
return "sidsearch";
}

String sid=stu.getSid();
Student mystu=studentService.getStudentBySid(sid);
model.addAttribute("STU", mystu);
return "sidresults";
}

@ExceptionHandler(StudentNotFoundException.class)
public String handlleSNFException(StudentNotFoundException ex,Model model) {
System.out.println("-----Local--handlleSNFExceptions()-------");
model.addAttribute("ErrMsg", "Student Id Not FFound");
Student stu=new Student();
model.addAttribute("mystudent", stu);
return "sidsearch";
}
}
```

11. BidSearchController.java

```
package com.coursecube.spring;

import java.util.List;
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.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
@SessionAttributes("mystudent")
public class BidSearchController {

@Autowired
BidValidator bidValidator;
@Autowired
StudentService studentService;

//Defining Command Object
@ModelAttribute("mystudent")
public Student createStudent() {
System.out.println("--------createStudent()-------");
Student stu= new Student();
return stu;
}

@GetMapping(value = "bidsearch")
public String showSearchPage() {
System.out.println("--------showSearchPage()-------");
return "bidsearch";
}

@PostMapping(value = "searchByBid")
public String searchStudentByBid(@ModelAttribute("mystudent") Student stu,BindingResult
results,Model model) {
System.out.println("--------searchStudentByBid()-------");

bidValidator.validate(stu, results);
if(results.hasErrors()) {
return "bidsearch";
}

String bid=stu.getBid();
List<Student> mylist=studentService.getStudentsByBid(bid);
model.addAttribute("MyStuLIST", mylist);

return "bidresults";
}

@ExceptionHandler(InvalidBatchIdException.class)
public String handlleBIDException(InvalidBatchIdException ex,Model model) {
System.out.println("-----Local--handlleBIDException()-------");
model.addAttribute("ErrMsg", "Batch Id is Invalid");
Student stu=new Student();
model.addAttribute("mystudent", stu);
return "bidsearch";
}

}
```

12. StudentService.java

```
package com.coursecube.spring;

import java.util.List;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public interface StudentService {
public Student getStudentBySid(String sid);
public List<Student> getStudentsByBid(String bid);
}
```

13. StudentServiceImpl.java

```
package com.coursecube.spring;

import java.util.ArrayList;
import java.util.List;
import org.springframework.stereotype.Service;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Service
public class StudentServiceImpl implements StudentService {

@Override
public Student getStudentBySid(String sid) {

System.out.println("SS-getStudentBySid()");
//Contact DAO
Student stu=null;
if(sid.equals("JLC-123") || sid.equals("JLC-999")) {
stu=new Student();
stu.setBid("B-99");
stu.setSid(sid);
stu.setSname("Srinivas");
stu.setEmail("sri@jlc");
stu.setPhone("12345");
}else {
throw new StudentNotFoundException();
}

return stu;
}
@Override
public List<Student> getStudentsByBid(String bid) {

System.out.println("SS-getStudentsByBid()");
//Contact DAO
List<Student> list=new ArrayList<>();
if(bid.equals("B-12") || bid.equals("B-99")) {
Student stu=new Student();
stu.setBid(bid);
stu.setSid("JLC-199");
stu.setSname("Srinivas");
stu.setEmail("sri@jlc");
stu.setPhone("12345");

list.add(stu);list.add(stu);
list.add(stu);list.add(stu);
list.add(stu);list.add(stu);
list.add(stu);list.add(stu);
list.add(stu);list.add(stu);
}         else {
throw new InvalidBatchIdException();
}

return list;
}
}
```

14. StudentNotFoundException.java

```
package com.coursecube.spring;

public class StudentNotFoundException extends RuntimeException {

}
```

15. InvalidBatchIdException.java

```
package com.coursecube.spring;

public class InvalidBatchIdException extends RuntimeException {

}
```

16. JLCWebConfig.java

```
package com.coursecube.spring;

import org.springframework.context.MessageSource;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.ReloadableResourceBundleMessageSource;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Configuration
@ComponentScan(basePackages = "com.coursecube.spring")
public class JLCWebConfig {

