.

CGI

Java Full Stack Developers (FSD)

Topics Index

Trainee : Bikash Jena

(23-8-2021)

Backend

1. Core java
2. OOPS
3. Encapsulation
4. Inheritance
5. Polymorphism
6. Abstract
7. Constructors
8. Two types of Constructors
9. Default constructor
10. Parameterized Contractors
11. Static Keyword
12. Static variables
13. Static methods

Without object creation we can access static variabls and static methos

Through class name. method name and class name. variables

1. Final
2. Strings
3. String class (final class)

Once we create string class object with new operator we can't modify

1. StringBuffer(final class)
2. StringBuilder(final class)
3. Interface (keyword)
4. Abstract class
5. Exception Handling
6. CheckedExceptions
7. Unchecked Exceptions
8. try{}
9. Catch()
10. Final
11. Throws
12. Throw
13. User defined Exceptions (extends Exception /RuntimeException)
14. JavaBeans
15. Collections Framework
16. java.util.Collection (root interface)
17. Set (store randomly, not allowed duplicates)
18. List (store order but will allowed duplicates)
19. Hashset ,TreeSet
20. ArrayList ,LinkedList, Vector (old API)
21. Iterator / ListIterator / Enumeration (All retrieve data form Collection Objects)
22. Generics
23. forEach advanced Loop
24. java.util.Map interface
25. HashMap (methods are non-synchronized)
26. Allow Null keys and Null values Keys
27. Hashtable(Old APi, all methods are synchronized)

Not allow Null Values and Null Keys

1. java.util.Collections (class) (sort method)
2. Comparable (java.lang)
3. compareTo() (Compare One object)
4. Comparator (java.util)
5. Compare(Compare two objects)
6. Multithreading (Interviews)
7. Multiple threads are running simultaneously CPU Ideal time reduce
8. We can create a thread two ways

Extends Thread

Runnable Interface

1. Java 8 Features
2. Lambda expression
3. Functional Interfaces
4. ForEach – Iterable
5. Stream API
6. Method References
7. Date
8. JDBC (Interviews)
9. Types 1, Type 2, Types 3
10. Type 4 Thin driver
11. Statements:

Statement

Prepared Statements

True/false =execute()

Int =executeUpdate()

ResultSet rs=executeQuery()

1. Database (RDBMS SQL (MySQL))

DML, DDL, TCL

1. Layer wage Arc

Controller - > Service - > DAO ->(database/MySQL)

1. JUNIT (Framework)

Java code Testing

Front End

1. HTML /HML 5
2. CSS /CSS3
3. Bootstrap (Bootstrap, CSS3, HTML5=for develops Responsive web pages)
4. JavaScript (Client side validation )
5. JSON

{

        "name": "Ram Technologies ",

        "description": "JDBC",

        "id": "COREJAVA"

    }

1. Nodejs (Installation for running Typescript and Angular Application )

C:\> npm -version

6.14.13

C:\Users\hp>node --version

v14.17.0

1. Typescript

npm install -g typescript

D:\CGI-UI-Development\08-Typescript>tsc --init

file with:

TS1-App.ts

console.log ("My Typescript First Program");

-- Compile TS file

D:\CGI-UI-Development\08-Typescript>tsc TS1-App.ts

Run Js file

D:\CGI-UI-Development\08-Typescript>node TS1-App.js

My Typescript First Program

1. Angular 7/8

1. frontend framework

2. angular is a framework for building single page application using typescript

3. developed by google

4. Module+Component based architecture

5. Angular contains command line interface that comes with comands that help us create and use in our project

6. To work with Angular CLI, we need to install NodeJS

7. commands:

1. ng version

2. npm install -g @angular/cli

3. ng new <project name>

4. ng g c <component name>

8. to run the project: ng serve

ng g s <service name>

9. Angular, by deafult, runs on port 4200

10. To change port: ng serve --port 4211

11. Concepts in angular Directives, data binding, services, dependency Injection

12. all dependencies/Libraries are available in node modules

13. for changing directory cd <Application name>

14. Angular default files

4. app.component.css

3. app.component.html (root html)

2. app.component.ts (root component)

1. module.ts (root module)

