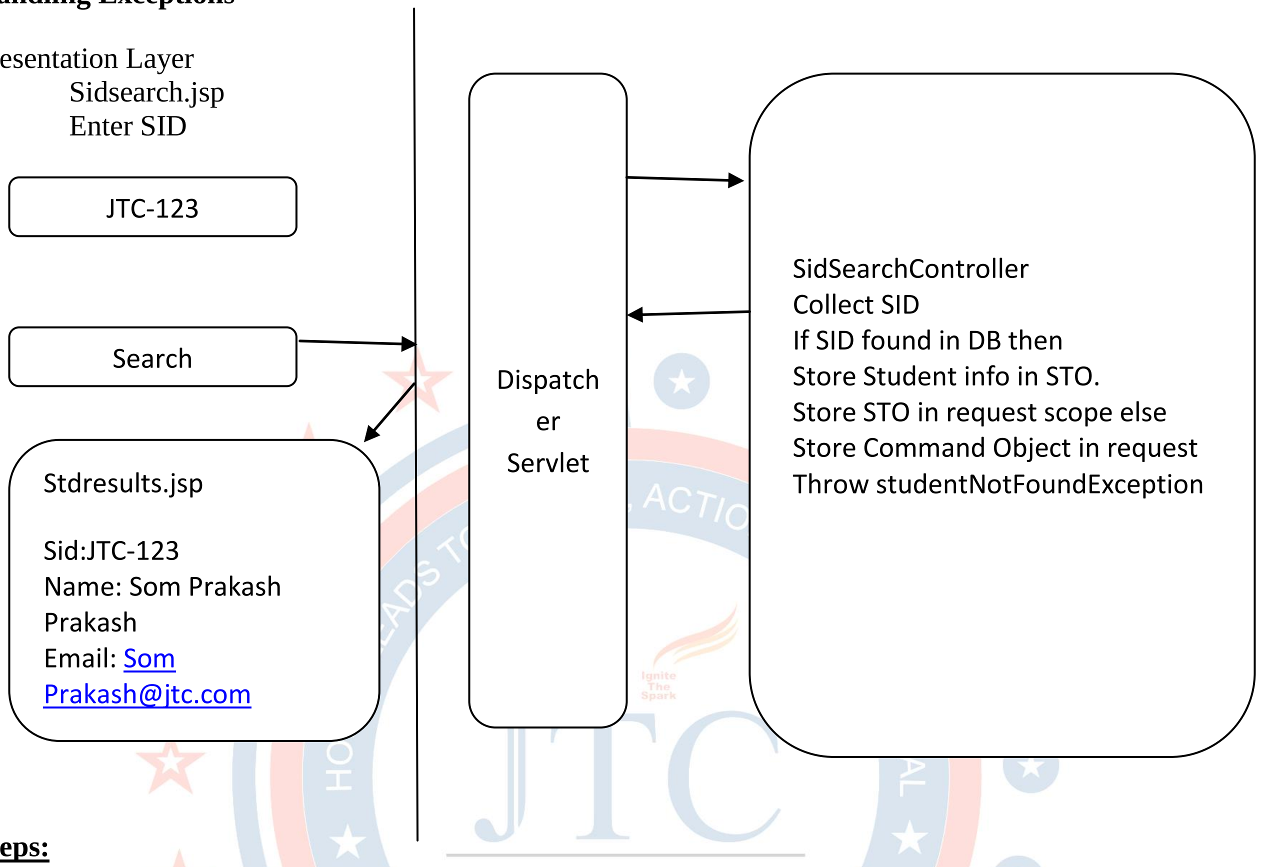
Handling Exceptions

Presentation Layer
Sidsearch.jsp
Enter SID


JTC-123
Search
Stdresults.jsp
Sid:JTC-123
Name: Som Prakash Prakash
Email: Som Prakash@jtc.com
Dispatcher Servlet
SidSearchController
Collect SID
If SID found in DB then
Store Student info in STO.
Store STO in request scope else
Store Command Object in request
Throw studentNotFoundException

<u>Steps:</u>

1) writhe the user defined Exception class.

```
    Public class student Not FoundException extends Runtime Exception {
            Private string sid;
            Public studentNotFoundException() {}
            Public studentNotFoundException(string sid) {
                        This.sid=sid;
            }
            Public string toString() {
                        String msg=”student not found”;
                        If(sid!=null) {
                                    Msg=”student id:”+sid+”not found”;
                        }
                        Return msg;
            }
            Public string getMessage() {
                        Return toString();
            }
}
```

2. When you get a problem in the controller then do the following in the controller

a. store the command object in the request object.

Req.setAttribute("sidSerchCommand",sidCommand);

b. Create and throw the user Defined Exception as per your requirement i.c inside required method of controller class.