@Bean
public InternalResourceViewResolver viewResolver() {
InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
viewResolver.setViewClass(JstlView.class);
viewResolver.setPrefix("/");
viewResolver.setSuffix(".jsp");
return viewResolver;
}

@Bean
public MessageSource messageSource() {
ReloadableResourceBundleMessageSource ms=new ReloadableResourceBundleMessageSource();
ms.setBasename("classpath:messages");
ms.setDefaultEncoding("UTF-8");
return ms;
}
}
```

Lab72: Files required

1. index.jsp	Same as Lab71
2. sidsearch.jsp	Same as Lab71
3. sidresults.jsp	Same as Lab71
4. bidsearch.jsp	Same as Lab71
5. bidresults.jsp	Same as Lab71
6. Student.java	Same as Lab71
7. SidValidator.java	Same as Lab71
8. BidValidator.java	Same as Lab71
9. messages.properties	Same as Lab71
10. MyGlobalExceptionHandler.java	**Newly Added in Lab72**
11. SidSearchController.java	**Updated in Lab72**
12. BidSearchController.java	**Updated in Lab72**
13. StudentService.java	Same as Lab71
14. StudentServiceImpl.java	Same as Lab71
15. StudentNotFoundException.java	Same as Lab71
16. InvalidBatchIdException.java	Same as Lab71
17. JLCWebConfig.java	Same as Lab71
18. JLCWebAppInitializer.java	Same as Lab71

10. MyGlobalExceptionHandler.java

```
package com.coursecube.spring;

import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */

@ControllerAdvice
public class MyGlobalExceptionHandler {

        @ExceptionHandler(StudentNotFoundException.class)
        public String handlleSNFException(StudentNotFoundException ex,Model model) {
                System.out.println("-----Global--handlleSNFExceptions()-------");
                model.addAttribute("ErrMsg", "Student Id Not Found");
                Student stu=new Student();
                model.addAttribute("mystudent", stu);
                return "sidsearch";
        }
        @ExceptionHandler(InvalidBatchIdException.class)
         public String handlleBIDException(InvalidBatchIdException ex,Model model) {
                System.out.println("-----Global--handlleBIDException()-------");
                model.addAttribute("ErrMsg", "Batch Id is Invalid");
                Student stu=new Student();
                model.addAttribute("mystudent", stu);
                return "bidsearch";
         }

}
```

11. SidSearchController.java

```
package com.coursecube.spring;

import javax.servlet.http.HttpServletRequest;
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.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
@SessionAttributes("mystudent")
public class SidSearchController {

@Autowired
SidValidator sidValidator;

@Autowired
StudentService studentService;

//Defining Command Object
@ModelAttribute("mystudent")
public Student createStudent() {
System.out.println("--------createStudent()-------");
Student stu= new Student();
return stu;
}

@GetMapping(value = "sidsearch")
public String showSearchPage() {
System.out.println("--------showSearchPage()-------");
return "sidsearch";
}
@PostMapping(value = "searchBySid")
public String searchStudentBySid(@ModelAttribute("mystudent") Student stu,BindingResult
results,Model model) {
System.out.println("--------searchStudentBySid()-------");

sidValidator.validate(stu, results);
if(results.hasErrors()) {
return "sidsearch";
}

String sid=stu.getSid();
Student mystu=studentService.getStudentBySid(sid);
model.addAttribute("STU", mystu);
return "sidresults";
}
}
```

12. BidSearchController.java

```
package com.coursecube.spring;

import java.util.List;
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.*;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
@SessionAttributes("mystudent")
public class BidSearchController {

@Autowired
BidValidator bidValidator;

@Autowired
StudentService studentService;

//Defining Command Object
@ModelAttribute("mystudent")
public Student createStudent() {
System.out.println("--------createStudent()-------");
Student stu= new Student();
return stu;
}
@GetMapping(value = "bidsearch")
public String showSearchPage() {
System.out.println("--------showSearchPage()-------");
return "bidsearch";
}

