What are **spring stereotypes?**

**@Component** is the main stereotype used to create beans.

Subtypes of component are **@Service – used to craete beans at service layer** (holds business logic)

**@Repository -used to create beans at DAO Layer.Deals with CURD operations**

**@Controller – used to create beans at Controller Layer** (Front controllers used to handle requests and give response)

JDBC Steps:

1.Load and Register the driver 🡪Class.forName(“com.mysql.jdbc.Driver “);

2.Create the connection ->Connection con=DriverManager.getConnection(url,username,password);

3.Create the statement -> Statement st =con.createStatement();

4.Execute the query ->ResultSet rs=st.executeQuery(query);

5.Close the connection

Servlets Example Code:

Servlet is a server component which is responsible to create dynamic Web Content.f

public class Add extends HTTPServlet{

service(req,res){

}

}

**How to execute maven manually?**

1.From cmd change directory to xml file of that pgm >mvn clean install

2.From s/w open tomcat -bin – copy path, from eclipse editor go to pgm – xml config n copy path(in form of war file)

3.s/w – tomcat – webapps – Paste copied xml loc

4.from cmd >startup.bat(This command is used to start tomcat server)

**Spring MVC :**

Summary:

(common processing logic)

|

Client request-> DispatherServlet ->HandlerMapping : Identify the controller.

-->Controller processing the logic and returns modelname

-->ViewResolver : convert logical name into physical name

---> View : what formate display the data like pdf, excel

**Flow /Architecture of Spring MVC**

How to create the spring container

// stand alone application

1. ApplicatonContext context = new ClasspathXMLApplicationContext();

= new AnnoationConfigApplicationContext();

// web application

2. ApplicationContext context = new XMLWebApplicationContext();

<http://localhost:8080/login.html>

when any request is send dispatcher servlet, object will be created

http:// localhost:8081/login.htm

steps:

1.Create a maven web project and add dependencies in pom.xml

2.Write web.xml

<webapp>

<servlet>

<s-n>dispather</s-n>

<s-c>org.springframework.web.DispatcherSerlet</s-c>

</servlet>

<urlpattern>

<s-n>dispather</s-n>

<url-patter>\*.htm or.mvc or .do or .action</url-pattern>

</urlpattern>

</webapp>

3.Write controller class

public interface Controller{

ModelAndView handleRequest(HttpServletRequest req, HttpServletResponse resp)

}

public HelloworldController implements Controller{

//override methods and write some business logic

public ModelAndView handleRequest(HSR req, HSR resp){

ModelAndView mav.setViewName mav = new ModelAndView();

Mav.seViewName ("/home");

return mav;

}

}

4. write dispatcher-servlet.xml file

a. configuration about HandlerMapping, controller, view resolver

<beans>

<bean id=”helloWorld” class=”HelloWorldController”>

<bean id=”viewResolver class=”------“>

<property prefix=”/WEB-INF/jsp”/>

<property suffix=”.jsp”>

</bean>

</beans>

DispatcherServlet always looks for <servlet-name>-servlet.xml file

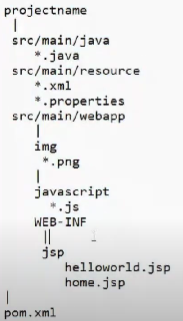
dispatcher-servlet.xml

5.Write jsps under /WEB-INF/jsp folder

Hello.jsp,home.jsp

6.Build the Application as war module

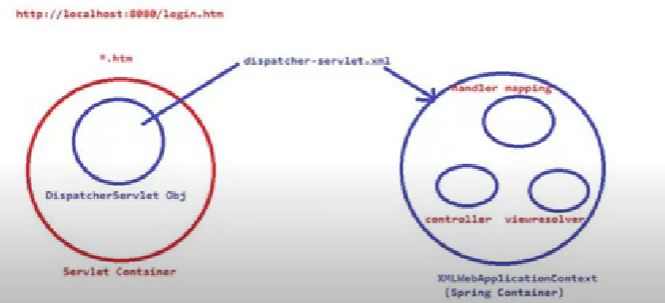
7.Deploy application

in web-inf only web.xml,jsp present

\*Dispatcher Servlet **object** created by **servlet container**

**Spring Container created by dispatcher servlet.**

\*If a class(here HelloWorldController)called by dispatcher then it is called Controller.