15. We make changes in the app.component.html and app.component.ts files to develop angular

inline template:

16. An inline HTML template for a component is defined using template config in @Component decorator

17. In angular we can perform component interaction

18. Interpolation ?.

Interpolation is used display a component

property in the respective view template with

double curly braces syntax

19. Concept like Routering can help to easily manage different URLs

20. To create routing module the command is ng generate module app-routing --flat --module=app

21. Data binding :

Data binding is a core concept in Angular and allows to define commutation between a component and the DOM

22. Property binding

<input type=”email” [value]=”email”;>

template: 'Name : <input [id]="myId" type="text value="Ram"',

public myId="textId";

23. Event binding :

For the DOM to the component

When a specific DOM event happened(eg : click ,change ,key ups

Call the specified method in the component

24. Two way data binding

ngModel is Directive :

Part of the Angular “FormsMoudle” and has to imported into your module manually .

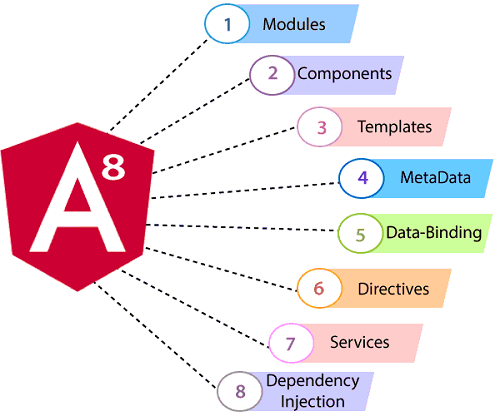
25. Service

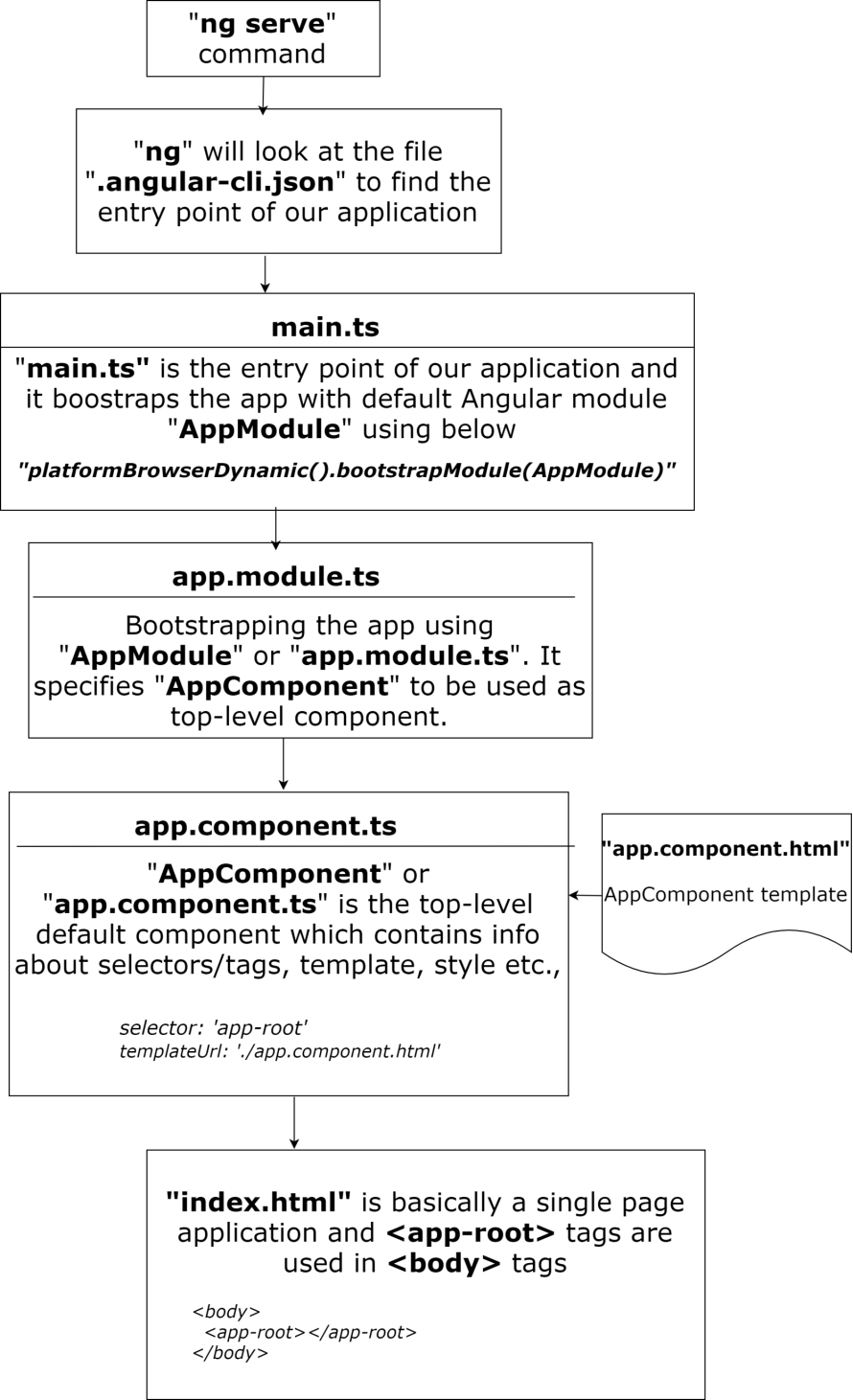
Why Service ?

