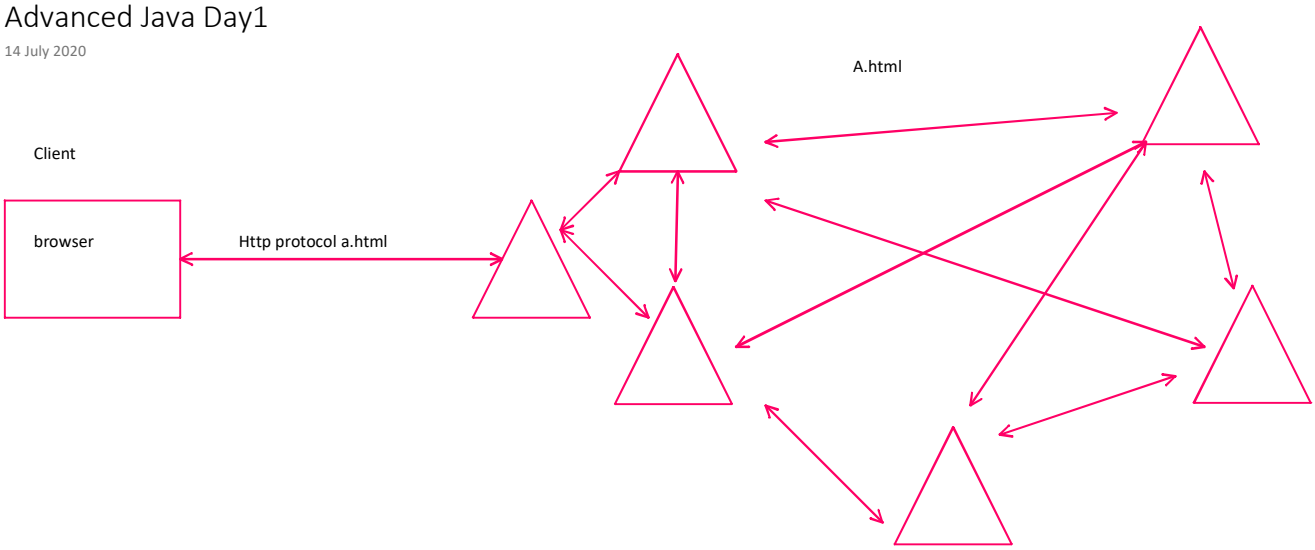
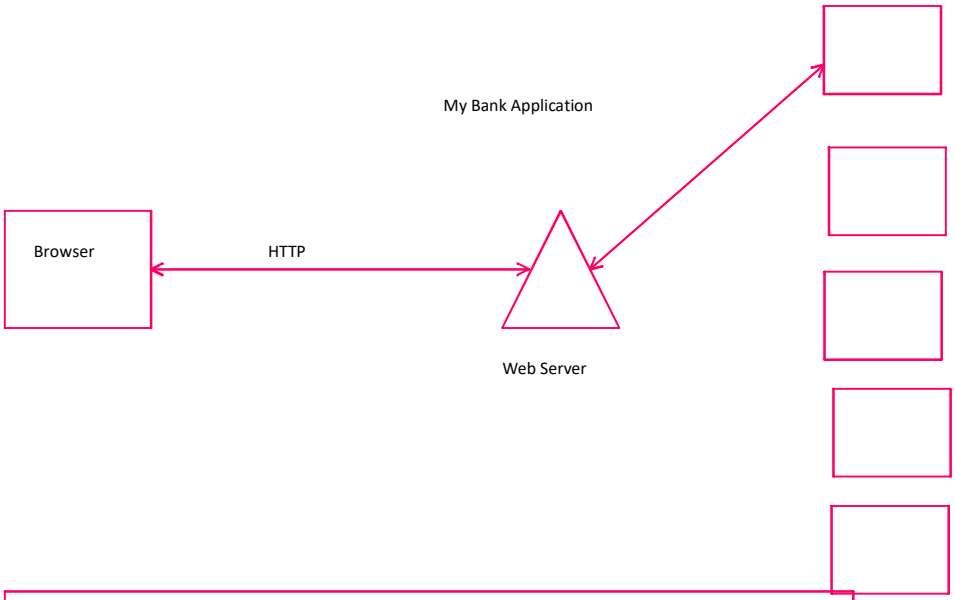
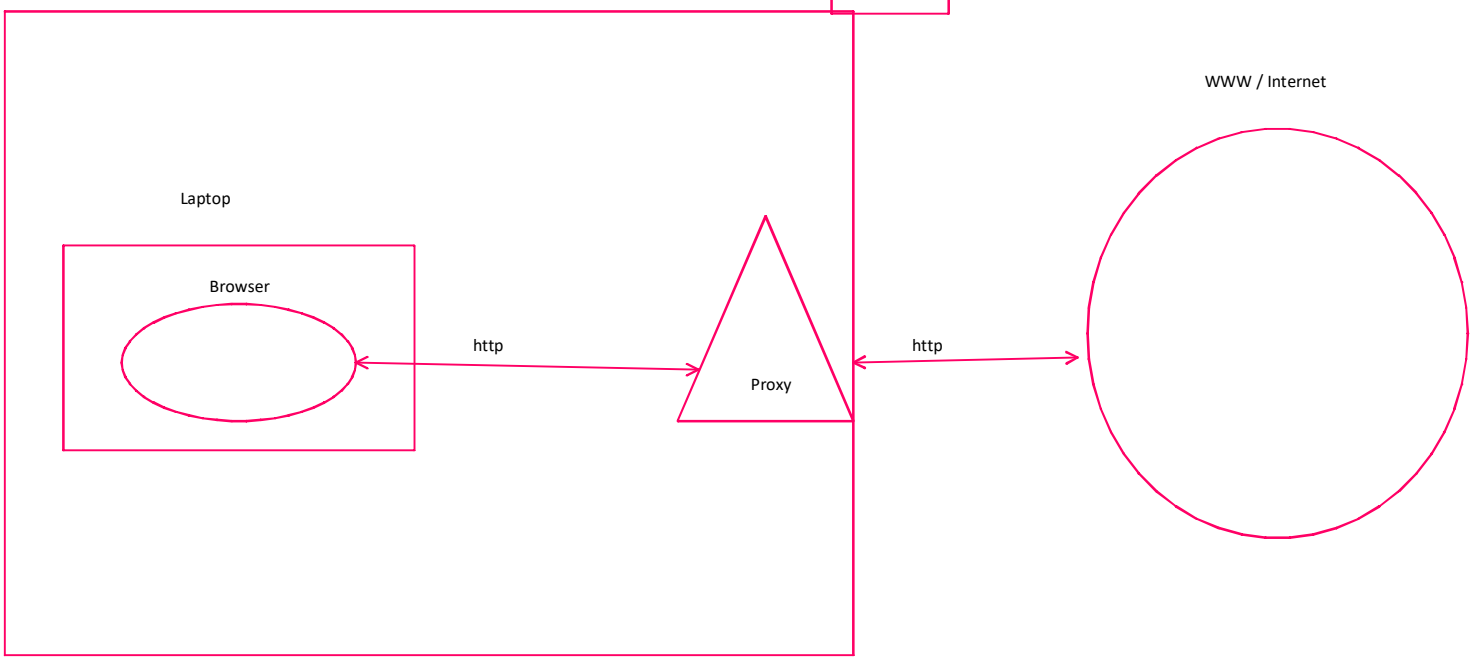
Advanced Java Day1

14 July 2020

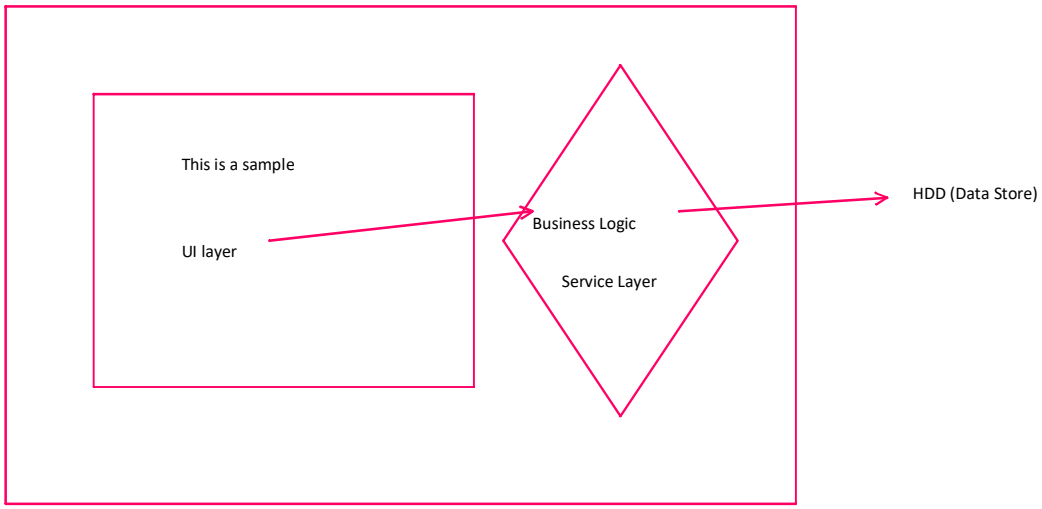


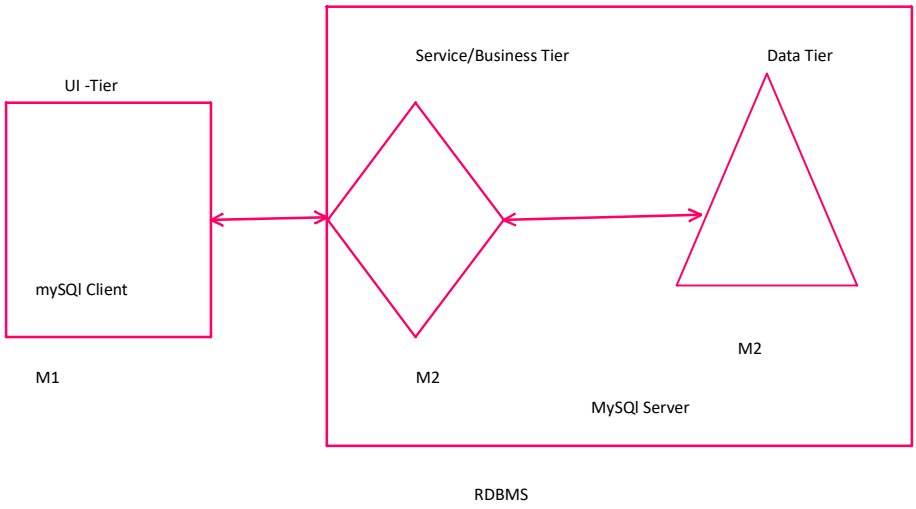
WEB of Interconnected Machines

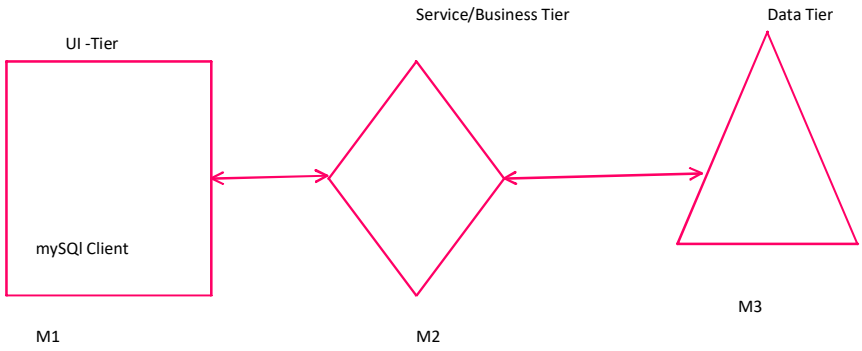


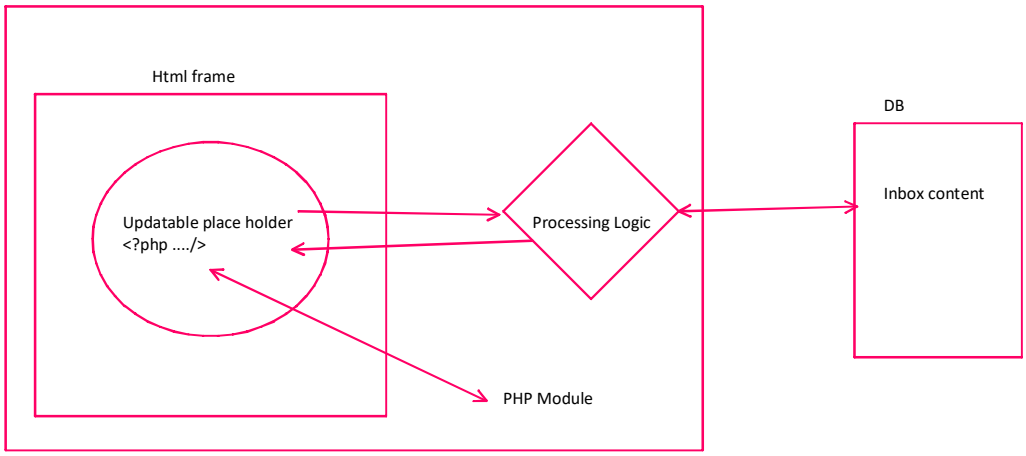


Notepad application

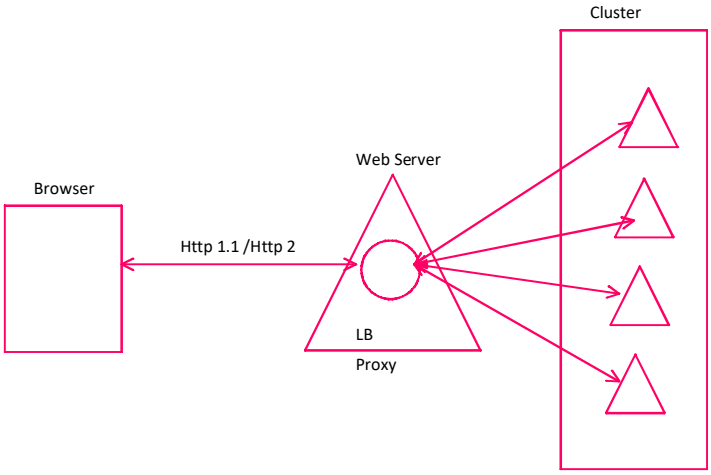

This is a sample
UI layer
Business Logic
Service Layer
HDD (Data Store)


