Spring MVC

Spring MVC /Spring web MVC
==========

important logics in web application developement
==========

a) request data gathering loigic (read form data(req params) , header values, misc info)
b) form validation logic (verifying the pattern and format of the form data)
c) request procssisng /b.logic (main the logics dealing calculations, analyzation)
d) PErsistence logic (like jdbc code/spring jdbc/orm/data code)
e) middlware services (like security , Tx mgmt , logging and etc..)
f) presentation logic (the UI code that makes enduser to supply inputs and to view outputs)
and etc...

Different models of java Web application devleopment
==========

Model1 architecture

=>HEre we use either servlet or jsp as the main web comps to place the above logics.. i.e if servlet comps are used then jsp comps will not be used and vice-versa.

=>It is more of Jsp based web application developement

note: It is old and suitable only for small scale projects (max of 10 webpages)

note:: Here we mix up mulitple logics logics in single servlet/jsp comp.. so code becomes clumzy.. becoz no seperate b/w logics..

Model2 architecture(MVC)

MVC1 MVC2

Develop web app as layered app having multiple comps developed in multiple technologies.

MVC
====

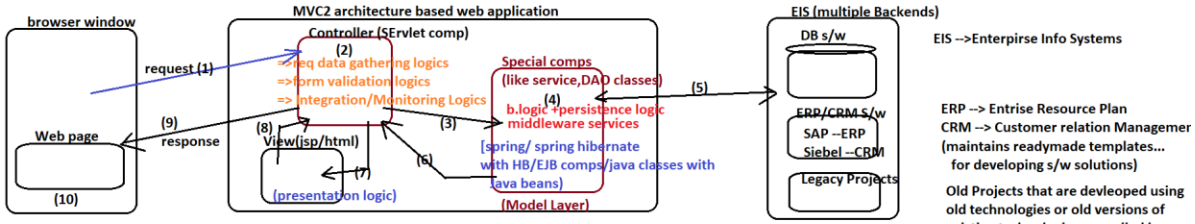
M --->Model Layer -->Data (b.logic+persistence logic) (Accounts officer)

V --->View Layer -----> represents Presenentation logic (for UI for enduser) (Beautician)
C --->Controller layer ---> Monitoring logics (controls the flow) (Traffic police /Super viser)

note :: MVC1 architecture we take sperate comps for model layer (like service,dAO classes) but we take singlecomp either servlet/jsp comp having both view , controller layer logics (MVC1 is good compare to model1 architecture towards seperation of logics, if want to get more seperation b/w logics then go for MVC2)

In MVC2 architeture we take sperate comps for view(html/jsp/...), seperate comp for controller (servlet/servlet filter) and also seperate comps for model(java classes+ java beans) will be taken.. to get more clean seperation b/w logics

MVC2 Architecture Diagram
==========



EIS -->Enterpirse Info Systems

ERP --> Entrise Resource Plan
CRM --> Customer relation Managemer (maintains readymade templates... for developing s/w solutions)

Old Projects that are devleoped using old technologies or old versions of existing technologies are called legacy

Co al project, C lang project, JAva 1.0 project , java 5.x project servlet 2.4 and jsp1.2 project.. spring 2.x project..

=>Multiple BakEnds togather is called as EIS

MVC2 architecture advantages
==========

(a) More layers.. So there is clean sepertion b/w logics
(b) Modifications done in one layer does not effect other layers
(c) Maintenance and enhacement of the web application becomes easy
(d) Parellel development is possible , So the productivity is good
(e) It is industry standard architecture to develop the Applications..

Limitations
==========

(a) Knowlege on multiple technologies is required.
(b) For parellel devleopment more programmers are required

Project Team
|--->part1 (presentation-tier developer)
|--->part2 (Business tier developers)

=>part1 of team uses servlet,jsp and other ul technologies support to develop view ,controller layer loigcs
=>part2 of team uses Spring,hiberante,EJB ,java classes and etc.. to develop model layer b.logoics , persistnece logics..

=>Since these two parties(part1,part2) can work parellel. So we can say parellel development possible.. These two parties of team can be there in the same company or can be there in the diff diff companies of two diff locaitions..

MVC3.... 6 are no way related java .. they are versions of asp.net mvc..

MVC2 Rules/Principles
==========

a) All operations must happen under control or monotoring of controller Servlet
b) Every comp of every layer is given to have specific logics, So place only those logics and do not place any addtional logics
c) There can be multiple view comps , multiple model comps (service, DAO clasess) but it is recomanded to take single Controller Servlet comp..
d) View camps should not interact with model comps directly and vice-versa.. they should inteact through controller servlet..

MVC = MVC2
==========

What is diff b/w Architeture and Design pattern?

Ans) => DesingPattern gives specific solution for specific problem of application development..
=> ARchitecture gives plan and flow of execution while developing s/w apps as layered Applications.. In the creation architecture, we use different Design patterns in different layers
MVC (MVC1,MVC2) is an architecture... that gives plan to develop web application layered Application..
In the development of each layer like M, V, C .. So many design patterns will be involved..
In Model layer, we use service ,DAO and etc.. patterns
In View layer , we use composite view ,view helper patterns
In Controller, we use frontcontroller, application controller patterns..

Why web f/w are given , when we can developed MVC2 archiecture web application driectly by using servlet,jsp techologies?

When we can develop MVC2 architecture based web applicaiton usisng servlet,jsp technologies directly.. then what is need of web application frameworks?

if we develop web application by using servlet,jsp techologies direclty the limitations are

(a) We need to develope all the logics of all the layers. No layer logics will be generated dynamically , So the burden on the programmer will be increased.
(b) we should rember implement MVC2 rules/principles (It is constucting the house on our own)
(c) We should manage navigation (flow between the comps explicitily)
(d) takes bit time to develope MVC2 rules based web application.
(e) we can not use other technologies in view layer other than jsp,html.. like themeleaf,volocity,freemarker and etc.. and etc..

(js is not a independnet technology becoz it can not be executed on its own..it must be embedded with html or jsp or etc... So angularjs, reactjs, jquery also falls under same category)

To overcome these problems.. we can web application frameworks... they are

(a) struts --> from apache outdated (outdated)
(b) spring mvc (part of spring f/w) ---> from interface21 (1)
(c) JSF ---> from sunMs /Oracle corp (2)
(d) ADF ---> from oracle corp (so costly)
(e) web work ---> from Open symphony

web mvc frameworks /web application frameworks provides abstraction layer on Servlet,jsp web techologies having following benifits

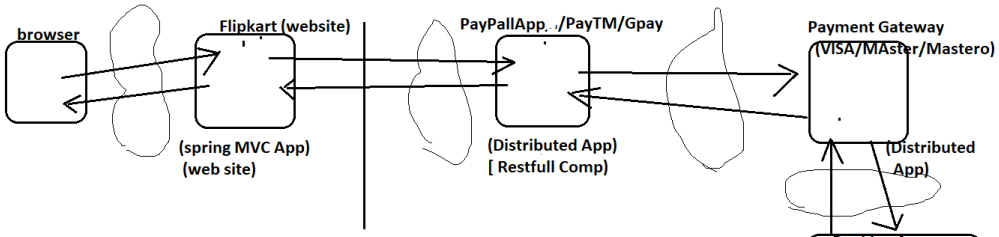
(a) we need not develop main controller comp.. becoz they are giving a pre-defined Servlet comp as built-in controller comp based on Frontcontroller design pattern..
(b) MVC f/w (Mainly FrontController) itself takes care of navigation among comps/layers..
(c) Most of the MVC2 rules/principles will implented automatically
(d) Productivity will be good..
(e) Most of MVC frameworks allows use different technologies in the view layer including jsp,html ....
(f) Developers feels light weight.. while developing applications...
and etc...

It is like purchasing ready made house from real estate company or builder..

note:: with the intergation of jquery/angularjs even small,medium scale web site develop is happening in java...

note:: if the web application is small , medium scale web application and do not want maintain it long time then go for jsp ,servlet based mvc2 web application development..
eg:: movie promotional web sites, car/bike model promotional websites and etc..
note:: if the web application is larget scale web application and we want to maintain it long time then go for mvc f/ws based web application development..
eg:: e-commerce web sites, banking websites.. , insurance websites.. , university websites.

Spring MVC Advantages
=====================
(a) MVC framework common advntages also available here...
(b) Allows to use different UI Technologies like jsp , html , theymeleaf, java classes.. and etc..
(we can even integrate all these UI Technologies based with JS and Js tools (jquery,ajax, angular js, angular,reactjs and etc..)
(c) Allows to develop mvc2 architecture based web application by having frontcontrller design pattern as implicit design pattern..
FrontController built-in Servlet comp is spring mvc :: org.sf.web.mvc.servlet.DispatcherServlet
(d) supports Internalization (118n) --> Displaying labels, numbers, dates, currency symbols accorring diff locales..
(lang + country)
(e) Easy integran with other modules of spring like aop,data,jdbc,orm ,Txmgmt, security and etc.... (we can take the support dependency injection)
(f) Interaction with Restfull Distributed Apps is simplified .. in spring mvc...



web application

a) Clients are web based (like browsers)

b) Generally Clients s/ws like browser s/w.. (not programmable Apps)

c) supports only http,https protocols

d) Enduser inteacts with them directly..

e) use sevlet,jsp directly or spring mvc, jsf, webwork to develop web applications

Distributed App

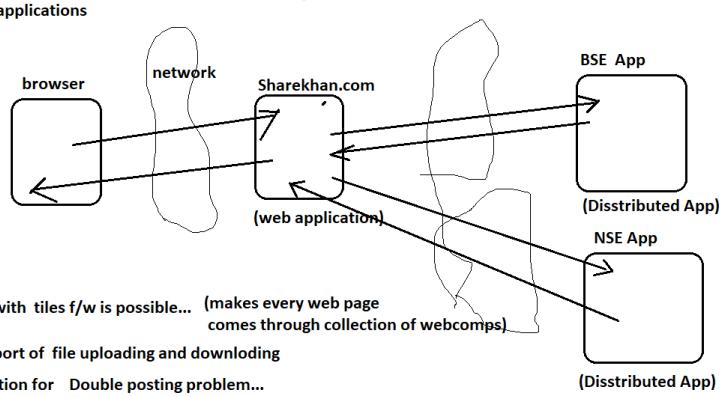
a) allows diff types of clients (web sites, desktop,mobile Apps, smart devices, IVR System and etc..)

b) all clients are programmable Apps

c) suports http,https, SOAP and other protocols

d) Enduser will not interact directly...

e) use EJB(old),RMI(old) , CORBA(old), webServices (SOAP/Rest)



f) Intergraion with tiles f/w is possible... (makes every web page comes through collection of webcomps)

g) Built-in support of file uploading and downloding

h) implicit solution for Double posting problem...

and etc...

---

we can develop spring mvc applications in four ways

==========

=>Using Xml driven cfgs (Declarative Approach) (old -- only in maitence projects)

=> Using Annotation driven cfgs (both in new and old projects)

=> 100% Code driven /Java Config driven cfgs (latest but not popular)

=> Spring boot driven cfgs (very new --> only in new projects)

Xml driven Spring MVC

==========

note:: Though Servlet comp is pre-defined its cfg in web.xml file mapping with url /url pattern is mandatory , othrerwise ServletContainer will not recognize that..

So DisatcherServlet cfg in web.xml (xml , annotation driven cfgs of spring mvc) in mantat other ServletContainer will not recognize that serlvet comp...

=> SErvlet ,jsp comps are managed by SErvletContainer+jsp Cotnainer.. where spring beans like service classes, DAO class, AOP classes, DataSource will be managed by Spring's IOC containers.. In spring MVC we need work with both containers..

=> if want to make Servlet comp taking diff urls based requests then should cfg either with directory match url pattern or extension match url pattern.

/x/y/* --->Directory match
/x/* ---> Directorh match
/* --->directoy match

*.do ---> extension match
*.htm ---> extension match

Assignments

==========

a) 3 types of url patterns (exact match, extension match, directly match)

b) <load-on-startup> (https://youtu.be/opQ5kqJgKE4)

c) Init params , context params and request params

=>In Spring MVC "DispatcherSevlet" is built-in front controller servlet
=>In JSF "FacesServlet" is built-in front controller servlet
=>In Struts "ActionServlet" is built-in front controller servlet
=>In webWork "FilterDispatcher" is built-in front controller SErvlet Filter

What is Front Controller DP?
==========

=FrontController is a special web comp of java web application.. that traps and taks either all requests or multiple requests , applies common system services like logging, auditing, security and etc.. , delegates the requests handler classes(java classes) , takes the results(model) from them , passes them to view comps (like jsp comps) and passes the formatted results to browser as response..

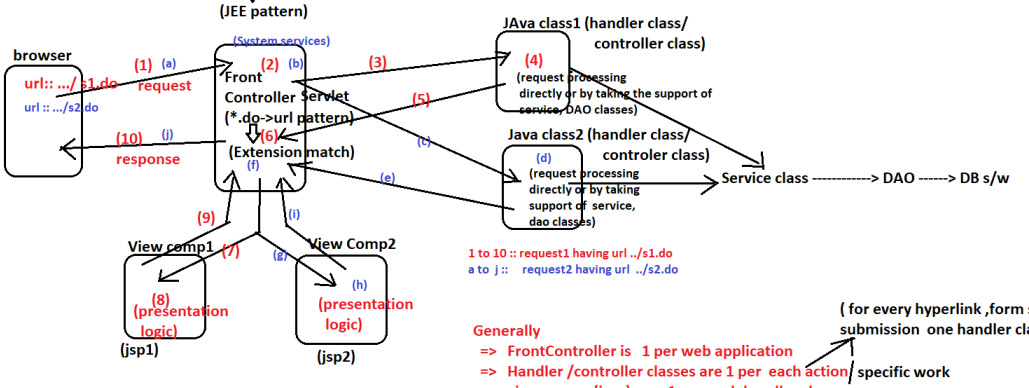
=>In a singleline, we can say Frontcontroller controls all the activities /navigation of the Application..

=> keeping b.logic or request processing logic in java classes is good pratice becoz they make logics more reusable and invokable from difftypes of clients.. but they can not take direct http requests from clients..

=> To overcome this problem we still write b.logic/request processing logics in java classes (handler classes/controller classes) but we develop one frontController web comp either as Servlet(good) /SErvlet Filter to trap all requests and to delegate to handler classes.. and also for taking results from handler/controller classes and to pass them to view comps for formatting results..

=>FrontController desging pattern always implemented in MVC2 architecture of web application development..

(JEE pattern)



Generally
=> FrontController is 1 per web application
=> Handler /controller classes are 1 per each action specific work
=> view comps (jsps) are 1 per each handler class

(for every hyperlink ,form submit .. any rquest submission one handler class will be taken)

(for exampe 10 handler classes means 10 view comps(jsp comps)

FrontControlller

note:: To allow SErvlet taking multiple requests or all requests.. it should be cfg either with Extention match or directory match url pattern

note:: Since FrontController is doing so many operations it is better to keep it ready during deployment of the web application itself .. (enable pre-instantation and pre-intialization of FC) by enabling <load-on-startup>

what is the diff b/w FrontController and Controller/Handler class?

Ans) FC-Front Controller is generally 1 per web application.. and it is a web comp (servlet/ Servlet filter) having system service logics to apply by trapping all or multiple requests..

=> Controller/Handler classes are normal java classes that are taken on 1 per each action/work basis (app contains more controller/handler classes) having request processing logic directly or indirectly..

What is the diff b/w MVC and Frontrcontrller ?

=>MVC/MVC2 is architecture to develop java web based web application as layered Application

=> FrontController is desing Pattern to develop "C->controller" layer of MVC2/MVC architecture with better pratices.. having total control on navigation.

note:: with MVC/MVC2 architecture , there is no FrontController ....

MVC2 /MVC architecture is like any company

FrontController is like CEO of the company

note:: In spring MVC Apps .. we do not develop FrontController .. we use readymade Frontroller called "DispatcherServlet" by cfg it in web.xml file.. as shown below..

web.xml

```
<web-app>
  <!-- servlet cfg -->
  <servlet>
    <servlet-name>dispatcher </servlet-name>
    <servlet-class>org.sf.web.servlet.DispatcherServlet </servlet-class>
    <load-on-startup>2</load-on-startup>
  </servlet>          >=0 should be given
  <!--Servlet maping -->
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern> *.htm </url-pattern>
  </servlet-mapping>  (Extension match url pattern)
</web-app>
```

jars in WEB-INF/lib folder

spring-web-<ver>.jar

spring-webmvc-<ver>.jar (DS is available here)

DS ACtions on the Deployment

(a) Deployment of spring mvc web application

(b) Loading of web.xml .. checking wellformness, validness and creating web.xml file InMemory MetaData

(c) Becoz of <l-o-s> eanbled on the DS, The pre-instantiation of DS takes place

(d) As part DS of initialization the init() of DS executes and this method creates IOC container of type XmlWebApplicationContext by taking WEB-INF/<DS logical name>-servlet.xml as spring cfg file by default (in our example /WEB-INF/dispatcher-servlet.xml)

(e) IOC container Loads spring bean cfg file (WEB-INF/dispatcher-servlet.xml) , checks its well-formness, validness and creates Its InMemoryMetaData..

(f) IOC container performs Pre-instanitation of singleton scope beans and also completes injections on them like service classes, dao clases, DataSources, handler classes and etc...

(g) keeps Singleton scope spring bean objs in the Internal cache of IOC container..

(h) keeps DispatcherServlet object in the Internal cache of ServletContainer..

Internal Cache of IOC container (Managed by DS)

id 1	service class obj ref
id 2	DAO class obj ref
id3	Handler class ob ref

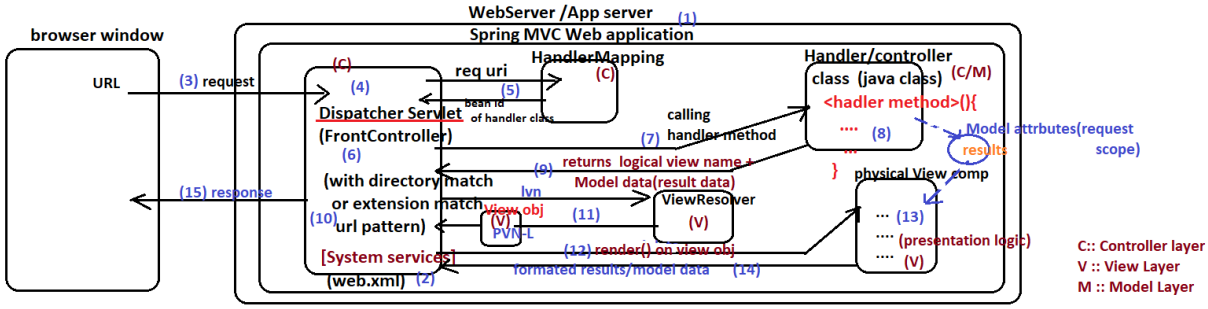
and etc..

InternalCache of ServletContainer

dispatcher	DispatcherServlet obj ref

Spring MVC is designed around FrontController "DispatcherServlet" i.e entire navigation or flow will be taken care by DispatcherServlet. The FrontController DispatcherSErvlet acts as single entry-Single exit point for all the requets and responses..

Spring MVC Flow (Diagram based)
==========



(1) Programmer deplyos the Spring MVC Web application in webServer or Application server

(2) All DispatcherServlet related Deployment Actions takes place (refere previous class) based in the cfgs done in web.xml file

(3) Enduser gives request to spring MVC web application using browser... having request url

(4) As FrontController "DS" (DispatcherServlet) traps and takes the request and applies common System services on the request like logging, auditing and etc...

(5) DS handovers the request to HandlerMapping by giving incoming request uri/url and HandlerMapping maps/links request url/uri with Handler /controller class and returns its bean id back to DS

(6) DS recieves the bean id of Handler class from HandlerMApping and submits it to DS Managed IOC container and gets Handler class object by called ctx.getBean(-,-) method

(7) Ds calls handler method on Handler class object

(8) handler method either directly process the request (or) process the request by taking the support Service, DAO classes...

(9) Handler class/Controller class gives logical view name + model data (result) to DS ..
(note: this logical name can mapped /linked with different technologies based view comps with out distrubing the source code handler classes)

(10) DS takes logical view name(lvn) and model data from handler/controller class.. and keeps model data in request scope as model attributes

(11) DS gives logical view name (lvn) to Viewresolver.. and viewResolver maps lvn to physical view name(PVN) and returns View Object having PVN and its Location back to DS
note:: View Resolver does not renders the view comp.....It will resolve/indetify physical name and Location.. and gvies to DS in the for View object..

12) DS calls render() on the recived View object to transfer/render the control to physical View comp

13) Physical View comp format the results(model data) using presenting logics by getting model data(results) from model attributes (request scope)

14) physical View Comp (UI comp) gives formatted data results (model data) back Frontcontroller DS

15) Ds sends formatted results to browser as response...

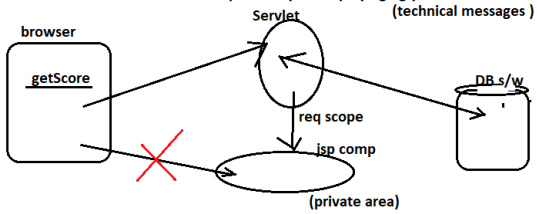
req -->DS ---> HM ---> beanId to DS --> DS get Handler obj --> DS calls handler method -->
DS gets lnv +model data keeps in model attribute -->DS gives LVN to ViewResolver---> ViewResolver gives View obj to DS--> DS calls render() on View object --> PVC --> DS ---> response to browser

=>Entire flow of Spring MVC is decided by the FrontController DS , So we need arrange comps as required for the Spring MVC flow... Some comps we need to develope manually (like handler classes/physical view comps and etc..) Some other comps are already availble as readymade classes So we need to cfg by injecting required inputs.. (like Handle mappings, view resolvers and etc..) So all these should be cfg as spring beans in spring bean cfg file.. (/WEB-INF/dispatcher-servlet.xml)

In real time is recomanded to place jsp pages in private area of the web application (either in WEB-INF or its sub folders) .Even spring MVC recomands to place jsp files in private area, The advntages are

a) we can hild the technology of websiste from endusers...(Secracy or security will be improved)
b) if jsp page getting request scope data from servlet comp..then request to jsp page can be avoided and we can eleminate the possibility of displaying ugly values..