Throw new student notFoundException(sid);

3. Use the @ExceptionHandler annotation for the method of controller class from where the exception is thrown.

@ExceptionHandler({StudentNotFoundException.class})

4. Configure simpleMappingExceptionResolver bean with the required exceptions thrown and the corresponding views to display.

```
<bean class="org.springframework.web.servlet.handler.simpleMappingExceptionResolver">
        <property name="exceptionMappings">
                <props>
                <prop key="com.jtcindia.spring.mvcf.studentNotFoundException"> sideserch </prop>
                </props>
                </property>
        </bean>
```

Jtc 70: Files required

Index.jsp	Sidsearch.jsp
Sidresults.jsp	SidSerchController.java
SidSearchCommand.java	SidValidator.java
StudentTO.java	SidNotFoundException.java
Messages.properties	Web.xml
Jtcindia-servlet.xml	

Index.jsp	Sidsearch.jsp
<%@ taglib prefix="c" uri=http://java.sun.com/jsp/jst/core%> <html><body> <h1>java Training center<br/> <a href="<c:cul value=k"sidsearch.jtc"/>"> Serch Student</a> </h1> </body> </html>	<%@ taglib prefix="form" uri=http://www.springframework.org/tags/form%> <html><body> <h1> java Training Center</h1> <h2> Student search Form </h2> <from:form action="searchStudent.jtc" method="post" commandName='sidSerchCommand"> <table> <tr> <td> Enter student ID </td> </tr> <tr> <td><form:input path="sid"/></td></tr> <tr><td><font color="red" size="5"> <form:errors path="sid"/>${exception} </font></td></tr> <tr><td> <input type="submit" value="Search"/> </td></tr> </table> </form:form> </body></html>

Sidresults.jsp

```
<html><body>
<h1> java Training Center</h1><br/>
<h1> search Results </h1>
<table>
<tr><td> student ID</td>
<td>${STO.sid}</td></tr>
<tr><td> Batch ID</td>
<td>${STO.bid}</td></tr>
<tr><td> student Name ID</td>
<td>${STO.sname}</td></tr>
<tr><td> Email ID</td>
<td>${STO.email}</td></tr>
<tr><td> phone No</td>
<td>${STO.phone}</td></tr>
</table></body></html>
```

SidSearchCommand.java

```
Package com.jtcindia.spring.mvc;
Public class sidSerchCommand {
Private string sid;
  // Setters and getters
}
```

Student TO.java

```
Package lcom.jtcindia.spring.mvc;
Public class student TO{
Private string sid;
Private string bid;
Private string sname;
Private string email;
Private string phone;
//Setters and Getters
}
```

Sid Validator.java

```
Package com.jtcindia.spring.mvc;
Import org.springframework.validation.*;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class SidValidator implements validator {
Public Boolean supports (class clazz) {
Return SidSerchCommand.class.equeals(clazz);
}
Public void validate(object obj, Errors errors) {
SidSearchCommand sidSearchCommand=(SidSerchCommand) obj;
String sid="sidSerchCommand.getsid();
If(sid==null||sid.length()==0) {
Errors.rejectValue("sid","errors.sid.required");
}else if(!sid.startsWith("jtc")) {
Errors.rejectValue("sid","srrors.sid.format1");
}else{
String p=sid.substring(4);
Try{
Int X=integer.parselnt(p);
If(x<100||x>999)
Errors.reject Value("sid","errors.sid.format2");
}catch (Exception e) {
Errors.rejectValue("sid","errors.sid.format3");
} } } } }
```