Components shouldn’t fetch or save data directly

@Injectable() service

That the new service imports the Angular Injectable symbole and annotates the class with @Injectable() decorator.

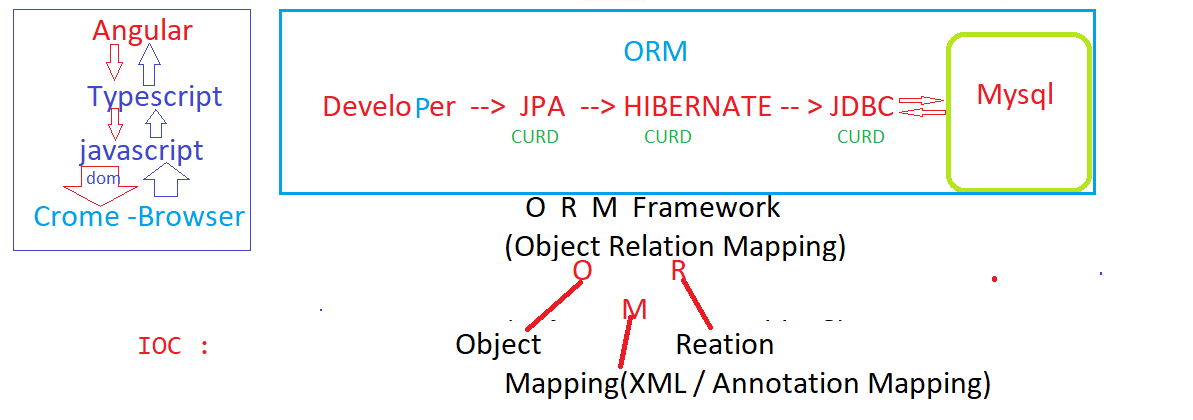


Angular Flow: 