UI -Tier
Service/Business Tier
Data Tier
mySQl Client
M1
M2
M2
MySQl Server
RDBMS


UI -Tier
Service/Business Tier
Data Tier
mySQl Client
M1
M2
M3



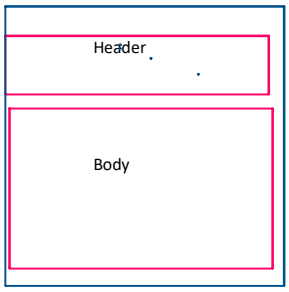
Apache httpd/Web Server



Synchronous
Asynchronous

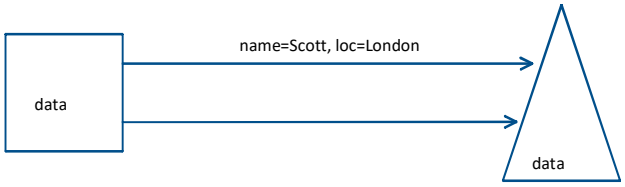
message

Http Message (Request/Response)



cache

GET http://developer.mozilla.org?name=Scott&loc=London



Get("name")-->Scott

Get("loc")-->London

Non Java Domain

1. Apache Web Server/nginx/IIS/iPlanet
2. PHP For Dynamic Content generation/Processing
3. CGI for the above purpose too

In Java Platform

1. Apache Tomcat/Undertow/Jetty
2. Servlets/JSP for Dynamic Content generation/Processing

Servlet --> Code Based approach
JSP --> Page based approach

JCP (java Community Process)

JCP Specs

Specifications

JCP (java Community Process)

Specifications

JCP Specs

API
(Application Programming Interface)

SPI
(Service Provider's interface)

Container: Runtime (where your specific type of applications run)-->

Gold fish, cat fish, all other sweet water fishes
Can you keep Shark in the same Box?

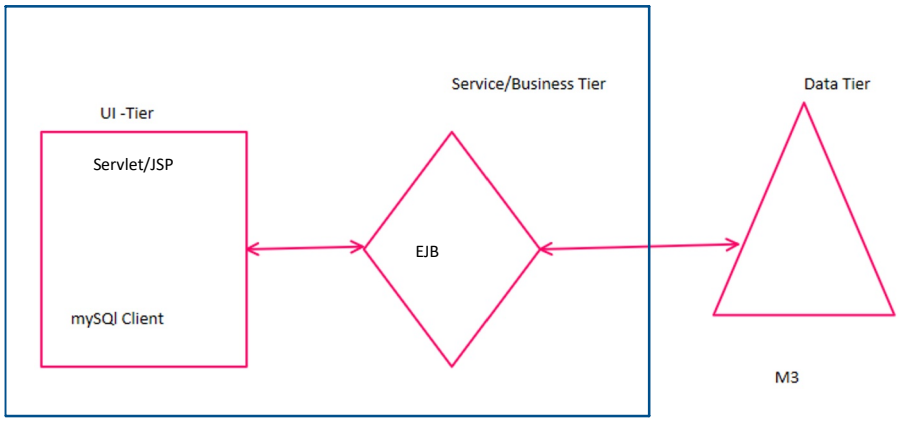
Enclosure for all Types
You need appropriate ENV for each type of Fish

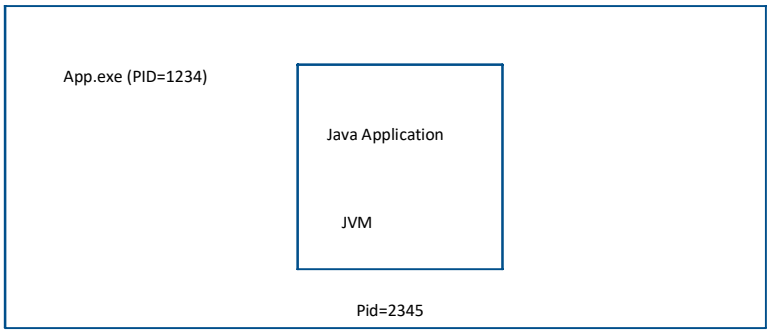
Container --> Enclosure+ENV

In java, You have various types of Application Components

You need JVM for all Types
You need component wise ENV
Runtime: JVM + component wise ENV--> Container

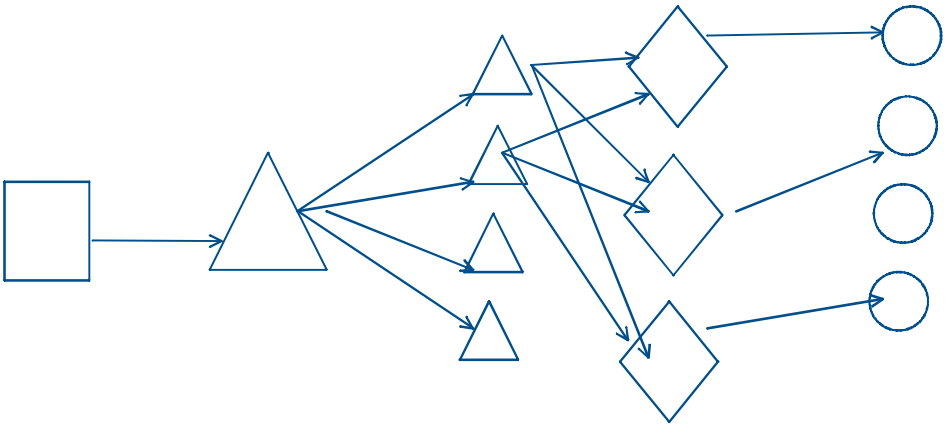
Main Based Applications--> Application Container (JRE-->JVM+java runtime)
Web Components -> Web Container (JRE+Servlet+JSP Runtime)
Applet--> Applet Container(Browser+JRE)
EJB -> EJB Container (JRE+EJB Runtime)



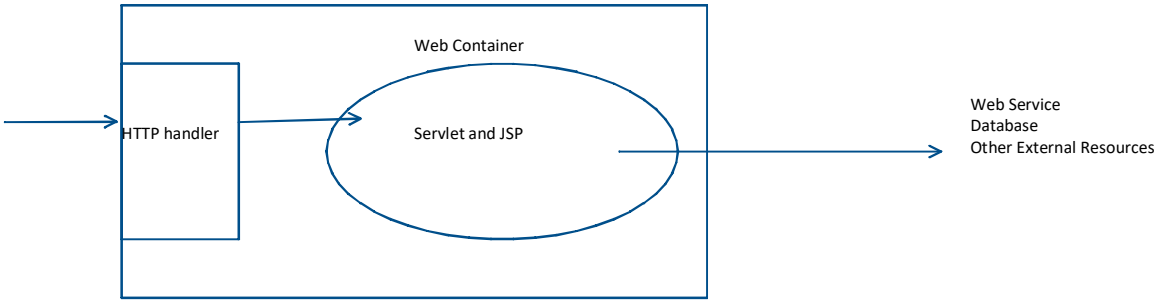


Java EE Framework (reusable software design, customisable)
Used to create Enterprise applications

It has support for UI Tier, Business Tier and Data Tier



Apache Tomcat (Web Server, with embedded Web Container) /Jetty/Undertow etc



Can Serve Web Applications created using Servlet and JSP
Part implementation of Java EE Specs (Web Profile)

Web Applications in Java are deployed as .war file e.g. MyApp.war

MyApp (Context Root)

WEB-INF
- classes
- lib
- Web.xml

Create the WAR Structure

- Code the components and place them in respective locations
- Servlets are kept in /WEB-INF/classes
- JSP/HTML/IMAGE etc are kept under context root or in sub folders
- Create the .war file using java 'jar' command
- Copy the .war file to deploy location of the Web Server
- Start the Web Server
- Use the appropriate URL to access the application
- http://server:port/ContextRoot/resource
- E.g. http://localhost:8080/MyApp/demo.html

Prerequisites:

1. Make sure you have installed Java and set JAVA_HOME and Path
2. Make sure you extracted Apache Tomcat in a convenient Location
3. And it starts....

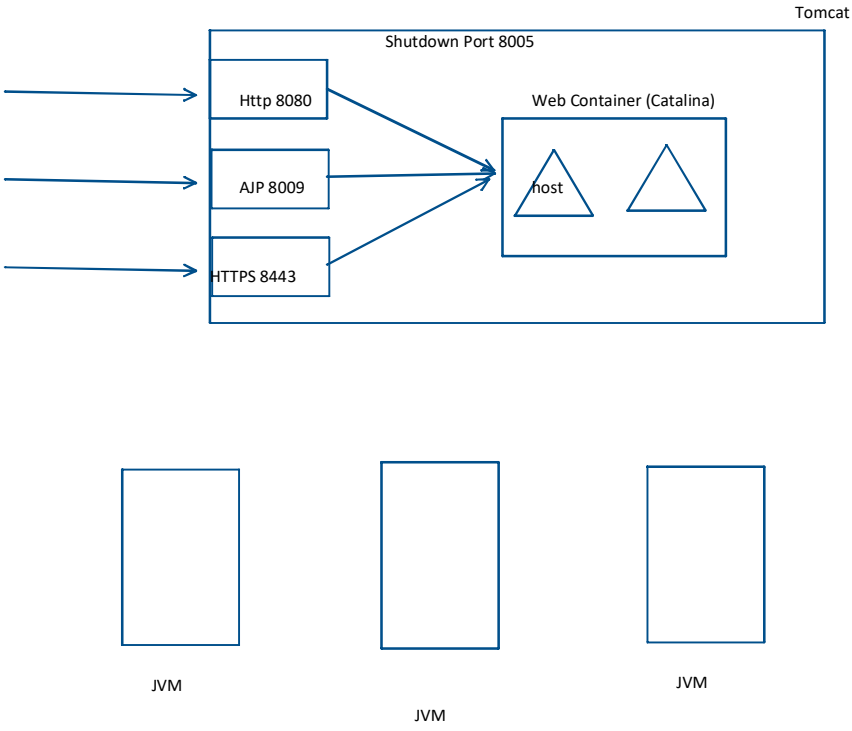
How to create the .war

1. Move inside MyApp Folder
2. Jar -cvf MyApp.war .

Task -- Make a 6 seater oval Dinning Table

1. Automated Machine
2. Manual Labourer
3. .....

Apache Tomcat Web Server Config Part



Shutdown.bat

XML Parser

Tomcat Startup

1. JMV starts
2. XML Parser parses the config files and converts the values to java objects/properties
3. The Server comes to ready status

Adv Java Day3

17 July 2020 09:10

1. JSP compilation

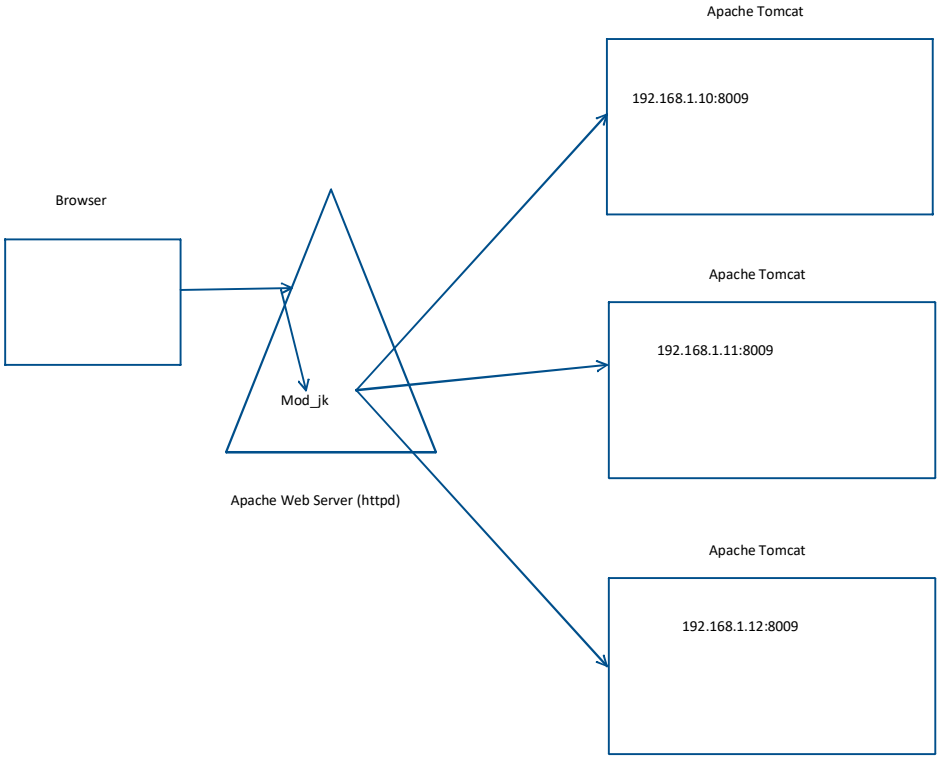
JSP (which Contains html) -->Web Container/JSP Engine converts the .jsp to Servlet--> Generates the Response to USER

GET/POST/PUT/DELETE ---> http Client

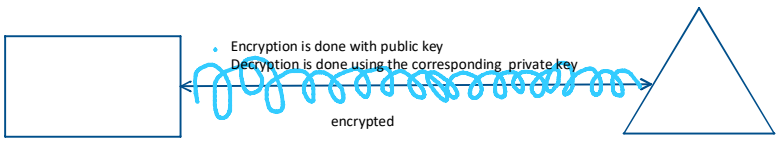
1. Browser (GET and POST using a Form)
2. Test clients like Postman to fire all http methods
3. Curl for the above tasks
4. You can write java code for http clients!!!
    a. Commons http client
    b. Java.net package
    c. Other 3rd party API like spring mvc, Jersey