SidSearchController.java

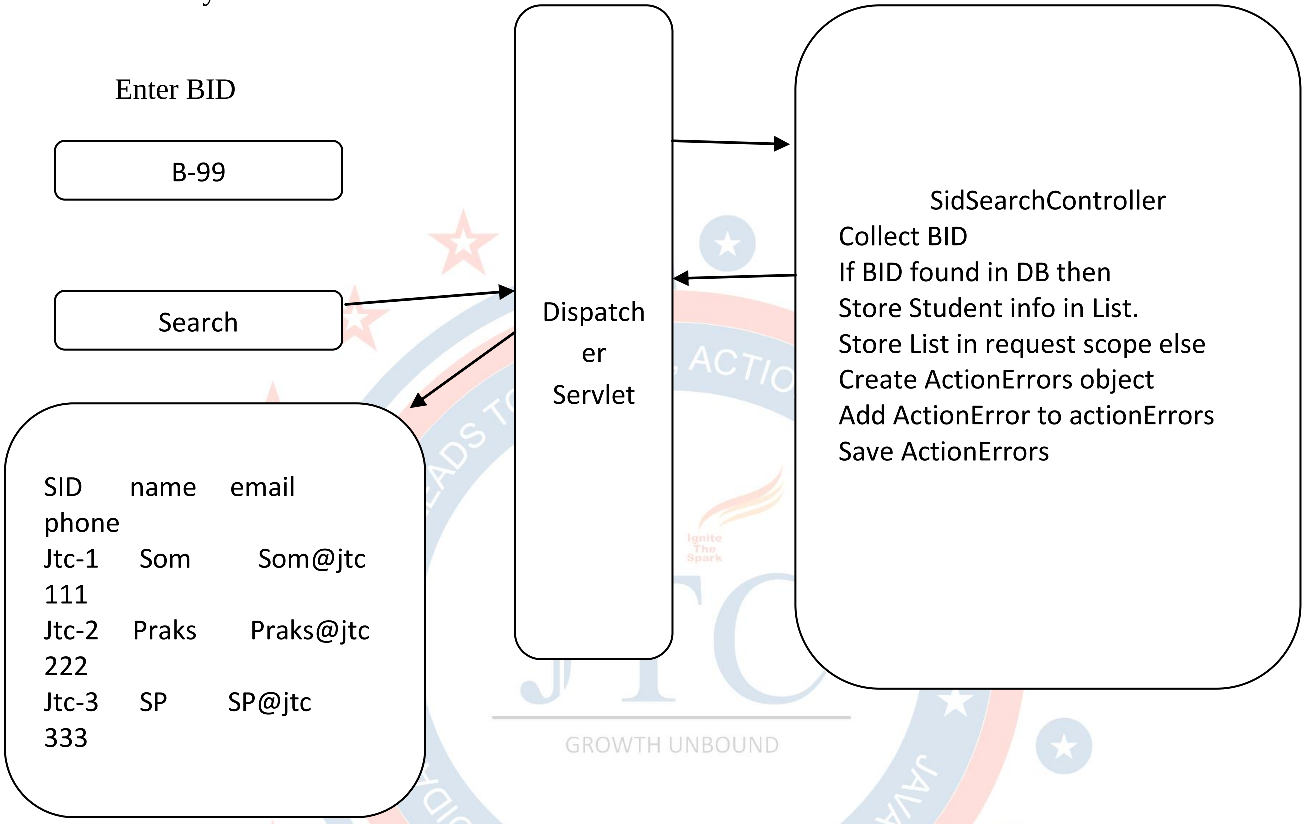
```
Package com.jtcindia.spring.mvc;
// IMPORT HERE
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
@Controller
Public class SidSearchController {
@Autowired
Private SidValidator sidValidator;
@RequestMapping(value=”/searchStudent”,method=RequestMethod.Post)
@ExceptionHandler({Student Not FoundException.class})
Public string searchStudent(
@ModelAttribute9”sidSearchCommand”) SidSearchCommand sidcommand,
Errors errors, HttpServletRequest req, model model) {
System.out.println(d”searchStudent”);
sidValidator.validate(sidCommand,errors);
if (errors.hasErrors()) {
system.out.println(errors.getErrorCount());
return”sidSearch”;
}
String sid=sidCommand.getSid();
String result=””;
System.out.println(sid);
If(sid.equals(jtc-123”) || sid.equals(“jtc-999”)) {
studentTO sto=new studentTO();
Sto.setSid(sid);          sto.setBid(“B-99”);               sto.setSname(“Som Prakash”);
Sto.setEmail(Som@jtc.com);                  sto.setPhone(“99999”);
Model.addAttribute(“STO”,sto);
//req.setAttribute(“STO”,sto);
Result=”sidresults”;
} else {
Req.setAttribute(“sidSearchCommand”,sidCommand);
Throw new studentNotFoundException(sid);
}
Return result;
}
@RequestMapping(value=”/sidsearch”)
Protected string showSearchPage(Map model) throws servletException {
system.out.println(“showSearchPage”);
SidSearchCommand sidCommand=new sidSearchCommand();
Model.put(“sidSearchCommand”,sidCommand);
Return”sidsearch”;
} }
```

SidNotFoundException.java	Jtcindia-servlet.xml

Package com.jtcindia.spring.mvc;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit : www.jtcindia.org
**/
Public class studentNotFoundException extends RuntimeException {
Private string sid;
Public studentNotFoundException() {}
Public studentNotFoundException(string sid) {
This.sid=sid;
}
Public string to string () {
String msg=”student not found”;
If(sid!=null) {
Msg=”student id:” +sid+”not found”;
}
Return msg;
}
Public string getMessage() {
Return toString();
}
}
 	
Messages.properties
Errors.sid.required=Student ID is Required
Errors.sid.format1=student id must starts with JTC
Errors.sid.format2=Only 3 Digits Allowed after JTC
Errors.sid.format3=Only Digits Allowed after JTC
	<?xml version=”1.0” encoding=UTF-8”?>
	<beans….>
	<context:annotation-config/>
	<context:component-scan base package=” com. Jtcindia.spring.mvc.SidValidator”/>
	<bean class=”org.springframework.web.servlet. view.internalResourceViewResolver”>
	<property name=”view class” value=”org.spring gramework.web.servlet.view.internalResourceView”/>
	<property name=”prefix” value=”/”/>
	<property name=”suffix” value=”.jsp”/>
	</bean>
	<bean id=”messageSource” class=” org. springframework.context.support.ResourceBundle MessageSource”>
	<property name=”basename” value=”messages”/>
	</bean>
	<bean
	class=”org.springframework.web.servlet.handler. simple Mapping ExceptionResolver”>
	<property name=”exceptionMappings”>
	<props>
	<prop key=”com.jtcindia.spring.mvc.studentNot Found Exception”>
	Sidsearch
	</prop></props></property>
	</bean></beans>

Handling Errors

Presentation Layer



Step 1: To handle the errors in controller class write following code.
Errors.rejectValue("bid","bid.notfound");