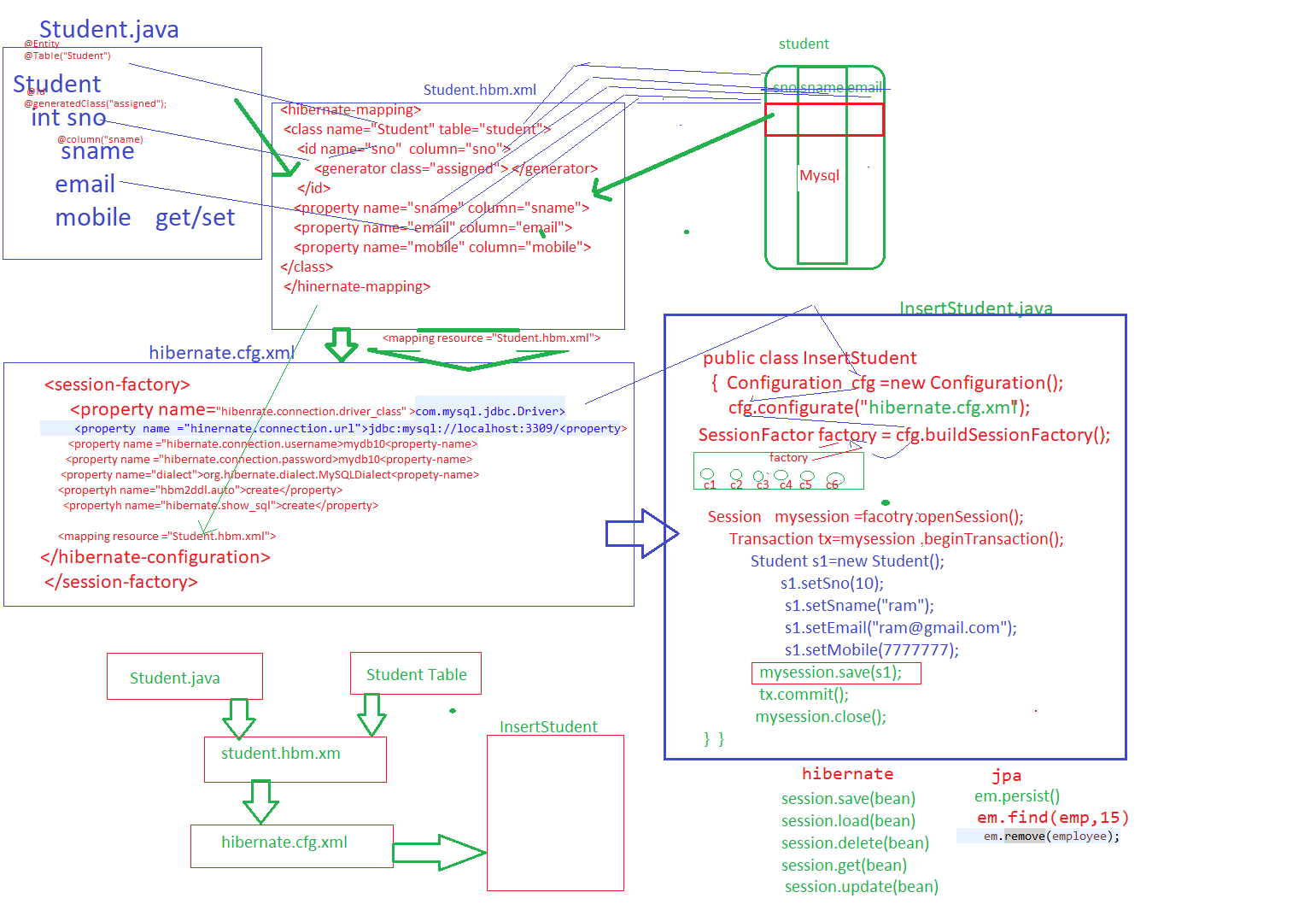
------------

Backend

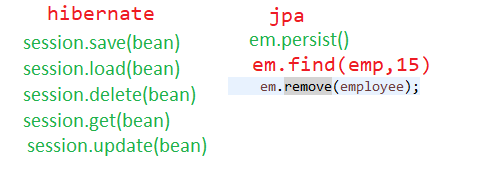
1. ORM Technologies
2. Object Relational Mapping
3. orm is used to create data in database without writing sql query again and again