@PostMapping(value = "searchByBid")
public String searchStudentByBid(@ModelAttribute("mystudent") Student stu,BindingResult
results,Model model) {
System.out.println("--------searchStudentByBid()-------");

bidValidator.validate(stu, results);
if(results.hasErrors()) {
return "bidsearch";
}

String bid=stu.getBid();
List<Student> mylist=studentService.getStudentsByBid(bid);
model.addAttribute("MyStuLIST", mylist);

return "bidresults";
}
}
```

Handler Interceptors

- Handler Interceptors will be invoked before and after the controller invocation for performing pre-propcessing and post-propcessing tasks.

- You can have one or more Handler Interceptors in the application.

- Steps to write Handler Interceptors:

  1) Write class by implementing HandlerInterceptor interface
  2) Override the following methods in your class
     a) public boolean preHandle(request,response,Object handler) throws Exception
     b) public void postHandle(request,response, Object handler,modelAndView)
     c) public void afterCompletion(request,response, Object handler, Exception

Ex:

```
public class MyInterceptor  implements HandlerInterceptor {

@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object
obj) throws Exception {

//Your Logic Here

return true;
}
@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response, Object obj,
ModelAndView modelAndView) throws Exception {

//Your Logic Here

}
@Override
public void afterCompletion(HttpServletRequest req, HttpServletResponse res, Object
handlerMethod, Exception ex) throws Exception {

//Your Logic Here

}
}
```

3) Register your handler interceptor with the Spring Container as follows:

a) Write the Application Config class by implementing **WebMvcConfigurer**.

b) Override the following method in the Config class
   public void addInterceptors(InterceptorRegistry registry)

c) Register the handler interceptor using the registry object

```
@Override
public void addInterceptors(InterceptorRegistry registry) {
registry.addInterceptor(new MyInterceptor1());
registry.addInterceptor(new MyInterceptor2()).addPathPatterns("/hai.jlc");
registry.addInterceptor(new MyInterceptor3());

}
```

Note :

♦ To apply the interceptor for all the incoming request URIs
  registry.addInterceptor(new MyInterceptor1());

♦ To apply the interceptor for specific incoming request URIs
  registry.addInterceptor(new MyInterceptor2()).addPathPatterns("/hai.jlc");

preHandle()

- This method will be called just before the controller method.
- This method returns Boolean value.
- When preHandle() method returns true then Interceptors or Controller in HandlerExecutionChain will be called.
- When preHandle() method returns false then Control will be returned to Client without invoking Interceptors or Controller in HandlerExecutionChain and No view will be rendered.

postHandle()

- This method will be called immediately after the controller method execution.

afterCompletion()

- This method will be called just before sending the response.

Using HandlerInterceptorAdapter

- HandlerInterceptorAdapter is subclass of HandlerInterceptor interface and overriding all three methods.
- You can write your interceptor class just by extending HandlerInterceptorAdapter class and you can override only the necessary methods out of the three.

Ex:

```
public class MyInterceptor3 extends HandlerInterceptorAdapter {

@Override
public boolean preHandle(request, response, object) throws Exception {

            //Your Logic Here

return true;
}

@Override
public void postHandle(request, response, object, modelAndView) throws Exception {

            //Your Logic Here
}
}
```

Lab 73: Handler Interceptor Example.

Lab70: Files required

1. index.jsp	**2. hello.jsp**
3. hai.jsp	**4. hellohai.jsp**
5. HelloController.java	**6. MyInterceptor1.java**
7. MyInterceptor2.java	**8. MyInterceptor3.java**
9. JLCWebConfig.java	**10. JLCWebAppInitializer.java**

1. index.jsp

```
<html>
<body>
 <br><h1>Java Learning Center<br/>
<a href="hello">Hello Guys</a><br/><br/>
<a href="hai">Hai Guys</a><br/><br/>
<a href="hellohai">Hello Hai Guys</a></h1>
</body>
</html>
```

2. hello.jsp

```
<html>
<body>
<br><h1>Java Learning Center  </h1> <br/><br/>
<h2> I am Hello JSP </h2>
<h2> I am Hello JSP </h2>
<h2> I am Hello JSP </h2>
</body>
</html>
```

3. hai.jsp