Step 2: Specify the key in property file.
Bid.notfound=Batch ID not Found

Step 3: Use <form:errors> tag in JSP to display the error messages.
<form:errors path="bid"/>

Jtc 71: Files required

Index.jsp	bidsearch.jsp
bidresults.jsp	bidSerchController.java
bidSearchCommand.java	bidValidator.java
StudentTO.java	Messages.properties
Web.xml	Jtcindia-servlet.xml

Index.jsp

```
<%@ taglib prefix="c" uri=http://java.sun.com/
jsp/jstl/core%>
<html><body>
<br><h1>java Training Conter<br/>
<a href="<c:url value="bidsearch.jtc"/>">search
student</a></h1>
</body></html>
```

Bidsearch.jsp

```
<%@ taglib prefix="form" uri=http://www.
springframework.org/tags/form%>
<html><body>
<h1> java Training Center</h1>
<h2>Student Search Form</h2>
<form:form action="searchStudent.jtc" method="
post"commandName="bidSearchCommand">
<table>
<tr> <td> Enter Batch ID</td></tr>
<tr><td><form:input path="bid" /></td></tr>
<tr><td><font color="red" size="5">
<form:errors path="bid"/>
</font></td></tr>
<tr><td><input type="submit"
Value="search"/></td></tr>
</table> </form:form> </body> </html>
```

Bidresults.jsp

```
<%@ taglib prefix="c" uri=http://java.sun.com/
jsp/jstl/core%>
<html><body>
<h1> java Training Center</h1>
<br/><h1>search Results</h1>
<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 items="${LIST}" var="STO">
```

bidSearchController.java

```
packave com.jtcindia.spring.mvc;
//IMPORT HERE
@Controller
Public class BidSerchController {
@Autowired
Private BidValidator bidValidator;
@RequestMapping (value="/searchStudent",
method=RequestMethod.POST)
Public string searchStudent (@ModelAttribute
("bidSearchCommand")
BidSerchCommand bidSearchCommand,
BindingResult errors, HttpServletRequest req) {
System.out.println("searchStudent");
bidValidator.validate (bidSearchCommand, errors);
if (errors.hasErrors()) {
system.out.println(errors.getErrorCount());
return"bidsearch";
}
String bid=bid SearchCommand.getBid();
String result="bid search";
System.out.,println(bid);
If(bid.equals("B-12"))||bid.equals("B-99)) {
List<student TO> list=new ArayList<student TO>();
Student to sto=new student TO();
Sto.setSid(JTC-123");      sto.setBid(bid);
Sto.setSname("Som Prakash");
sto.setEmail(Som@jtc.com);
Sto.getPhone("99999");
List.add(sto);                    list.add(sto);
// Add more student TO object result="bidresults";
Req.setAttribute("LIST",list);
} else {
Result="bidsearch";
Errors.rejectValue("bid","bid.notfound");
}
Return result;
}
@RequestMapping (value="/bidsearch")
Protected String showSearchPage(Map model) throws
servletException {
system.out.println("showSearchPage");
```

`<tr> <td>${Sto.sid}</td>`
`<td>${STO.bid}</td>`
`<td>${STO.sname}</td>`
`<td>${STO.email}</td>`
`<td>${STO.phone}</td>`
`</tr> </c:forEach> </table> </body> </html>`
	BidSearchCommand bidSearchCommand=new BidSearchCommand();
	Model.put("bidSearchCommand",bidSearchCommand);
	Return"bidsearch";
	} }

BidSearchCommand.java

```
Package com.jtcindia.spring.mvc;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class BidSearchCommand {
Private string bid;
// Setters and Getters
}
```

StudentTO.java

```
Package com.jtcindia.spring.mvc;
Public class student TO {
Private string sid;
Private string bid;
Private string sname;
Private string email;
Private string phone;
// setters and Getters
}
```

BidValidator.java

```
Package com.jtcindia.spring.mvc;
Import org.springframework.validation.*;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
Public class BidValidator implements validator {
Public Boolean supports (Class clazz) {
Return BidSearchCommand.class.equals(clazz);
}
Public void validate (object obj, Errors errors) {
BidSearchCommand bidSearchCommand=(BidSearchCommand) obj;
String bid=bidSearchCommand.getBid();
If(bid==null||bid.length()==0){
Errors.rejectValue("bid","errors.bid.required");
} else if (!bid.starts With("B")) {
Errors.rejectValue("bid","errors.bid.format1");
} else {
String p=bid.substring(2);
Try{
Int x=integer.pargelnt(p);
If(x<10||x>99) {
Errors.rejectValue("bid","errors.bid.format2");
}
} catch (Exception e) {
```

```
Errors.rejectValue("bid","errors.bid.format3");
}
}
}
}
}
}
```