Jakarta Tomcat(Catalina)-->given to Apache Foundation---> Apache Tomcat (Catalina)

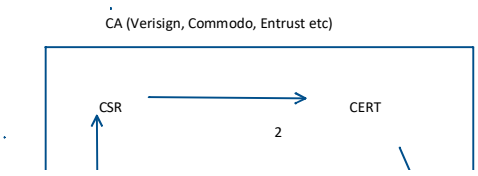
AJP--> Apache JServe Protocol



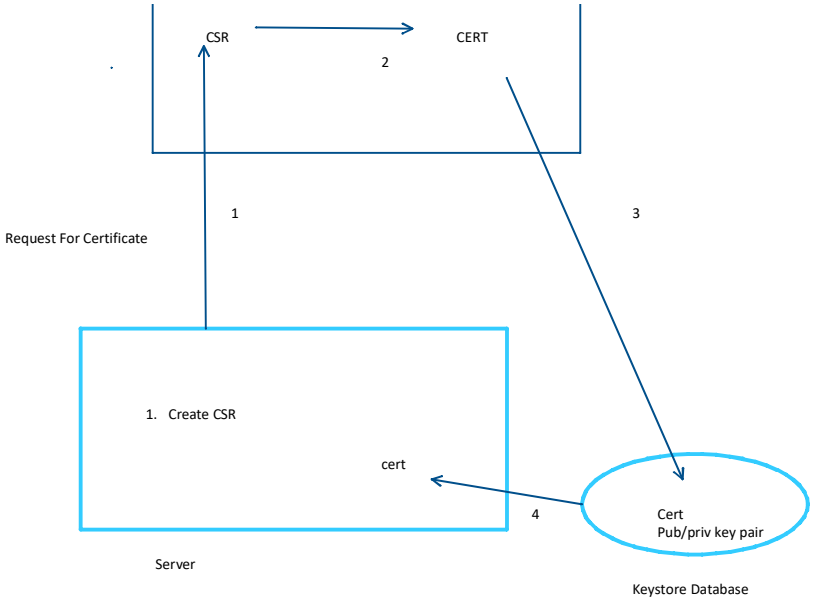
SSL (HTTPS)



SSL uses a combination of public /private key
Public key is distributed to clients and is used to encrypt the message.
The private key is not distributable, it is used to decrypt the message which has been encrypted by the corresponding public key
The certificate is sent to the client along with the public key



1. Your Server needs a digital certificate to prove its valid identity (CA Cert)
2. So, You apply to a CA (Certifying Authority) with a CSR (Certificate Signing Request)
3. CA ISSUES a certificate and you import the same to your key store
4. You configure this keystore to be used by your server
5. Now the server is ready to support communication using SSL (HTTPS)



Self Signed Certificate
(You are the CA and provide guarantee)

Usage of Keytool to create Self Signed Certificate and covert the keystore to pkcs12 format

1. Keytool -genkey -alias mykey -keystore mykeys.jks -keyalg RSA -validity 365
2. keytool -importkeystore -srckeystore mykeys.jks -destkeystore mykeys.jks -deststoretype pkcs12

After you have created the certificate and the keystore, open tomcat/conf/server.xml and locate connector for port 8443 and uncomment it. The configuration should match the one given below:

```
<Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
        maxThreads="150" SSLEnabled="true" secure="true" sslProtocol="TLS" keystoreFile="c:/ssl-temp/mykeys.jks"
keystorePass="welcome1">

  </Connector>
```

Save the file after you have updated the configuration and restart apache tomcat. Test the certificate using

https://localhost:8443/

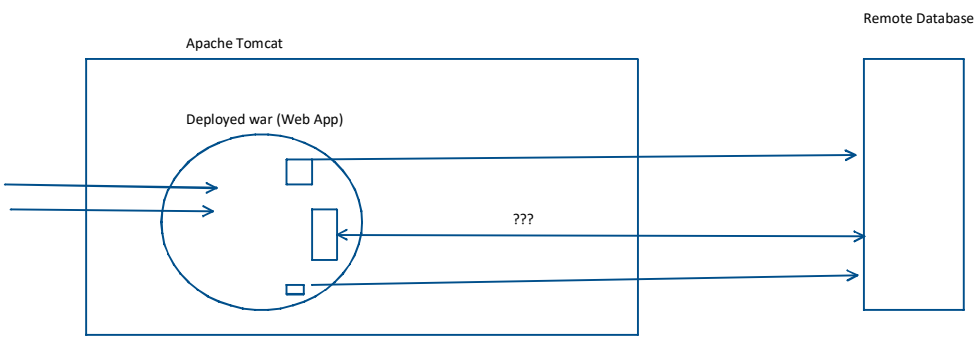
The browser will raise warning, indicates the certificate installation is successful.

Automatic Redirection to port 8443 even if you access http port

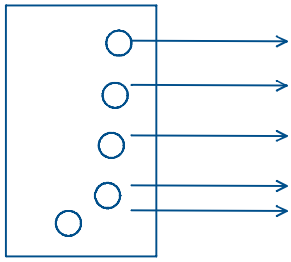
Add the following snippet to you web applications web.xml

```
<security-constraint>
<web-resource-collection>
<web-resource-name>App</web-resource-name>
<url-pattern>/*</url-pattern>
</web-resource-collection>
<user-data-constraint>
<transport-guarantee>CONFIDENTIAL</transport-guarantee>
</user-data-constraint>
</security-constraint>
```

Database connection with Apache Tomcat



1. We use JDBC
2. Use connection Pool (Data Source)
    a. Collection of pre initialized reusable Connection Objects



Connection Pool

JNDI (Java Naming and Directory Interface)

Data Source on Apache Tomcat

1. To Create a Datasource in Apache Tomcat, we need
    a. Configure the Database and its JDBC Driver
    b. Configuration for Apache Tomcat

Step1: initialize database with the given data
Run MYSQL database
Log into the MYSQL Client
C:\> mysql -u <user> -p

Mysql> create database demo;
>use demo;

- Paste the content from the given sample .sql file.

Step2: install the mysql jdbc driver in tomcat/lib folder (copy the driver .jar here)

Step 3: add the following snippet to tomcat/conf/context.xml to look as follows:

```
<Context >
<Resource name="jdbc/ds" auth="Container"
        type="javax.sql.DataSource" username="root" password="root"
        driverClassName="com.mysql.jdbc.Driver"
        url="jdbc:mysql://localhost:3306/demo"
        maxTotal="8"
        />
<!--You have some more lines here-->
</Context>
```

Step4: Deploy the given Application to Apache Tomcat
Copy the given sample TestDataSource.war to tomcat/webapps/

Step5: restart Apache Tomcat and access the application using the following url:
http://host:port/TestDataSource/EmpInfo.html

What we did today

Tomcat Architecture
SSL on Tomcat
Database/DataSource on Tomcat with Testing
HTML tags

Adv Java Day4

18 July 2020 10:05

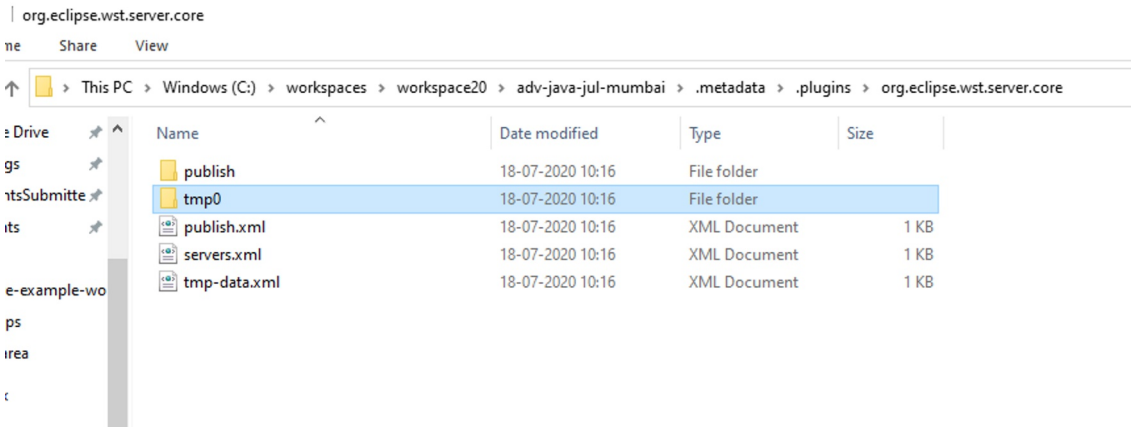
Deploy Applications from Eclipse/STS to Apache Tomcat

2 ways of deployment
1. Export the project as war file and deploy the war file (this is done in prod deployment)
2. Integrate Apache Tomcat with Eclipse/STS and deploy the app from IDE (dev deployment)

Integration of Apache Tomcat With Eclipse JEE/STS

The integrated Tomcat does not run from the install location. It is created in your workspace and the server runs from workspace.

Default Location of Apache Tomcat in workspace:



Hands On: external stylesheet

Header : "Customer Details" with solid border 2px and blue color, text colour white, background dark blue

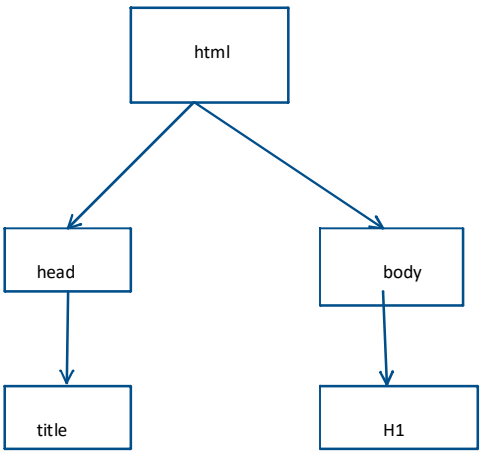
Table : headers : custId, Cust Name, Country, Contact and some dummy data (6 rows), header back ground green, padding 10px, border solid orange, each even row with light gray background

Entire content at the centre of the screen

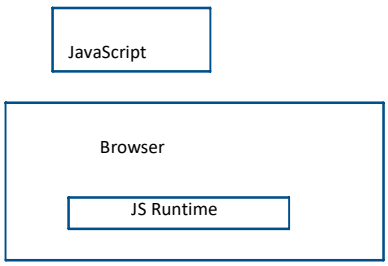
Done!

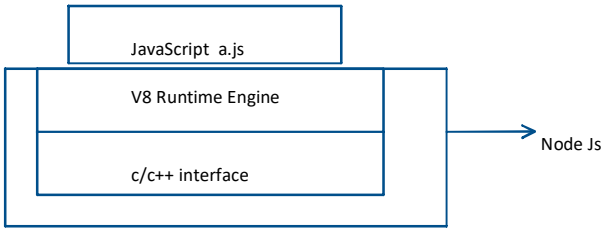
Java Script
What is DOM?

```
html
<html>
<head>
	<title>Demo Page</title>
</head>
<body>
	<h1>This is a Demo page</h1>
</body>
</htm>
```



1. **JavaScript was created for html**


JavaScript
Browser
JS Runtime


JavaScript a.js
V8 Runtime Engine
c/c++ interface
Node Js

Adv Java Day 5

20 July 2020 08:32

Recap of Previous topics

1. How Web Works
2. Web Server
3. Static and Dynamic Page and How they are created
4. Servlet/JSP
5. Http Protocol (header, body, methods and response status codes)
6. Apache Tomcat Configuration
7. SSL on Tomcat
8. DataSource and how it is created on Tomcat
9. How to Use Eclipse IDE, Integrate Apache Tomcat with eclipse/STS
10. Create Dynamic Web Project
11. Html, CSS
12. JavaScript

Now what I expect from You:

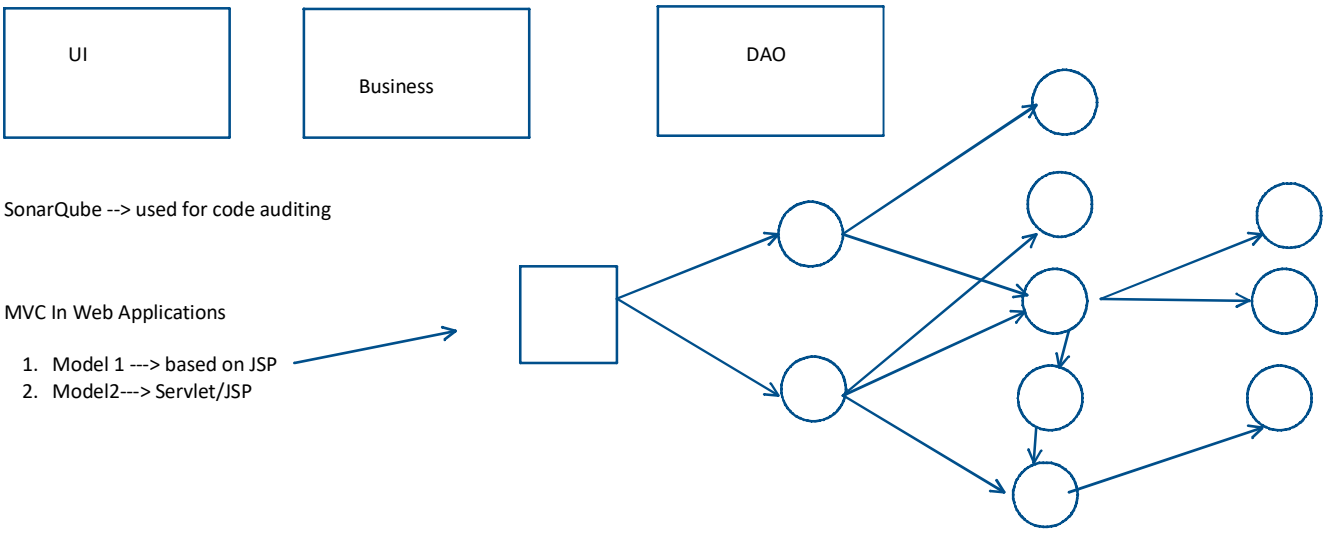
Knowledge of

1. **Core Java Basics**
2. **Exception Handling**
3. **Collections (Set, List, Map at least)**
4. **Concepts of Thread**