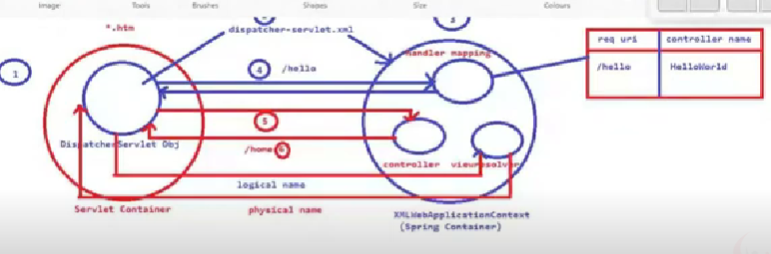


XML Web Application Context

(Spring Container)

Inside handler mapping it maintains 1 table

|  |  |  |
| --- | --- | --- |
| Req uri | Controller name |  |
| /hello | HelloWorld |  |



1.Dispatcher Servlet obj always looks for dispatcher-servlet.xml file

2.The beans handler mapping ,controller,view resolver will be created in Spring Container.

3.The DS(Dispatcher Servlet )sends request to Handler Mapping(HM) & based on uri (/hello)HM gives Controller name(HelloWorldcontroller)

4.DS calls Controller,the Controller always gives logical view name to View Resolver

5.VR (View Resolver)converts logical view name to physical view name

6.The DS(Disptcher Servlet) display name in view layer

**What is use of Logical View?**

If we want to upgrade the project, it’s not required to change the controllerlogic.That why it returns in Logical View format. Only VR logic to be changed

--------------

**<mvc:view-controller path="/" view-name="helloworldStatic" />**

This line is used to execute the view name without calling the controller

Generally, we will be used to display static data while landing page

view-name will be given to ViewResolver and ViewResolver will convert into the physical name

helloworldStatic.jsp

**\*Instead of implementing Controller we use @Controller**

**Ex Class Hello implements Controller{ }**

**@Controller**

**Class Hello{}**

**---For Handler Mapping we use @RequestMapping**

**Drawbacks of xml:**

**=========**

1. Developer should learn XML, XSD, DTD .. language

2. It is not a type safety

3. Switch java to XML is difficult

4. conditional logic we can't write in XML

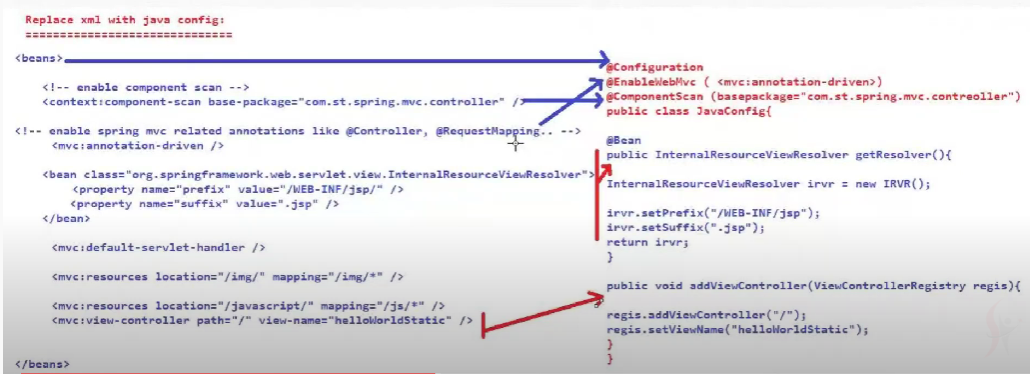
5. no maintainability and readability

**To overcome all these problems we should use java configuration:**

DispatherServlet by default will be looking for dispatcher -servlet.xml file

Now we don't have an XML file then how DispatherServle will load the MvcConfig class

We have to provide configure it manually in web.xml, so that DispatcherServlet will load the MvcConfig class



<init-param> - used for particular servlet

<context-param> - used for aall servlets

**Example : Spring MVC with no xml file**

**1. dispather -servlet.xml ---- replacement with java config**

**2. web.xml ➖➖➖➖➖ replace with java config**

Q) without web.xml how the web container can identify the DispatcherServlet class

Sol :

ServletContainerIntializer (when tomcat start it will looks for web.xml, if not found then

|extends for ServletContainerIntializer)

SpringServletContainerIntializer

|extends

WebApplicationIntializer

|extends

AbstractContextLoaderListner

|extends

AbstractDispatherservletIntializer

|extends

AbstractAnnotationConfigDispatherservletIntializer

public class webconfig extends AbstractAnnotationConfigDispatherservletIntializer{

public class<?> getRootConfigClasses(){

return null;

}

public class<?> getServletConfigClasses(){

return new Class<?>[]{MvcConfig.class};