Messages.properties	Jtcindia-servlet.xml

Errors.bid.required=Batch ID is Required
Errors.bid.format1=Batch id must starts with B
Errors.bid.format2=Only 2 Digits Allowed after B
Errors.bid.format3=Only Digits Allowed after B
Bid.notfound=Batch ID not Found
	<?xml version="1.0" encoding="UTF-8"?>
	<beans….>
	<context:annotation-config/>
	<context:component-scan base package="com. Jtcindia.spring.mvc"/>
	<bean id="bidValidator" class="com.jtcidia.spring. mvc.BidValidator"/>
	<bean class="org.springframework.web.servlet. view.internalRsourceViewResolver">
	<property name="viewClass" value="org. springframework.web.servlet.view.internalResource view"/>
	<property name="prefix" value="/"/>
	<property name="suffix" value=".jsp"/>
	</bean>
	<bean id="messageSource" class="org. springframework.context.support.Resource Buldel MessageSource">
	<property name="basename" value="messages"/>
	</bean>
	</beans>

Main Elements of Spring MVC

Presentation Layer Components

1. JSP/EL/JSTL
2. Spring Form Tag Library
3. Message Bundles
4. Commands
5. View Resolvers

Controller Layer Components:

1. DispatcherServlet

2. HandlerMappings
3. Handler IntercePtors
4. Controllers

Presentaion Layer Components

JSP/EL/JSTL

You have already studied.

Spring form Tag Library

<form:form/>	<form:input/>	<form:Jtcel/>	<form:password/>
<form:hidden/>	<form:checkbox/>	<form:checkboxes/>	<form:errors/>
<form:option/>	<form;options/>	<form:radiobutton/>	<form:radiobuttons/>
<form:select/>	<from:textarea/>		

Message Bundles

- You can centralize Labels and Error messages in message bundle.
- You can write one or more message bundles in your web application.
  Syntax:
  Basename CountryISOCode languageISOCode. Properties
  Basename LanguageISOCode.properties
  For accessing any message from message bundle. You need to register ResourceBundleMessageSource in the spring configuration Document with basename.

```
<bean id=”messageSource”:
        Class=”org.springframework.context.support.ResourceBundleMessageSource”>
                <property name=”basename” value=”messages”/>
</bean>
```

Accessing Labels

Ex:
Messages.properties
Jtc.title=JTC (EN)
Messages hi.properties
Jtc.title=JTC (HI)
Use the required keys in JSP using JSTL tag as follows;
<%@ taglib prefix=”fmt” uri=http://java.sun.com./jsp/jstl/fmt%>
<fmt:message key=”jtc.title”/>

Accessing Error messages

Messages.properties

Errors.username.required=username is required.
Errors.required={0} is required.
Errors.length={0} length must be between {1} and {2}

In validate () method

1. Errors.rejectValue("username","errors.username.required",null,"Username is Mandatory");
   Displays Username is Required

2. Errors.rejectValue("username","uname.errors.required",null,"Username is Mandatory");
   Displays Username is Mandatory

3. Errors.rejectValue("username","errors.required.required",object []{"UName"},"Username is Mandatory");
   Displays UName is Required

4. Errors.rejectValue("username","errors.length",new object []{"Username","5","9"},"Defalut value");
   Displays Username length must be between 5 and 9

5. Errors. reject Value("password","errors.length",new object[] {"password","4","8"},"Defalut value");
   Displays password length must be between 4 and 8

In JSP

<form:errors path="username"/>
<form:errors path="password"/>

Commands

- Command is mainly used to store client submitted data.
- Command class is simple java Bean class with private Variables and public setter and Getter methods.
- Client submitted data will be stored in command object by calling setter methods.
- Data in the command object will be collected by calling getter methods and will be populated into JSP form elements.

Command Validations

- Write the validations required for your command data in a separate Validator class with the following Steps.
    1. Write your validator class by implementing validator interface which is avaiJtcle in org.springframework.validation package.
    2. Override the following two methods.
       Boolean supports (class<?> clazz)
       Void validate(Object command, Errors errors)
    3. Write the Code inside the supports () method to check whether correct command is used or not.

4. Write the code inside the validate () method to validate input data.
5. When any input and is voiliting validation rules the add the error messages using the following methods.

   Void reject Value(String,string,object [],string)
   Void reject Value(String,string)

View Resolvers

- Spring MVC provides the view Resolver which resolve view logical names to actual wiews.
- Fillowing are the list of view Resolver provided in Spring MVC

1. internalResourceViewResolver
2. XmlViewresolver
3. ResourceBundleViewResolver
4. contentNegotiationgViewResolver
5. BeanNameViewResolver
6. UrlBasedViewResolver
7. FreeMarkerViewResolver
8. VelocityViewResolver
9. JasperReportsViewResolver
10. XsltViewResolver

Internel Resource view Resolver

Ex1:
```
<bean class="org.springframework.web.servlet.view.internalResourceViewResolver">
<property name="viewClass"
          Value="org.springframework.web.servlet.view.internalResorceView"/>
<property name="prefix" value="/"/>
<property name="suffix" value=".jsp"/>
</bean>
```

Ex 2:
```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="viewClass" value="org.springframework.web.servlet.view.jstlView"/>
<property name="peofix" value="/"/>
<property name="suffix" value=".jsp"/>
</bean>
```