Layered Design:

1. UI or Interaction Layer --> User Interacts with this Layer
2. Business Layer --> Core Task is done here e.g. Access data from database, calculate TAX etc
3. Data layer--> The code or Strategy to access data from external Data Source

GOF Design pattern

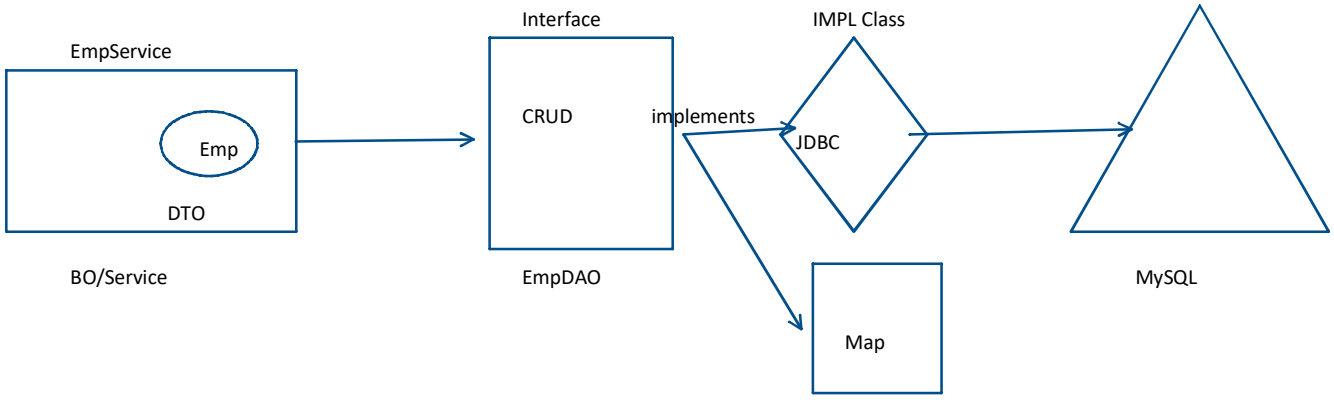


SonarQube --> used for code auditing

MVC In Web Applications

1. Model 1 ---> based on JSP
2. Model2---> Servlet/JSP

DAO (Example)



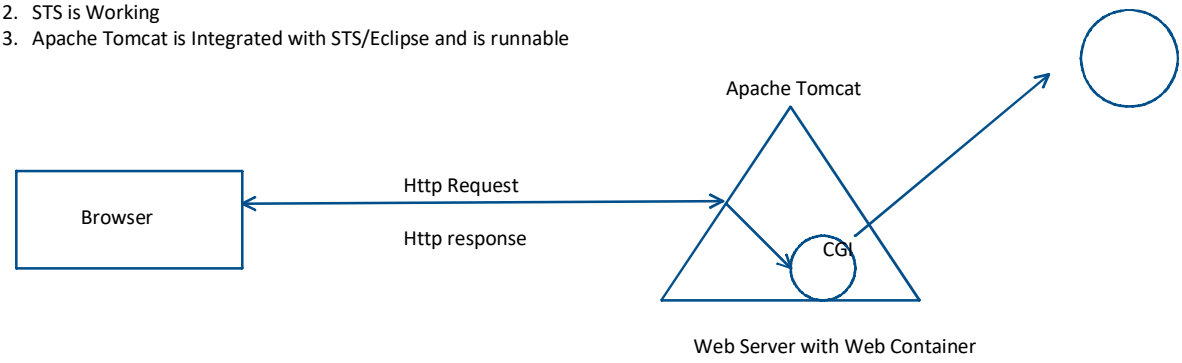
Hands On:

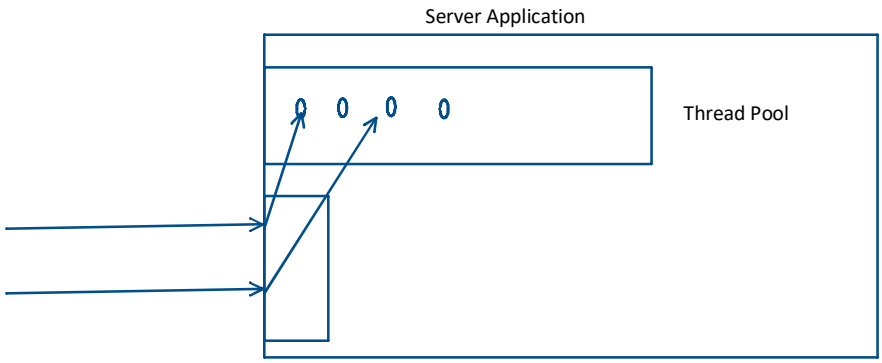
1. Implement a Map based Dao with Service or Business Object for a Student with the given fields. Dao should have CRUD and List methods.

Student details:
rollNo:int (unique Id)
studentName:String
Course:String

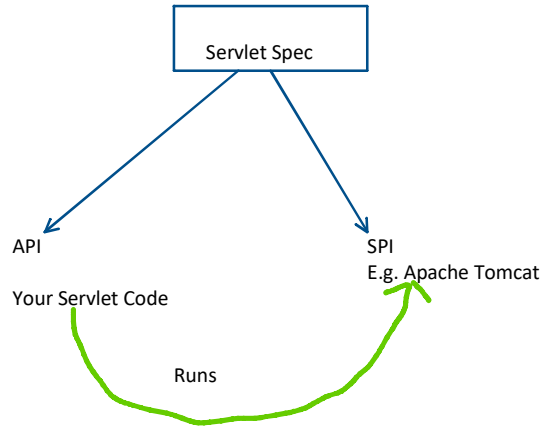
For Servlets, Please make sure

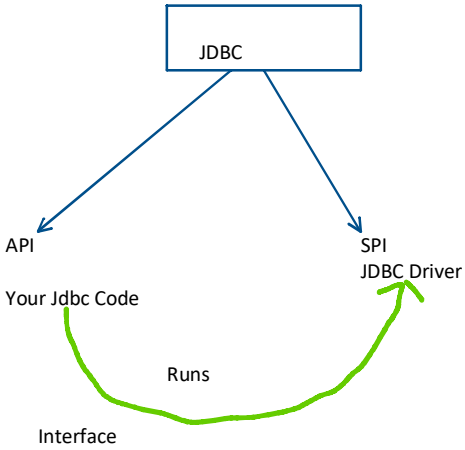
1. Java path is set
2. STS is Working
3. Apache Tomcat is Integrated with STS/Eclipse and is runnable





Servlet Specs

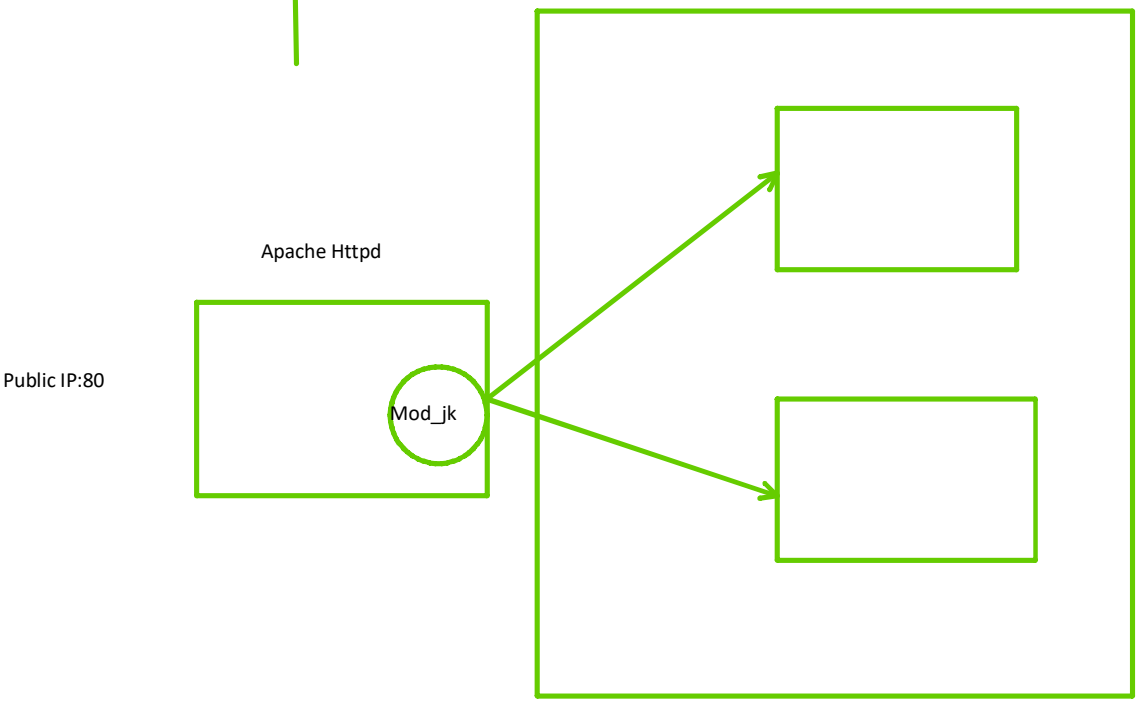




Interface

Connection conn=DriverManager.getConnection();

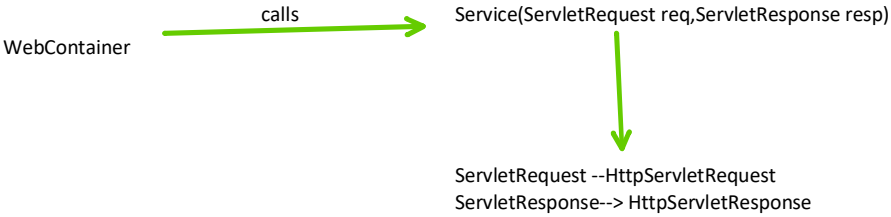
Web Module : 2.2,2.3,2.4,2.5, 3.0,3.1,3.2,4



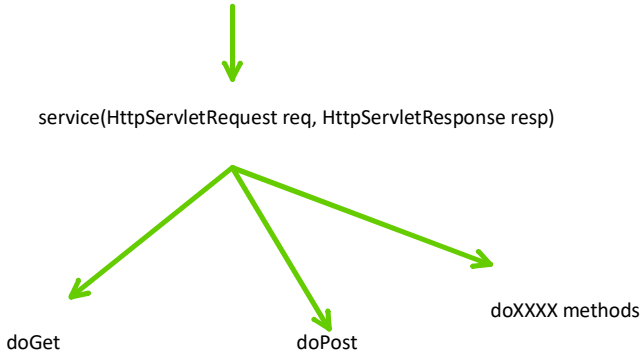
HttpServlet

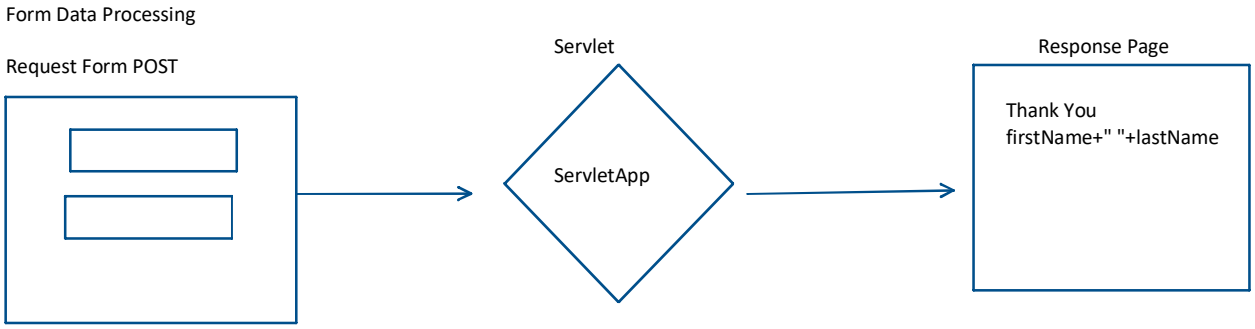
Http GET -----> In Http Servlet it call doGet

Http POST -----> In Http Servlet it call doPost



ServletRequest --HttpServletRequest
ServletResponse--> HttpServletResponse





Add 2 numbers and display the result on browser

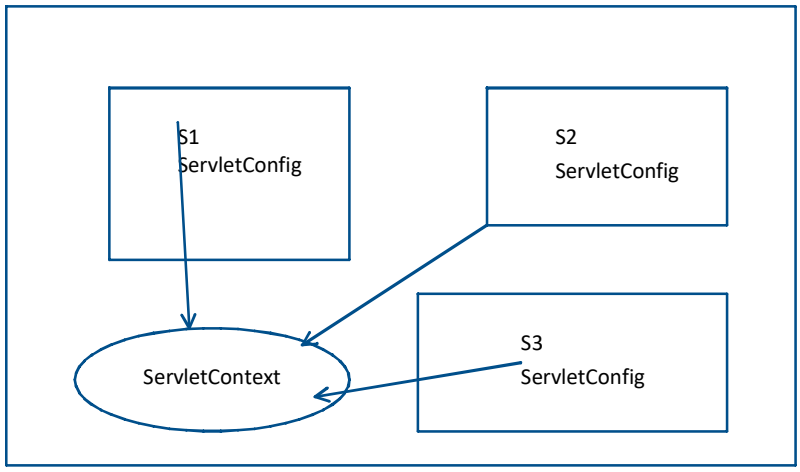
Adv Java Day 6

21 July 2020 09:36

1. Create an html Form to submit the numbers
2. Write Servlet Life Cycle methods init and destroy in the same servlet and show me
3. Write SOP in init and destroy so that we can see the output in the server console.āāĀ