```
<html><body>
<br><h1>Java Learning Center  </h1> <br/><br/>
<h2> I am Hai  JSP </h2>
<h2> I am Hai  JSP </h2>
<h2> I am Hai  JSP </h2>
</body></html>
```

4. hellohai.jsp

```
<html>
<body>
<br><h1>Java Learning Center  </h1> <br/><br/>
<h2> I am Hello Hai  JSP </h2>
<h2> I am Hello Hai  JSP </h2>
<h2> I am Hello Hai  JSP </h2>
</body></html>
```

5. HelloController.java

```
package com.coursecube.spring;

import java.util.Map;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
@Controller
public class HelloController {

@GetMapping("/")
public String showIndexPage() {
System.out.println("---------showIndexPage()---------");
return "index";
}
@GetMapping("/hello")
public String showHello(Map map) {
System.out.println("---------showHello()---------");
System.out.println(map);
return "hello";
}

@GetMapping("/hai")
public String showHai() {
System.out.println("---------showHai()---------");
return "hai";
}

@GetMapping("/hellohai")
public String showHelloHai() {
System.out.println("---------showHelloHai()---------");
return "hellohai";
}
}
```

6. MyInterceptor1.java

```
package com.coursecube.spring;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public class MyInterceptor1 implements HandlerInterceptor {

@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object obj)
throws Exception {
System.out.println("\n MyInterceptor1 -> preHandle " + obj);
return true;
}

@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response, Object obj,
ModelAndView modelAndView) throws Exception {
System.out.println("MyInterceptor1 -> postHandle " + obj);
}
@Override
public void afterCompletion(HttpServletRequest req, HttpServletResponse res, Object
handlerMethod, Exception ex)
throws Exception {
System.out.println("MyInterceptor1 -> afterCompletion " + handlerMethod + "\t" + ex);
}

}
```

7. MyInterceptor2.java

```
package com.coursecube.spring;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */
public class MyInterceptor2 implements HandlerInterceptor {

@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object obj)
throws Exception {
System.out.println("MyInterceptor2 -> preHandle " + obj);
return true;
}

@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response, Object obj,
ModelAndView modelAndView) throws Exception {
System.out.println("MyInterceptor2 -> postHandle " + obj);
}

@Override
public void afterCompletion(HttpServletRequest req, HttpServletResponse res, Object
handlerMethod, Exception ex)
throws Exception {
System.out.println("MyInterceptor2 -> afterCompletion " + handlerMethod + "\t" + ex);
}

}
```

8. MyInterceptor3.java

```
package com.coursecube.spring;


import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.springframework.web.servlet.ModelAndView;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */

public class MyInterceptor3 extends HandlerInterceptorAdapter {
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object obj)
throws Exception {
System.out.println("MyInterceptor3 -> preHandle " + obj);
return true;
}

@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response, Object obj,
ModelAndView modelAndView) throws Exception {
System.out.println("MyInterceptor3 -> postHandle " + obj);
}
}
```

9. JLCWebConfig.java

```
package com.coursecube.spring;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
import org.springframework.web.servlet.view.InternalResourceViewResolver;
import org.springframework.web.servlet.view.JstlView;
/*
* @Author : Srinivas Dande
* @company : Java Learning Center
* */

@EnableWebMvc
@Configuration
@ComponentScan({ "com.coursecube.spring" })
```

```
public class JLCWebConfig  implements WebMvcConfigurer {
@Bean
public InternalResourceViewResolver viewResolver() {
System.out.println("viewResolver");
InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
viewResolver.setViewClass(JstlView.class);
viewResolver.setPrefix("/WEB-INF/myjsps/");
viewResolver.setSuffix(".jsp");
return viewResolver;
}
@Override
public void addInterceptors(InterceptorRegistry registry) {
System.out.println("addInterceptors");
registry.addInterceptor(new MyInterceptor1());
registry.addInterceptor(new MyInterceptor2()).addPathPatterns("/hai");
registry.addInterceptor(new MyInterceptor3());
}
}
```