Ex 3:
```
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="viewClass" value="org.springframework.web.servlet.view.jstlView"/>
<property name="peofix" value="/WEB-INF/pages/"/>
<property name="suffix" value=".jsp"/>
</bean>
```

Ex 4:

```
<bean class=”org.springframework.web.servlet.view.InternalResourceViewResolver”>
<property name=”viewClass” value=”org.springframework.web.servlet.view.TileJstlView”/>
</bean>
```

<u>Cantroller Layer Components</u>

DispatcherServlet

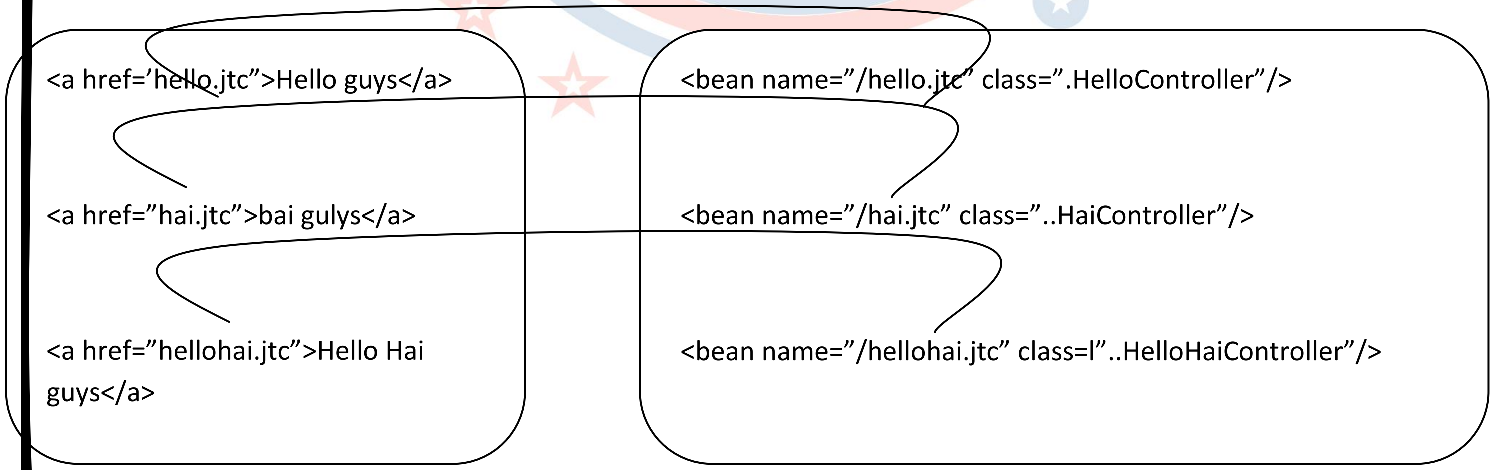
1. DispatcherServlet is the Front Controller Component which is responsible for the following:
    - Receives the incoming request
    - Process the request completely
    - Delivers the response.

HandlerMappings

1. Handler Mappings are responsible for identifying corresponding controller for incoming request URI.
2. All the Handler Mappings provided are subclasses of HandlerMapping interface.
3. Following are the various handler Mappings provided in spring MVC;
    - BeanNameUrlHandlerMapping
    - DefaultAnnotationHandlerMapping
    - simpleUrlhandlerMapping
    - controllerBeanNameHandlerMapping
    - ControllerClassNameHandlerMapping

BeanNameUrlHandlerMapping

- This is the default Handler Mappings which will be registered by the spring container Automatically.
- This Handler Mapping checks whether any bean avaiJtcle whose name is same as incoming request URI.