ServletConfig --> one ServletConfig Object Per Servlet
ServletContext --> One Servlet Context per Web Application

web.xml

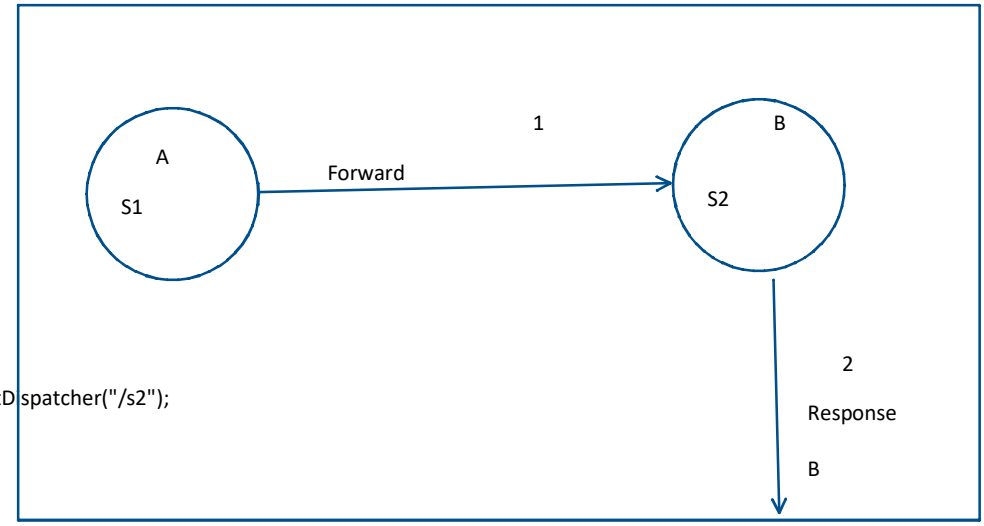


1. Create A New Dynamic Web Project
2. Add 2 Servlets S1 and S2
3. S1 reads param from web.xml and displays it in response
4. S2 reads param from web.xml and displays it in response

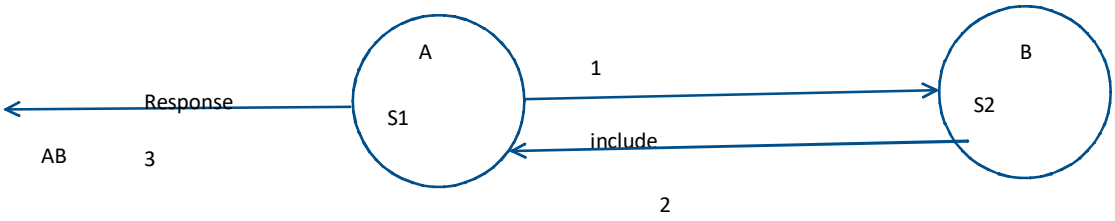
"<h1>Value from S1 Servlet Config is" +value+"</h1>"

RequestDispatcher

forward()
Incluсde()



RequestDispatcher rd=request.getRequestDispatcher("/s2");
rd.forward(req,resp);



Scoped Objects

ServletRequest			

HttpSession
ServletContext

setAttribute("name",Object);

getAttribute("name")

Response.sendRedirect("page/resource address");

S1---forwards--s2 → http://www.google.co.in

Response.html

S1-->Req.setAttribute("server","google");

S2--->
If(server.equals("google"){
resp.sendRedirect("http://www.google.co.in");
}else{

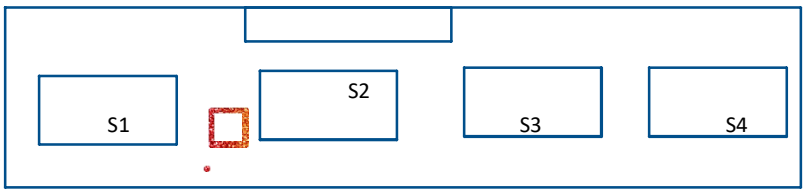
resp.sendRedirect("respponse.html");

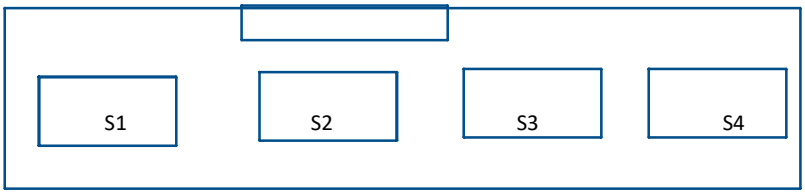
}

HttpSession --> On Server Machine

Cookie --> Client Machine

Session.setAttribute("name",Object)





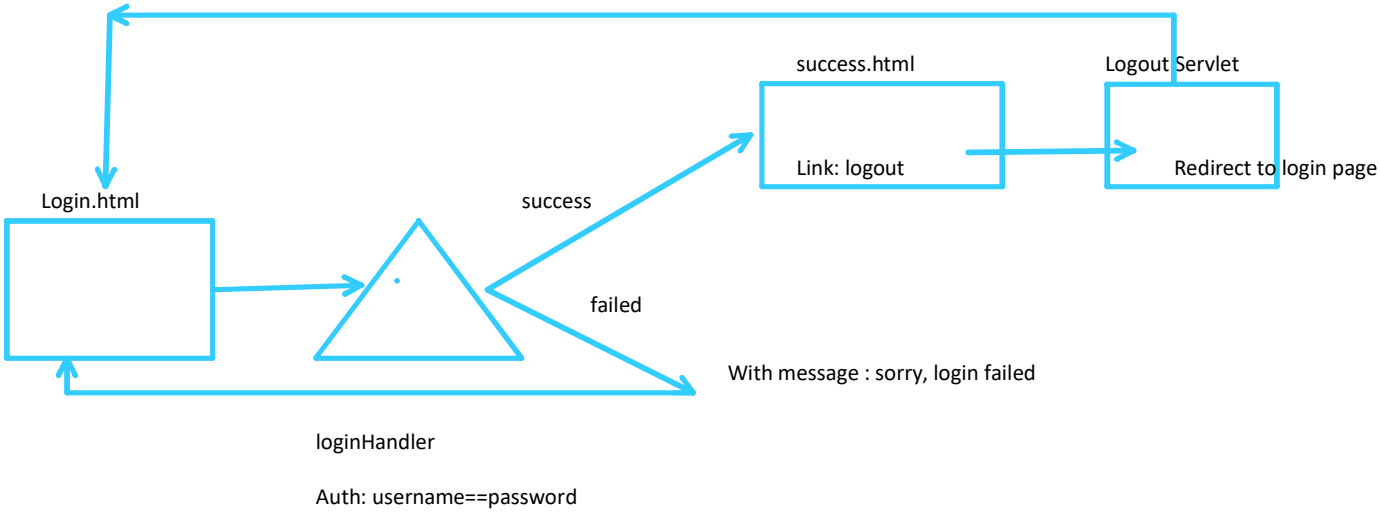
HttpSession session=req.getSession(true);
Session.setAttribut("itemName","itemValue");

Till now we learnt (15:00 hrs)

ServletConfig and its configuration in web.xml
ServletContext and its configuration in web.xml

RequestDispatcher (forward, include)
Response.sendRedirect
HttpSession
Inter Servlet Communication using request.setAttribute

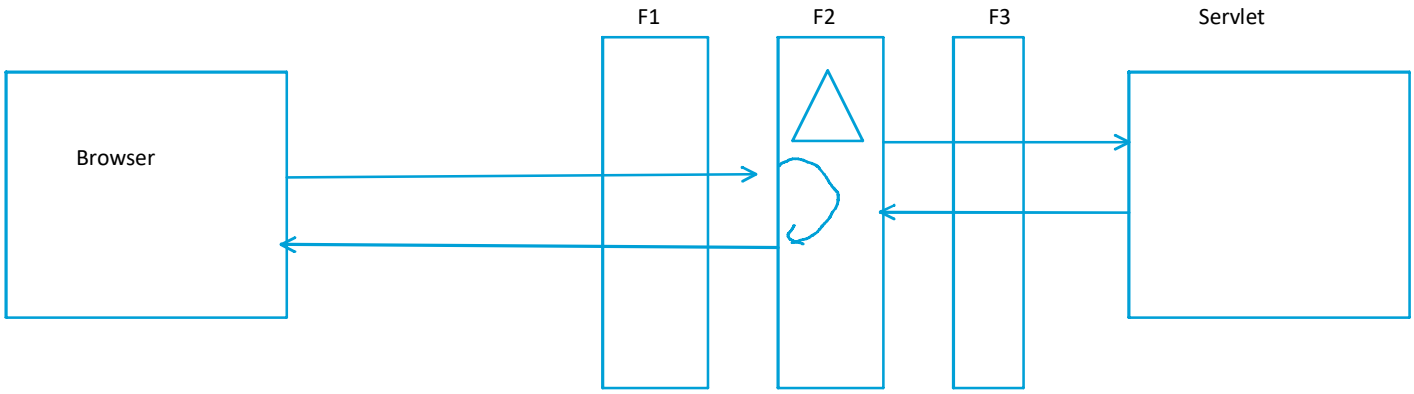
Let's Create Login Application:



Time allotted : 45-50mins

Adv Java Day 7

22 July 2020 09:33



Class LoggingFilter implements Filter{

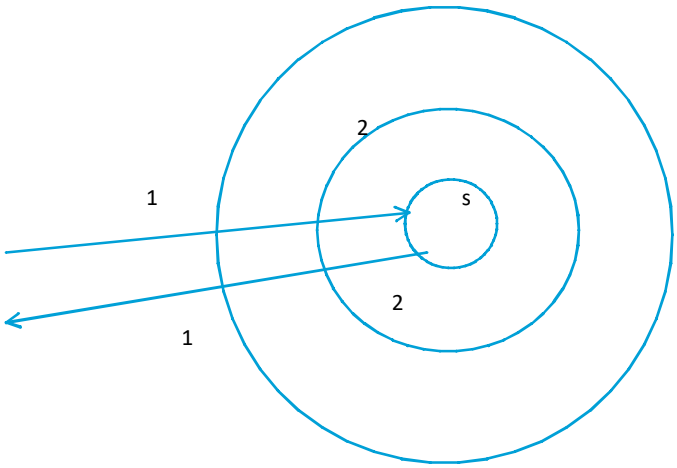
Public void doFilter(...,...){

filterChain.doFilter();
}

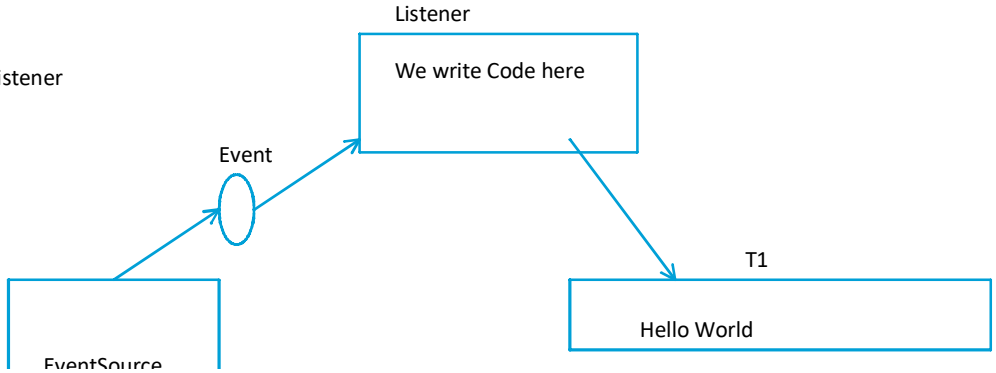
}

/demo/add

Hands on: Create a Filter and configure to intercept the requests to the already created/deployed servlet(s).



Event and EventListener



Hands on for Servlet Listener

1. Create a new Dynamic Web Project
2. Create a Simple HttpServlet with doGet method
3. Create an HttpSessionListener impl
4. Create an HttpSessionAttributeListener
5. Inside the servlet create a session, add some attributes remove attributes and the invalidate the session
6. Check the console log for Sysout you wrote in Listeners.

@Override

A.jsp

```
<body>
    <h1><%="Welcome To JSP"%></h1>
    <%!String message="Welcome To Tomcat Server";%>
    <h1><%out.println(message); %></h1>
</body>
```

```
class A_jsp.java {

String message="Welcome To Tomcat

public int add(int a, int b){

return (a+b);
}


_jspService(....){

out.write(Welcome To JSP);

out.println(message);

Out.write(add(2,4));

}



}
```

Create JSP

Header: Welcome to JSP

Define a method

```
public int add(int a, int b){

return (a+b);
}
```

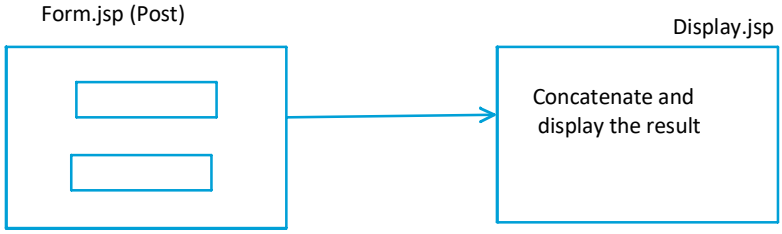
Method called from expression tag and Scriptlets

Declaration Tag: <%! Variable/methods %>
Expression Tag: <%=.....%>
Scriptlets : <% %>

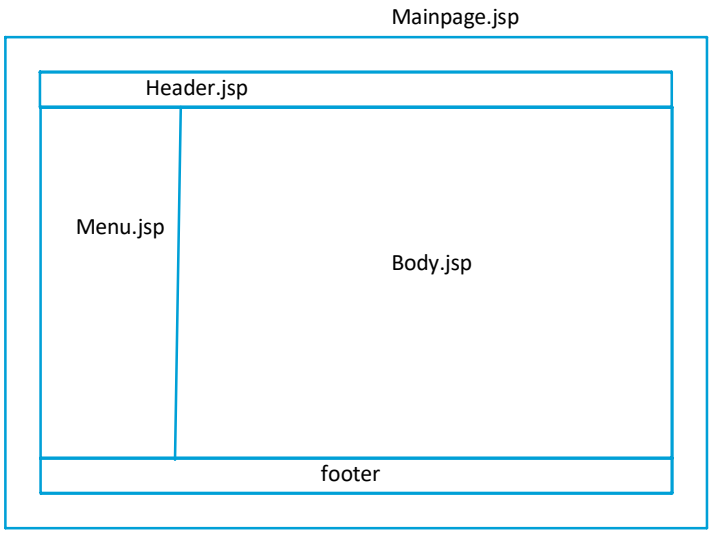
```
<%=add(2,4)%>
```

<h1>Hello User </h1> 10 times using a for loop

Form.jsp (Post)



request.getParameter("param");