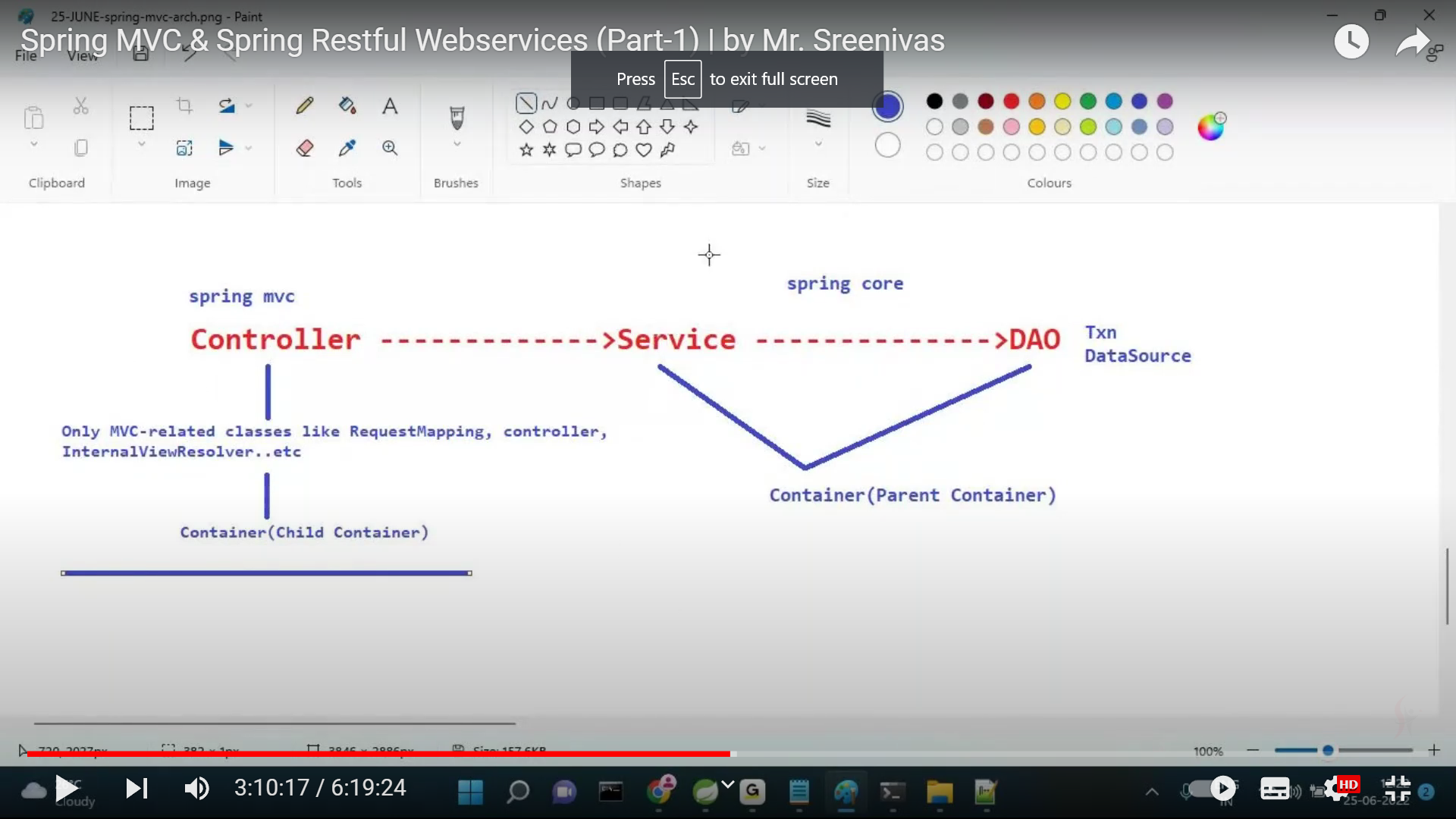
}

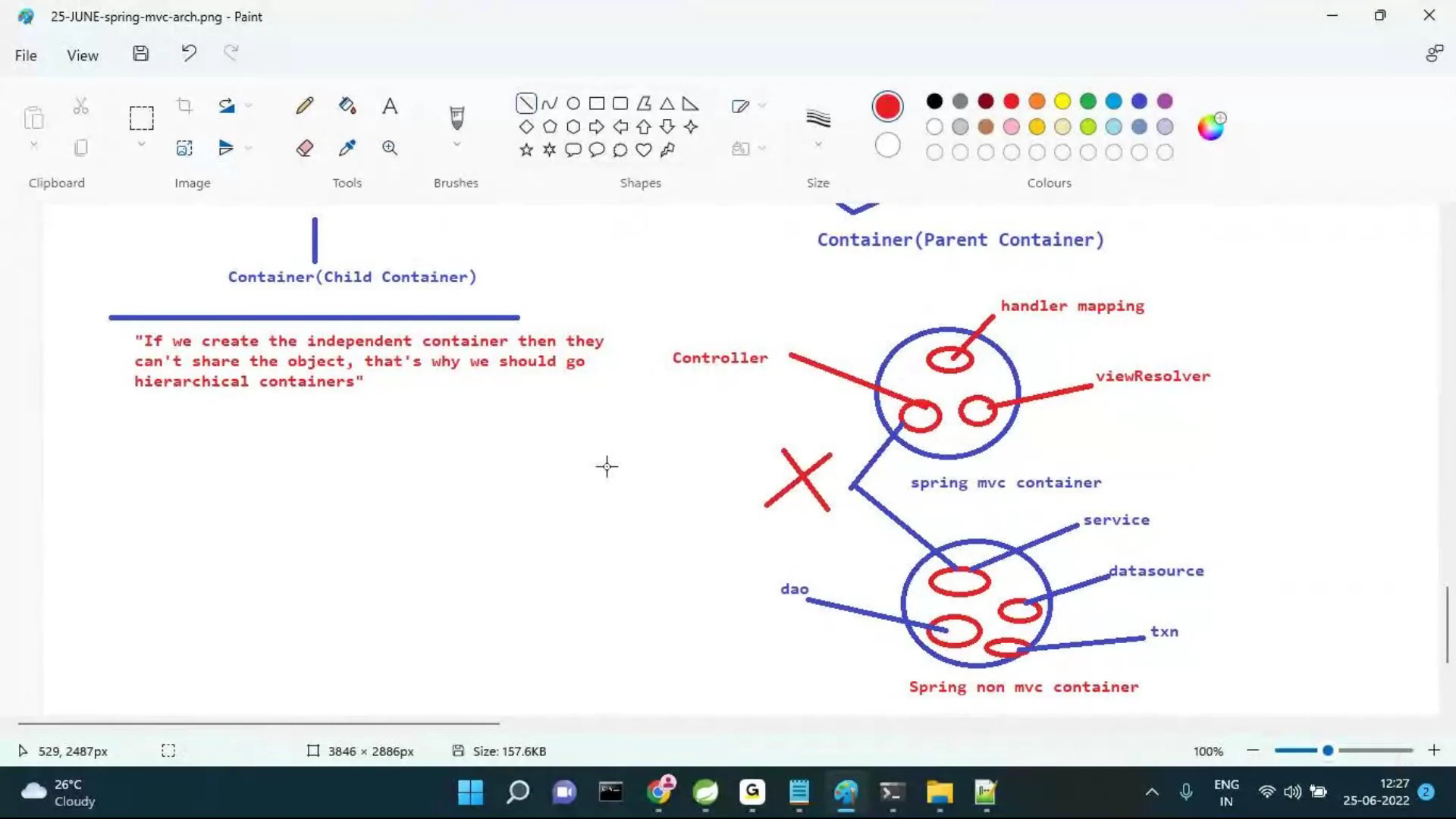
public String[] getServletMapings(){

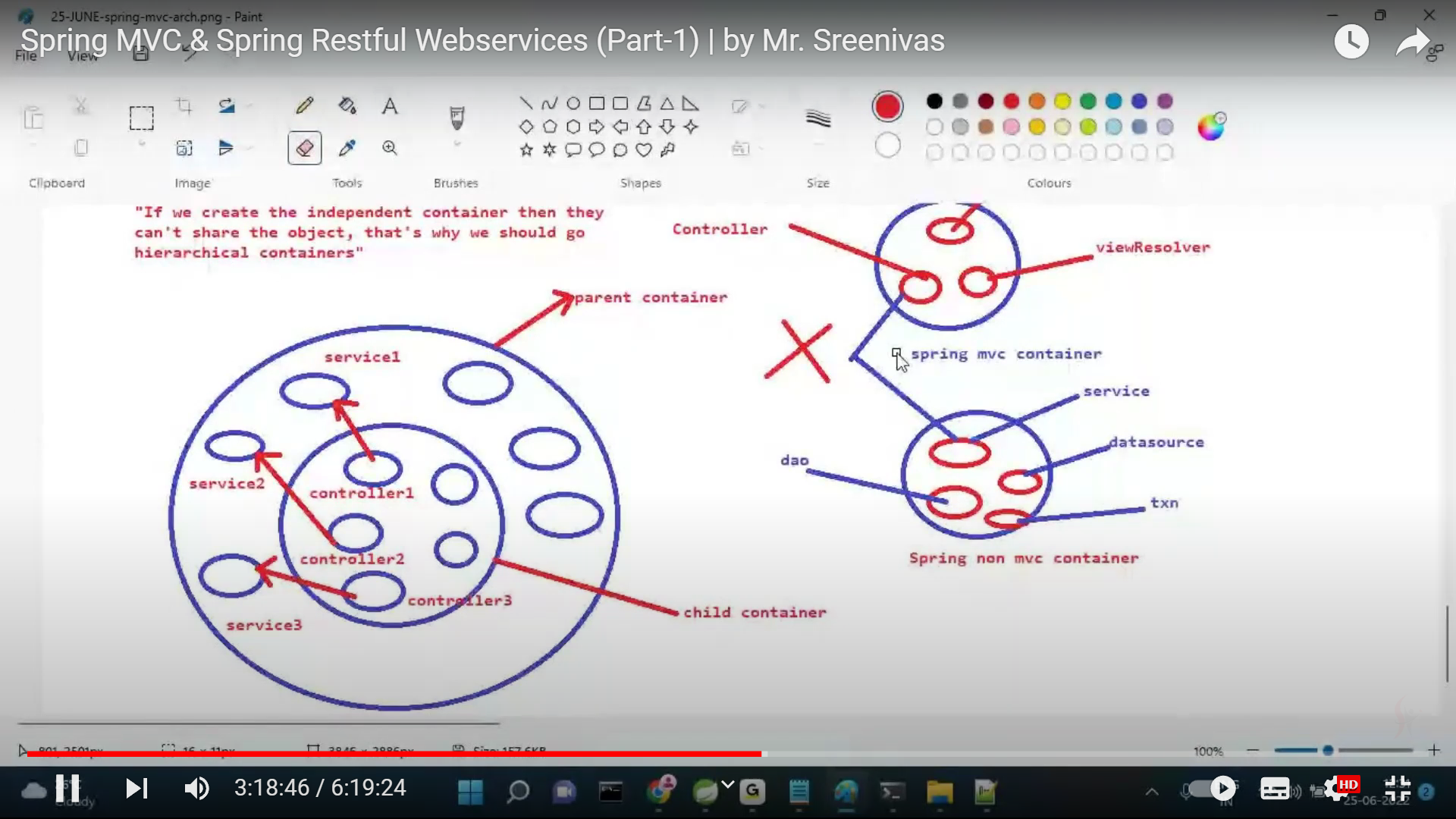
return new String[ ]{"/"}

}

1.xml mvc 2.java mvc 3.no dispatcher -servlet xml example







Example 4 : spring mvc example on parent and child container

Controller -->Service-->DAO

child container : :

#1. spring MVC related beans like HandlerMapping, Controller, and ViewResolver will be created in

child container

#2. child container will be created by DispathchserServlet with the help of dispatcher-servlet.xml file or java config(MvcConfig.java)

ParentContainer :

#1. Non spring MVC classes like Service, Dao, Security, Txn, Datasource..etc will be created in

Parent Container

#2. ContextLoaderListener is responsible for creating the parent container with the help of

ApplicationContext.xml