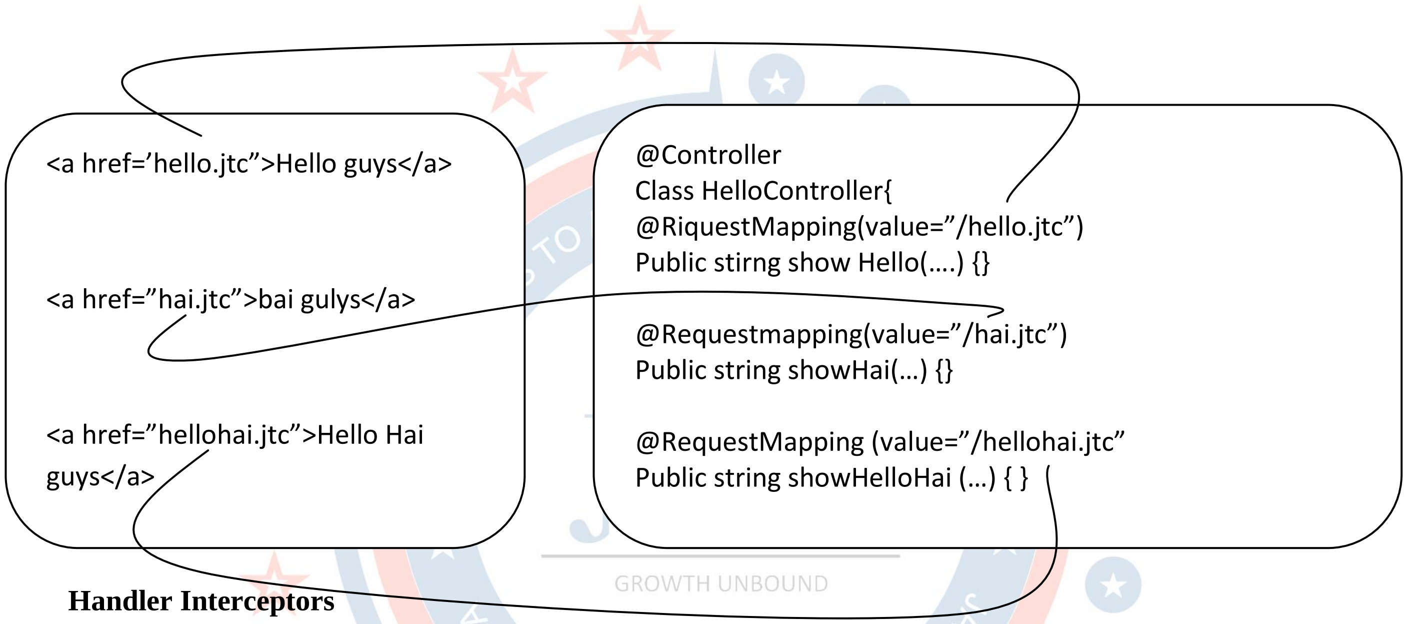


```
<a href=’hello.jtc”>Hello guys</a>
<a href=”hai.jtc”>bai gulys</a>
<a href=”hellohai.jtc”>Hello Hai guys</a>
```

```
<bean name=”/hello.jtc” class=”.HelloController”/>
<bean name=”/hai.jtc” class=”..HaiController”/>
<bean name=”/hellohai.jtc” class=l”..HelloHaiController”/>
```

DefaultAnnotationHandlerMapping

- This is the default handler Mappings which will be registered by the spring Container Automatically.
- This Handler Mapping checks whether any method avaiJtcle avaiJtcle in the controller class whose @RequestMapping value is same as incoming request URI.


<a href='hello.jtc">Hello guys</a>
<a href="hai.jtc">bai gulys</a>
<a href="hellohai.jtc">Hello Hai guys</a>
@Controller
Class HelloController{
@RiquestMapping(value="/hello.jtc")
Public stirng show Hello(....) {}
@Requestmapping(value="/hai.jtc")
Public string showHai(...) {}
@RequestMapping (value="/hellohai.jtc"
Public string showHelloHai (...) { }

Handler Interceptors

1. Handler Interceptors will be invoked before and after the controller invocation for performing pre-propcessing and post processing tasks.
2. You can have one or more Handler Interceptors in the application.

Steps to write Handler interceptors

1. Write your interceptor class by implementing Handlerinterceptor interface
2. Mark your Interceptor class with @Component annotation.
3. Override the following methods in your Interceptor class:
   - Public Boolean preHandle(request,response, object handler,)
   - Public void postHandle(request,response, object handler, modelAndView)
   - Public void afterCompletion(request,response, object handler, Exception e)

Ex:

```
@Component
Public class MyInterceptor1 implements HandlerInterceptor {
```

```
Public Boolean preHandle(HttpServletRequest req, HttpServletResponse res,object obj) {
        Return true;
}
Public void postHandle(HttpServletRequest req.HttpServletResponse res,object obj,ModelAndView mav)
{
Public void after Completion (HttpServletRequest req,HttpServletResponse res, object obj, Exception e) {
} }
```

You can con configure your Interceptor classes in spring configuration Docment in two ways:

a. To apply the Interceptor for all the incoming request URIs

```
        <mvc:interceptors>
            <bean class="com.jtcindia.spring.mvc.myinterceptor1"/>
        </mvc:interceptors>
```

b. To apply the Interceptor for specific incoming request URI

```
        <mvc:interceptors>
                <mvc:interceptor>
                        <mvc:mapping path="/hai.jtc"/>
                        <bean class=com.jtcindia.spring.mvc.myInterceptor1"/>
                </mvc:interceptor>
        </mvc:interceptors>
```

preHandle()

- This method will be called just before the controller method.
- This method returns Boolean value.
- When preHandl () method returns true then interceptors or controller in HandlerExecutionjhChain and No view will be rendered.

postHandle()

This method will be called immediately after controller method execution.

afterCompletion ()

This method will be called just before sending the response.