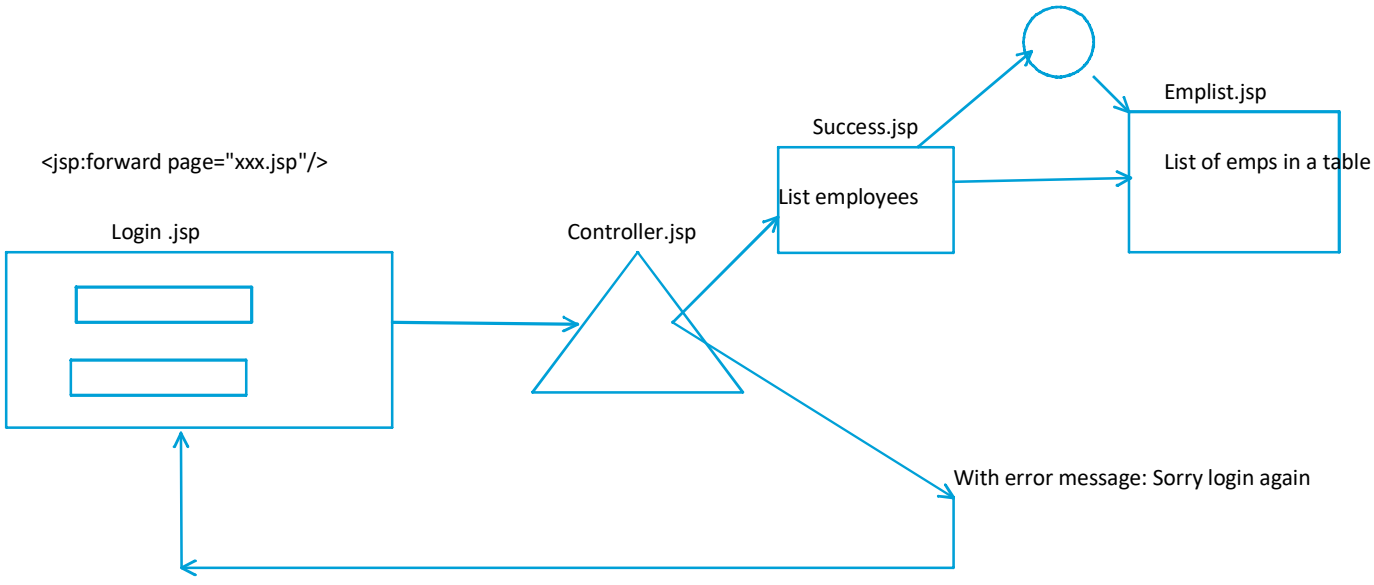


<%@ includcde file="demo.html"%>

<jsp:include page="demo.hdml"/>

<jsp:forward page="demo.hdml"/>

Directives
Scripting Elements
JSP Actions



Create the HttpSessionApp again, but this time use only JSP
Create a front login Page and display logged in user name in all navigated pages.

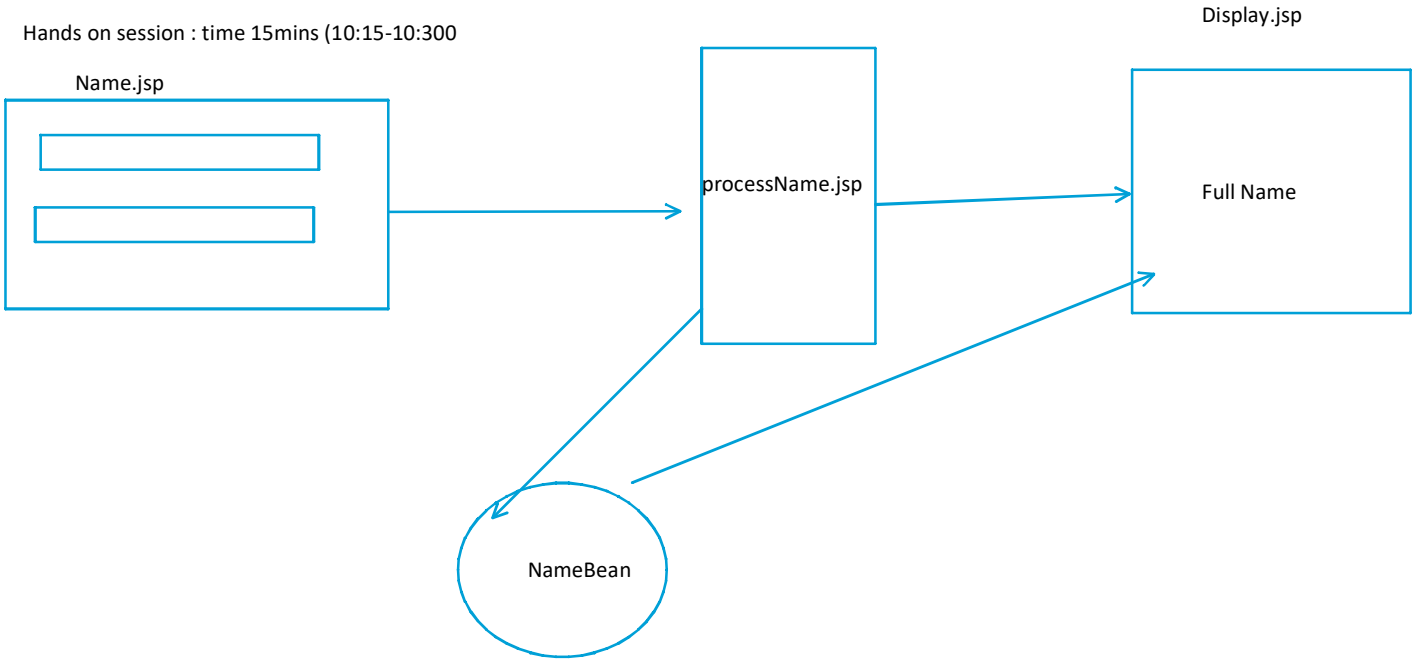
Login page, 3x shopping items page, 1 checkout page

Adv Java Day 8

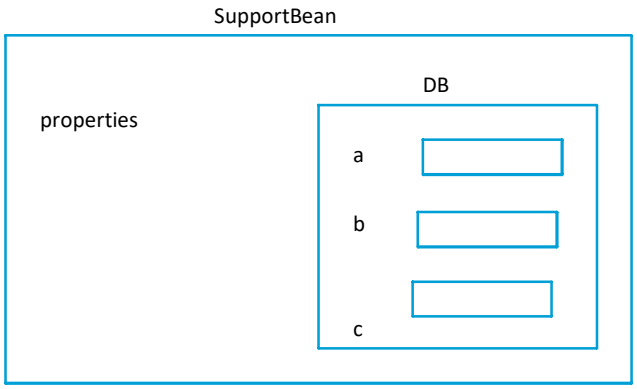
24 July 2020 09:43

1. JSP with Java Beans

Hands on session : time 15mins (10:15-10:300



<jsp:setProperty property="*" name="x"/> -----> name of Java Beans Property==name of input field



Hands On : JSP Based application : TechSupport (1.5Hrs)

Servlets/JSP EMP Crud based application Use Web Module 2.5 and mysql

1. Enterprise Application (J2EE/JEE)
2. Framework
3. Declarative Programming
4. DI and how it is done in Spring Framework
5. Spring Application Context or DI Container

J2EE

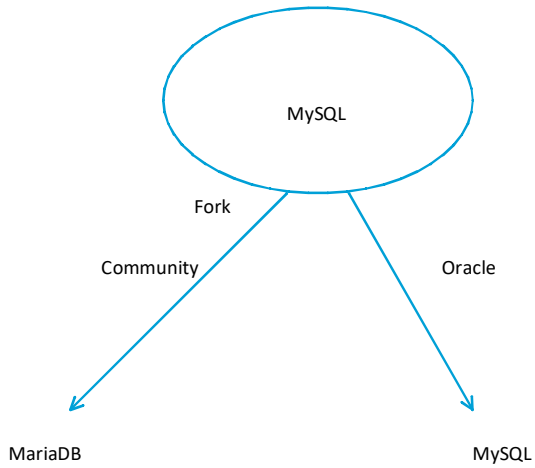
Declarative Programming
Robust Components
Declarative Security

```
class  Emp{

private Address address;
private Name name;

Public Emp(Address address, Name name){
------
-----
}

Public Emp(){}

Setter for Name and Address
Getter for Name and Address
}


Address addr= new Address();

3 line
Name name=new Name();
2 lines


Emp e= new Emp(addr, name);


Emp e2= new Emp();
E2.setAdrress(addr);
E2.setName(name);
```

Adv Java Day 9

27 July 2020 08:28

1. Database Setup --Done
2. Maven -->
3. Spring Framework
    a. Spring Core
    b. Spring jdbc
    c. Spring MVC
    d. Spring REST



What is Maven?

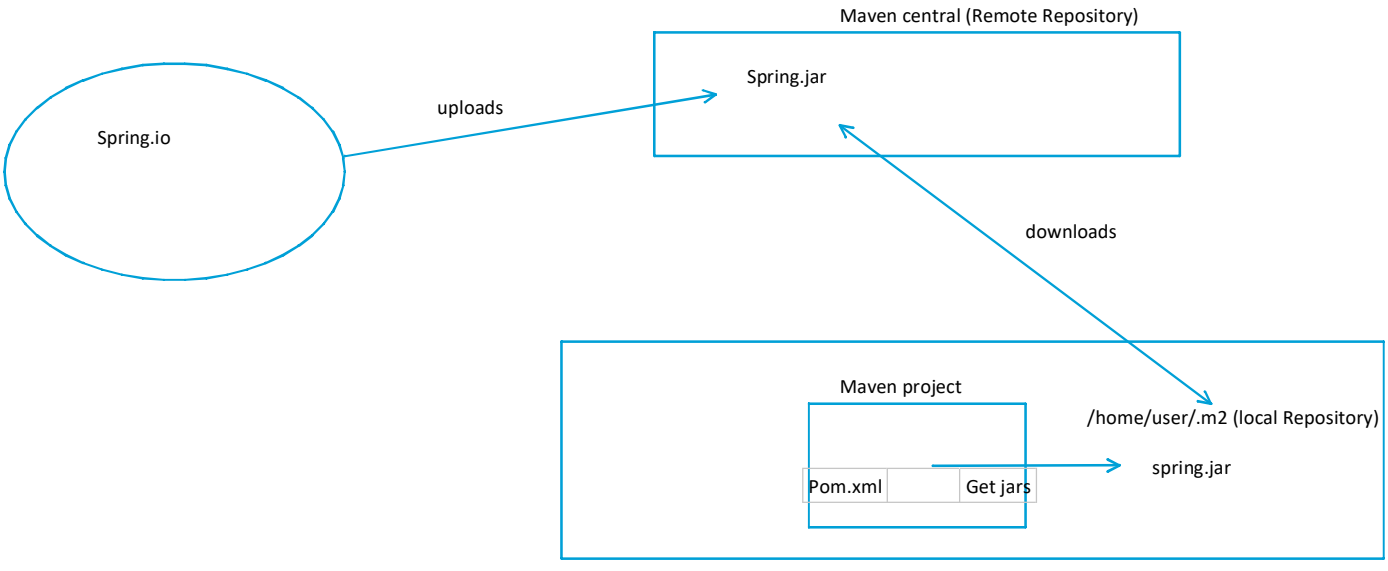
Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

From <https://maven.apache.org/>

Maven Project Structure

App-Root
    src
        Main
            Java
                Your packages for java class
            Resources
                Your config file (classpath)
        Test
            Java
            Resources

    Pom.xml



Your Laptop with Maven installed

Maven Co ordinates ---> Unique identifiers for .jar files

groupId:artifactId:version

groupId: company/project domain/subdomain (generally)

artifactId: project Id
Version: your release version

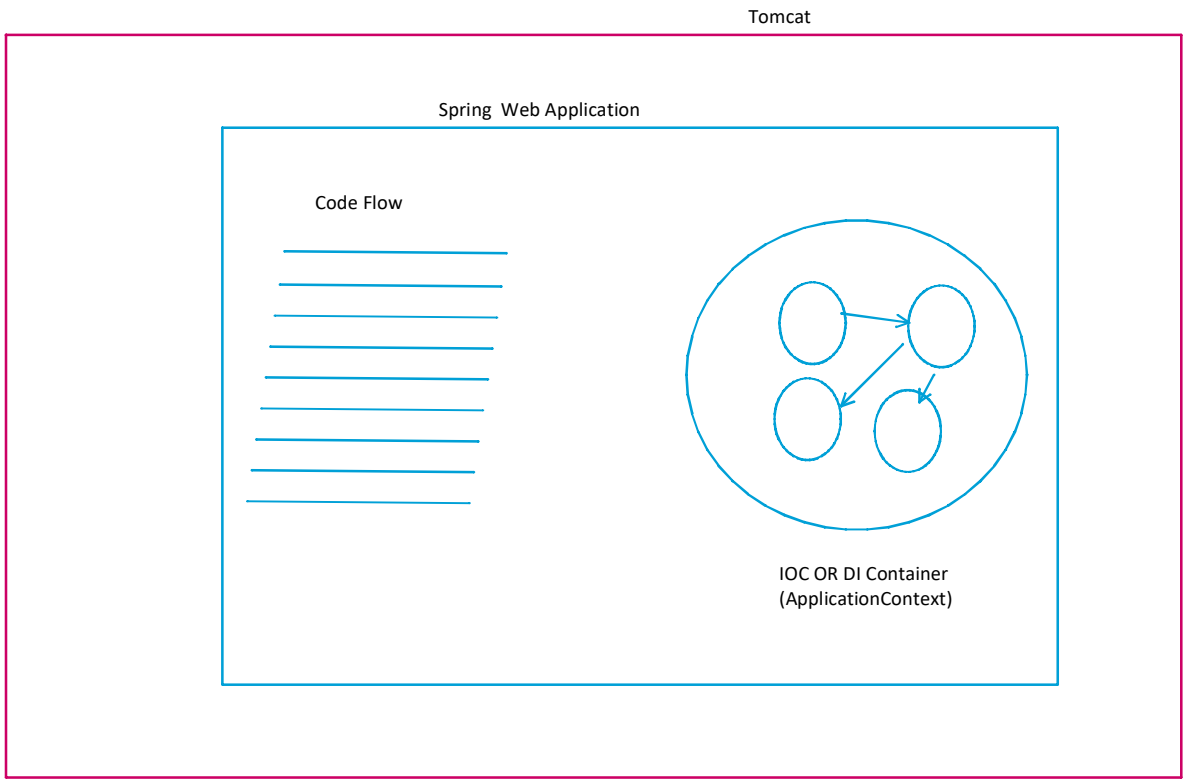
Demo: Create a war with a simple Servlet using maven in eclipse

1. Create a Maven project with war as package
2. Manage dependency for project using POM file (pom.xml)
3. Build the .war file
4. Deploy it in Apache Tomcat

Maven has GOALS (task, what maven should do?)