WEB-INF
|--->pages
|---->result.jsp



Xml driven spring MVC configurations (Code based flow)

Controler/Handler class

=>It is java class that implements org.sf.web.servlet.mvc.Controller (I) ...
=> Every controller must be cfg as spring bean in spring bean cfg file having bean id
=> It contain b.logic directly to process the request as Model layer comp or it can talk service, DAO classes to get results from them as Controller layer comp (recomanded)
=> In xml driven cfg we take this class on 1 per each action/work basis..

```
//Controller/Handler class                          Invasive
public class MyController implements Controller(I) {

 //Handler method                                   (j)
  public ModelAndView handleRequest(HttpServletRequest req,
                                    HttpServletResponse res)throws Exception{
    return new ModelAndView("result","sysDate", new Date());
  }          (k)
}
```

LVN (must be string) — model attr name (must be String) — (model attr value-can be any object)

in dispatcher-servlet.xml (b)

```
<bean id="mc" class="pkg.MyController"/>
          (i)
```

various types of MVC controllers (Only in xml driven cfgs)

Readymade Controller classes to simplify diff usescases

(a) AbstractController --> for hyperlinks
(b) AbtsrractCommandController --> for form submission with out validions
(c) SimpleFormController --> form form submission with validations
(d) MultiAtionContrller --> to handler multiple hyper links /multiple submits button requests
(e) AbstractWizardFormContrlller --->To display and process form that is coming in chain and etc..
(x) Removed from Spring 4.x for supporting more annotation based programming

---

Maintence projects of spring MVC ---> xml cfgs or xml +annotation cfgs
new Projects of spring mvc ---> spring boot MVC or xml + annotations cfgs

HandlerMapping

=> This is a java class implementing org.springframework.web.servlet.HandlerMapping(I)
=> This comp is useful to map incoming uri/urls with Handller /controller classes i.e it each incoming uri with one handler class .... and returns that handler class bean id back to DS ...
=> We generally do not get the need of developing user-defined Handler mappings.. becoz we have lots of ready made handler mappings classes.

eg:: =>SimpleUrlHandlerMapping (best in xml driven spring mvc cfgs)
=>BeanNameUrlHandlerMapping (default in xml driven cfgs)
=>ControllerClassNameHandlerMapping (removed from spring 4.x onwords)
=>DefaultAnnotationHanlderMapping (removed from spring 5.x -->given for annotation driven cfgs)
(default for annotation driven cfgs before spring 5.x)
=>RequestMappingHandlerMapping (best for annotation driven cfgs .. default also from spring 5)

=>Every HandlerMapping must be cfg in spring bean cfg file

in dispatcher-servelt.xml (b)

```
<bean id="suhm" class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
   <property name="mappings">                          java.util.Properties
      <props>                                  (g)
         <prop key="/home.htm">mc </prop>
      </props>
   </property>
</bean>
```

key = incoming request uri; value = handler/controller class bean id

ViewResolver
============
=> It s java class that implements org.springframework.web.servlet.ViewResolver(I)
=> This comp takes logical view name from DS and gives View Object (View(I) impl class obj) having the physical view name location..
=>This useful to achieve loose coupling towards changing UI technology of View layer with out effecting other comps..
=> We have lots of readymade ViewResolvers ... to fufill our requirements.. So no need of developing user-defined ViewResolver....

=>InternalResourceViewResolver (useful to use jsp/servlets comps of private area as view comps)
=>UrlBasedViewResolver (Allows to take any technology view Comp)
=>ResourceBundleViewResolver (Allows to collect info about physical view comps and location from properties file)
=>XmlViewResolver (Allows to collect info about physical view comps and location from xml file)
=>BeanNameViewResolver (allows to take java classes view comps)
=>TilesViewResolver (allows to tiles f/w for views)
and etc...

note:: In different usecases.. we use diff ViewResolvers...

Every ViewResolver must be cfg in spring bean cfg file as spring bean.

In dispatcher-servlet.xml (b)
=====================

```
<bean id="irvr" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
   <property name="prefix" value="/WEB-INF/pages/"/>
   <prperty name="suffix" value=".jsp"/>
</bean>
```

(m)

Recived logical view name(lvn) from DS and appends prefix(location) and suffix (extension) of phyisical view comp and returns View object having that physical view comp name and location

View Object (n)
(o) /WEB-INF/pages/result.jsp
prefix suffix
lvn

/WEB-INF/pages/result.jsp (physical view comp)

```
<h1> welcome to Spring MVC </h1>
model data:: ${sysDate}
```
(p)

web.xml
=========

web.xml (b)
========

```
<web-app>
  <!-- servlet cfg -->
  <servlet>
    <servlet-name>dispatcher </servlet-name>
    <servlet-class>org.sf.web.servlet.DispatcherServlet </servlet-class>
    <load-on-startup>2</load-on-startup>
  </servlet>
  <!--Servlet maping -->
  <servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern> *.htm </url-pattern>
  </servlet-mapping>
  <welcome-file-list>
    <welcome-file>index.jsp </welcome-file>
  </welcome-file-list>
</web-app>
```

>=0 should be given

(Extension match url pattern)

(d)

DispatcherServlet obj
(Frontrcontroller)
(f) (h)
*.htm
(l) (n)
(q)
browser as response
(r)

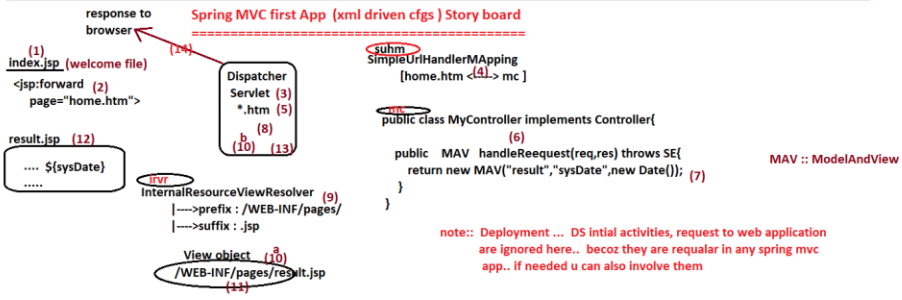
our web app name is ::
FirstMVCApp

(a) deployment → becoz of <load-on-startup>
(b) DS intial operations (all pre-instantations and injections on DS, spring beans takes place)
(c) request generation from browser
http://localhost:3030/FirstMVCApp

index.jsp (default welcome in most of the servers)

```
<jsp:forward page="home.htm"/>
```
(e) (generates implicit request to spring mvc web app based on the url home.htm)

Spring MVC first App (xml driven cfgs) Story board
==========



note:: Deployment ... DS intial activities, request to web application are ignored here.. becoz they are requalar in any spring mvc app.. if needed u can also involve them

Steps to develop spring MVC First Web application which displays the private area web comp as home page

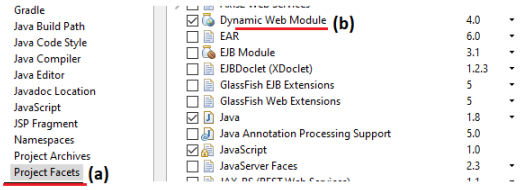
==========

step1) make sure that Tomcat server is cfg with Eclipse IDE

step2) create Gradle Project... name:: MVCProj1-Xml-FirstApp-ShowingHomePage

step3) Perform following operations on Gradle Project to make it as web application.. Project

RTC On Proj (Right click on Project) ----->properites -----> porject facects ---->select dynamic web module : 4.0



windows +shift +s for snipper tool.

step4) add the following jar files or dependencies to the build.gradle

build.gradle

==========

note: The jars added using maven/gradle build tool to the web application will placed automatically in classpath/buildpath and also in WEB-INF/lib folder

```
plugins {
    // Apply the java-library plugin to add support for Java Library
    id 'war'
}
repositories {
    // Use jcenter for resolving dependencies.
    // You can declare any Maven/Ivy/file repository here.
    jcenter()
}
dependencies {
   // https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api
  implementation group: 'javax.servlet', name: 'javax.servlet-api', version: '4.0.0'
  // https://mvnrepository.com/artifact/org.springframework/spring-webmvc
 implementation group: 'org.springframework', name: 'spring-webmvc', version: '5.2.9.RELEASE'


}
```

step4) add web.xml file and configure DispatcherServlet and welcome file..

RTC on WEB-INF --->new ---> others ---> xml ---> .... (web.xml)

```
<web-app>

  <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
      <servlet>
            <servlet-name>dispatcher</servlet-name>
            <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
            <load-on-startup>2</load-on-startup>
      </servlet>

      <!-- Map all requests to the DispatcherServlet for handling -->
      <servlet-mapping>
            <servlet-name>dispatcher</servlet-name>
            <url-pattern>*.htm</url-pattern>
      </servlet-mapping>

      <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
      </welcome-file-list>

</web-app>
```

ctrl +space and select disptacherservlet to get this code dynamically

(<DS logical name>-servlet.xml)

</web-app>

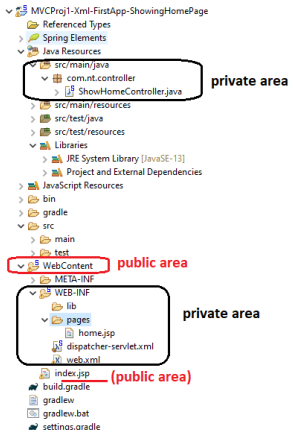
(<DS logical name>-servlet.xml)

step5) add spring bean cfg file as dispatcher-servlet.xml in WEB-INF folder

RTC on WEB-INF ---> new ---> others -->spring bean file ---> dispatcher-servlet.xml ---> .....

step6) create "pages" folder WEB-INF folder to make main jsp pages/files in private area..

=>pages is not standard folder name, So u can take any name..
=>WEB-INF,classes,lib ,web.xml names standard names. and we can not change them



step7) create packages in src/main/java folder and complete the development...

src/main/java (write the source code)
|--->com.nt.controller

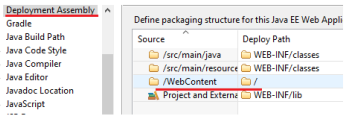
note:: keep welcome file(index.jsp) always in public area (webcontent folder)

step8) Run the Application...

RTC on Project --->
run as -->run on server --->
select tomcat...

note:: In Gradle , by default webcontent folder does not participate in deployment.. So we perform deployment assembly expcitly..

RTC on Proj ---> properties ----> Deployment Assembly -----> Add ----> folder --->webcontent ---> .....



=>In standard web application directory structure WEB-INF and its sub folders are called private area...
i.e only ServletContainer can use them.. (not the outsiders) .. where the outside of Area of of web WEB-INF folder (like webcontent folder data) comes under prublic area.. i.e any one can access that area..

We can pass inputs to SErvlet comp in two ways

a) request params (form data)
--> if the inputs are non-technical and expected from enduser through browser like name,age,gender and etc..
--> will be stored automatically in request object ..
--> use request.getParamter(-) method to read them

b) init params/context params
->if the inputs are technical inputs and expected from programmers through web.xml file like driver class name, jdbc url, dbuser and etc...
-> will be stored in SErvletConfig object (init params), Servletcontext obj(context params)
=> use cg.getInitParameter(-) or sc.getInitParameter(-) to read these values
cg--->Servletconfig object
sc -->ServletContet object.

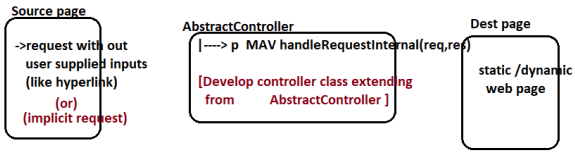
note:: init params are specific to one servlet /jsp comp
note : context params are common for all web comps of a web application..

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/cfgs/mycfg.xml</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
</servlet>
```

Fixed init param of DS to specify the name and location of spring bean cfg file

note:: In xml driven spring MVC .. developing controller class directly by implementing pkg.Controller(I)
is bad pratice.. becoz we have different readymadate controller classes for differ usecases...
like AbstractController, MultiActionController and etc...

AbstractController
==========

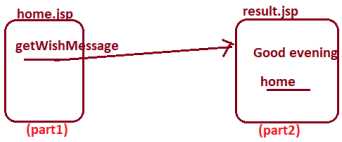


usecases:: listAllEmployees
getAllfriends getCurrentTredingjobs getWishMessage showCoranaReport

(these hyperlinks may or may not have query string supplied inputs .. but user-supplied inputs)

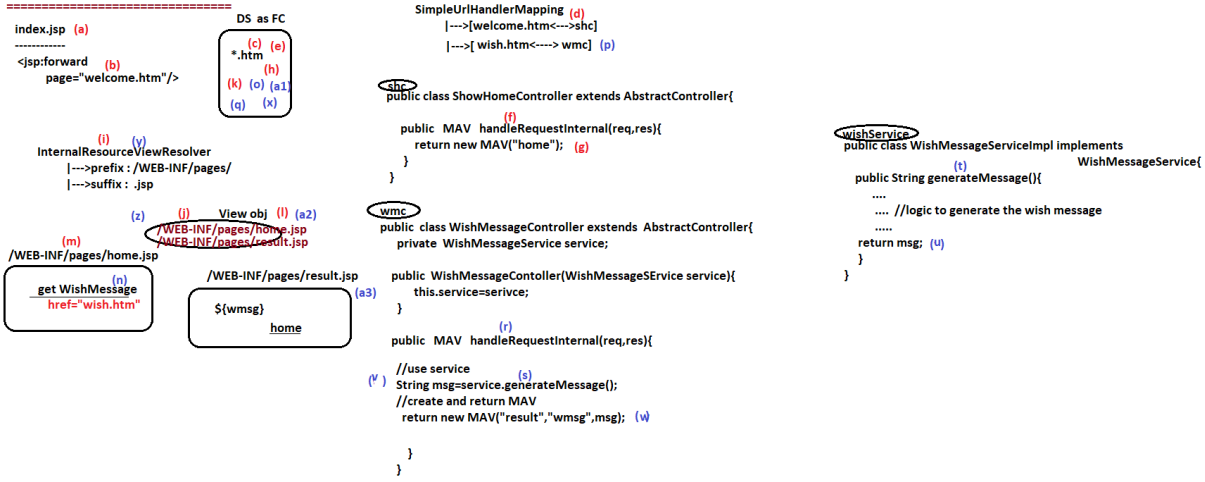
 get WishMessage (or)

getBalance



2 Controllers (AbstractController)
|--->ShowHomeController (To show home page with out any request processing)
|--->WishMessageController (To show result page with message after request processing)

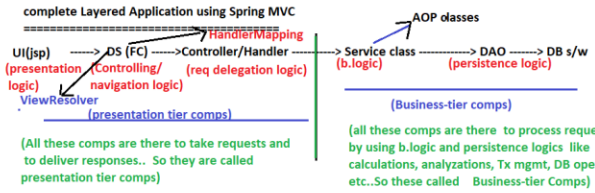
StroryBoard of WishApp Applicaiton
==========



//java 10 feature local variable type inference (compiler decides the data type based on the initial values)
// using var we can not go for compound variables decl .. we can not declare two variables in same line
// var type variables can not be intialized with null ... but can be intialized simple values.. like 0,0,0 and etc.
//var type can be taken only for local variables..

Different controller type Controller classes are having diff handler methods ... but DS calls all those methods .. through handleRequest(req,res) method.

AbstractController -----> handleRquestInternal(req,res)
AbstractCommandController ---> handle(req,res,.. ,...)
SimpleFormContrller ---> onSubmit(-,-,-,-)
and etc..



=>We need to develop the presentation-tier , business tier comps having loose coupling i.e the degree of Dependency should less..[This indicates we should be in a position to use Spring in each tier (presentation tier or business tier irrepspective spring is used or not in another another)

=>To implement the above concept, we need to take two IOC cotnainer 1 for presentation tier comps and another one for business-tier comps..

=> DS created IOC container for presentation tier comps

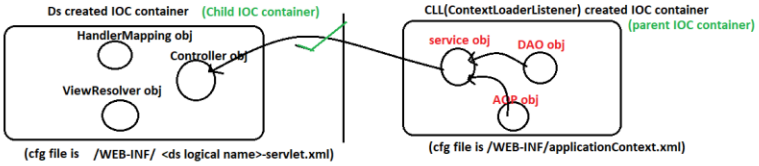
=> ContextLoaderListener created IOC container for business-tier comps

It is built-in Servletcontext Listener

|---> It creates IOC container when ServletContext obj is created (during the deployment of web application) (by calling contextInitialized(-) method)

|---> It stops /closes same IOC container when ServletContext obj is destoyed [by calling contextDestoyed(-) method) (When web applicaiton is stopped or reloaded/undeployed)

note: The ContextLoaderListener created IOC container takes /WEB-INF/applicationContext.xml as the default spring bean cfg file..



=> Since we need to inject Service class obj to Controller object.. So we need to take CLL created IOC container that manages business tier comps as parent IOP container and DS created IOC container that manages presentation tier comps as the child container...

note:: Parent IOC container Spring Bean managed can be injected to child IOC container spring bean managed.. but reverse is not possible..

=>XmlWebApplicationContext IOC container is self Intelligent IOC container .. becoz when it is created and if it notices another IOC container of same type is already available or created then it make already available/created IOC container as parent IOC container and current created IOC conttainer as child Container...

Since CLL creates first IOC container of XmlWebApplicationContext and DS creates Second IOC container of same type , So the DS created IOC container automatically becomes Child IOC contanier by taking CLL created IOC container as parent IOC contanier..

Objects creation order during the deployment of web application

=>Listener classes objs
=>ServletContext object
=> FilterConfig objs (if filters are cfg)
=> Filter objs (if filters are cfg)
=> ServletConfig objs (if SErvlets are cfg)
=> <load-on-startup>enabled Servlet class objs

web.xml for two IOC containers

(a) -->Deployment

(c)Servletcontext obj

(d) CLL created IOC container (parent)

(b) CLL object

(e) (InMemory MetaData) /WEB-INF/applicationContext.xml

(f) pre-instatation of singleton scope beans and injections)

```
<web-app>
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener </listener-class>
  </listener>

  <servlet>
    <servlet-name> dispatcher</servlet-name>
    <servlet-class> org.sf.web.mvc.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>2 </load-on-startup>
  </servlet>
  <servlet-mapping>
    <servlet-name> dispatcher</servlet-name>
    <url-pattern>*.htm </url-pattern>
  </servlet-mapping>

</web-app>
```

(g) DS obj creation

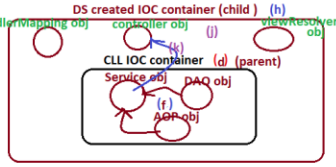
(child)

(h) DS created IOC container

(i) InMemory MetaData /WEB-INF/dispatcher-servlet.xml

(j) pre-instantiation of singleton scope beans

note:: Some old versions of existing servers like weblogic 8/9 if DS comp <load -on-startup> value 0/1 ..there is possiblity of creating DS object first .. CLL obj.. due to this DS IOC container becomes parent CLL IOC container becomes Child IOC cintainer and the whole Service object injection to Controler will be failed.. So it recoamaded to take <load-on-starrtup> value as 2.. So that CLL object will always be created before DS object...



(l) keeping singleton scope spring objs in Internal cahes of both IOC containers..

Q ? here both ds and ContextLoaderListener are predefined and given by spring fw only na then if we change presentation tier tech agai n we need to develop that ContextLoaderListener class na? where is loose coupling here?

Ans) We can work with ContextLoaderListener .... though we are not cfg DS in web.xml becoz ... still spring is used in business tier.. So we can still add spring jar files..

Q) if we use non containered (non-spring) presentation technologies..ohw we can inject business tier objects

Ans) Use the plugins supplied by those presentation technologies.

(like JSF or only sevlet,jsp or jsp/html agngular and etc..

```
<web-app>

  <context-param>
    <param-name>contextConfigLocation</param-name>
          <param-value>/WEB-INF/business-beans.xml</param-value>
  </context-param>

      <!-- Bootstraps the root web application context before servlet initialization -->
      <listener>
            <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
      </listener>

  <!-- The front controller of this Spring Web application, responsible for handling all application requests -->
      <servlet>
            <servlet-name>dispatcher</servlet-name>
            <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
            <init-param>
              <param-name>contextConfigLocation</param-name>
              <param-value>/WEB-INF/presentation-beans.xml</param-value>
            </init-param>
            <load-on-startup>2</load-on-startup>
      </servlet>

      <!-- Map all requests to the DispatcherServlet for handling -->
      <servlet-mapping>
            <servlet-name>dispatcher</servlet-name>
            <url-pattern>*.htm</url-pattern>
      </servlet-mapping>

      <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
      </welcome-file-list>

</web-app>
```

Given to change the name of location of for parent IOC container spring bean cfg file... but the recomanded name and default name is /WEB-INF/applicationContext.xml

Given to change the name of location of spring bean cfg file for child container... but the recomanded name and default name is /WEB-INF/<Dslogicalname>-servlet.xml

Can we same name for Init params and context params?

Ans) Yes , Possible becoz init params allocate memory in ServletConfig object.. and they are specific to each SEr
Context params allocate Memory in ServletContext obj and they are viisble in all web comps of web applica
like servlet comps, jsp comps, listeners, filter and etc...

Different HandlerMappings

=>All handler mappings are the classes implementing pkg. HandlerMapping(I)
=>These are useful to map incoming request urls to controller/handler classes..

Different handler Mappings

=>BeanNameUrlHandlerMapping (default in xml driven cfgs)
=>SimpleUrlHandlerMapping (mostly used in xml driven cfgs)
=>ControllerClassNameHandlerMapping (removed from spring 5.x) 4.x
=>DefaultAnnotationHandlerMapping (default in annotation driven cfgs upto spring 4.x and removed in spring 5.x)
=>RequestMappingHandlerMapping (default in annotation driven cfgs from spring 5.x)
and etc...

BeanNameUrlHandlerMapping

=>It is Default , if no handler mapping is cfg in xml driven cfgs..
=> It will map Incoming request uri with that handler/controller who is having incoming request uri as the bean id.

incoming request uri is :: /welcome.htm

in dispatcher-servlet.xml

```
<bean class="pkg.BeanNameUrlHandlerMapping"/>

<bean id="/welcome.htm" class="pkg.ShowHomeController"/>
```

ControllerClassNameHandlerMapping

=>Removed from spring 4.x and 5.x becoz its confusing functionality..

=> takes incoming uri .. removes the extension .. makes first letter as upper case letter and appends Controller then search Controler class name having that name to map the request..

welcome.htm --------maped to----------> WelcomeController
home.htm ----------maped to----------> HomeController

(TEst in spring mvc 3.x setup)

SimpleUrlHandlerMapping

=> Allows to map different incoming uris with different controller by taking the support of "mappings" property of type "java.util.Properties".

In applicationContext.xml

```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="welcome.htm">shc</prop>
            <prop key="wish.htm">wmc</prop>
        </props>
    </property>
</bean>

    <!-- contrIller classes for SimpleUrlHandlerMapping -->
  <bean id="shc" class="com.nt.controller.ShowHomeController"/>
  <bean id="wmc" class="com.nt.controller.WishMessageController">
    <constructor-arg ref="wishService"/>
  </bean>
```

HandlerMapping Chaining

=> if we cfg multiple HandlerMappings.. they will work having chaining among them.. i.e if first HandlerMapping fails to locate controller class...then it passes to next handler mappings... in the order of cfg..