Note: If there are any non mvc classes present in the classpath, DispatcherServlet looks for ParentContainer

if it is created then only child container will be created.

Example 6: spring mvc multiple dispatcherservlets

DispatherServler1 ---- child container1 --- mvc

DispatherServlet2 ---- child container2 -- rest api

Note : According to project requirements there will be many Dispatcher Servlets would be there but

Parent container is only one. This parent container beans like service, dapo, datasource, txn..ete

will share to all the child containers.

-->In web.xml we have to configure 2 DispatcherServlets with separate url patterns

/\*.mvc--- MVCDispatherServlet

/\*.rest – RestDispatcherServlet

1. Introduction about MVC

2. MVC Architecture

3. MVC examples

a. Spring MVC with XML approach - web.xml and dispatcher-servlet.xml

b. Spring MVC with XML and java config approach - web.xml and java config (replacement of dispatcher-ser

c. Spring MVC with java config (no xml file - replacement of web.xml with java config and

replacement of dispatcher-servlet with java config

d. spring mvc with parent and child container

spring mvc with multiple DispatcherServlets

\*Root Config class created by ContextLoaderListener

\*MVC Config created by Dispatcher Servlet

Example 8 : XML configuration with java config

Problem Statement :how to load parent container and child container without web.xml

with web.xml

parent container-is created by ContextLoaderListener using applicationContext.xml file

child container- is created by Dispatcherservlet using dispatcher-serviet.xml file

without web.xml

The developer has to write manual code in the configuration class

parent container:

\*When we implement WebApplicationInitializer, servletContainerInitializer,SpringServletcontainer Initializer..developer need to manually write this code for dispatcher servlet.