Compile
Clean
Package
Install
test

Hands On : Create a Simple Maven project with 1 Servlet and generate the .war file



Task : (using Maven Project)
Aim: Dependency Injection (No Spring)

1. Create a class Message
   class Message {
   private String header;
   private String body;

   }

2. Create a class

class Mail{
private String toAddress;

private String fromAddress;
private Message message;

}

Create an Application with Mail populated with all the attributes and print the mail message using mail Object.

<packaging>jar</packaging>

Hands On : create the following in Spring using xml configuration

class Trainer{
private String name;
private Participant participant;

}

class Participant{
private String name;
private String city;
private LunchBox lunchBox;

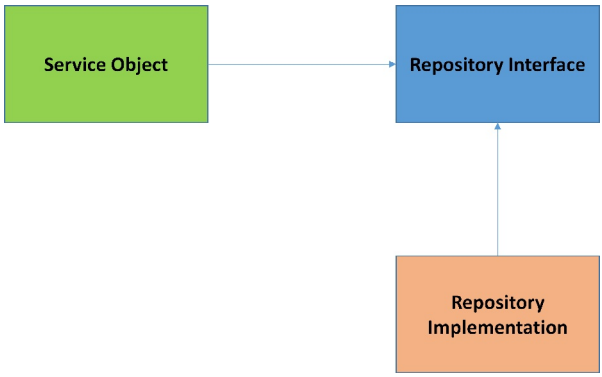
}

class LunchBox{
private String item1;
private String item2;

}

1. Convert Each class into Java Bean
2. Configure bean wiring in context XML
3. use Spring to display the items of Lunch Box for a participant under the Trainer.

Hands on:



class Emp{
private int empId;
private String name;
private String city;
private double salary;
}

public interface EmpDao{

public String save(Emp e);

}

class EmpService{

public String registerEmp(int id, String name, String loc, double salary){

//you have to call save method from EmpDao}

}

class EmpDaoMockImple implements EmpDao{

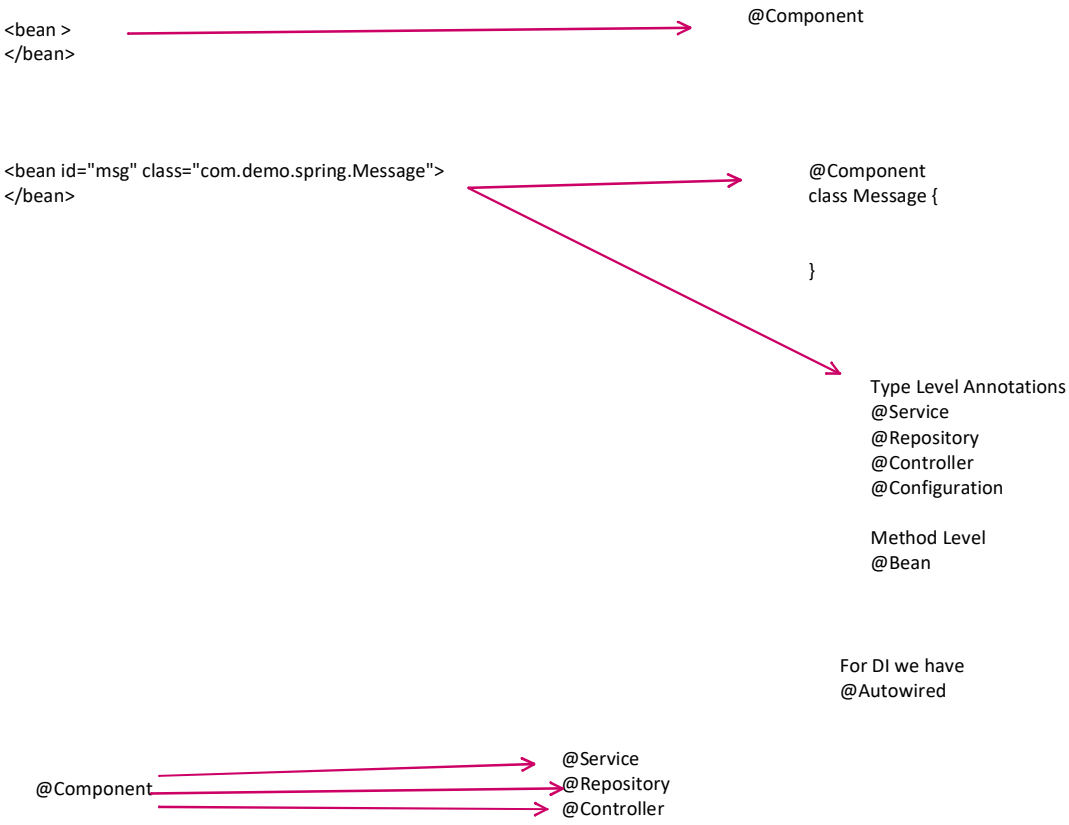
your code goes here;

}

Spring Annotations:

@Component

Spring Annotations:



For Bean Scope ---> @Scope

Bean Naming

Every Bean must have a name
If annotated with type level annotation, the default name is class name with first letter lowercase

If annotated with method level annotation, the default name is method name

Adv Java Day 10

28 July 2020 09:29

Spring Framework

1. Software Library to build customised Enterprise Application
2. Uses Java Beans as Major component for Programming
3. Has an IOC or Dependency Injection Container
4. The DI Container is at Application Scope (within Application)
5. DI Container is accessed using ApplicationContext Object
6. Use either xml or Annotation Based configuration
7. It uses Setter and Constructor Based Injection

Hands On:

```
class PrinterApp {

private Writer writer;

public void print(String message){

writer.write(message);
}

}
```

extract the Writer interface
Use PlainTextWriter Class as implementation for Writer Interface

Create the Demo Application to print some plain text on to the console

Next Topic : Bean Annotation

When Dev Writes the source code, he can use
@Component
@Service/@Repository/@Controller etc

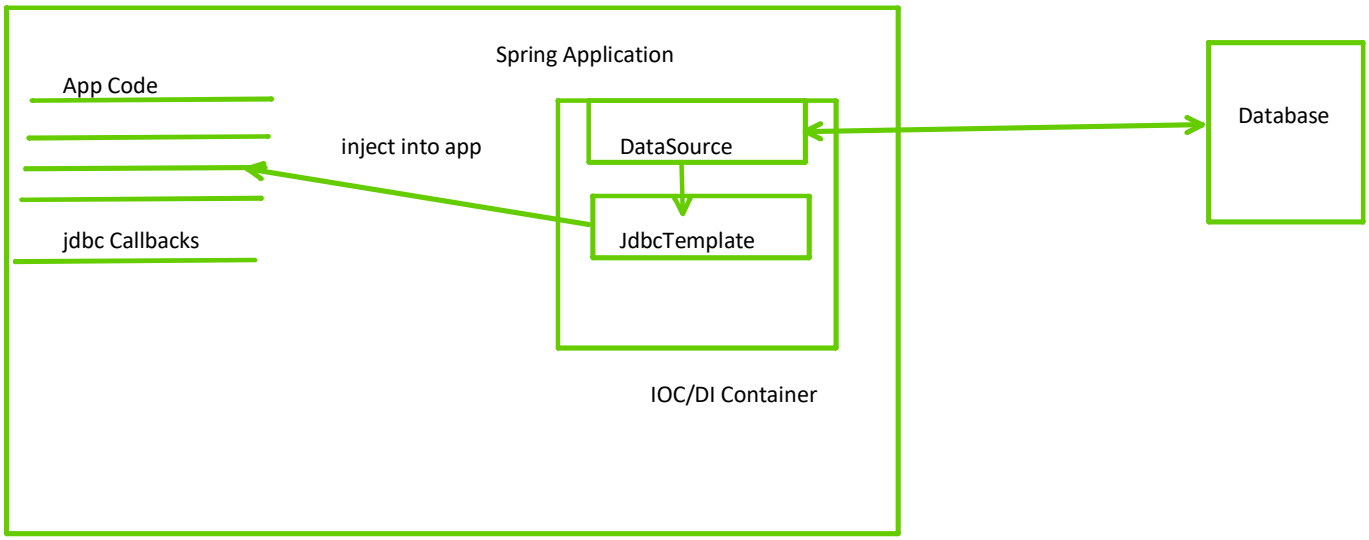
```
@Component
class Message{

}
```

But If the source file is not available or you need to create a Spring Bean from 3rd party class file (in a .jar)

```
@Bean
public Message message(){
return new Message();
}
```

Next Topic: Spring JDBC and DAO Implementation

1. There is no Checked Exception
2. Spring DAO has its own exception Hierarchy
3. it uses JdbcTemplate with Spring jdbc Callback



Spring JDBC Hands on Done

Home Assignment:

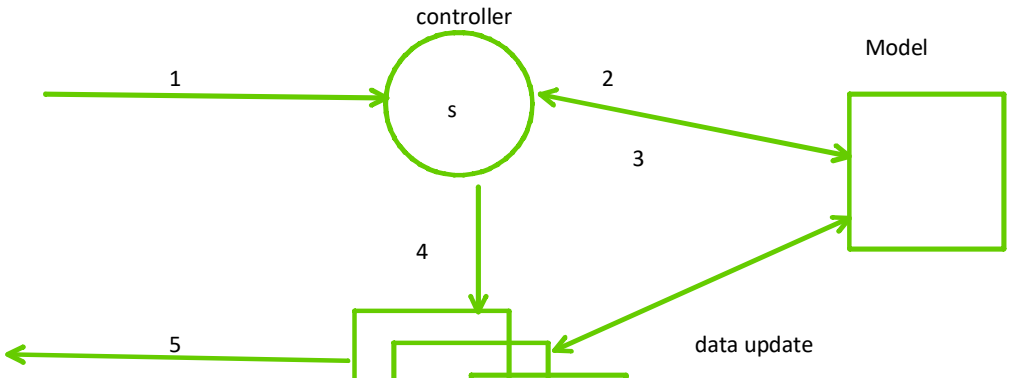
implement the following Dao Interface using SpringJDBC

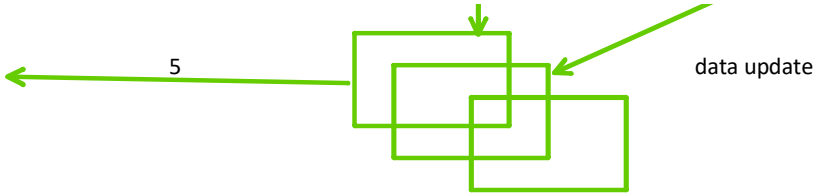
```
public interface EmpDao {

	public String saveEmp(Employee e);

	public List<Employee> getAll();

	public Employee findById(int empid);

}
```

```
public class Employee {

	private int empId;
	private String name;
	private String city;
	private double salary;

	//constructors
	//setter and getter
}
```

Spring MVC:

MVC Pattern (Model2)





Spring MVC helps create Web Applications based on MVC (Model2)

Handler Mapping in Spring MVC

/demo ---> DemoController
/saveData--->DbController
/delete --> EmpDeleteController

ViewResolver

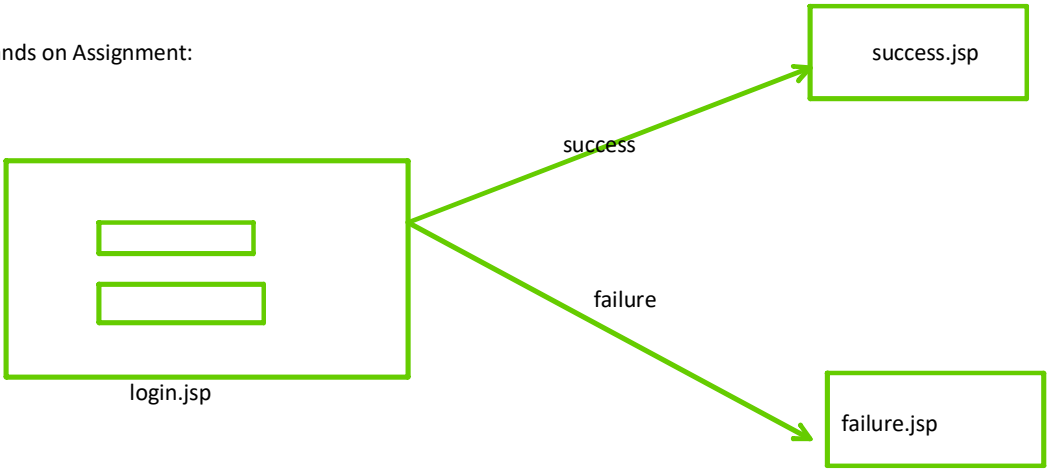
login ---->/pages/login.jsp
logout--->/pages/logout.jsp
catalogue-->/pages/cat.pdf