```
<!-- Handler Mapping -->
  <bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/>
  <bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
            <prop key="welcome.htm">shc</prop>
            <prop key="wish.htm">shc</prop>
        </props>
    </property>
  </bean>
```

=>While working multiple handlemappings.. if any ambiguity problem comes for any incoming request uri i.e one incoming uri is mapped with two diff controller classes by using two different handler mappings.. then the 1st configured handler mapping linked controller will be picked up..

Incoming request uri is :: welcome.htm (a)

```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
   <property name="mappings">
      <props>
         <prop key="welcome.htm">shc</prop>   (b)
      </props>
   </property>
</bean>

<bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"/>

<bean id="shc" class="com.nt.controller.ShowHomeController1"/>   (c)
<bean id="/welcome.htm" class="com.nt.controller.ShowHomeController"/>
```

1st HM

2nd HM

---

if want give prority to another handlermapping , not based on order of cfg then we specify that prority order by injecting numeric value to "order" property of HandlerMappings.. which indicates high value indicates low priority and low value indicates high priority .

incomling request uri :: welcome.htm (a)

```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
     <props>
        <prop key="welcome.htm">shc</prop>
     </props>
  </property>
  <property name="order" value="2"></property>
</bean>

<bean class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping">
  <property name="order" value="1"/>          (b) --> becoz of low priority value
</bean>
```

default order value is :: 65535 (big value)

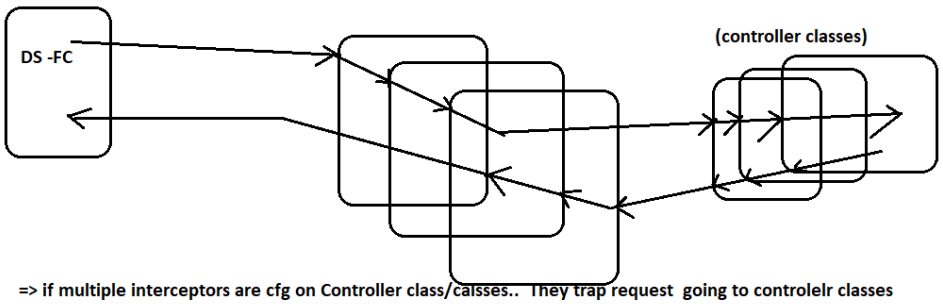
```
<!--Controller for SimpleUrlHandlerMapping -->
<bean id="shc" class="com.nt.controller.ShowHomeController1"/>

<!-- Controllers for BeanNAmeUrlHandlerMapping-->          (c)
<bean id="/welcome.htm" class="com.nt.controller.ShowHomeController"/>
```

Handler Interceptors (These are like servlet filters to controller classes)

==========

=>These are extension hooks to Controller/Handler classes to add pre/post logics having ability to enable or disable with out the touching the source code of Controller/Handler classes..



=> if multiple interceptors are cfg on Controller class/calsses.. They trap request going to controlelr classes in the order of cfg...and they trap response coming from controller classes in reverse order...

=> To develop Interceptor class our classe must implement spring supplied pkg.HandlerInterceptor (I)

(In spring 5 it is given java 8 interface having default methods)

HandlerInterceptor methods are
==========
default void afterCompletion(req,res,Object handler, Exception ex) throws Exception
|-->Executes after rendering the view comp ..useful for clean up operations like removing data from session scopes
default void postHandle(req,res,Object handler, MAV mav) throws Exception
|-->Executes after executing the controlelr..useful for post processing logics like converting model attrbute values to different formats like chaning number formats, currecy symbols.. date formats and etc..
default boolean preHandle(req,res,Object handler) throws Exception
|-->Executes before executing the controlle..useful for pre processing logics like checking browser type, .. checking timings, Authenticaiton and etc...
if this method returns "true" then Controller of flow of execution will execute.. otherwise the controller will not execute..

=>Upto Spring 4 Developer prefered developing handler /controller class by extending from HandlerInterceptorAdapter class which is a pre-defined class implementing HandlerInterceptor (I) provioding null method definitations for all the methods... Show we can override only those methods in which we are intrested in...

=>From spring5 .. we can develop Interceptor class directly by implemeting HandlerInterceptor(I) becoz it is given Java 8 interface with default methods.. So we can overiride only those methods in which we are intrested in..

Procedure to add Inteceptor that allows request between 9 am to 5pm to Application

step1) keep any old App ready (like WishApp)

step2) Develop handler interceptor..

```
public class TimingsCheckingInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest req, HttpServletResponse res, Object handler)
            throws Exception {
        var hour=0;
        RequestDispatcher rd=null;
        //get Current hour of day..
        hour=Calendar.getInstance().get(Calendar.HOUR_OF_DAY);
        //check the timings
        if(hour<9 || hour>17) {
            rd=req.getRequestDispatcher("timeout.jsp");
            rd.forward(req,res);
            return false;
        }
        else
            return true;
    }

}
```

step3) cfg Handler class in spring bean cfg file (dispatcher-servlet.xml)

```
<!-- Interceptors cfg -->
   <bean id="tci" class="com.nt.interceptor.TimingsCheckingInterceptor"/>
```

step4) Link Interceptor with Controler classe(s) using HandlerMapping.. support.

```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
   <property name="mappings">
      <props>
         <prop key="welcome.htm">shc</prop>
         <prop key="wish.htm">wmc</prop>
      </props>
   </property>
   <property name="interceptors">
      <array>
         <ref bean="tci"/>
      </array>
   </property>
</bean>
```

step5) develop that timeout.jsp (in public area ---> In webcontent folder outside WEB-INF folder)

timeout.jsp
==========

```
<h1 style="color:red;text-align:center">Request must be given between 9am to 5 pm</h1>
   <br>
  <a href="welcome.htm">home</a>
```

Task:: Develop interceptor checking the browser type... (allow only Chrome browser requests)?

Q) I want to apply inteceptor only on specific bunch of controller .. not on all controllers then how should we do?

Q) Why interceptors cfg is taking place through handler Mappings...

Problem::

```
interface I1{
   void m1();
   void m2();
   void m3();
}

public class App1 implements I1{
   ...
   ... we must implement all the 3 methods..
   ...
}
```

Solution1:: (Legcay)
========== (Adapter class)

```
public class I1Adapter implements I1{
   public void m1(){ } //null method
   public void m2(){ } //null method
   public void m3(){ } //null method
}

public class App1 extends I1Adapter{
   ... //no need of overriding all the 3 methods
   ... // we can implement only one method in which are intrested in
}
```

Solution2: (java 8 interface with default methods) (Best)
==========

```
interface I1{
default void m1(){ .. }
default void m2(){...}
default void m3() {..}
}

public class App1 implements I1{
   ...
   ... // override that method in which u r intrested in..
   ...
}
```

=> javax.servlet.Servlet(I) is still normal java8 interface ... by keeping older versions compability in mind..

why we are cfg interceptors with HandlerMappings, why not with Controller classes directly?

=>we generally apply interceptors on multiple controllers .. HanlderMapping comp is the place where we map multiple incoming request uris with multiple controller.. So we prefer HandlerMappings to apply interceptors on multiple controllers..

How to apply spefic interceptors on specific controller classes ?

Ans) Map related controloer classes with their incoming request uri using seperate HandlerMapping and link interceptor on controller classes using that handlerMapping..

```
<bean class="SimpleUrlHadlerMapping">
  <property name="mappings"
     <props>
       <prop key="home.htm">shc </prop>
       <prop key="emps.htm">ec </prop>
     </props>
  </property>
  <property name="inteceptors">
    <list>
       <bean class="pkg.TimingChekingInterceptor"/>
    </list>                    (inner bean)
  </property>
</bean>
```

```
<bean class="SimpleUrlHadlerMapping">
  <property name="mappings"
     <props>
       <prop key=" wish.htm">wmg</prop>
       <prop key="exam.htm">exc</prop>
     </props>
  </property>
  <property name="inteceptors">
    <list>
       <bean class="pkg. BrowserCheckingInterceptor"/>
    </list>                    (inner bean)
  </property>
</bean>
```

ViewResolvers

==========

=> These are given to resolve/indetify the phyiscal view comp name and location based on the given logical view name.. It returns View object having the name and location of phyiscal view comp

=> All ViewResolver are implementation classes of org.sf.web.servlet.ViewResolver (I)

eg:: InternalResourceviewResolver
UrlBasedViewResolver
ResourceBundleViewResolver
XmlViewResolver
TilesViewresilver
and etc..

=>View objects are the objects of classes implementingorg.springframework.web.servlet.View(I). On this View Object render(-) method will be called in order pass the control to physical UI comp..

eg: InternalResourceView
JstlView
XlstView
TilesView
VolociyView
and etc...

UrlBaseViewResolver

==========

=> capable of resolving /indentifying physical view name and location of any technology .. but we must cfg View class explicity...

for locating jsp/servlet comps of private area cfg "InternalResourceView" or JstlView as the View class

for working Tiles env.. cfg "TilesView" as the view class

for working with freemarker env.. cfg "FreeMarkerView" as the view class..

eg::
```
<bean class="org.springframework.web.servlet.view.UrlBasedViewResolver">
    <property name="viewClass" value="org.sf.web.servlet.InternalResource"/> (or)
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/WEB-INF/pages/"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

View

matically

==========

eclipse {

What is the diff b/w JstlView and InternalResourceView classes?

=>InternalResourceView makes the programmer to added jstl jar files to the WEB-INF/lib folder only when jstl tags are used in the jsp pages otherwise not required.

=>JstlView makes the programmer to added jstl jar files(2) to the WEB-INF/lib folder irrespective of jstl tags are used or not in jsp page..

=>JstlView class extends from InternalResourceView class

```
eclipse {
        wtp {
                component {
                        resource sourcePath: "/WebContent", deployPath: "/"
                }
        }
}
```

if the view class not cfg while working with UrlBasedViewResolver then we get

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'org.springframework.web.servlet.view.UrlBasedViewResolver#0' defined in ServletContext resource [/WEB-INF/dispatcher-servlet.xml]: Initialization of bean failed; nested exception is java.lang.IllegalArgumentException: Property 'viewClass' is required

JSTL -->Jsp Standard tag library :: It set of Jsp tablibraries having bunch tags in each jsp taglibrary -->These are useful to make jsp comps/pages java code less jsp pages or sciptless jsp pages ...

JSTL Taglibraries are

1)Core 2) Sql c) fomatting d) xml e)functions

note:: every JSTL jsp taglibrary is identity with its taglib uri which is fixed taglib uri ::

Core ---> (c) ---> http://java.sun.com/jsp/jstl/core
sql ---> (sql) ---> http://java.sun.com/jsp/jstl/sql
formatting ---> (fmt) ---> http://java.sun.com/jsp/jstl/fmt
xml ---> (x) ---> http://java.sun.com/jsp/jstl/xml
functions ---> (fn) ---> http://java.sun.com/jsp/jstl/fn

we add jar files to work with JSTL tags/tag libraries

```
// https://mvnrepository.com/artifact/javax.servlet/jstl                    (jstl.jar)
//implementation group: 'javax.servlet', name: 'jstl', version: '1.2'

// https://mvnrepository.com/artifact/taglibs/standard                      (standard.jar)
//implementation group: 'taglibs', name: 'standard', version: '1.1.2'
```

To JSTL tags .. we need import tablibray url to the jsp page using <%@taglib ...%> by specifying user-defined prefix.

result.jsp (Not fixed, So useful to differetiate the jsp tags of two different jsp taglibraries when both have same names with diff functionalaties

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

        <h2 style="color:red;text-align:center"> Result message is :: <c:out value="${wmsg}"/> <br> </h2>
```

JSTL +EL to display scoped data on to the browser..

InternalResoruceViewResolver

=> Sub class of UrlBasedViewResolver ... but given to only to use the private are servlet/jsp comps as view comps... by taking "InternalResourceview" or "JstlView" as the default view class..

=>This can not used to work with different view Technologies... like Tiles, Volocity and etc... This can use only Servlet,jsp,html files of private area as the view comps..

=>Here View class cfg is optional.. if jstl jars are added then it will "JstlView" as default view class otherwise it will "InternalResourceView" as the view class..

note:: Both UrlBasedViewReoslver and InternalResourceViewResolver does not support ViewResolverChaining..
Even ResourceBundleViewResolver, XmlViewResolver also do not support view resolver chaining...

note:: BeanNameViewResolver(to take java classes as view classes), TilesViewResolver supports ViewResolverChaining...

ResourceBundleViewResolver

=>Allow cfg view class name , url seperetly for each logical view name (lvn) by taking the support of properties file (ResourceBundle)..

=>It allows to cfg diff technology view comps for different logical view names and also to place diff view comps in different locations with different extensions..

=>Default properites file name is views.properties (src/main/java folder) .. if any other name or location is taken we must explictly configure it.

views.properites (src/main/java)

```
# <lvn>.(class) = <fully qualified view class name>
 home .(class) = org.sf. web.servlet.InteranlResourceView
#<lvn>.url =  name and location ,extension of physical view comp
     home .url=/WEB-INF/html/ first_page.html

 # add for more lvns
 result. (class) =org.sf.web.servlet.JstlView
 result.url=/WEB-INF/pages/show_result.jsp
```

While working UrlBasedUrlViewResolver or InternalResourceViewREsolver the limitations are

=> All view comps must be there in same location having same prefix and must have same extension i.e same technology

=> There is tight coupling lvn and physical view file name

=> All view comps should be rendered using same view class

note: To overcome these problems use ResourceBundleViewResolver or XmlViewResolver

in dispatcher-servlet.xml

```
<bean class="pkg.ResourceBundleViewResolver"
   <property name="basename"  value="views"/>
</bean>
```

(or)

ResourceBundleViewResolver takes src/main/java/views.properties as the default properties file name and location.

In Normal Eclipse Dyanmic web applicaito it is going to be src/views.propeties.

```
<!-- View Resolver -->
  <bean class="org.springframework.web.servlet.view.ResourceBundleViewResolver">
    <!--  <property name="basename" value="views"/> -->
     <property name="basename" value="com/nt/commons/views"/>
  </bean>
```

XmlViewResolver

=> same as ResourceBundleViewResolver ... but allows to take inputs from another xml file..
=> The default name is /WEB-INF/views.xml file.. if any other file name or location is taken u must specify.. explicity using "location" property..

WEB-INF/views.xml (seperate xml file)
==========

```
<beans ...>   lvn                                                    View class name
  <bean id="home"   class="org.springframework.web.servlet.view.InternalResourceView">
      <property name="url"  value="/WEB-INF/html/first_page.html"/>
  </bean>
            fixed                physical view comp name and Location
            name
  <bean id="result"  class="org.springframework.web.servlet.view.JstlView ">
      <property name="url"  value="/WEB-INF/pages/show_result.jsp"/>
  </bean>
</beans>
```

dispatcher-servlet.xml
==========

```
<bean class="org.springframework.web.servlet.view.XmlViewResolver">
    <property name="location"  value="/WEB-INF/views.xml"/>
</bean>
```

```
<bean class="org.springframework.web.servlet.view.XmlViewResolver">
    <!--  <property name="location" value="/WEB-INF/views.xml"/> -->
     <property name="location"  value="/WEB-INF/cfgs/views.xml"/>
</bean>
```

=>we can avoid giving extension words in the incoming uris.. if we cfg DS with "/" url pattern.. This make DS and FrontController cum DefaultServlet i.e if not one is taking this will trap and take the request..

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <load-on-startup>2</load-on-startup>
</servlet>

<!-- Map all requests to the DispatcherServlet for handling -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

Incoming request uris in dispatcher-servlet.xml

```
<bean class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
        <props>
          <prop key="welcome">shc</prop>
          <prop key="wish">wmc</prop>
        </props>
    </property>
</bean>
```

Limitations of xml driven spring MVC Application development

=>Controller/Handler classes are invasive

=> handler methods signatures are fixed.. though we are not using most of the params like req,res we need to take them..

=> Handler method return type is ModelAndView object.. (fixed).. Though we want give only "lvn" still we need to take complete MAV object..

=> We can not develop single controller class for multiple uses cases... by placing multiple handler methods

Annotation cfgs driven Spring WebMVC web application devleopment

step1) Take controller classes as normal java classes having @Controller annotations and link them with spring bean cfg file using <context:compent-scan ..../>

step2) add handler methods with flexible singnature (return type, method name, params are flexible) and add @RequestMapping annotation on the top of handler method specifying incoming request uri/url..

step3) Cfg DefaultAnnotationHandlerMapping(upto 4.x) or RequestMappingHandlerMapping (from 5.x) as Hanldermapping in spring bean cfg file (if not cfg also.. they become default)

step4) cfg u r choice ViewResolver..

Sample Controller class

```
package com.nt.controller;

@Controller
public   class ShowHomeController{

  @ReqestMapping("/welcome")
  public  String  showHomePage(){
    return "home";
  }

}
```

Thum rule in annotation driven programming

=>use-defined classs using streo type annotaitons and link them with spring bean cfg file using <context:componet-scan .../>

=>pre-defined classes using <bean> tags of spring bean cfg file..

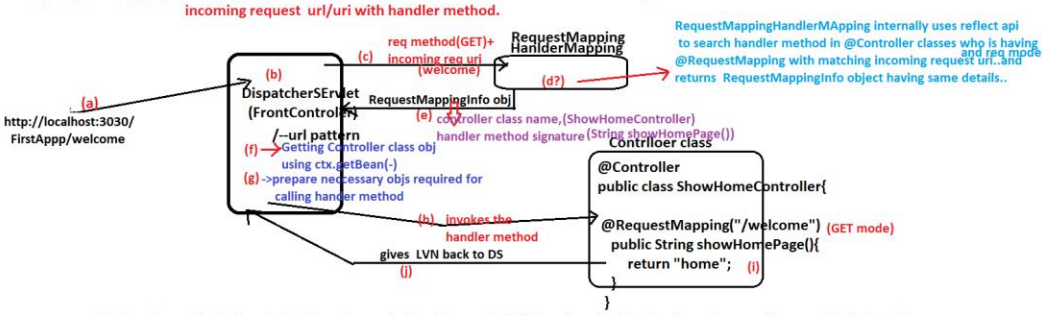
in dispatcher-servlet.xml

```
<beans ....>
   <!--Handler Mapping -->
   <bean class="RequestMappingHandlerMapping"/>

    <!--view resolver
    <bean class="pkg.InternalResourceViewResolver">
         <p  name="prefix" value="/WEB-INF/pages/"/>
         <p  name="suffix" value=".jsp"/>
    </bean>

  <context:component-scan base-package="com.nt.controller"/>
</beans>
```

@Controller :: To mark java class as spring bean cum web controller/hander class
@RequestMapping :: To mark java method of controller as handler method and also to map incoming request url/uri with handler method.



The handler methods of controller classes in annotation driven spring MVC can have flexible signatures.. however there are list of allowed parameter types and allowed return types.. (i.e we can not take other than them)

possible parameter types
=========================
=>WebRequest, NativeWebRequest
=>javax.servlet.ServletRequest, javax.servlet.ServletResponse
javax.servlet.http.HttpSession
and etc...
for more parameter type refer this...
https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/web.html#mvc-ann-arguments

most of them are given by keeping restfull web services in mind

possible return types
=====================
@ResponseBody
HttpEntity<B>, ResponseEntity<B>
String
View ,viod
@ModelAttribute
java.util.Map, org.springframework.ui.Model and etc...

most of them are given by keeping restfull web services in mind

for more return types refer
https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/web.html#mvc-ann-return-types

Annotations cfgs driven spring MVC App development advantages
==
(a) controller classes are non-invasive
(b) Handler methods are having flexible signatures..
(c) we can take controller classes on 1 per related use-casses basis by having multiple handler methods..
and etc..

---

jsps (as needed)
DS (1 per project)
controller class (1 per related use-cases)
service class (1 per module)
DAO (1 per Db table or 1 per related db tables)

In Spring MVC Annoation driven cfgs..

Story board of WishApp WEb application using Annotation driven cfgs
==

Child IOC container
==================

index.jsp (a)

<jsp:forward page="welcome" /> (b)

home.jsp (m)
---
get Wish Msg (n)

result.jsp (d1)
${wmsg}

DS -FC (c) (f) (r)
(url pattern:/)
(h) (o)
(k) (y) (b1)

InternalResourceViewReslover
|-->prefix: /WEB-INF/pages/
|--->suffix: .jsp (i) (z)

View object (j) (c1)
/WEB-INF/pages/home.jsp (l) (a1)
/WEB-INF/pages/result.jsp

(d?) (p?)
RequestMappingHanlderMapping (default)

@Controller
public class WishMessageController{
@Autowired
private IWishMessageService service;
@RequestMapping("/welcome") (e)
public String showHome(){
return "home"; (g)
}
@RequestMapping("/wish") (q)
public MAV showWishMsg(){
// use service (s)
(w) Stirng wmsg=service.generateWishMsg(); (t)
return new MAV("result","wmsg",msg); (x)
}
}

Parent IOC container
==================

@Service("wishService")
public class WishMessageSerivceImpl implements IWishMessageService{ (u)
public String genateWishMessage(){
....
....
retun msg; (v)
}
}

=>if the handler method is not giving logical view name (like method return type is void) then the incoming request uri /request path of that handler method will be taken as lvn automatically .. For this internallly uses RequestToViewNameTranslator ...