Using HandlerInterceptor Adapter

- HandlerInterceptor Adaptor 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:

```
@Component
Public class MyInterceptor 1 exteuds Handlerlnterceptor Adapter {
Public Boolean preHandle(HttpServletRequest req, HttpServletResponse res,Object obj) {
    Return true;
}
```

Public void postHandle(httpServletRequest req.HttpServletResponse res, object obj, modelAndView mav) {
} }

Jtc72: Files required

Index.jsp	HelloController.java
MyInterceptor1.java	MyInterceptor2.java
MyInterceptor3.java	Web.xml
Jtcindia-servlet.xml	

Index.jsp

```
<%@ taglib prefix="c" uri=http://java.sun.com/jsp/jstl/core%>
<html><body>
<br><h1>java Training Conter<br/>
<br/><a href="hello.jtc">Hello guys</a>
<br/><a href="hai.jtc">Hai guys</a>
<br/><a href="hellohai.jtc">Hello Hai guys</a>
</body></html>
```

MyInterceptor3.java

```
Package com.jtcindia.pring.mvc;
Import javax.servlet.http.HttpServletRequest;
Import javax.servlet.http.HttpServletResponse;
Import
org.springframework.web.servlet.handle.*;
Import org.springframework.stereotype.
component;
Import org.springframework.web.servlet.
ModelAndView;
@Component
Public class Mylnterceptor3 extends
handlerlnterceptorAdapter {
public Boolean prehandle(httpServletRequest req,
httpServletResponse res,
Object obj) throws Exception {
System.out.println("/nMylnterceptor3 >
preHandle");
Return true;
}

Public void postHandle (HttpServletRequest req,
HttpServletResponse res,
Object obj, ModelAndView mav) throws
Exception {
System.out.println("mylnterceptor3>
```

HelloController.java

```
Package com.jtcindia.spring.mvc;
Import java.util.map;
Import org.springframework.stereotype.
controller;
Import org.springframework.web.bind.
annotation.RequestMap ping;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
@Controller
Public class HelloController {
@RequestMapping(value="/hello.jtc")
Public string showHello(map model) {
System.out.println("showHello()");
Return "hello";
}
@RequestMapping(value="/hai.jtc")
Public string showHai() {
System.out.prinntln("showHai()");
Return "hai";
}
@RequestMapping(value="/hellohai.jtc")
Public string show helloHai(Map model) {
System.out.println("showHelloHai()");
Return "hellohai";
}
}
```

```
postHandle");
}
}
```

MyInterceptor1.java

```
Package com.jtcindia.spring.mvc;
Import javax.servlet.http.httpServletRequest;
Import javax.servlet.http.HttpServletResponse;
Import org.springframework.stereotype. component;
Import org.springframework.web
Servlet.handlerlnterceptor;
Import org.springframework.web.servlet.Model
AndView;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
@Component
Public class Mylnterceptor1 implements
Handlerlnterceptor {
Public Boolean prehandle(httpServletRequest req,
HttpServletResponse res,
Object obj) throws Exception {
System.out.println("\nMylnterceptor1 > preHandle");
Returen true;
}
Public void postHandle(HttpServletRequest req,
httpServletResponse res,
Object obj, ModelAndView mav) throws Exception
System.out.println9"mylnterceptor1 > postHandle");
}
Public void afterCompletion(HttpServletRequest req,
HttpServletResponse res, opject obj, Exception e)
Throws Exception {
System.out.println("MyLnterceptor1>
afterCompletion");
}
}
}
```

MyInterceptor2.java

```
Package com.jtcindia.spring.mvc;
Import javax.servlet.http.HttpServletRequest;
Import javax.servlet.http.HttpServletResponse;
Import org.springframework.stereotype.Component;
Import
org.springframework.web.servlet.Handlerinterceptor;
Import org.springframework.web Servlet.Model and
view;
/*
*@Author: Som Prakash Rai
*@Company : java Training Center
*@ visit          : www.jtcindia.org
**/
@Component
Public class Mylnterceptor2 implements
Handlerlnterceptor {
Public Boolean prehandle(httpServletRequest req,
HttpServletResponse res,
Object obj) throws Exception {
System.out.println("\nMylnterceptor2 > preHandle");
Returen true;
}
Public void postHandle(HttpServletRequest req,
httpServletResponse res,
Object obj, ModelAndView mav) throws Exception
System.out.println("mylnterceptor2 > postHandle");
}
Public void afterCompletion(HttpServletRequest req,
HttpServletResponse res, opject obj, Exception e)
Throws Exception {
System.out.println("MyLnterceptor2>
afterCompletion");
}
}
}
```

Jtcindia-cervlet.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns=http://www.springframework.org/schema/beans
            Xmlns:xsi=http://www.w.3.org/2001/XMLSchema-instance
            Xmlns:p=http://www.springframework.org/schema/p
            Xmlns:mvc="http://www.springframework.org/schema/mvc"
            Xmlns:context=http://www.springframework.org/schema/context
            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:annotation-config/>
<context:component-scan base-package="com.jtcindia.spring.mvc"/>
<mvc:interceptors>
              <bean class="com.jtcindia.spring.mvc.myinterceptor1"/>
              <mvc:interceptor>
                       <mvc:mapping path="/hai.jtc"/>
                       <bean class="com.jtcindia.spring.mvc.myinterceptor2"/>
               </mvcf:interceptor>
               <bean class="com.jtcindia.spring.mvc.myinterceptor3"/>
</mvc:interceptors>
<bean
             Class="org.springframework.web.servlet.view.InternalResourceViewResolver">
             <property name="viewClass"
                          Value="org.springframework.web.servlet.view.jstlView"/>
              <property name="prefix" value"/WEB-INF/pages/"/>
              <property name="suffix" value="jsp"/>
        </bean>
</beans>
```