InternalResourceViewResolver

prefix|logical view Name|suffix ==Full View Path

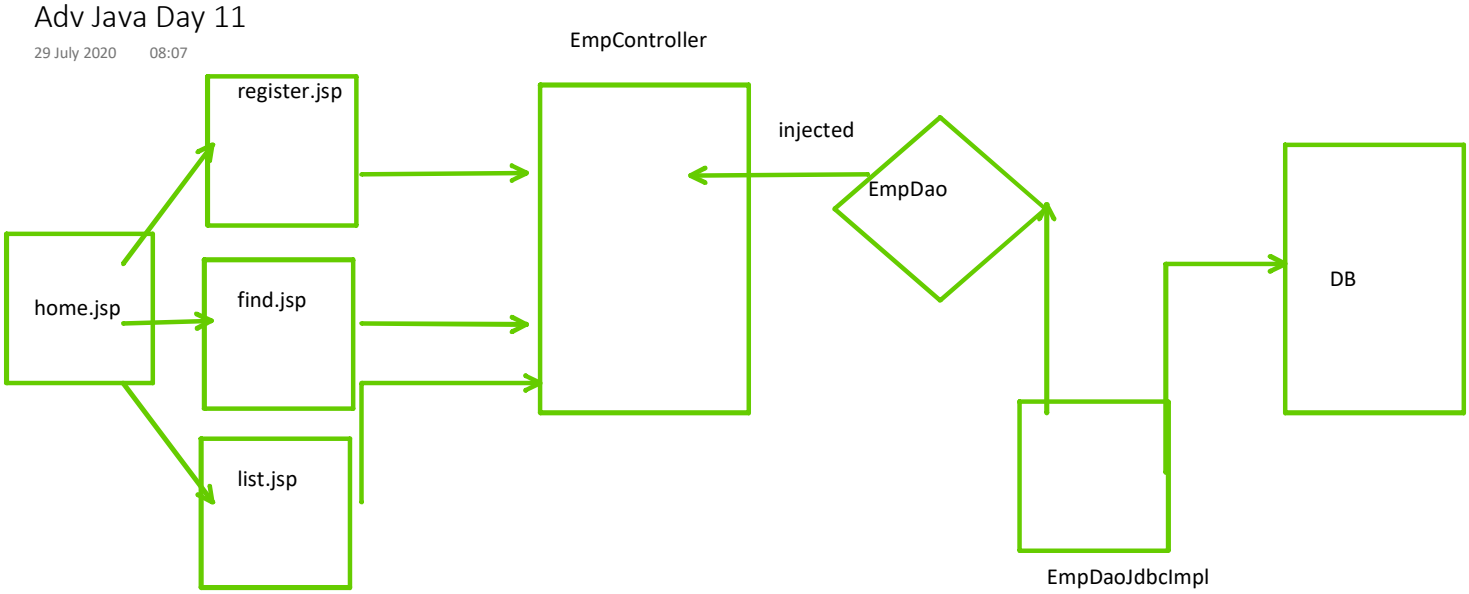
/WEB-INF/jsp/+hello+.jsp==/WEB-INF/jsp/hello.jsp

Hands on Assignment:



Adv Java Day 11

29 July 2020 08:07

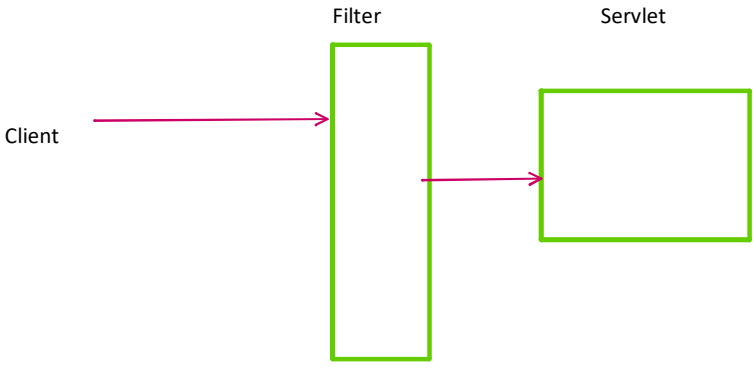


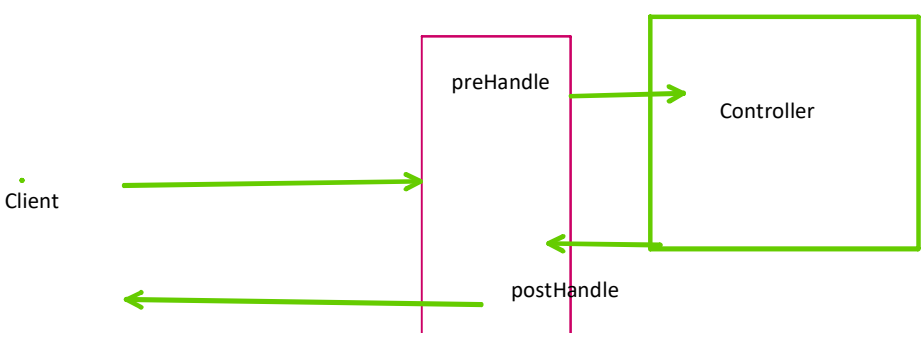
Spring dependency : spring-context, spring-jdbc, spring-tx, spring-webmvc, mysql-connector-java

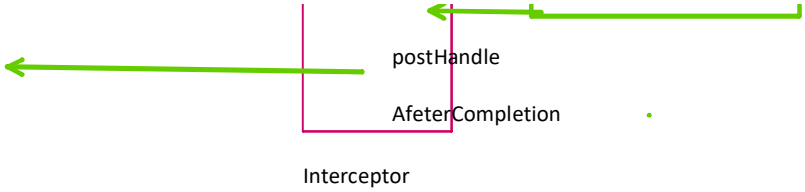
```
@ControllerAdvice
create class GlobalExceptionhandler{

@ExceptionHandler(RuntimeException.class)
	public String handleMyException(RuntimeException ex) {
		System.out.println("Global  Scoped Exception Handler");
		return "failed";
	}


}
```

In Servlets API We have Filters

In Spring We have Interceptors







HandlerInterceptor

1. Create the Interceptor class which implements HandlerInterceptor {

3 methods

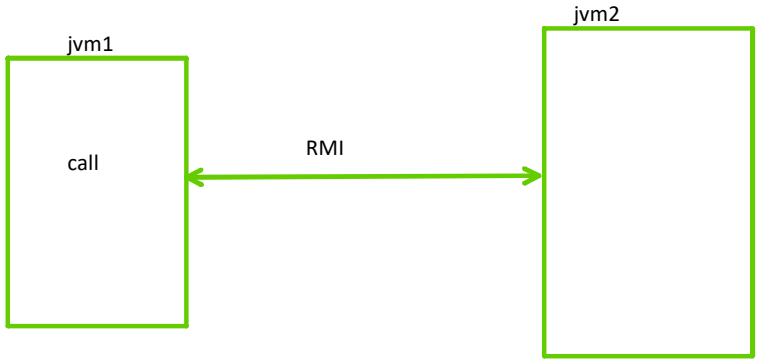
}

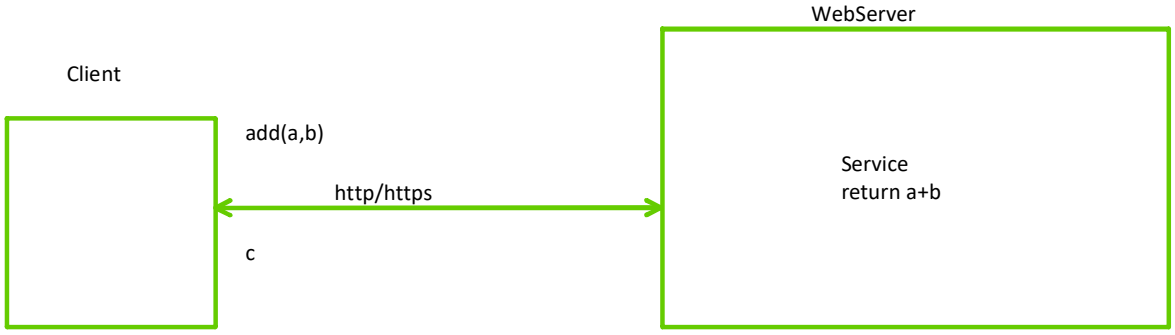
Register the interceptor in your WebConfig class as follows:

```
public class WebConfig implements WebMvcConfigurer{

      //other methods

      @Override
      public void addInterceptors(InterceptorRegistry registry) {
            registry.addInterceptor(new Interceptor Object ...);
      }
}
```

Business Method calls

Local --> using Object Reference
Remote-->





Web Services

SOAP Web Services --> exchange data only in XML Format (SOAP Message)--> Specification

REST Web Services --Exchange data in any transport format (XML,TEXT,JSON,STREAM,MULTIPART etc etc etc)

REST Uses all Http verbs (GET, POST,PUT,DELETE,HEAD, OPTIONS ...)

You Map
HTTP Methods ---> Serve rside methods

1. Create a maven war project with Spring mvc dependency
2. Create a Controller like one given below

```
@RestController
public class GreetController{

@RequestMapping(path="/greet", method=RequestMethod.GET)
public String greet(){
return "Welcome to REST";
}

}
```

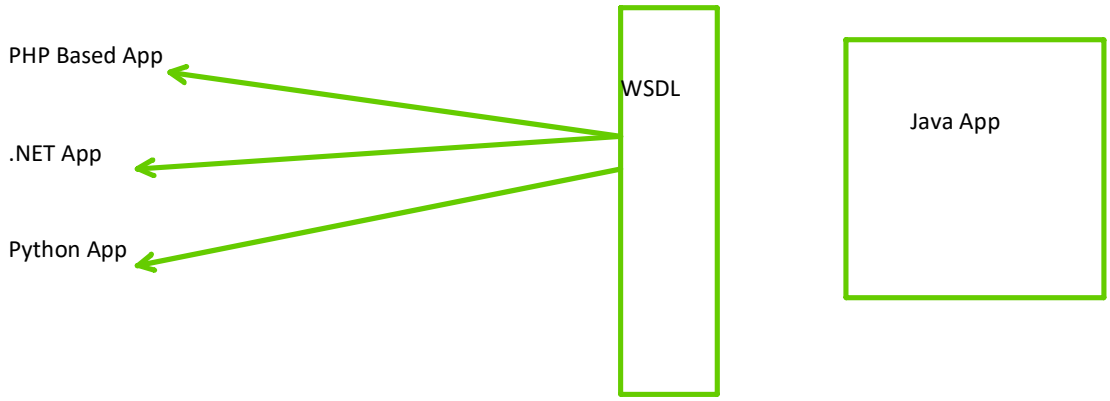
Test URL: http://localhost:8080/AppAname/greet

Adv Java Day 12

30 July 2020 09:34

Why Web Services?

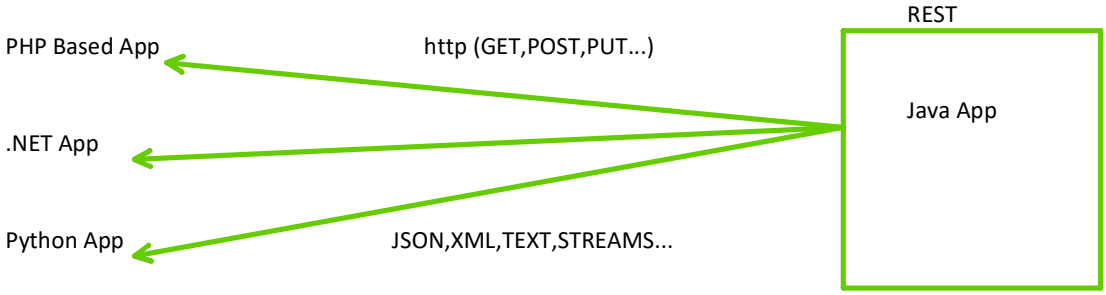
SOAP STD



data Exchange : only XML

But Modern Day :JSON,TEXT, STREAMS ...

REST



RMM (Richardson Maturity Model) for REST

Modern Apps

gaming Console

....

Query String http://localhost:8080/App/demo?a=20&b=30
Form Data
Multipart
Raw data (mime types)

@RequestParama ("xx") reads query params
@PathVariable("yy") reads patha variable in uri

CREATE READ

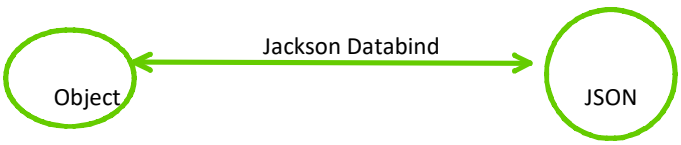
Implement DELETE and PUT (UPDATE)

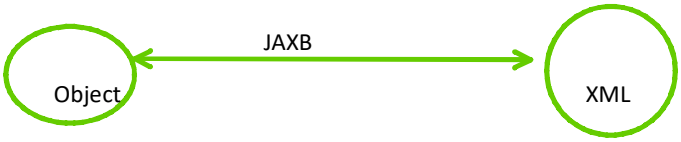
Spring 5 onwards

@RequestMapping(method=RequestMethod.GET) --->@GetMapping
@RequestMapping(method=RequestMethod.POST) --->@PostMapping

@RequestMapping(method=RequestMethod.DELETE) --->@DeleteMapping

@RequestMapping(method=RequestMethod.PUT) --->@PutMapping





JQuery:

1. JavaScript Library
2. small and fast
3. manipulates html DOM
4. used for responsive UI and AJAX Calls

jQuery Syntax:
$(selector).action()

```
$(document).ready(function(){

alert("Jquery is Ready");
}
);
```

Selector

1. element selector
2. id selector
3. class selector