```
@RequestMapping(value="/countries",method=RequestMethod.GET)
public void fetchCountries(Model model) {
    .... //code
    ...
}
```

here "countries" will be taken as LVN , So looks for /WEB-INF/pages/countries.jsp as Physical View Name

---

To pass model data + lvn to DS to Do not take MAV as parameter of hanlder method.. take it as the return type of handler method...

```
@RequestMapping(value="/countries",method=RequestMethod.GET)
public void fetchCountries(ModelAndView mav) {
    Set<String> countriesList=null;
    //use Service
    countriesList=service.getAllCountries();
    //adding multiple model attribues
    mav.addObject("listInfo", countriesList);
    mav.addObject("operation", "countries");
    mav.addObject("countriesCount", countriesList.size());
    mav.setViewName("show_results");
}//method
```

MAV is not part of possible parameter types for handler method...

(Bad pratice will not work)

```
@RequestMapping(value="/countries/add",method=RequestMethod.GET)
public Map<String,Object> fetchCountries() {
    Set<String> countriesList=null;
    Map<String,Object> map=new HashMap();
    //use Service
    countriesList=service.getAllCountries();
    //adding multiple model attribues
    map.put("listInfo", countriesList);
    map.put("operation", "countries");
    map.put("countriesCount", countriesList.size());
    return map;
}//method
```

problems in this signature::

(a) we need to create Map object manually to add model data
(b) No control on lvn..since multiple words are in the request path (countries/add) then it takes "add" lvn and "countries" as the folder name overall it lookes for "/WEB-INF/pages/countries/add.jsp"

Taking Model /ModelMap/Map as the return type of the handler method is having fillowing limitations.. So not reocmanded to take them as retunr type
(a) We should create their objects manually in roder to model data
(b) Some times getting their impl class names/sub class names to create obj is bit complex
(c) We do not get control on lvn , more ever request path of handler method becomes lvn

note:: It recomanded to take same as the parameter types of the handler methods the reasons are
(a) DS will create those objects ... so our hanlder method just recive them as parameter to keep model data
(b) Need not to know impl /sub class names of the above to create obejcts..
(c) By taking return type as string , we can get controller on lvn

Best signature of for handler method is
==========

```
@RequestMapping(value="/countries",method=RequestMethod.GET)
public String fetchCountries(Model model) {    (Best)
    Set<String> countriesList=null;
    //use Service
    countriesList=service.getAllCountries();
    //adding multiple model attribues
    model.addAttribute("listInfo", countriesList);
    model.addAttribute("operation", "countries");
    model.addAttribute("countriesCount", countriesList.size());
    //return lvn
    return "show_results";
}//method
```

Rules to develop Controller/Handler class

a) Class must be a public class.. (To make IOC container loading and instnating the class)
b) We must add @Controller annotation to make it as spring bean cum web controller
c) we can take mulitple public methods as handler methods by adding @RequestMapping annotaitons specifying request PATH and request mode/method (GET,POST --> Spring MVC)

d) Multiple handler methods can not same PATH in same controller class and same req mode/method

```
@Controller
public class MyController{

 @RequestMapping("/welcome");          #Invalid
 public void showHome(){                (default GET)
   ..
 }
 @RequestMapping("/welcome");          (default GET)
 public void showResult(){
   ..                                   throws excetopn java.lang.IllegalStateException: Ambiguous mapping
 }
}
```

note:: Spring MVC is given to develop web applicaitons having capability to take requests from browser.. Browser can give only two modes/methods of request they are (GET,POST).. So the handler methods of Spring MVC controller can also deal with only GET,POST mode requests..

=>Http supports actually 8 modes requests.. from different types of clients ..
=>browser as client , can send only "GET","POST" mode requests.

All the 8 methods/modes are request are
GET,POST,HEAD,PUT,DELETE,TRACE, PATCH,OPTIONS ,CONNECT (reserved)

=>While working Restfull webServices we will use all the 8 modes or max mode requests.. becoz as web Service comp it should allow requests from diff types of clients .. (not just browser)...

browser, postman,swagger, mobileApp, Desktop App, Soap ui and etc..

e) request PATHs are case-sensitive

```
@Controller
public class MyController{

 @RequestMapping("/WELCOME");
 public void showHome(){                # valid
   ..
 }
 @RequestMapping("/welcome");           does not throw any exception..
 public void showResult(){
   ..
 }
}
```

f) different handler methods of a controller can have same request PATH with diffrent request methods /modes (1 for GET, 1 POST)

```
@Controller
public class MyController{

 @RequestMapping(value="/welcome",method=RequestMethod.GET);
 public void showHome(){                                          #Valid
   ..
 }
 @RequestMapping(path="/welcome",method=RequestMethod.POST);
 public void showResult(){
   ..
 }
}
```

input jsp /html (UI code)

```
<form action="welcome" method="POST">
   <input type="submit" value="submit"> <!-- POST mode request -->
</form>
<br>
<a href="welcome">go</a>  <!-- GET mode request -->
```

404 error :: if the request resource is not found (incomin url not matching with handler method request path)
405 error :: incoming request url is matching with handler method request path .. but request mode/method (GET/POST) not matching..
500 :: Any Exception in the execution..

g) One handler method can have multipe request paths

```
@Controller
public class MyController{

 @RequestMapping(value={"/welcome","/login","/home"})
 public void showHome(){
   ..
 }
}
```

html/jsp/ui page to generate requests

```
<a href="welcome">go1</a>  <!-- GET mode request -->
<a href="login">go2</a>  <!-- GET mode request -->
<a href="home">go3</a>  <!-- GET mode request -->
```

h) if no request path is specified in @RequestMapping the default will be "/" and default request mode/method GET

```
@Controller
public class MyController{

 @RequestMapping        //here  PATH is "/"  req mode/method  is "GET"
 public void showResult(){
   ..
 }
}
```

"/" indicates that the handler method takes intial request that comes to home page/welcome page with out any reqeust PATH .. (i.e hander method becomes default handler method)

equal to

```
@RequestMapping (value="/")
public void showResult(){
  ..
}
```

) At max how many handler methods of a controller class can have same request paths with different request modes/methods ?
(spring MVC setup)

Ans) As of now (2) 1 with GET mode , 1 with Post mode

) can we take two handler methods of a controller class with default request path ("/")?

Ans) yes one with GET mode and another one with POST mode. (same mode can no be taken)

```
@RequestMapping   //  "/"+ GET mode
public  String  showHomePage1() {
      System.out.println("TestController.showHomePage1()-GET");
      return "home_page";
}

      @RequestMapping(method=RequestMethod.POST)     //  "/" + POST mode
      public  String  showHomePage2() {
            System.out.println("TestController.showHomePage2()=POST");
            return "home_page";
      }
```

htm/jsp/ui code
==========

```
<form action="./" method="POST">
   <input type="submit" value="submit">  <!-- POST mode request -->
</form>
<br>
<a href="./">go</a>   <!-- GET mode request -->
```

"." rpresents current web applicaiton..

) can we cfg one handler method with different modes/methods request ?

```
@RequestMapping(value="/first", method= {RequestMethod.GET,
                                         RequestMethod.POST
                                          })
public  String  showHomePage1() {
      System.out.println("TestController.showHomePage1()");
      return "home_page";
}
```

html/jsp/ui code
==========

```
<form action="first" method="POST">
   <input type="submit" value="submit">  <!-- POST mode request -->
</form>
<br>
<a href="first">go</a>   <!-- GET mode request -->
```

I) if two handler methods of two Controller classes are having same request with diff functionalaties then we can request them ?
(Very important)

PATH and mode

how

Problem code or invalid code

```
@Controller
pubic class StudentController {

 @RequestMapping("/register")
 publicString saveData(){
   ....
  }
}
```

```
@Controller
public class EmployeeController{

@RequestMapping("/register")
 public  String  saveEmp(){
   ...
  }
```

Valid code or SolutionCode

(Global path)

```
@Controller
@RequestMapping("/student")
pubic class StudentController {

 @RequestMapping("/register")
 publicString saveData(){
   ....
  }
}
```

global path

```
@Controller
 @RequestMapping("/employee")
public class EmployeeController{

@RequestMapping("/register")
 public  String  saveEmp(){
   ...
  }
```

/student/register --> goes to saveData() of StudentController

/employee/register --> goes to saveEmp() of EmployeeController

jsp/html/ui code
==========

```
<a href="student/register">registr student</a>
<a href="employee/register">registr employee</a>
```

) How can we chain one handler method with another method ?

m) How can we chain one handler method with another method ?
(or)
How can we redirect one handler method request antother hander method?

```
@Controller
public class CRUDOperationController{

 @RequestMpping("/remove")
  public   String  deleteStudent(){
       .... . //some code  to delete record
      .....
    return "redirect:display";
  }

  @RequestMpping("/display")
    public   String   viewStudents(){
         .... . //some code to all recores
        .....
      return "report_page";
    }

@RequestMpping("/modify")
  public   String updateStudent(){
       .... . //some code to update reccord
      .....
    return "redirect:display";
  }

@RequestMpping("/registr")
  public   String insertStudent(){
       .... . //some code to update reccord
      .....
    return "redirect:display";
  }

                }
```

HEre Souce and destination handler methods can be there either in the same controller or in different controller class..

redirect:<uri> internally uses response.sendRedirect(-) method i.e request goes to next hanlder method of same or different controlelr class after having one network round trop with browser..

---

n) new Annotations are give from spring 4.x as alternate @RequestMapping annotaion

=>@GetMapping(value="/first") --> /first+GET mode equals to @RequestMapping(value="/first" method=RequestMethod.GET)
=>@PostMapping(value="/first") --> /first+POST mode equals to @RequestMapping(value="/first" method=RequestMethod.POST)
other @PutMapping ,@DeleteMapping and etc.. are there but they related to Restfull webServices

note:: @xxxMapping annotation given spring 4.x are method level annotions.. i.e they can not be applied at controller class level for giving global path..

```
@GetMapping(value="/welcome")
public  String showHome() {
      return "home";
}
```

note:: we can not apply multiple spring 4.x @XxxMapping annotations on same handler method..then only first one will be taken.

```
@PostMapping("/first")                    takes   /first  with POST mode request  and gives  error for
  @GetMapping("/first")                             /first with GET  mode request..
  public  String  showHomePage1() {
        System.out.println("TestController.showHomePage1()");
        return  "home_page";
  }
```

```
<form action="first" method="POST">   (success)
   <input type="submit" value="submit"> <!-- POST mode request -->
</form>
<br>
<a href="first">go</a>   <!-- GET mode request -->
          (error 405)
```

---

if u want to inject ServletContainer created objects to Controller class like ServletConfig ,ServletContext, request,response and etc..
we can go @Autowired/@Inject/@Resource or we can take as paarameters of handler method

```
@Controller
public class TestController {
      @Autowired
       private ServletContext sc;   // 1 per web application
       @Autowired
      private  ServletConfig cg;  //1 per Servlet DS

      //@RequestMapping(value="/first", method= RequestMethod.GET)
      @GetMapping("/first")
      public  String  showHomePage1(HttpServletRequest req , HttpServletResponse res) {
            System.out.println("TestController.showHomePage1()");
             System.out.println(sc.getClass());
             System.out.println(cg.getClass());
             System.out.println(req.getClass());
             System.out.println(res.getClass());
            return  "home_page";
```

Data Rendering to View (Passing model Data from Controller Class to View comps throguh DS)
==========

a) Using ModelAndView object (old style and not good) (request scope)
b) Using Map Object (request scope) — Both will use Map Colelction intenally So bad becoz of their load factory 0.75
c) Using ModelMap object (request scope)
d) Using Model Object (request Scope) — (Best) (Good becoz of less memory cosumption)
e) Using request object (request scope) — (bad makes to use Servlet api)
f) Using Session object (Session scope)
g) Using SErvletContext obj (application scope) — (Bad becoz of their scope usage of servlet api)

allocates memory future elements

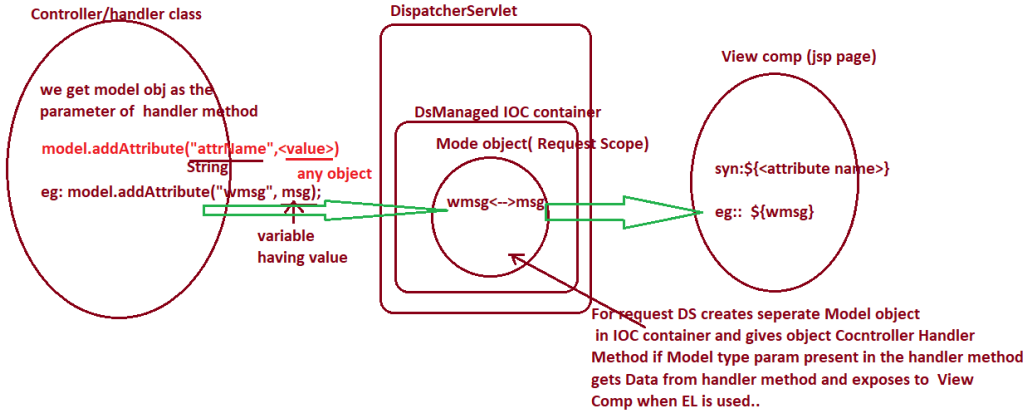
=>Since request to request inputs are different so generated outputs will also be different.. So Data rendering is always good through request scope

=>Jsp pages can read and use that model data either using EL or JSTL +EL

HashMap object initial capacity :: 16 (elements)
Load factor : 0.75 i.e 16 * 0.75=12 (Threshold number)

Once 12 element is added the Hash Map capacity becomes double 16 +16 =32

Using Model obejct
==========



Example
==========

```
@Controller
public class DataRenderingController {
	@GetMapping("/welcome")
	public String	processData(Model model) {
		System.out.println("DataRenderingController.processData()");
		System.out.println(model.getClass());
		model.addAttribute("msg","welcome");
		return "display";
	}
```

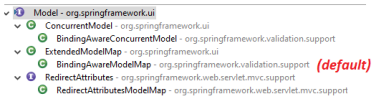
dipslay.jsp
==========

```
<%@ page isELIgnored="false"%>

<h1 style="color:blue;text-align:center"> Result page ---> result:: ${msg } </h1>
```

}

The Spring MVC takes BindingAwareModelMap as the default Impl class of Model(I) and this having capability of allocation Memory in only for required elements.. without bringing load factory into picture, so we can "Model object" based data rendering is the best pratice .



Working arrays ,collections as the model data

==========

In Handler class

==========

```
@GetMapping("/welcome")
    public String    processData(Model model) {
        String names[]=new String[] {"raja","rani","suresh","mahesh"};
        List<String> fruitsList=List.of("apple","banana","grapes","cherries");  //java 9 .. immutable List
        Set<Long> phonesSet=Set.of(999999L,888888L,77777L);  //java 9 .. immutable List
        Map<String,Integer> ageMap=Map.of("raja",30,"rani",31,"suresh",40);
        //add them as model attributes
        model.addAttribute("namesInfo",names);
        model.addAttribute("fruitsInfo",fruitsList);
        model.addAttribute("phonesInfo",phonesSet);
        model.addAttribute("agesInfo",ageMap);

        return "display";
    }
```

display.jsp

==========

```
<%@ page isELIgnored="false"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/core"  prefix="c"%>
<b>names  are </b> <br>
  <c:forEach var="name" items="${namesInfo}">
       ${name} <br>
  </c:forEach>
  <hr>
 <b>fruits are </b> <br>
 <c:forEach var="fruit" items="${fruitsInfo}">
       ${fruit} <br>
 </c:forEach>
 <hr>
 <b>phones are </b> <br>
 <c:forEach var="ph" items="${phonesInfo}">
       ${ph} <br>
 </c:forEach>
 <hr>
 <b>ages </b> <br>
 <c:forEach var="age" items="${agesInfo}">
       ${age.key} ---> ${age.value}  <br>
 </c:forEach>
```

Collection with java Bean object like ListDTO or listBO /ListEntities

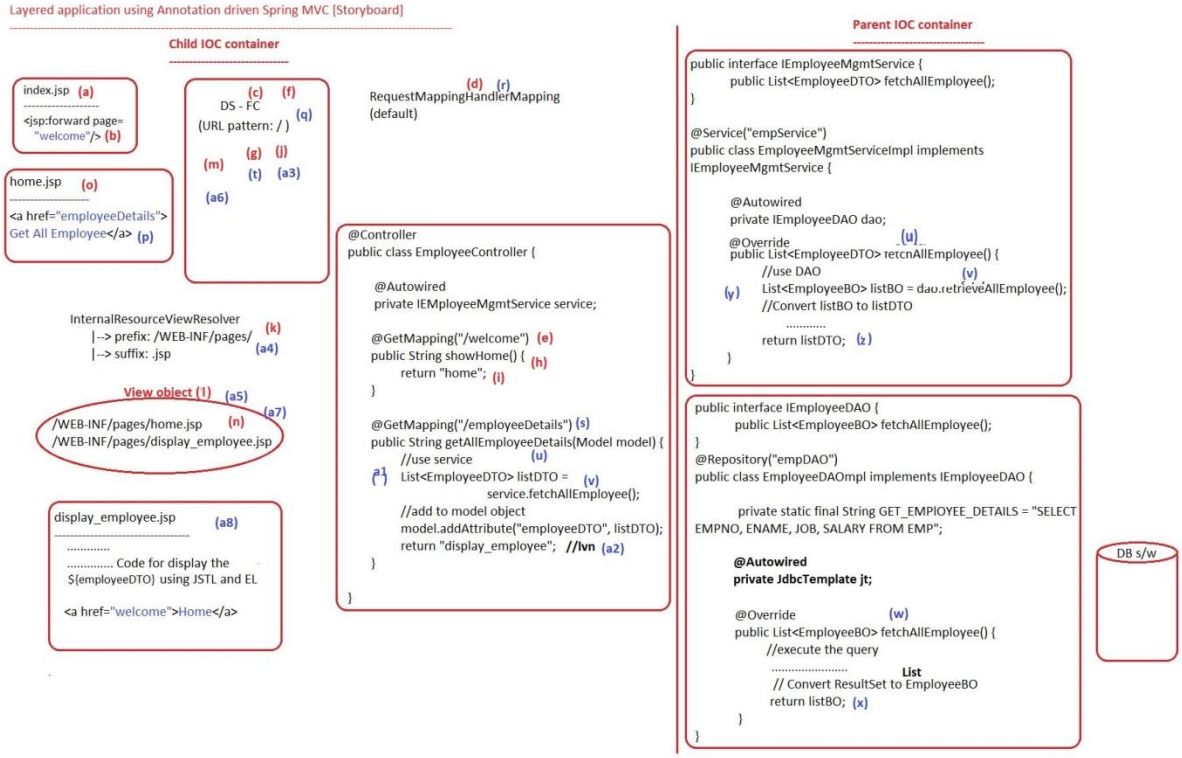
```
@GetMapping("/welcome")
 public String    processData(Model model) {
       //preare ListDTO
       List<StudentDTO> listDTO=List.of(new StudentDTO(101,"raja","hyd"),
                                        new StudentDTO(102,"rajesh","vizag"),
                                   new StudentDTO(103,"suresh","delhi")
                                        );
       //add to mode attribute
       model.addAttribute("studList",listDTO);
 return "display";
 }
```

display.jsp

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


<c:choose>
  <c:when test="${studList ne null && !empty studList }">
    <table border="1" align="center">
       <tr> <th>sno</th><th>sname</th><th>sadd </th></tr>
       <c:forEach var="dto"  items="${studList}">
         <tr>
           <td>${dto.sno} </td>
           <td>${dto.sname} </td>
           <td>${dto.sadd} </td>
         </tr>
       </c:forEach>
    </table>
  </c:when>
</c:choose>
```

Layered application using Annotation driven Spring MVC [Storyboard]



Why can not we take single java bean class as Model class for layer to carry Data?
why we need seperate DTO ,BO classes?

Ans1) BO/Entity class/Domain class objects should have only persistable or persistent data.. Sometime we need extra properties to hold like serialNo, grossSalary,netSalary and etc... extra data , So take seperate DTO class
Ans2) Sometime we need to roun up or roundown values or we need correct values before giving to DB or after collectiong from DB ... This needs seperate DTO
Ans3) Spring Data, Spring ORM , HB and etc.. ORM env.. have got ability to genrate DB table dynamically based on DB class/Entity class proeprtes , keeping extra properties in BO class itselef may create addtional cols in DB table..

```
<!-- Get SErverManaged JDBC DataSoruce obj from Jndi registry of Underlying SErver -->
 <bean id="jofb" class="org.springframework.jndi.JndiObjectFactoryBean">
    <property name="indiName" value="java:/comp/env/DsJndi"/>
 </bean>
```

JndiObjectFactoryBean is pre-defined ServiceLocator given by Spring API having capability to get Datasource object from underying Server Managed Jndi regitry based on the given Jndi name (java:/comp/env/DsJndi) and makes that recieved DataSource obj as the Spring Bean. in IOC cintainer... (Resultant object)

note:: To create SErver Managed jdbc con pool for oracle in Tomcat server of Eclipse IDE.. add the the following <Resource > tag under <Context> of context.xml file..

```
  <Resource auth="Container" driverClassName="oracle.jdbc.driver.OracleDriver" maxIdle="10"
      maxTotal="20" maxWaitMillis="3000" name="DsJndi" password="manager"
      type="javax.sql.DataSource" url="jdbc:oracle:thin:@localhost:1521:xe" username="system"/>
```

note:: Spring JDBC automates entire persitence logic.. but we can customize that peristence logic in the place we want by taking the support of Callaback interfaces... Callaback interfaces related callback methods exposes the JDBC objects as params.. we take those params /jdbc objs and we customize the logics as we want.
eg1:: RowMapper ----> To customize single record manipulation
ResultsetExtractor ---> To customize multiple records manipulation
and etc..

note:: the callback methods of callback interfaces will be called automatically by JdbcTemplate methods by passing the interanaly jdbc objs .. as agruments.. So that we can use those objs for customizing persistence logic by writing jdbc code.. like convrt RS single record BO object and connverting RS multiple records ListBO objs..

```
List<EmployeeBO> listBO=jt.query(GET_ALL_EMPLOYEES,new EmployeeRowMapper());
```

collects DS---> gets Con object from pool ---> creates PS having given Query --> executes Query gets RS ---> takes given EmployeeRowmapper obj-> calls extractData(RS) method on that --> gets ListBO having EmployeeBO obj ->returns same listBO back as return value.