// this is to load root context

XmlWebApplicationContext rootApplicationContext = new XmlWebApplicationContext();

rootApplicationContext.setConfigLocation( "classpath: /applicationContext.xml") ;

// add listener for root context

ContextLoaderListener listener = new ContextLoaderListener(rootApplicationContext);

servletContext.addListener(listener);

Child Container:

XmlWebApplicationContext childApplicationContext = new xmlWebApplicationcontext();

childApplicationContext. setConfigLocation("/NEB-INF/dispather-serviet .xm1");

DispatcherServiet dispatcherServiet = new DispatcherServiet (childApplicationContext) ;

// add dispatcher servlet to the main servlet context

ServletRegistration.Dynamic dispatcher = servletContext.addserviet("dispatcher", dispatcherServlet);

dispatcher. addMapping("/");

------------------------------------------------------

URI: https: //www.amazon.in/s?bookid=1234/bookname=springrest – QueryString -- @RequestParam annotation is used to read the values from querystring

URI:https://ww.amazon-in/1234/springrest ---PathParam --- @PathVariable -- is used to read the values from path params



To extract data from the uri we should use @RequestParam and @PathVariable

1. Querystring :

>@RequestParam is used to extract the query string data from the uri

>In QueryString data is optional, if you will not send then default value will come

2. Path Params :

-->@PathVariable is used to extract the uri path data from the uri

-->In Path params data is mandatory, if you will not send date then 404 resource not found error will come

How to read the data from Http Request Body

-->Genrally Http request body will send in the form of http post method

-->ModelAttribute willt care to read the form data and bind into Java Bean

-->During binding to java bean if any errors found then BindingResult will be executed which will hold

all the error details



-->Spring MVC is not using to develop web applications because of below reasons

1. we need web container is required-licence

2. processing time will take care

3. not light weight

Today Session:

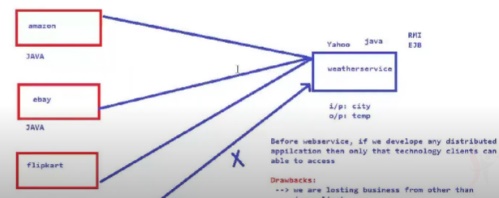
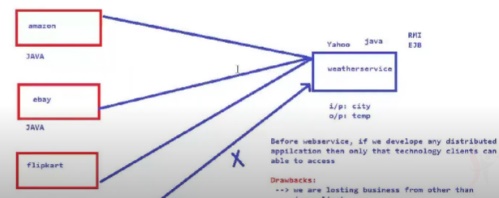
1. JAXB

2. JACKSON

3. Importance of Http in Rest API

4. Spring Rest Examples

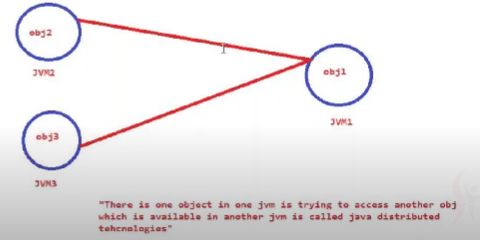
1. CRUD Operation

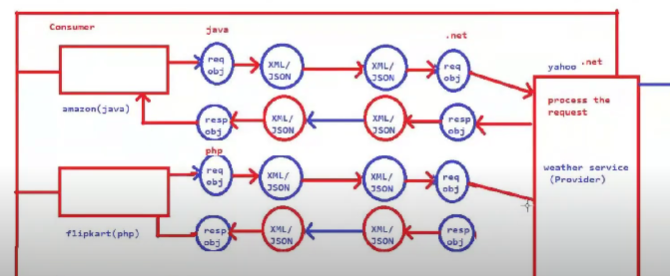


Yahoo will depend on amzon,ebay,flipkart for weather condns which is restricted to only java technology.So losing business from other than java clients.