=>To apply bootstrapping (Bunch of css,js libraries)
step1) write following <link> tag by collecting from www.getBootstrap.com

```
<head>
  <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
</head>
```

step2) know the style classes and apply them using "class" attribute different tags..

Code for taking printcout of the webpage
==========

```
<h1><a class=" text-danger text-center" href="JavaScript:doPrint()">Print</a></h1>

</div>
<script language="JavaScript">
    function doPrint(){
        frames.focus();
        frames.print();
    }
</script>
```

Passing Data from view comps to Controller (Data Binding)

(a) Using forms
- |---> Traditional html forms
- |--->Spring MVC jsp froms (spring mvc supplied jsp tags)
- |---> UI Forms (jquery/angularJs / Angular Forms/ReactJs forms)

[This is useful to carry to enduser supplied dynamic data]

(b) Using Request params appended to the Query String
(hypler link having href url with query string)

href="<path>?<param1>=<val1>& <param2>=<val2>"

[This useful carry application supplied static/dynamic data]

edit delete

This hyperlinks will carry appliation suplied pk value as addtional Data..

(c) Using JavaScript (js based ajax/ jquery/angular/....)
=> To make any comp/ any event sending request and data to web application

eg1:: when page is loaded onLoad event should the send the request to get countries to sleect box
eg2:: when country selected in select box request go with data and should should bring states into another select box

(a) Using forms

=>In spring MVC working with forms will have two phases
(a) Initial Phase (talks about displaying from page throguh handler method)
- |--->Generally its "GET" mode request
- |---> We can design form either using tradional html tags or spring mvc tags(best)
- |---> In the handler we create and keep model(new)/command (old) class obj with empty data /initial Data..
- |---> This model/command class is java bean having property names matching with form comp names (text box, radio buttons and other names)
- |-->if form page form is designed using spring MVC supplied jsp pages them Model class obj initial value can be displyed as the initial values of form comps (like text boxes) .. with tradtional html tags this is not possible

Method in handler /controler class

```
@controller
public class StudentController{
@GetMapping("/register") (d?)
public String showForm(Map<String,Object> map){
    Student st=new Student(); //model/Command class obj
(g) map.put("stf rm",st);
    return "student_form"; (h)
}
```

lvn of the form page..

command class/Model class

```
public class Student{
  private int sno;
  private String sname;
  private String sadd="hyd"
  //getters && setters
    ....
    ....
    ....
}
```

tradtional html tags based form page

```
<form action="register"  method="POST">
    student number ::  <input type="text" name="sno"/>
     student name ::  <input type="text" name="sname"/>
      student address ::  <input type="text" name="sadd"/>
    <input type="submit"   value="register student"/>
</form>
```

not recomaded to use these tag becoz we we not put model object in text boxes of form page as initial value..

[supports 1 way binding i.e only form submission to Model object)

(or)

spring MVC supplied jsp form tags based form page

student_form.jsp (WEB-INF/pages) (m)

```
<%@taglib uri=" ....." prefix="frm"%>                    (n)
  <frm:form  action="register"  method="POST"  modelAttribute="stfrm">
     sno :: <frm:input type="text"  path="sno"/>
     sname :: <frm: input type="text" path="sname"/>
     sadd    :: <frm: input type="text" path="sadd"/>
      <input type="submit"  vlaue="register">
  </frm:from>                                  (p)
```

(collects data from model class obj properties)

These spring supplied tags supports two way binding i.e while displaying form page model obj inital values will be bound to text boxes as intial vlaues and when form submitted .. them form data will be stored into model object.

b) Post back request (submitting form page and processing requst)

=> Here form data will be read and will be stored in Model object by doing neccessary convertions..

=> Can create or locate form Model object to to perform form binding /request wrapping(nothing but writing form data to Model object)

=> we need handler method controller as shown below..

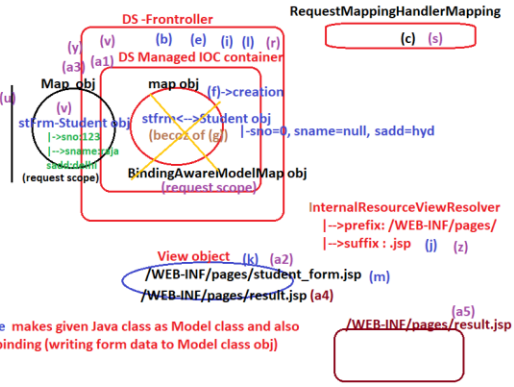
```
@Controller
public class StudentController{
  @PostMapping("/register")   (t?)
  public  String    processForm(@ModelAttribute("stFrm")Student stud,
                                Map<String,Object> map){
                        (w)
    ....
    ... // porces request directly or using service
    ...
     .. // add results as model attributes
    return "result";  (x)
  }
```

@ModelAtribute makes given Java class as Model class and also performs form binding (writing form data to Model class obj)

request url::

(a) (GET)

http://localhost:3030/FormsApp/register.htm



<a >go it is not error request goes to previous url (using which this page launched)

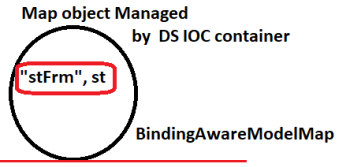
```
<frm:form method="Post" modelAttribute="stFrm">
    ...
    ...  //when form is submitted the request goes to previous url (the url using which the form page is launched
}                                                                  like /register)
```

What is the difference Trandition html form and spring mvc jsp form tags?

tradtional html forms	spring mvc jsp forms
a) given by w3c	a) given by spring framework
b) supports one way binding (form to java object/req obj)	b) Suport two way binding (java object to form and form to java object)
c) Initial values of form comps can not be collected from java obj	c) can be collected..
d) comes to brwser as it is (as html tags)	d) internally jsp tags will be converted into html tags with intial collected from java obj (model obj)

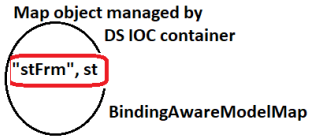
Different ways of writing handler method for initial phase request (GET mode request)

```
@GetMapping("/register")
public  String  showForm(Map<String,Object> map){
   Student st=new Student(); //model/command class obj
    st.setSadd("hyd");
   map.put("stFrm",st);
   return "student_form";
 }
```



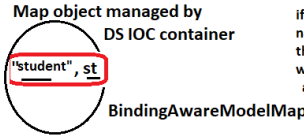
(or)

```
@GetMapping("/register")
public  String  showForm(@ModelAttribute("stFrm") Student st){
    st.setAdd("hyd");
    return "student_form";
 }
```



(or)

```
@GetMapping("/register")
public  String  showForm(@ModelAttribute      Student st ){
    st.setAdd("hyd");
    return "student_form";
 }
```

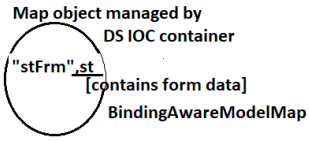


if we do not provide model attribute name in @ModelAtribute(-) the model/command class name will be taken as default model attribute name (but first letter becomes lowercase letter)

Different ways of creating hanlderMethod for POST back request

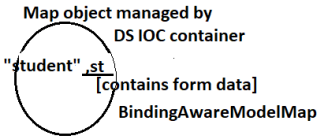
=>use this for (a),(b) versions of initial phase request

```
    @PostMapping("/register")
a)  public   Stirng  processForm(Map<String,Object> map,
                                 @ModelAttribute("stFrm") Student st){
       .....
       ....  //logic to process request  and to results to  map as model attribute
       ....
      return "result";
   }
```



=>use this for (c) version of initial phase request

```
    @PostMapping("/register")
b)  public   Stirng  processForm(Map<String,Object> map,
                                 @ModelAttribute   Student st){
       .....
       ....  //logic to process request  and to results to  map as model attribute
       ....
      return "result";
   }
```



The default Scope of Map object and Model class obj/Form class obj/Command class obj is request scope..

Form DataBinding(writing form data to Model class obj)
performs operations in the following order (POST mode)

a) Searches handerMethod that ready to handle Post back request based request Mode(POST) and requestPath(like "/register") [Basically seraches for handlerMethod with @PostMapping("/register")]

b) Based @ModelAttribute annotation basaed parameter it create Model class obj either default name (method parameter name(st)) or specified name (like stFrm) as model attribute name also keep it in Map object

c) reads form data using req.getParameter(-) methods .. performs necessary convertions using convertor and PropertyEditors and writes from data to Model class object (that is created above)

(d) calls Postback request handler method having the Map object , model class obj as the arguments..

---

refer :: MVCProjAnno11-DataBinding

note1: In tradtion html tags based from page the default request mode is "GET" where in spring mvc supplied jsp tags based forms the default request mode is "POST" .

note:: Spring MVC is supplying two Jsp Taglibraries

a) Generic Taglibrary --->talib uri :: *http://www.springframework.org/tags*
recomanded prefix :: spring

b) Form tag library ----> taglib uri :: *http://www.springframework.org/tags/form*
prefix :: form

Q)What is form backing obj?
ans) Model /command class obj is aslo called as formbaking object.. becoz it is having data that is required to populate to form comps..

Q)what if model class properties not matching with form comp names ?
ans) only matched comp values will be bound to Model class obj

Data Binding Using Request Parameetrs (Sending data from UI to Controller through DS)

=>sending data as req params in query string in the form key =value pairs.

?key1=value1&key2=val2&key3=val3

=>keys are req param names
=>values are req param values | (Both are case-sensive)

eg:: http://localhost:3030/FirstMVCApp/register?sno=101&sname=raja

To read these values in servlet programming

```
int no=Integer.parseInt(req.getParameter("sno")); //gives 101
String name=req.getParameter("sname"); //gives raja
```

=>GET-FORM data also comes to Server as req params and stored into request object automatically
=> GET -explicitly query string having req params will be stored into request obj automatically
=> POST-form data comes to server as request body/payload but will be stored into request obj automatically.

To read these values spring mvc controler Use @RequestParam

=>@RequestParam reads the given req param value and puts hanler method parameter

syntax1:: @RequestParam("key") <Data type> <var>
syntax1:: @RequestParam <Data type> <var> (key and var name must match)

```
eg:: @GetMapping("/links")
     public String processForm(Map<String,Object> map,
                               @RequestParam("sno") int no,
                               @RequestParam("sname") String name)
```

```
@GetMapping("/links")
eg:: public String processForm(Map<String,Object> map,
                               @RequestParam int sno,
                               @RequestParam String sname)
```

http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sname=karan

---

=>if we give request params with wrong case then we get 400 error (bad request)

```
@GetMapping("/links")
eg:: public String processForm(Map<String,Object> map,
                               @RequestParam int Sno,
                               @RequestParam String Sname)
```

http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sname=karan

=>If we give request params with wrong case then we get 400 error (bad request)

```
@GetMapping("/links")
eg:: public String processForm(Map<String,Object> map,
                               @RequestParam int Sno,
                               @RequestParam String Sname)
```

http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sname=karan

---

=>Addtional request params in the query String will be ignored... with out any error/exception

```
@GetMapping("/links")
eg:: public String processForm(Map<String,Object> map,
                               @RequestParam int Sno,
                               @RequestParam String Sname)
```

http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sname=karan &sadd=hyd ...

(The addtional request param)

---

if one request param is having multipe values key1=val1 & key1=val2&key1=val3
then we read by using simple/wrapper type method param or array/List/Set type method param

eg:: http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sname=karan&sadd=hyd&sadd=vizag

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam int sno,
                           @RequestParam String sname,
                           @RequestParam Set/List<String> sadd) {
    System.out.println("request params:::"+sno+"  "+sname+" "+sadd.get(0)+"...  "+sadd.get(1));
    return "show_params";
}
```

(or)

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam int sno,
                           @RequestParam String sname,
                           @RequestParam String sadd) {
    System.out.println("request params:::"+sno+"  "+sname+" "+sadd;
    return "show_params";
}
```

holds multiple values as the comma seperate list of String values like vizag,hyd

---

=>While creating request prams in the query string of request url and while reading them througn handler methods the order of params do not matter...

http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sname=karan&sadd=hyd&sadd=vizag
(or)
http://localhost:3030/MVCProj11-DataBinding-Forms/links?sadd=hyd&sadd=vizag&sno=1001&sname=karan

both are sname

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam List<String> sadd,
                           @RequestParam int sno,
                           @RequestParam String sname) {
    System.out.println("request params:::"+sno+"  "+sname+" "+sadd.get(0)+"...  "+sadd.get(1));
    return "show_params";
}
```

(or)

Both are same

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam int sno,
                           @RequestParam String sname
                           @RequestParam List<String> sadd,
                           ) {
    System.out.println("request params:::"+sno+"  "+sname+" "+sadd.get(0)+"...  "+sadd.get(1));
    return "show_params";
}
```

=>If @RequestParam is specified in the handler method and that request param is not passed through query String of request url then we 400 (bad request error) .. becoz the "required" param of @RequestParam is having "true" as the default value indicating request param must be supplied .. otherwise exception will come...
To disable that compulsion of supplying req param we can take required=fase in @RequestParam. This time it will tak null as default value when req param is not given in the url.. To assign default value any request param then we take "defaultValue" param of @RequestParam

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam List<String> sadd,
                           @RequestParam int sno,
                           @RequestParam(required = false,defaultValue = "anonymous") String sname) {
```

URL :: http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sadd=hyd&sadd=vizag

⇩

since s name is not given and required=false,defaultValue is taken So "anonymous" will be stored into sname method param

URL :: http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=1001&sadd=hyd&sadd=vizag&sname=raja

⇩

The sname method param holds "raja"

---

if we pass mismathed data along with req param value then happens?

http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=xyz

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam int sno){
  ...
}
```

raises the exception... 400 bad reeqeust
(java exception:: [org.springframework.web.method.annotation.MethodArgumentTypeMismatchException: Failed to convert value of type 'java.lang.String' to required type 'int'; nested exception is java.lang.NumberFormatException: For input string: "xyz"]

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam(required=false) int sno){
  ...
}
```

raises the exception... 400 bad reeqeust
(java exception:: [org.springframework.web.method.annotation.MethodArgumentTypeMismatchException: Failed to convert value of type 'java.lang.String' to required type 'int'; nested exception is java.lang.NumberFormatException: For input string: "xyz"]

What happens for the following request url and handler method?

http://localhost:3030/MVCProj11-DataBinding-Forms/links?sno=

here default of sno is "" and it can not be converted into numeric value.

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam(required=false) int sno){
  ...
}
```

raises the exception... 400 bad reeqeust
(java exception:: [org.springframework.web.method.annotation.MethodArgumentTypeMismatchException: Failed to convert value of type 'java.lang.String' to required type 'int'; nested exception is java.lang.NumberFormatException: For input string: "xyz"]

To solve the above exception

http://localhost:3030/MVCProj11-DataBinding-Forms/links

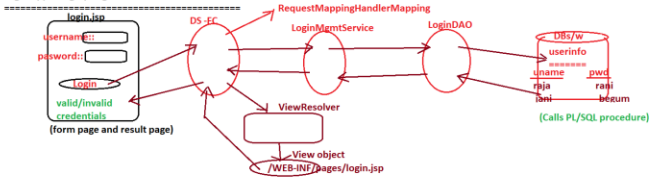
⇩

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam(required=false) Integer sno){
  ...
}
```

(or)

```
@GetMapping("/links")
public String getLinksData(Map<String,Object>map,
                           @RequestParam(required=false) String sno)
  ...
}
```

Login App using Spring MVC Annotation driven env..



note:: In spring MVC form launching (initialphase request-GET) and form submission(post back request-POST) will be handled by Same controller with two different handler methods having same rquest path/uri with two different modes of request (GET,POST).

The reusable logics that are required in multiple modules of Project will be placed in Db s/w as stored procedures /functions...

```
CREATE OR REPLACE PROCEDURE P_AUTHENTICATION
(
  USERNAME IN VARCHAR2
, PASSWORD IN VARCHAR2
, RESULT OUT VARCHAR2
) AS
 CNT NUMBER(4);
BEGIN
  SELECT COUNT(*) INTO CNT FROM USERINFO WHERE UNAME=USERNAME AND PWD=PASSWORD;

  IF(CNT<>0)THEN
    RESULT:='VALID CREDENTIALS';
  ELSE
    RESULT:='INVALID CREDENTIALS';
    END IF;
END;
```

procedure to develop SpringMVC web application using Eclipse +maven

step1) create maven project by selecting "maven-archetype-webapp" as the archetype (Project templates)

File menu--> maven project --> select "maven-archetype-webapp" as archetype --> quickstart -->for standalone Apps
provide details webapp -->for web application
groupId :: nit
artifact Id :: MVCProj12-LoginApp-layedApp
version :: contine default value
package :: not required

-->next->finish

step2) Do following things on the Project

=> Open Project propeerties change java version to 13/14/15, and [ ] Dynamic web module 4.0 in Project Facets section

=>Open Order and export tab in Project builpath and select unselected checkboxes..

step3) Add the following dependencies(jar files) in pom.xml file

jstl1.2, spring mvc 5.2.9 , servlet-api 4.0.1 (collect from mvnrepository.com (from maven tab))
spring jdbc 5.2.9

step4) create packages for soucecode in src/main/java folder and place html,jsp, spring cfg file in webcontent and its sub folders accordingly

step5) run the Application..

=>SimpleJdbcCall internally uses JdbcTemplate and simplifies PL/SQL procedure call by Generating query based on given ProcedureName, Map of In param names and values .. and also calls Pl/SQL procedure and return Map object having out param names and values..

sjc (SimpleJDbcCall)
|-->Ds (Dependent obj)
|-->setProcedureName(-) :: give PL/SQP procedure name
|-->Map obj having In Param names and values
|-->execute(map obj)

Map obj <-- returns
(Gives out param names and values)

```
<jee:jndi-lookup id="ds" jndi-name="java:/comp/env/DsJndi"/>
```

Internally uses "JndiObjectFactoryBean" to get DS object from underlying server jndi registry based given jndi name and also makes the recieved DS object as spring Bean with the bean id "ds".

```
<bean id="dsTxMgmr" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
   <property name="dataSource" ref="ds"/>
</bean>
```

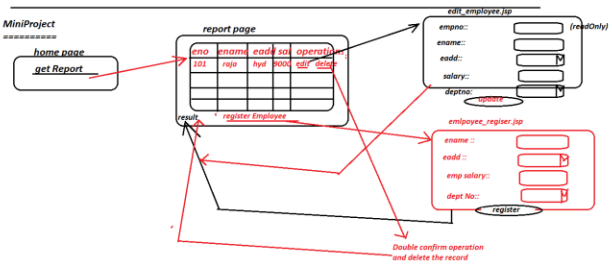
Takes care of begining Tx, commiting Tx and and rollback Tx automatically...
=>if every thing is smooth in the @Transactional method then it commits Tx ..If something gone in the method (if exception is raised) rollbacks Tx.

note:: if b.methods are dealing bunch of related non-select persistence operations then we definetly need TxMgmt support to apply do everything or nothing principle on the code.. (like transfer money withdraw, deposite operations)

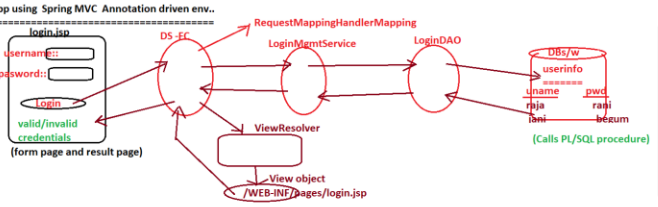
note:: if b.methods are dealing select persistence operations then it recomanded to enable TxMgmt support on the b.method to make DB s/w only giving "committed data" to Application.

In context.xml of Tomcat server (Server Managed con pool in tomcat server)

```
<Resource auth="Container" driverClassName="oracle.jdbc.driver.OracleDriver" maxIdle="10"
   maxTotal="20" maxWaitMillis="3000" name="DsJndi" password="manager"
   type="javax.sql.DataSource" url="jdbc:oracle:thin:@localhost:1521:xe" username="system"/>
```



Login App using Spring MVC Annotation driven env..
==========



note:: In spring MVC form launching (initialphase request-GET) and form submission(post back request-POST) will be handled by Same controller with two different handler methods having same rquest path/uri with two different modes of request (GET,POST).

The reusable logics that are required in multiple modules of Project will be placed in Db s/w as stored procedures /functions...

```
CREATE OR REPLACE PROCEDURE P_AUTHENTICATION
(
 USERNAME IN VARCHAR2
, PASSWORD IN VARCHAR2
, RESULT OUT VARCHAR2
) AS
 CNT NUMBER(4);
BEGIN
  SELECT COUNT(*) INTO CNT FROM USERINFO WHERE UNAME=USERNAME AND PWD=PASSWORD;

  IF(CNT<>0)THEN
    RESULT:='VALID CREDENTIALS';
  ELSE
    RESULT:='INVALID CREDENTIALS';
  END IF;
END;
```

procedure to develop SpringMVC web application using Eclipse +maven
==========

step1) create maven project by selecting "maven-archetype-webapp" as the archetype (Project templates)

File menu---> maven project ---> select "maven-archetype-webapp" as archetype --> provide details
groupId :: nit
artifact Id :: MVCProj12-LoginApp-layedApp
version :: contine default value
package :: not required
-->next->finish

quickstart -->for standalone Apps
webapp -->for web application

step2) Do following things on the Project

=> Open Project propeerties change java version to 13/14/15, and [ ] Dynamic web module 4.0 in Project Facets section

=>Open Order and export tab in Project builpath and select unselected checkboxes..

step3) Add the following dependencies(jar files) in pom.xml file

jstl1.2, spring mvc 5.2.9 , servlet-api 4.0.1 (collect from mvnrepository.com (from maven tab))
spring jdbc 5.2.9

step4) create packages for soucecode in src/main/java folder and place html,jsp, spring cfg file in webcontent and its sub folders accordingly

step5) run the Application..

=>SimpleJdbcCall internally uses JdbcTemplate and simplifies PL/SQL procedure call by Generating query based on given ProcedureName, Map of In param names and values .. and also calls PI/SQL procedure and return Map object having out param names and values..

sjc (SimpleJDbcCall)
|--->Ds (Dependent obj)
|--->setProcedureName(-) :: give PL/SQP procedure name
|--->Map obj having In Param names and values
|--->execute(map obj)

Map obj <-- returns
(Gives out param names and values)

```
<jee:jndi-lookup id="ds" jndi-name="java:/comp/env/DsJndi"/>
```

Internally uses "JndiObjectFactoryBean" to get DS object from underlying server jndi registry based given jndi name and also makes the recieved DS object as spring Bean with the bean id "ds".

```
<bean id="dsTxMgmr" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
   <property name="dataSource" ref="ds"/>
</bean>
```

⇩

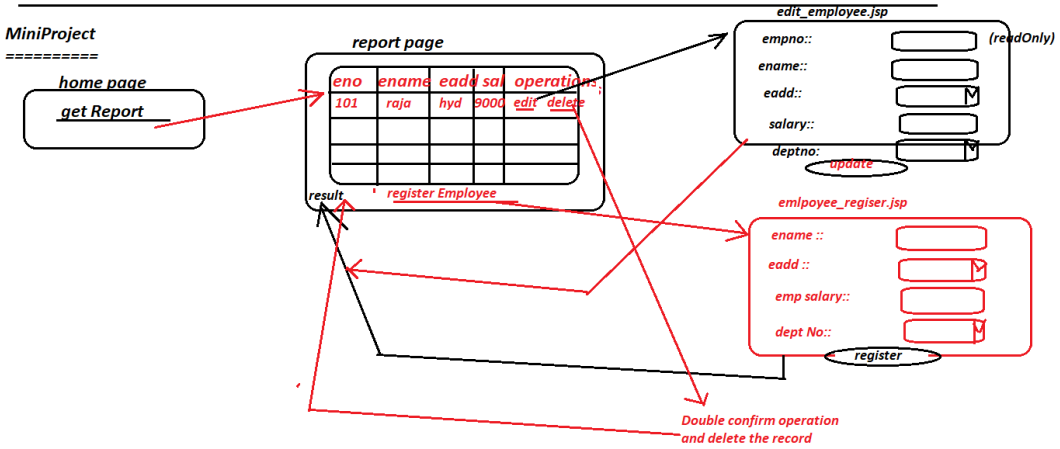
Takes care of begining Tx, commiting Tx and and rollback Tx automatically...
=>if every thing is smooth in the @Transactional method then it commits Tx ..if something gone in the method (if exception is raised) rollbacks Tx.

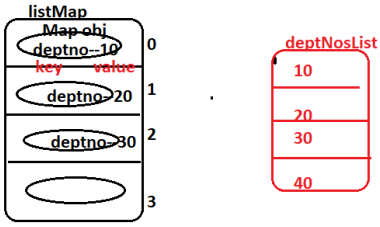
note:: if b.methods are dealing bunch of related non-select persistence operations then we definetly need TxMgmt support to apply do everything or nothing principle on the code.. (like transfer money withdraw, deposite operations)

note:: if b.methods are dealing select persistence operations then it recomanded to enable TxMgmt support on the b.method to make DB s/w only giving "committed data" to Application.

In context.xml of Tomcat server (Server Managed con pool in tomcat server)
==========

```
<Resource auth="Container" driverClassName="oracle.jdbc.driver.OracleDriver" maxIdle="10"
   maxTotal="20" maxWaitMillis="3000" name="DsJndi" password="manager"
   type="javax.sql.DataSource" url="jdbc:oracle:thin:@localhost:1521:xe" username="system"/>
```





=>While working radion buttons, checkboxes , select boxes, list boxes we can not get their bunch of of initial values as dynamic from the bound the model class properties becoz they are designed hold selected values , not all the dynamic intial value.. So we need construct dynamc initial vlaues from as reference data by @ModelAttribute method having List collction as the return type.. and we neeed specify that modele attribute name as "items" value for the above comps..

```
In Controller class
================
@ModelAttribute("deptsInfo")  //constructing reference data/initial for select box
      public List<Integer>  populateDeptNos(){
      //use service
      return service.fetchAllDeptNos();
}
```

In form page
==========

```
  <tr>
      <td>Employee deptno :: </td>
      <td>  <frm:select path="deptNo">
            <frm:options items="${deptsInfo}"/>
          </frm:select>

          </td>
      </tr>
```

@ModelAttribute is multipurpise annotation
a) To make java bean as model ro command class in
initial phase ,post back handler methods

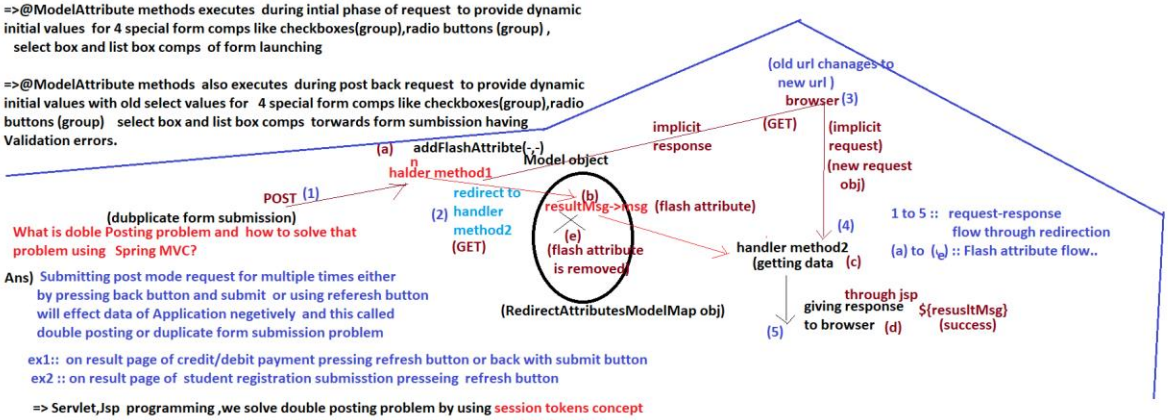
```
public   String showForm(@ModelAttribute Employee emp){
  ...
}
```

b) To provide addtional refernce data as dynamic intial values for
checkboxes (group), radio buttons (group) , select box and list box commps

```
@ModelAttribute("deptsInfo")  //constructing reference data/initial for select box
      public List<Integer>  populateDeptNos(){
      //use service
      return service.fetchAllDeptNos();
}
```

=>@ModelAttribute methods executes during intial phase of request to provide dynamic initial values for 4 special form comps like checkboxes(group),radio buttons (group) , select box and list box comps of form launching

=>@ModelAttribute methods also executes during post back request to provide dynamic initial values with old select values for 4 special form comps like checkboxes(group),radio buttons (group) select box and list box comps torwards form sumbission having Validation errors.



(dubplicate form submission)

What is doble Posting problem and how to solve that problem using Spring MVC?

Ans) Submitting post mode request for multiple times either by pressing back button and submit or using referesh button will effect data of Application negetively and this called double posting or duplicate form submission problem

ex1:: on result page of credit/debit payment pressing refresh button or back with submit button
ex2 :: on result page of student registration submisstion presseing refresh button

=> Servlet,Jsp programming ,we solve double posting problem by using session tokens concept
=> In spring MVC programming ,we solve double posting problem by using PRG Pattern.
(Post Redirect Get pattern)

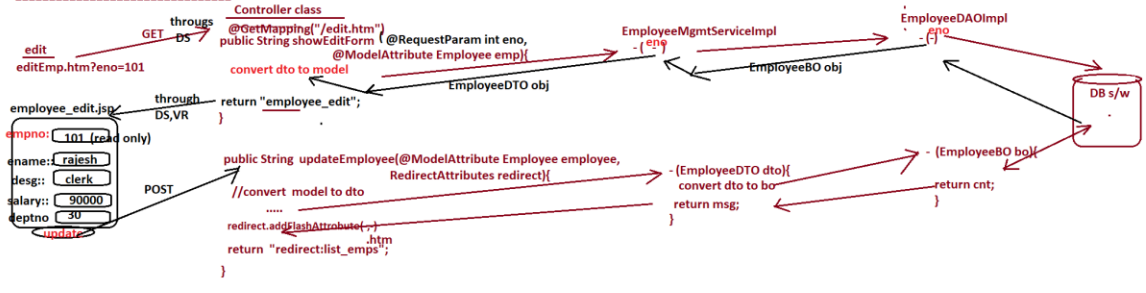
=>PRG Pattern say the request given to post back request handler method (POST) should be redirected GET request mode handler method ,In this process the redirectin activity chanages url in the browser address bar to GET mode request handler method request path.. So we press refresh button GET request executes which harmless.. (i.e POST request of form submit will not execute for refresh button)

note:: Flash attribute is the model attribute that allocates memory in RedirectAttributes object to make data across the source handler method and target handler method of redirection.. after that data will be deleted from RedirectAttributes object. (see the above diagram)

```
@PostMapping("/saveEmp.htm")  //for post back request
    public  String  saveEmployee(RedirectAttributes map, @ModelAttribute("empFrm") Employee emp){
        System.out.println("EmployeeController.saveEmployee()");
        EmployeeDTO dto=null;
        String result=null;
        List<EmployeeDTO> listDTO=null;
        //convert model to dto
        dto=new EmployeeDTO();
        BeanUtils.copyProperties(emp,dto);
        //use Service
        result=service.registerEmployee(dto);
        //keep in results  in flash attribute (special Map object)
         map.addFlashAttribute("resultMsg",result);
        //return lvn
        return "redirect:list_emps.htm";

    }
```

Edit/ Update operation In MiniProject



Form Validations in spring MVC Application

=>Verifying the pattern and format of the form data is called form validation

There are two types of form validations

- **client side form validations**
  - use js or other ui technologies to perform form validations
- **server side form validations**
  1. Using Programatic Approach
     - -->we develop seperate validator class having form valdation logic
     - ->allows to enable server side form validations only when client side form validations are not done
  2. Using Annotation Approach
     - -> we use validation api, hibernate validator api annotations here..

     [allowing to enable server side form validations only when client side form validations are not done is not possible]

=>The best pratice of form validation is write both client side and server side form validations but enable server side form validations only when client side form validations are done not (i.e if java script of browser is disabled)

steps to develop server side form validation logic (Programatic Approach)

step1) create properties file having form validation erorr messages and cfg file in dispatcher-servlet.xml

validation.properties

```
# validaion error messages for programatic approach
emp.ename.required=employee name is required
emp.ename.length=employee name must have min 5 chars and max 10 chars
emp.job.required=employee desg is required
emp.job.length=employee desg must have min 5 chars and max 9 chars
emp.sal.required=employee salary is required
emp.sal.range=employee salary range must be between 10000 to 100000
```

in dispatcher-servlet.xml *Fixed bean id*

```
<bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">
   <property name="basename" value="com/nt/commons/validation"/>
</bean>
```

step2) Develop Validator class by implementing Validator(I) and write logic in validate(-,-) method

errors obj (BindException obj)

ename	ename is reugired
...	....
...	....

```
@Component("empValidator")
public class EmployeeValidator implements Validator {

    @Override
    public boolean supports(Class<?> clazz) {
        return clazz.isAssignableFrom(Employee.class); //useful to check whether expected command/model is recieved
                                                       // to validator class or not
    }
```

```
@Override
public void validate(Object target, Errors errors) {
      Employee emp=null;
      //type casting on command class obj
      emp=(Employee)target;
      //write form validation logic
      if(emp.getEname()==null || emp.getEname().isEmpty()) {  //required rule
            errors.rejectValue("ename","emp.ename.required");  //errors obj hods property name, error message collected proepries file
      }
      else if(emp.getEname().length()<5 || emp.getEname().length()>10) {  //length
            errors.rejectValue("ename","emp.ename.length");
      }

      if(emp.getJob()==null || emp.getJob().isEmpty()) {  //required rule
            errors.rejectValue("job","emp.job.required");
      }
      else if(emp.getJob().length()<5 || emp.getJob().length()>10) {  //length
            errors.rejectValue("job","emp.job.length");
      }

      if(emp.getSal()==null ) {  //required rule
            errors.rejectValue("sal","emp.sal.required");
      }
      else if(emp.getSal()<10000 || emp.getSal()>100000) {  //length
            errors.rejectValue("sal","emp.sal.range");
      }

   }//method
}//class
```

step3) cfg Validator class in spring bean file using <context:componentScan> and also inject to controller class using @Autowired

in dispatcher-servlet.xml

```
<context:component-scan base-package="com.nt.controller,com.nt.validator"/>
```

In EmployeeController.java

```
@Autowired
private EmployeeValidator validator;
```

step4) ***Write following code in postback handler methods of both employee registraion and editing.***

In
```
@PostMapping("/saveEmp.htm")  //for post back request
public String saveEmployee(RedirectAttributes redirect,
                           @ModelAttribute("empFrm") Employee emp,
                           BindingResult errors)  method
```

(or)

In
```
@PostMapping("/editEmp.htm")
public String updateEmployee(@ModelAttribute Employee employee,
                             RedirectAttributes redirect,
                             BindingResult errors)  method
```

write this code before invoking the service class methods

```
//peform form validations
if(validator.supports(employee.getClass()))
      validator.validate(employee, errors);

//if form validation errors are there.. launch form page
if(errors.hasErrors())
      return "employee_register";
```

Errors(I)
⇧ extends
BindingResult(I)
⇧ implements
BindException(c)

step5) place <frm:errors> in form pages to display form validation error messages

```
<frm:errors cssClass="text-danger" path="ename"/>
<frm:errors cssClass="text-danger" path="job"/>
<frm:errors cssClass="text-danger" path="sal"/>
```

write in form pages aganist text boxes..

logical name or property name given in validator class

note:: DS creates empty "errors" obj (BindException obj) gives to validator class and gets errors object with validator error messages and sends to UI (jsp) by keeping in request scope, the <frm:errors> tag internally reads the error messages from "errors" object

Spring MVC Application can have 3 types of errors

1) Form validation errors
2) typemismatch errors
3) application errors
   logic

2)typemismatch errors

=>will come when form data can not be stored in model class property ...
place typeMismatch.<property>=<msg> in properties file... to get error message

```
#typemismatch error messages
typeMismatch.sal=salary must be numeric value..
```

3) application errors (This are b.logic errors)
   logic

In controller class post back method like saveEmployee(-,-,-) or editEmployee(-,-,-) method

```
//b.logic error or application logic errors
if(emp.getJob().equalsIgnoreCase("netaji") || emp.getJob().equalsIgnoreCase("actor"))
      errors.rejectValue("job", "blocked.jobs");
```

in validation.properties file

```
#application logic error messages
blocked.jobs=netaji and actor desg employees are not allowed to register
```

```
@PostMapping("/editEmp.htm")
 public String  <method>(RedirectAttributes redirect,
                         @ModelAttribute Employee employee,
                          BindingResult errors) {
   ....
   ...
 }
```

=> While taking handler method signature @ModelAttribute Model Classname and BindingResult errors parameters should taken back .. otherwise we get problem while dealing typeMistmatch errors..

Enabling Server side form validations only when client side form validations are not done

step1) add hidden box in form pages (employee_register.jsp, employee_edit.jsp)

```
<frm:hidden path="vflag"/>
```

step2) Take propeerty in model class to hold hidden box value..

In Employee.java (model class)

==========

```
private String  vflag="no";

public String getVflag() {
        return vflag;
    }
public void setVflag(String vflag) {
    this.vflag = vflag;
}
```

step3) write statement in java script indicating client side java script is executed

```
frm.vflag.value="yes";  // changing "no" to "yes" in JS indicating client side
                        //form validations are done
```

step4) Write the following in post handler methods saveEmp(-,-,-), updateEmployee(-,-,-) methods .. to perform server side validation based vflat value

```
if(emp.getVflag().equalsIgnoreCase("no")) {  //enable server side form validations only
                                             //when client form  validations are not done
    //peform form validations
    if(validator.supports(emp.getClass()))
        validator.validate(emp, errors);
}
```

---

note:: To disable JS in chrome :: menu : --- settings ---> search for java script ---> site settings ---> java script --> blocked

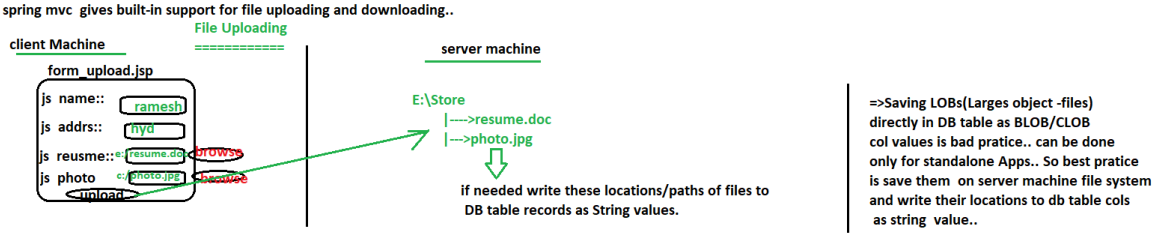
note:: To disable JS in FireFox :: type about:config in browser --> accept risk ---> search for java script --> javascript.enabled --->false

File uploading and downloading
==========

=> Sending files from client machine file system to Server file system is called file uploading and reversse is called filedownloading

usecases :: matrimony apps , job portal apps, social networking apps, profile mgmt apps and etc..

=> spring mvc gives built-in support for file uploading and downloading..



steps to perform file uploading
==========

step1) take file uploading comps in form page

```
<input type="file" name="resume">
<inpu type="file" name="photo"/>
```

step2) design form page having POST request mode/method (to carry unlimited amout of data) and also having "multipart/form-data" as the "enctype"

```
<frm:from action="...." method="POST" modelAtribute="..." enctype="mulipart/form-data">
  ....
  ....
      <input type="file" name="resume">
      <inpu type="file" name="photo"/>
</frm: form>
```

To send signal to server that form page sends diff types or MIME types of content

step3) Develop model class/command class having properties to hold uploaded files content (as InputStreams)
take the properties of type "MultiPartFile" (org.springframework.web.multipart)

```
public class JobSeekerInfo{
  private String jsName;
  private String jsAddrs;
  private MultiplePartFile resume;
  private MultiplePartFile photo;
  //setters && getters
    ...
    ..
    .
```

step4) Write logic in Post back handler method of controller class by using streams.. to recive uploaded files content from model class object and write them to server machine file system

use... IOStream file copy operations.. refer MVCProj14-AnnoFileUploading-Downloading

jar files springwebmvc -5.2.9 ,
servlet api 4.0.1
jstl 1.2
commons-fileupload 1.4
commons-io 2.8.0

collect from mvnrepository.com

step5) cfg "CommonsMultipartResolver" as spring bean in dispatcher-servelt.xml having fixed bean id to alert spring MVC env.."multipartResolver" and DS to recieve multipart MIME data along with the request..
note:: this allows to specify restrictions related to file uploadings...

```
<!-- Resolver cfg -->
  <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver"/>
```

fixed bean id

Alerts DS and other comps to check wheather current request is comming with multipart/form-data enctype or not .. if comming it makes DS and other comps to recived the the uploaded files content to store them in the Model class object MultipartFile type properties..

```
<!-- Resolver cfg -->
  <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
     <property name="maxUploadSize" value="#{350*1024*1024}"/>
     <property name="maxUploadSizePerFile" value="#{20*1024*1024}"/>
  </bean>
```

To apply restrictions uploading file sizes.. will be applied all the files..

(SPEL is used there to perform arithmetic and logical operations in xml file)

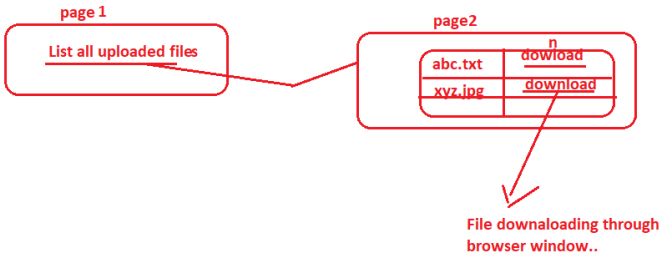
note:: To apply restrictions on file types and file size on specific files we need to write IO Streams customized logic in controller classes.

sample code in controller class

```
File resumeFile=new File(resumeFileName);
   if(resumeFile.length()>10*1024*1024)
        throw new IllegalStateException(" invalid filesize");
   if(resumeFileName.endsWith(".pdf") || resumeFileName.endsWith(".dat"))
        throw new IllegalStateException("invalid file type");
```

SPEL --->Spring expression Language
syntax :: #{<expr>} note:: can be used in "value" attribute of spring bean cfg file also in @Value() annotation ..

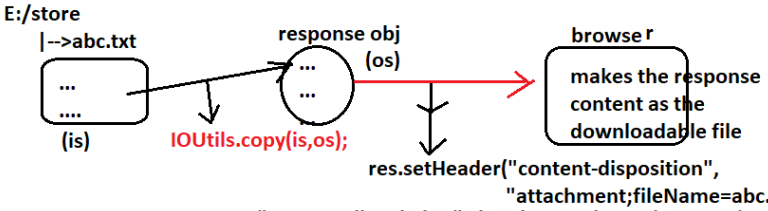
FileDownloading



Controller classes gets the ServletContainer managed ServletContext ,ServletConfig objs through @Autowired kept on class level instance instance variable becoz they are same across the multiple requests .. and gets request,resposse objs as the parameters of handler methods becoz they will change request to request.

In Controller class
==========
```
@Autowired
 private ServletContext sc;
@Autowired
private  ServletConfig cg;
  @Get/PostMapping("/handle")
 public String  handleForm(HttpServletRequqest req,HttpServletResponse res){
     ...
     ...
 }
```

=>The basic thumb rule of file downloading is get file content into response object and give instruction to browser for making that file content as the downloadable file..



note:: "content-disopistion" gives instruction to browser how to handle the recieved the content ..
=> if the value is "inline" then it dipslays the recived content on the browser itself
=> if the value is "attachment" then it makes the recived content as downloadable file

if centain file types are not having file MIME types like zip files, rar file, icon file and etc..
then we can universal/generic mimetype called "application/octet-stream".

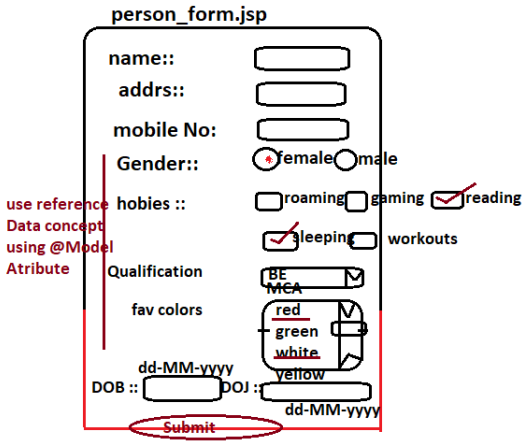
=>if the handler method returns "null" as the LVN then DS thinks that response is going to browser directly from handler method.. So no need of involving ViewResolvers..