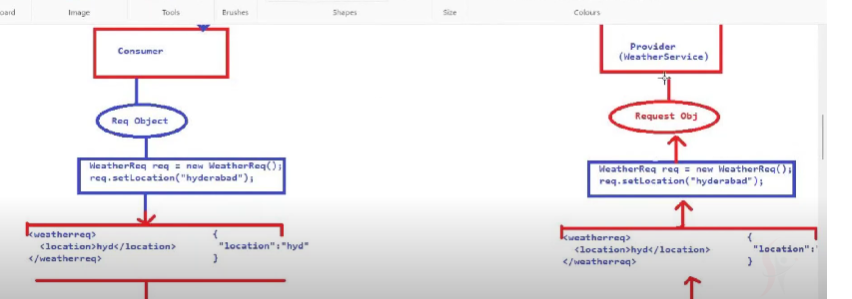
To overcome these problems all companies like IBM,Microsoft,Oracle sit together introduced a concept called

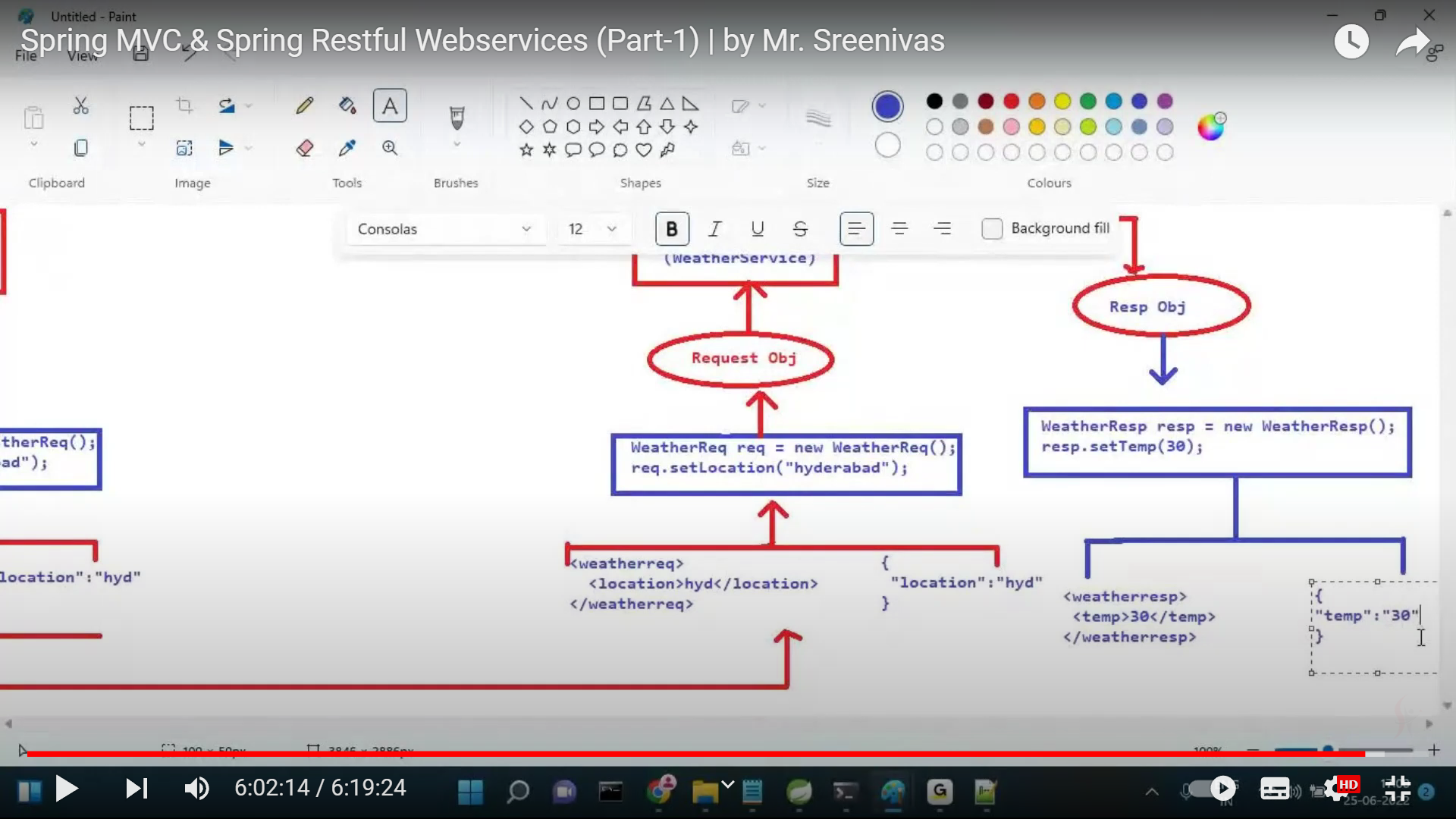
**Webservices.**

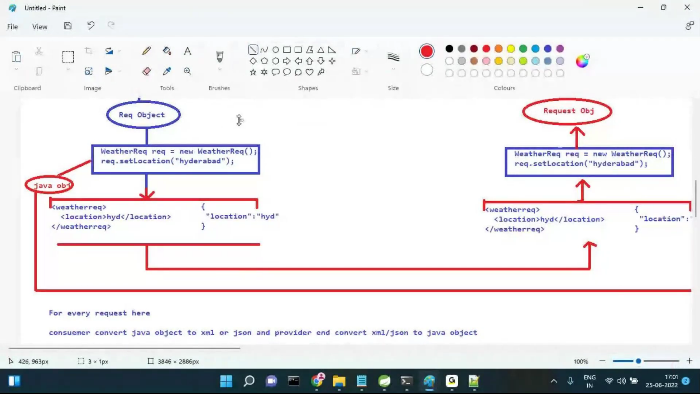


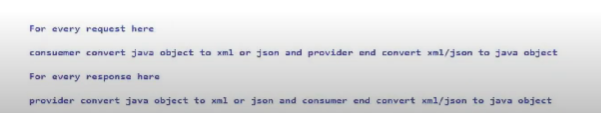


Response of flipcard in form of php bcz flipcard understands only php.









**Who will convert XML to java object and java Object to XML -- Sol : JAXB**

**Who will convert json to java object and java Object to json -- Sol : Jackson**

Who will convert XML to .net object and .net Object to XML -- Sol : .netxB

Who will convert json to .net and .net object to json -- Sol : .netCKSON

Who Will convert XML to php object and php Object to XML -- Sol + phpXB

Who will convert json to php object and php object to json Sol : phpCKSON

**Webservices:** If Producer knows some technologies and Consumer knows other technologies,the communication i.e., req send from producer is converted to XML/JSON and that is converted to consumer understandable lang .similarly response from Consumer converted to XML/JSON and that is converted to Producer understandable language.This is known as Webservices

Note : In Webservices there are 2 terminologies

1. Service provider 2. Service consumer

1. Service provider :

To develop service provider UI/presentation layer is not required, because always it is expecting

input as either xml or json

2. Service consumer : To develop service consumer may or may not requried UI/presentaion layer depends on requirement

XML:

<product>

<id>1234</id>

<name>mobile</name>

</product>

XML welformed ( follow the correct sytax)