```
@GetMapping("/download")
      public  String  downloadFile(@RequestParam("fname") String fileName,
                                   HttpServletResponse res)throws Exception {
            //get path of donwloadable file
            String filePath=sc.getInitParameter("uploadStore")+"/"+fileName;   // gives E:/store/abc.txt
            //create File object pointing to the file to be dowloaded
            File file=new File(filePath);
            //set file length as response content length
            res.setContentLengthLong(file.length());
            //get MIME type of the file and make it as response content type
            String mimeType=sc.getMimeType(filePath);
            res.setContentType(mimeType==null?"application/octet-stream":mimeType);
            //create InputStream pointing to the flle to be downloaded
            InputStream is=new FileInputStream(filePath);
            //create OutputStream pointing  response obj
            OutputStream os=res.getOutputStream();
            //Give instruction to browser to make the recived content as downloadable
            res.addHeader("Content-Disposition","attachment;fileName="+fileName);   //makes browser to display download
dialog box
            //copy file content to response obj
            IOUtils.copy(is, os);
      //take lvn as null
            return null;
      }
```

=>Working with @ModelAttribute for dynamic initial Data and @InitBinder for cfg PropertyEditor for Spring MVC Application to get used in DataBinding

=> To bind Enduser supplied different pattern date values from the form page to Model class obj properties/attributes the built-in covertors or Propertyeditors not sufficent.. So we need work customer editors.. To link to these Propertyeditors we need to use @InitBinder methods..

=>@ModelAttrbute(-) methods are useful to display dynamic initial values form radio buttons(group),checkboxses(group), select box , list box ..



Model class for the formpage

```
public  class Person{
  private String name;
  private  String addrs;
  private  long mobileNo;
 private   String  gender="female"
 private  String[] hobies=new String[]{"sleeping",
                                       "reading"};
 private  String  qlfy;
 private  String[]  colors=new String[]{"red","white"};
private  java.util.Date dob,doj;
//get setters && getters
....
 ...
}
```

```
<frm:radiobuttons path="gender" items="${genderList}"/>
```

gives multiple radion buttons that are grouped into single unit

grouped radio buttons name. The model class property name to which selected radio button value should be bound

The ModelAttrbute name that acts as referenceData to display bunch of radio buttons with values and labels (collected from @ModelAttribute("genderList") method of controller class..

Internal generated code

```
<input type="radio" name="gender" value="female" checked> female
<input type="radio" name="gender" value="male" > male
```

note:: @ModelAttrbute methods executes for every request.. i.e for the initial phase request that launches form page and for the post back request that submits the form page . these methods supplied reference data is very useful while displaying form page normally or with validation error messages becoz in both cases the radio buttons, checkboxes , list boxes, select boxes should get dynamic initia values...

=>By default only MM/dd/yyyy pattern date values will be bound with Model class obj java.util.Date type properties by taking the support of built-in property editors. for other date patterns we need to register Custom PropertyEditors explicity with ServletRequestDataBinder object (It is resposible to bind form data to Model class obj) by using @InitBinder method as shown below..

```
@InitBinder
public void myInitBinder(ServletRequestDataBinder binder) {
        System.out.println("PersonController.myInitBinder()");
        SimpleDateFormat sdf=new SimpleDateFormat("dd-MM-yyyy");
        binder.registerCustomEditor(Date.class, new CustomDateEditor(sdf, true));
}
```

property type

Readymade Property editor to convert given pattern String date values to java.util.Date class obj

note:: @InitBinder method executes in form launching to display initial values form comps accroding to property editor convertion.. also excutes in form submission to convert the form comp supplied String values as required for model class property.

---

note:: we can use html5 comps in spring mvc form tag library by specifying "type" attribute in <frm:input > tag

```
<tr>
    <td>DOB </td>
    <td><frm:input type="date" path="dob"/> </td>   (the default pattern is yyyy-MM-dd)
</tr>
<tr>
    <td>DOJ </td>
    <td><frm:input type="date" path="doj"/> </td>
</tr>
<tr>
    <td>salary range </td>
    <td><frm:input type="range" min="10000" max="200000" path="salary"/> </td>
</tr>
```

I18n in spring MVC

=>Making our App working for different locales is called working with I18n
=> We can take care the following things while dealing with I18n
a) presentation labels
b) date,time formats
c) number formats
d) currency symbols
e) indentation

=>spring mvc give built-in support to work with I18n

step1) Prepare multiple Locale specific Properties files having base file with common keys
====== and different values collected from google Translator

App.properties (base)
App_fr_FR.properties
App_de_DE.properties
App_hi_IN.properties
App_zh_CN.properties

=>base name should be placed in all proeprties file
=>All properties files shuold have same keys
but values can be changed.

step2) Configure base properties file in dispatcher-servlet.xml

```
<!-- Cfg propertie file -->
  <bean id="messageSource" class="org.springframework.context.support.ResourceBundleMessageSource">
    <property name="basename" value="com/nt/commons/App"/>
  </bean>
```

step3) Enable I18n on spring MVC App by cfg "SessionLocaleResolver" having fixed bean id "localeResolver"

```
<!-- Cfg Resolver to enable I18n on the Application -->
  <bean id="localeResolver" class="org.springframework.web.servlet.i18n.SessionLocaleResolver">
    <property name="defaultLocale" value="en"/>
  </bean>
```

Enables I18n on the spring MVC web application,
also takes default locale in this order

=> Session attribute "locale" value (if not there then)
=> defaultLocale property value (injtected) (if not thee then)
=> request header "accept" value.

note:: if given locale specific properties file is not
avaliable then it take base properties as default file..

step4) Cfg "LocaleChangeInterceptor" to change I18n settings for every request based locale info that is
coming along with request having given request param name..

```
<!-- Cfg LocalChangeInterceptor to chanage Local settings for every request -->
  <bean id="lci" class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">
    <property name="paramName" value="lang"/>
  </bean>
```

step5) Map the above Handler Interceptor to Controller class through handler Mapping

```
<!-- Handler Mapping -->
  <bean
 class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping">
    <property name="interceptors">
      <array>
        <ref bean="lci"/>
      </array>
    </property>
  </bean>
```

step6) Read message from active properties file by specifying keys and display them on the webpage..
using <spring:message> tag.
(Belongs to generic tag library)

Spring MVC with tiles
==========

=>A tile is logical partiation in the web page like header tile, footer tile, body tile and etc..

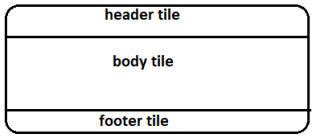
=> Tiles framework is third party framework to built web bapages having tiles with layout control and composite view design pattern..

=> Composite view design pattern says every web page comes as the output given by multiple webcomps (view comps) togather..

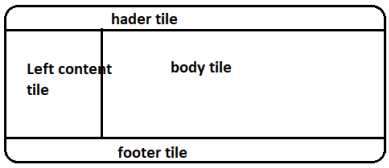
=> Layout control means.. all web pages are designed based on the common layout having tiles and these tile values are supplied by different jsp pages(view comps) more every. if modify any thing in the layout/template page , it will reflect to all the web pages..

Popular layout are
==========

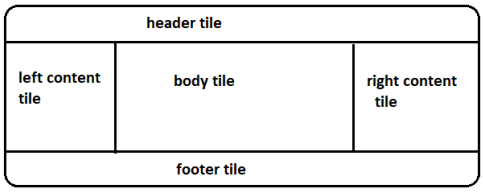
classic layout



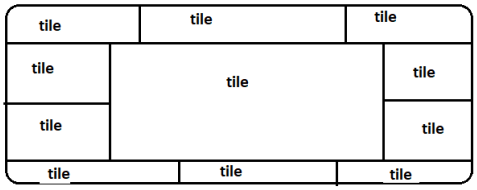
Two column Layout
==========



Three column Layout
==========



circular layout
==========



Advantages of tiles framework
==========

a) Layout control becoz of layout page/template page
b) Compsite view design pattern implementation
c) Uniform look for web pages
d) we can integrate this framework with struts,jsf ,spring mvc and etc..
e) Inheritence b/w tile definitations is possible.

and etc..

Spring MVC with Tiles
==========

step1) First decide how many web pages are the in website (4 web pages)

step2) Design layout page/template page having tiles with the support tiles tag library.

/WEB-INF/pages/layout.jsp (classic layout) (j) (x)
==========

```
<%@taglib uri="....." prefix="tiles"%>

<table border="0" width="100%" height="100%">
  <tr height="20%">
    <td><tiles:insertAttribute name="header"/>
  </tr>
  <tr height="70%">
    <td><tiles:insertAttribute name="body"/>
  </tr>
  <tr height="10%">
    <td><tiles:insertAttribute name="footer"/>
  </tr>
</table>
```

(j)
Executes this layout.jsp having following tile values to display the web page..

header ---> /WEB-INF/pages/header.jsp
footer ---> /WEB-INF/pages/footer. html
body ---> /WEB-INF/pages/page1_body.jsp

(x)
Executes this layout.jsp having following tile values to display the web page..

header ---> /WEB-INF/pages/header.jsp
footer ---> /WEB-INF/pages/footer. html
body ---> /WEB-INF/pages/page2_body.jsp

step3) Activate Tiles framework by cfg "TilesConfigurer" as spring bean in dispatcher-servlet.xml file

step4) Develop tile definitations in tile definitation cfg file specifying one tile definitation for each web page..

WEB-INF/tiles.xml
==========

```
<tiles-definitions>
  <definition name="page1Def"  template="/WEB-INF/pages/layout.jsp"  >
    <put-attribute name="header"  value="/WEB-INF/pages/header.jsp"/>
    <put-attribute name="body" value="/WEB-INF/pages/page1_body.jsp"/>
    <put-attribute name="footer" value="/WEB-INF/pages/footer.html"/>
  </definition>
  <definition name="page2Def"  template="/WEB-INF/pages/layout.jsp"  >
    <put-attribute name="header"  value="/WEB-INF/pages/header.jsp"/>
    <put-attribute name="body" value="/WEB-INF/pages/page2_body.jsp"/>
    <put-attribute name="footer" value="/WEB-INF/pages/footer.html"/>
  </definition>
  <definition name="page3Def"  template="/WEB-INF/pages/layout.jsp"  >
    <put-attribute name="header"  value="/WEB-INF/pages/header.jsp"/>
    <put-attribute name="body" value="/WEB-INF/pages/page3_body.jsp"/>
    <put-attribute name="footer" value="/WEB-INF/pages/footer.html"/>
  </definition>
  <definition name="page4Def"  template="/WEB-INF/pages/layout.jsp"  >
    <put-attribute name="header"  value="/WEB-INF/pages/header.jsp"/>
    <put-attribute name="body" value="/WEB-INF/pages/page4_body.jsp"/>
    <put-attribute name="footer" value="/WEB-INF/pages/footer.html"/>
  </definition>
</tile-definitations>
```

improved code using tiles inheritence
==========

```
<tile-definitations>
  <definition name="baseDef"  template="/WEB-INF/pages/layout.jsp"  >
    <put-attribute name="header"  value="/WEB-INF/pages/header.jsp"/>
    <put-attribute name="body" value=""/>
    <put-attribute name="footer" value="/WEB-INF/pages/footer.html"/>
  </definition>
  <definition name="page1Def"  extends="baseDef">
      <put-attribute name="body" value="/WEB-INF/pages/page1_body.jsp"/>
  </definition>
  <definition name="page2Def"  extends="baseDef">
      <put-attribute name="body" value="/WEB-INF/pages/page2_body.jsp"/>
  </definition>
  <definition name="page3Def"  extends="baseDef">
      <put-attribute name="body" value="/WEB-INF/pages/page3_body.jsp"/>
  </definition>
  <definition name="page4Def"  extends="baseDef">
      <put-attribute name="body" value="/WEB-INF/pages/page4_body.jsp"/>
  </definition>
</tile-definition>
```

step5) Cfg "TilesViewResolver" to render web pages based on tile definitions..

in dispatcher-servlet.xml
==========

```
<bean  class="org.springframework.web.servlet.view.tiles3.TilesViewResolver"/>
```

step6) Develop Controller class(es) having "tile definitation" name as the lvn in the handler methods

```
@Controller
public class PersonController{

@GetMapping("/page1")
 public  String showPage1(){
    return "page1Def";  // tile definitation name as the lvn
 }

@GetMapping("/page2")
 public  String showPage2(){
    return "page2Def";  // tile definitation name as the lvn
 }

@GetMapping("/page3")
 public  String showPage3(){
    return "page3Def";  // tile definitation name as the lvn
 }

@GetMapping("/page4")
 public  String showPage4(){
    return "page4Def";  // tile definitation name as the lvn
 }

}//class
```

```
<context:component-scan basePackage="com.nt.controller"/>
```

step7) develop all view comps (html, jsp files) having logics...

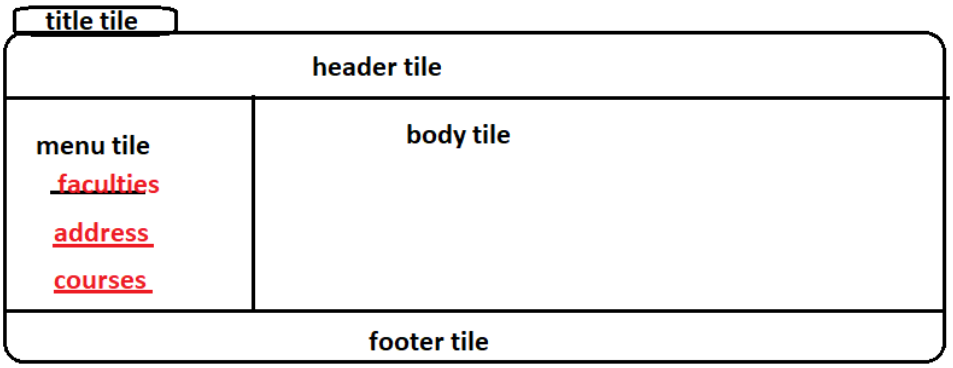
http://loclahost:3030/MVCProj17-Tiles/page1.htm

http://localhost:3030/MVCProj17-Tiles/page2.htm

Spring MVC with tiles example web application

==========

Two colum layout



Working with ViewResolver and Views
==========

=>In Spring MVC View an abstract entity i.e we can take any thing as view comps like
jsp files,servlet comps,html files,thymeleaf freemarker, volocity, spring beans (java classes) and etc..

=> To make Java classs as View comp we need to make java class implementing "View(I)" and
should cfg that class as spring bean .. we should also "BeanNameViewResolver"

=>To render response in the form of pdf docs or excel docs we need to use third party apis
like POI (excel) , IText (pdf) in our comps.. writing java code in jsp pages is very bad practice..
So prefere taking java class/spring beans as view comps to place third party code..

=> Developing java classes/ spring beans as view comps , it is recomanded to not implement
View(I) directly.. prefer extending them from AbstractXxxView classes which internally takes
care of common logics .. The classes are like Abstract Xls View , AbstractPdfView and etc...

```
@Component("excelView")
public class MyExcelView extenes AbstractXlsView{

public void buildExcelDocument(Map<String,Object> model,
                               org.apache.poi.ss.usermodel.Workbook workbook,
                               HttpServletRequest request,
                               HttpServletResponse response) throws Exception{
    ....
    ...
  }
}
```

```
@Component("pdfView")
public class MyPdfView extends AbstractPdfView{

 public          void buildPdfDocument(Map<String,Object> model,
                        com.lowagie.text.Document document,
                        com.lowagie.text.pdf.PdfWriter writer,
                        HttpServletRequest request,
                        HttpServletResponse response)throws Exception{
      ....
      ....
   }
}
```

In dispatcher-servlet.xml
==========

```
<bean class="pkg.BeanNameViewResolver"/>  // supports ViewResolver chaining
//other view resolvers cfg like InternalResourceViewResolver
```

=>Takes the lvn given by Controller class to
DS and seaches for view comp class/spring bean whose bean
id is matching with lvn , then takes that View class obj
to call render(-) method on it ..This render(-) intenrally
calls buildXxxDocument(-,-,-,-,-...) methods

Example Application
==========

web page (jsp page)

Report as Excel DOc --> taking java class /spring bean view comp.. output should come as Excel Doc as (uses poi api)

Report as PdfDoc --> taking java class /spring bean as view comp.. output should come as pdf Doc (uses iText api)

extra dependencies

```
<!-- https://mvnrepository.com/artifact/org.apache.poi/poi -->
<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi</artifactId>
    <version>4.1.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.lowagie/itext -->
<dependency>
    <groupId>com.lowagie</groupId>
    <artifactId>itext</artifactId>
    <version>4.2.2</version>
    <type>pom</type>
</dependency>
```

100%code Driven spring MVC

==========

=> Avoid xml cfgs completely..

=> web.xml file alternate is WebapplicationInitializer class
=> dispatcher-servlet.xml file alternate is WebMVCAppConfig class (@Configuration class)
=> applicationContet.xml file alternate is RootAppCofing class (@Configuration class)

Basic Thumb rule for 100% Code driven Apps

==========

=> Cfg user-defined classes as spring beans using stereo type annotations and link them with @Configuration class using @ComponentScan annotation
=> Cfg pre-defined classes as spring beans using @Bean methods of @Confiugration class..

WebMVCAppConfig.java
==========

```
package com.nt.config;

@Configuration
@ComponentScan(basePackages="....")
@EnableWebMVC
public class WebMVCAppConfig{

  ...
  .... more other beans using @Bean methods..
  ....      for ViewResolvers, TilesConfigurer and etc..

 }
```

RooAppConfig.java
==========

```
@Configuration
@ComponentScan(basePackages="....") (or)
@Import({PersistenConfig.class,ServiceConfig.class,...}
public class RootAppConfig{

  ....
  .... @Bean methods
  ....      for DS, JdbcTemplate and etc..
}
```

@EnableWebMVC Automatically registers multiple pre-defined classes with DS managed IOc container they are..

This class registers the following HandlerMappings:

RequestMappingHandlerMapping ordered at 0 for mapping requests to annotated controller methods.
HandlerMapping ordered at 1 to map URL paths directly to view names.
BeanNameUrlHandlerMapping ordered at 2 to map URL paths to controller bean names.
HandlerMapping ordered at Integer.MAX_VALUE-1 to serve static resource requests.
HandlerMapping ordered at Integer.MAX_VALUE to forward requests to the default servlet.

Registers these HandlerAdapters:

RequestMappingHandlerAdapter for processing requests with annotated controller methods.
HttpRequestHandlerAdapter for processing requests with HttpRequestHandlers.
SimpleControllerHandlerAdapter for processing requests with interface-based Controllers.

Registers a HandlerExceptionResolverComposite with this chain of exception resolvers:

ExceptionHandlerExceptionResolver for handling exceptions through ExceptionHandler methods.
ResponseStatusExceptionResolver for exceptions annotated with ResponseStatus.
DefaultHandlerExceptionResolver for resolving known Spring exception types

Registers an AntPathMatcher and a UrlPathHelper to be used by:

the RequestMappingHandlerMapping,
the HandlerMapping for ViewControllers
and the HandlerMapping for serving resources
Note that those beans can be configured with a PathMatchConfigurer.

3 ways of configuring servlet comp
==========
a) Declarative approach (<servlet>,<servlet-mapping> tags) [for Predefined Servlet comps where xml cfgs are allowed]
b) Annoation[@WebServlet] approach (for user-defined servlet comps)
c) Programatic Approach/Java code or config Appraoch /Dynamic Registration approach
[Using sc.addServlet(-,-) method] [For pre-defined servlet comps where xml cfgs are not allowed]

=>in 100%Code , Spring Boot mode of Applications development there is no possiblity of working with web.xml file So we need to use Programatic Approach to cfg/register predfined DispatcherServlet comp class.

WebApplicationInitializer class to cfg DS, ContextLoaderListener in spring MVC App as alternate web.xml file
==========

MyWebAppInitializer.java
==========
```
package com.nt.initializer;
.....
public class MyWebAppInitializer implements org.sf.web.WebApplicationInitializer{
                                                    (h)
  public void onStartup(ServletContext sc) throws ServletException{
    //create parent IOC Container
    AnnotationConfigWebApplicationContext parentCtx=
(i) pre-instantiation                    new AnnotationConfigWebApplicationContext();
of singleton scope  parentCtx.register(RootAppConfig.class);
 beans in          //create ContextLoaderListener object having Parent IOC container
Business tier       ContextLoaderListener listener=new ContextLoaderListener(parentCtx);
                    //register ContextLoaderListener with ServletContainer
                     sc.addListener(listener);  (j)
                    ------------------------------------------------------
                    //create child Container
                    AnnotationConfigWebApplicationContext childCtx=
                              new AnnotationConfigWebApplicationContext();
                (k) childCtx.register(WebMvcAppConfig.class);
  pre-instantiation /Create DispatcherServlet obj having Child IOC container
of singleton scope  DispatcherServlet servlet=new DispatcherServlet(childCtx);
 beans in           //register DispatcherServlet with SErvletContainer
presentation tier     ServletRegistration.Dynamic dyna=sc.addServlet("dispatcher",servlet);
                      dyna.addMappings{"/");
                      dyna.setLoadOnStartup(2);
                                                                                  (l)
                    }//method
}//class
```

(a) Deployment of Spring MVC App
(b)Identifies no web.xml file and also identifies no @WebServlet sevlet comps
(c) creates One Servletcontext obj

WEB-INF/lib folder of spring MVC App
[-->*.jar (spring-web-<ver>.jar and etc..)
(d) SErvletCotnaiener looks for META-INF/Services/javax.servlet.ServletContainerInitializer file in all the jar files that are added to WEB-INF/lib folder and finds it spring-web-<ver>.jar file
javax.servlet.ServletContainerInitializer (file) (META-INF/services)
org.springframework.web.SpringServletContainerInitializer (e)
Servletcontainer loads this ,cretes obj and calls onStartup(-,-) having ServletContext obj
(f)
p v onStartup(Set<..> set, ServletContext sc){
... calls onstartup(sc) on our WebApplicationInitializer class by searching (g) it in CLASSPATH (WEB-INF/classes)
}
note:: SpringServletContainerInitializer class Delegates the ServletContext obj to any WebApplicationInitializer(I) implementations present on the application classpath, i,e the onStartup(-,-) of SpringServletContainerInitializer class internally calls onStartup(-) our WebApplicationInitializer(I) Impl class onStartup(-) method..

AbstractAnnotationConfigDispatcherServletInitializer is pre-defined WebApplicationInitializer to register a DispatcherServlet and use Java-based Spring configuration.Implementations are required to implement:
getRootConfigClasses() -- for "root" application context (non-web infrastructure) configuration.
getServletConfigClasses() -- for DispatcherServlet application context (Spring MVC infrastructure) configuration.
getServletMappings() ---> To provide url pattern to DS cfg..

note:: The onstartup(-) of AbstractAnnotationConfigDispatcherServletInitializer internally takes care of creating both parent,Child IOC contaieners and ContextLoaderListener and DS registrations having IOC container and etc.. This methods internally calls the above 3 methods to get inputs like RootAppConfig , WEbMVcAppConfig configuration classes and DS url pattern.

Spring Boot MVC

==========

=> gives embedded servers

->apache Tomcat

->eclipse jetty

->undertow

=> 100% Code based configurations (xml cfgs are avoided)

=> AutoConfiugration

=> Live Server Realoding (Here chnages done in source will be recongnized internally)

=> Most of the spring WEB flow comps will be gnerated internally through auto configuration

=> We can work both with Embedded Servers and extenal servers

=> We can develop web application either as standalone Apps(jar file) (Embedded sErvers) and deployable web applications(war file) (Embedded servers/ Extenal servers)

and etc..

Thumb rule

==========

=>cfg user-defined classes as spring beans using streo type annotations

=> cfg pre-defined classes as spring beans using @Bean methods of @Configuration classes , if they are not coming through AutoConfiguration (we can give inputs to AutoConfiguration process using application.properties file)

Spring Boot MVC takes care of following thing internally

==========

a) DispatcherServlet cfg with "/" url pattern by giving one ready made web applicaiton Initializer class in the Deployment Directory Structure

b) RequestMappingHandlerMapping will come as handler mapping automtically.

c) So many ready SErvletFilters to show web pages having error messages..

d) Embeded Server (default tomcat)

e) InternalResoruceViewResolver by collecting the inputs from application.properties file. (prefix,suffix)

and etc..

note:: Controller classes, service classes ,DAO classes,AOP classes and etc.. should be developed by the Programmers..

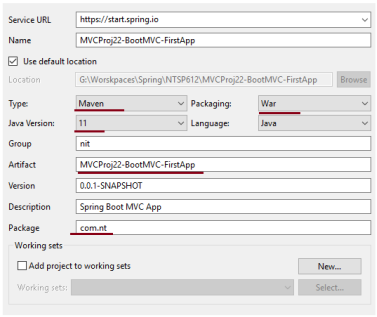
if u r system is not having external server installation then only use Embedded servers otherwise avoid them becoz Embedded servers do not suport connection pooling ,security , logging and etc.. features..

PRocedure to develop First Spring BOOT MVC Application

==========

step1) create Spring Starter Project having the following details..

File menu----> new ---> Spring Starter Project--->

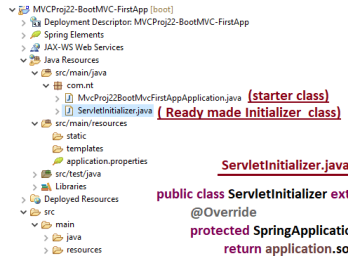


step2) Add tomcat-jasper jar file.... to pom.xml

```
<!-- https://mvnrepository.com/artifact/org.apache.tomcat.embed/tomcat-embed-jasper -->
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <version>9.0.39</version>
</dependency>
```

This is not requried if u r running the application on External Tomcat server.. Required only while working with Embedded Tomcat server..

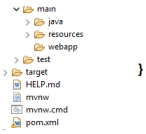
step3) Understanding Generated Directory structure of Spring Boot MVC Application..



Implmenents WebApplicationInitializer(I)

ServletInitializer.java

```
public class ServletInitializer extends SpringBootServletInitializer {
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(MvcProj22BootMvcFirstAppApplication.class);
```




```
}
```

```
protected SpringApplicationBuilder configure(SpringApplicationBuilder application) {
        return application.sources(MvcProj22BootMvcFirstAppApplication.class);
    }
```

note:: In Spring Boot mvc .. we can work only one IOC container..i.e there is no provision to work with two IOC container as we did in other approaches..

note:: In Spring Boot MVC Applications , there is no provision to write Business tier or presentation tier comps outside the spring env.. So two IOC containers provision is not given and not required.

step3) Develop the source code According u r requirment.. by keeping Autoconfiuration in mind

ShowBrowserController.java

```
@Controller
public class ShowBrowserController {

    @GetMapping("/welcome")
    public String  showHomePage() {
        //return lvn
        return "home";
    }

    @GetMapping("/browser")
    public String  getBrowserName(Map<String,Object> map,HttpServletRequest req) {
        //get browser s/w name
        String brName=req.getHeader("user-agent");
        //keep browser name as model attribute
        map.put("browser",brName);
        //return lvn
        return "result";
    }
}
```

application.properties

```
#ViewResolver cfg
spring.mvc.view.prefix=/WEB-INF/pages/
spring.mvc.view.suffix=.jsp
```

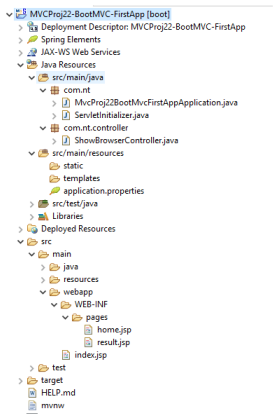
home.jsp

```
<h1 style="color:blue;text-align:center"><a href="browser">Get Current Browser name</a>
</h1>
```

result.jsp

```
<h1 style="color:red;text-align:center">Result page </h1>

<b><i>current browser name:: ${browser } </i></b>
<br>
<a href="welcome">home</a>
```



step4) Run the web Application on external server..

In pom.xml ---> comment tomcat-embedded-jasper dependecy tag.. (becoz that is only for embedded Tomcat server)

Right click on the RPoject ----> Run on Server --->choose Tomcat Server (any server)---> .....

step5) Test the web application

http://localhost:3030/MVCProj22-BootMVC-FirstApp
Project name as context path

note:: if u r running spring booty MVC Application on external server.. there is no need of placing main(-) method starter/main /Configuration class where @SpringBootApplicaton is coming..

Running Spring Boot MVC Application on Embedded Tomcat server

a) make sure tomcat-embedded-jasper jar/dependency is added to pom.xml file

b) make sure that main(-) is placed in main Confiuration/starter class where @SpringBootApplication is coming..

```
@SpringBootApplication
public class MvcProj22BootMvcFirstAppApplication {

    public static void main(String[] args) {
        SpringApplication.run(MvcProj22BootMvcFirstAppApplication.class, args);
    }

}
```

c) Change the embedded tomcat server port number.. using application.properties file.. (default port is :: 8080)

In application.properties

```
#Embedded tomat server port number
server.port=4040
```

d) Run the Application as spring Boot App/Application
by using main class where @SpringBootApplication is placed (Here applocation runs standalone App using embeded tomcat]

java

e) Test the web application request path

http://localhost:4040/welcome (type this url by opening browser window)

No Context path here.. note::default welcome file will not work here (Embeded Tomcat)

note:: ServletInitializer.java file is not required while running the Application as standalone App using embedded Tomcat

To make spring boot mvc app working with server managed jdbc con pool.. add the following entry in application.properties file

```
#To work server managed jdbc con pool
spring.datasource.jndi-name=java:/comp/env/DsJndi
```

Spring boot MVC gives InternalResourceViewResolver as the default ViewResolver to supply inputs that view resolver like prefix and suffix ,we need to write following entries in application.properties

```
spring.mvc.view.prefix=/WEB-INF/pages/
spring.mvc.view.suffix=.jsp
```

To view other than InternalResourceViewResolver in spring boot mvc Application , we need to cfg that ViewResolver as @Bean method in any configuration class

```
@Bean
public TilesViewResolver createVR() {
      return new TilesViewResolver();
}
```

=>In Spring Boot MVC the following special classes should be cfg as spring beans using @Bean methods becoz they will not come through auto configuration

a) Resolvers like CommonMultipartResolver(In file uploadin), SessionLocalResolver(I18n)
b) Configurers like TilesConfigurer
c) Other than InternalResourceViewResolver like TilesViewResolver,BeanNameViewResolver, ResourceBundleViewResolver and etc..

and etc..

The addtional ViewResolvers that are added as @Bean Methods will always added before the AutoConfiguration based InternalResourceViewResolver to support View Resolver Chaining..

Thymeleaf
==========
=> Another UI Technology that is build on the top of html for dynamic web pages
=> only html gives static web pages .. html tags+ thymeleaf tags gives dynamic webpages . It is light weigher alternate to heavy weight jsp pages..
=> In the execution of jsp page lot of memory and lot of cpu time is required becoz interanlly translated in jsp equalent servlet comp and creates multiple implicit objs (9) if used or not used.. for all these things lot of memory and cpu time required.
=>To overcome the above problems of jsp pages use thymeleaf as lightweight alternate for rendering dynamic webpages...

=> We need to write thymeleaf tags in html tags.. by importing thymeleaf namespace by specifying its namespace uri..

```
<html    xmlns:th="https://www.thymeleaf.org">
  ...
</html>
```

=>Spring boot gives built-in support of thymeleaf UI by giving
default prefix is <classpath>/templates/ (classpath here is src/main/resources folder)
default suffix is .html

=> To use thymeleaf in spring boot add this starter to pom.xml

```
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-starter-thymeleaf -->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-thymeleaf</artifactId>
   <version>2.3.4.RELEASE</version>
</dependency>
```

=>To execute thymeleaf code Thymeleaf engine is required which will become becoz this jar file.. This engine converts thymeleaf tags to html code.. and sends to browser as response..

=> Standard jsp tags identified with fixed prefix called <jsp: xxxx> and similary thymeleaf tags are indentified with <th: xxx> prefix

=> Popular symbols in thymeleaf programming
@ --->to specify Location (/<globalpath>/<request path>)
$ ---> To read data from container managed scopes like model attributes
* ---> To bind/link data to form comps (useful only in thymeleaf forms)

a. To read data from model attributes/container managed scopes
-> for primitive/wrapper/String type model attrbutes th:text="${<attribute name>}"
-> for Object type model attrbutes th:text="${<objectName>}"
-> for getting property values from Object type model attrbutes th:text="${<objectName>.<property name>}"
-> for looping through arrays/collection th:each="<counter variable>:${<collection/array type attribute>}"

b. To display images (To link image file to <img> tag)

```
<img  th:src="@{/path}">
```

c. To Link/map with hyperlink

```
<a  th:href="@{/path}"> xxxx </a>
```

d.To Link /map css file

```
<link rel="stylesheet"  th:href="@{/path}">
```

e. To Link /map java script fle

```
<script type="text/javascript"  th:src="@{/path}">
   ....
</script>
```

f) Thymeleaf forms are birectional forms i.e they support DataBinding(writing form data to model class obj) and DataRendering (writing handler methods supplied model attributes/model class obj data to form comps)

=> To specify action url <form th:action="@{global path/request path}" >
= > For binding model class obj data/model attributes data to form comps (for form backing object operation)
-> <form th:object="${model attribute name/object name of model clas}"> (alternate to <frm:from modelAttribute="....">)
=> For binding model class obj data /model attributes data to form comps
<input type="text" th:field="*{<model class property name/model attrobute name>}"> (alternate to <frm:input path="..."/>)

What is the difference b/w <th:text> and <th:field> tags in thymeleaf
==========
th:text is given to read data from Contaner from different scopes and to display them on browser.. like reading and displaying model class obj data and model attrbutes data..
th:field is given to bind model class obj property values /model attribute values(reference data) to form comps

---

Converting MiniProject to Thymeleaf UI based Application
==========
step1) add thymeleaf starter to pom.xml file

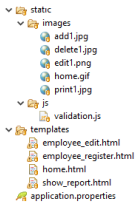
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

step2) Provide global path to controller class

```
@Controller
@RequestMapping("/employee")  //global path
public class EmployeeController {
  ...
  ...
  ...
```

}

step3) copy js,images folders of webapp/webcontent to "static" folder of src/main/resources folder and WEB-INF/pages folder jsp files to template folder of "src/main/resources" folder , change .jsp extension to .html



Refer MVCProj27

step4) modify "template" folder .html files code thymeleaf code from html code.

step5) Rum the Application...

Pagination using Spring boot mvc + spring Data JPA

==========

=>The process of displaying huge no.of records page by page having page numbers as hyperlinks with next , previous , first ,last links called pagination

=>Spring Boot MVC and Spring Data gives built-in support for pagination..

To do pagination

==========

a) Take Spring data jpa repository extending from PagingAndSortingRepository<T> which gives

Page<T> findAll(Pageable pageable)

|--->carries pageNo, pageSize as inputs

List<T>=(getContent())

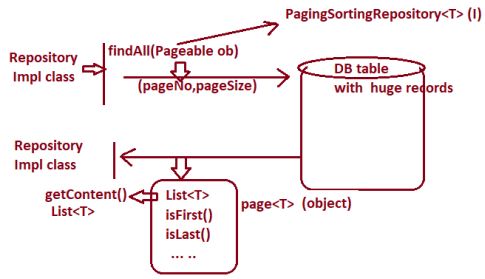
isFirst() ,isLast()

getTotalRecords()

getTotalPages()

getCurrentPage()

In Pagable Object pageNo input is O based index.



b) we should make sure request url is always coming with query String having page,size as the req param values..

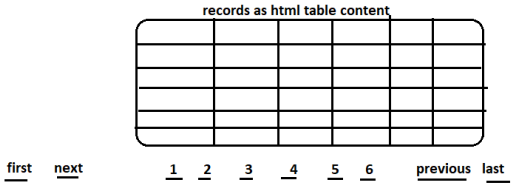
http://localhost:3030/MVCProj28/report_paging.htm?page=... &size=...

Becoz Spring MVC DispatherServlet can create Pageable object dymamically by gathering pagesize from "size" and pageNo from "page" fixed request parameter..

```
@GetMapping("/report_paging.htm")
public    String    showReportWithPagination(Map<String,Object> map,
            @PageableDefault(page=0,size=5)  Pageable pageable ){
 //use Repo
  Page<Employee>  page=empRepo.findAll(pageable);
  //keep page object in model attrbute
   map.put("page",page);
  //return lnv
  return  "emp_report";
 }
```

@PageableDefault makes DispatcherServlet to take given values as the default values for pageNumber,pageSize while construting Pageable object ,if the request is not given having page,size request params.

c) use JSTL tags in jsp page or thymeleaf tags to read page Object data and to display report having page number and other hyperklinks like next,previous and etc..



Exception Handling In spring boot mvc
==========

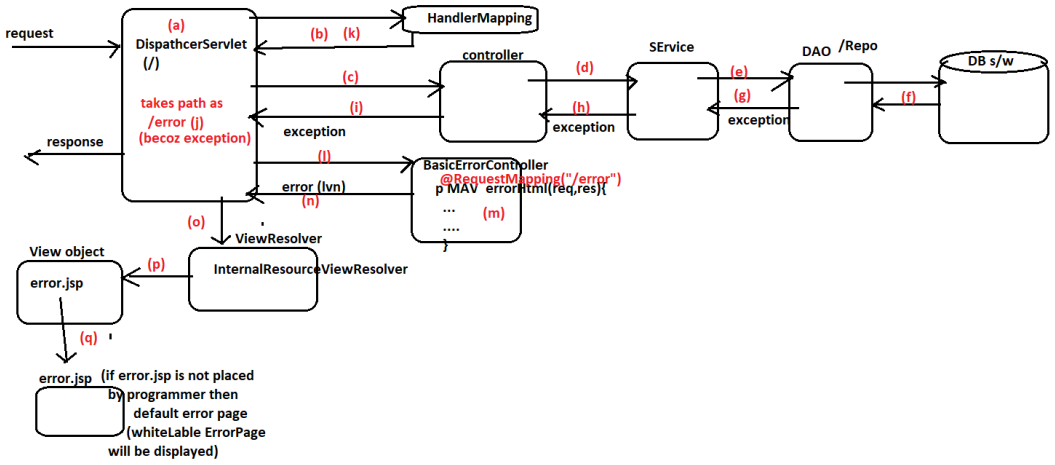
spring boot mvc gives built-in support for Exception handling .. alos default error pages to display the error messages.. if these error messages are not use-friendly.. we can customize them by taking our own error pages..

=> In spring mvc / spring boot mvc the exceptions raised in any layer will come Dispatcher Servlet through exception propopagation and DS internally uses the controller called BasicErrorController(impl class of ErrorController(I)) to render error pages with default error messages on to the browser (with support of errorHml() or error() handler methods) having request path "/error"

flow of execution
==========

=>DS calls handler method of our controller class through HandlerMapping
=>DS gets problem(Exception) from Controller (controller's own exception or propagated from DAO,Service class)
=> DS maps the request to BaiscErrorController through HandlerMapping having request "/error" and errorHtml() or error() handler method take the request and render default error page

note:: All the operations in DS (DispatherServlet takes place from doDispatch(-,-) that is called through doService(-,-) method)



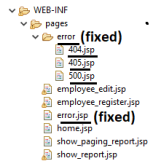
=>To cfg custom error page for better user-friendly uri
==========

=>add error.jsp in the place u r view resolver pointing to (like prefix value of View solver cfg)

webapp
|---->WEB-INF
|---->pages
|---->error.jsp (fixed name)
|---->home.jsp

in application.properties
---
spring.mvc.view.prefix=/WEB-INF/pages/
spring.mvc.view.suffix=.jsp

=>Instead taking same error page for all error codes we can design specific page for specific error code having <errorcode>.jsp as the file name is /error folder.



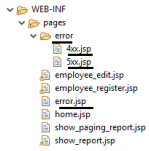
error code numbers are fixed

HttpStatusCodes
==========
100 -199 :: Information (1xx)
200-299 :: success (2xx)
300-399 :: redirection (3xx)
400-499 :: Incomplete (client side problems) (4xx)
500-599 :: Server side error (5xx)

4xx,5xx are error status codes (400-599)

if specific error code page is not found in "error" folder then it goes to common error page (error.jsp)

=>we can design specific error code page having names like 4xx.jsp , 5xx.jsp to same error pages for all 4xx errors and 5xx errors..



error.jsp
=======

```
%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1" isELIgnored="false"%>
  <h1 style="color:red;text-align:center"> Intenral problem -->try after some time  </h1>
  <br>
   status  ::  ${status}
```

4xx.jsp

```
 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

    <h1>Internal problem--Client side error</h1>

    status :: ${status}
```

---

we can create CustomException class linking with our choice error response status code and we can use that exception in our application.. as shown below..

---Custom exception class-----

```
package com.nt.exception;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(code = HttpStatus.SERVICE_UNAVAILABLE)
public class EmployeeNotFoundException extends RuntimeException {
        public EmployeeNotFoundException() {
              super();
        }
        public EmployeeNotFoundException(String msg){
              super(msg);
        }
}
```

In Service class handler method
==========

```
  //use DAO
Optional<EmployeeBO> opt=empRepo.findById(id);
if(opt.isEmpty())
      throw new  EmployeeNotFoundException("problem");
```

Embedded Servers in spring Boot MVC
==========

=> As of now spring boot mvc is supporting only 3 embedded and multiple(all) external server
   a) apache tomcat (default)
   b) Eclipse jetty
   c) Jboss undertow

   In development and testing of spring boot mvc apps "Embedded" servers will be used.. in Staging(UAT) and production env.. "External Servers" will be used..

To cfg jetty server with spring boot mvc application
==========

a) add spring-boot-starter-jetty dependency

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-jetty</artifactId>
</dependency>
```

also add this for jsp pages
==========

```
<dependency>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>apache-jsp</artifactId>
</dependency>
```

b) remove all entries related tomcat.. exclude from spring-boot-starter-web

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
        <exclusions>
            <exclusion>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-core</artifactId>
            </exclusion>
            <exclusion>
                <groupId>org.apache.tomcat.embed</groupId>
                <artifactId>tomcat-embed-websocket</artifactId>
            </exclusion>
        </exclusions>
    </dependency>
```

Do it from Dependency hierarch tab..
go to tomcat-embedded-core/websocket jar file right click --> exclude

---

Configuraging undertow server with spring boot mvc application
==========

step1) add spring-boot-starter-undertow dependency in pom.xml

```
<!-- https://mvnrepository.com/artifact/org.springframework.boot/spring-boot-
starter-undertow -->
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-undertow</artifactId>
</dependency>
```

step2) same as jettly server..