=========================================

1. per xml only one root element

2. for start element, end element is mandatory

3. xml elements are case sensitive

4.xml data should be readable/parsable

JAXB:

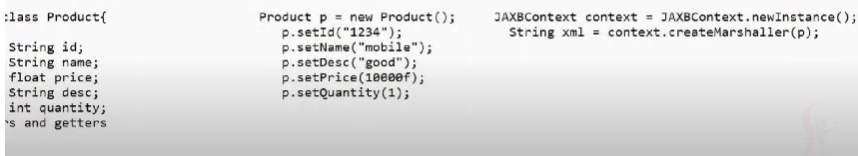
=========

**The process of converting java object to XML and XML to java object is called JAXB**

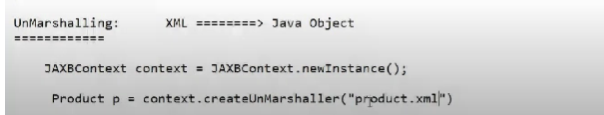
Marshalling : The process of converting java object to xml

Unmarshalling : The process of converting xml to java object

Ex: for marshalling



Ex:Unmarshalling



For every request, consumer end marshalling (java obj to xml) and provider end unmarshalling (xml to java obj)

For every response, provider end marshalling(java object to xml) and consumer end unmarshalling(xml to java obj)

Note : In rest api's development spring framework internally will be using jaxb to taking care of

marshalling and unmarshalling. i.e. developer no need to write this code manually.

JSON

==========

>JS0N stands for java script object notation

>It is derived from javascript but not depending on javascript

>Json is light weight and human readable text

>Json mime type is “application/json"

>Json file name extension with .json

>Json data is always in key value pair

Syntax : {key1:value1,key2:value2,……}

ex: {"id":”123”, “name”: ‘’mobile” , “desc”:null, “isDelivered” : true}

-->In Json key is always String, but value will be String, Number, Null, boolean(true/false), Object

-->Json data types:

1String 2. Number 3. Null 4) array 5. Object 6)Boolean(true/false)

-->json data always in curly ({})braces and array data in square brackets []