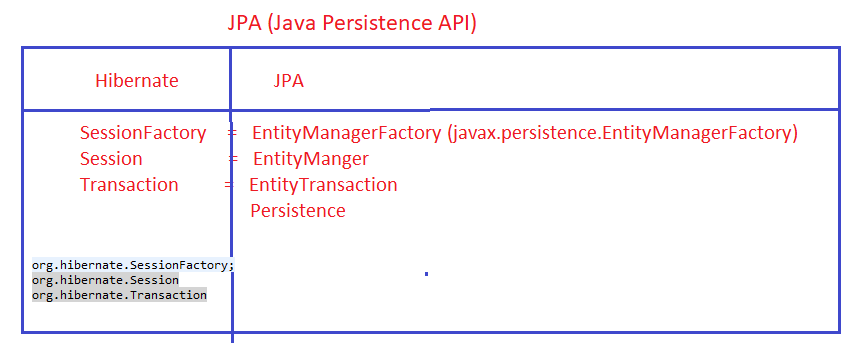


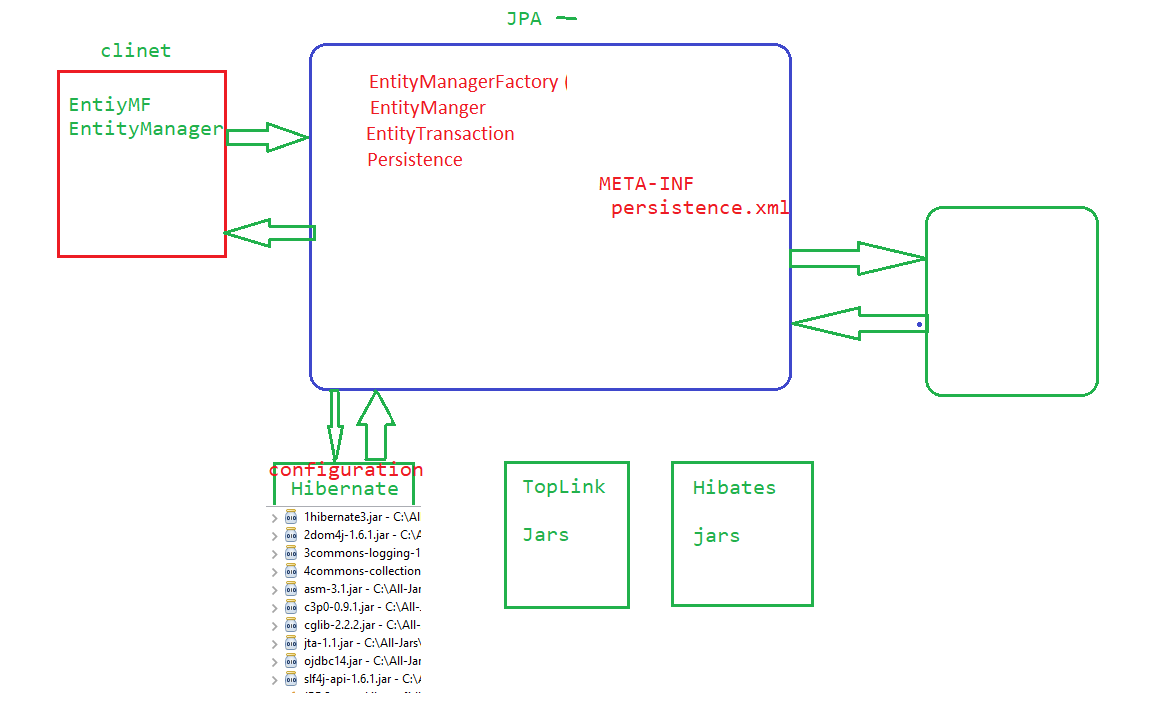
1. Hibernate :



1. JPA







18.Maven

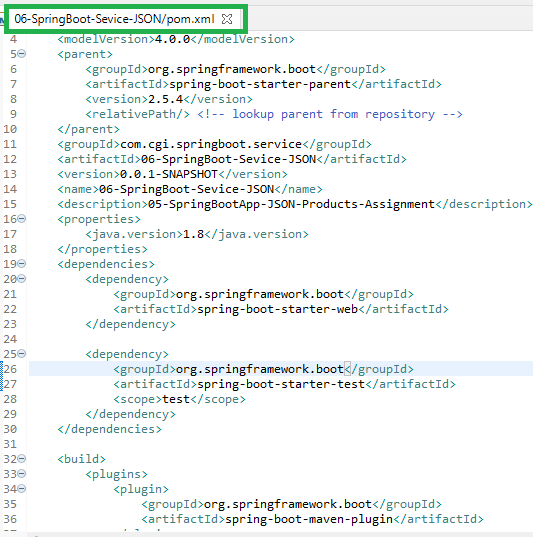
Maven is build tool

Pom.xm (Project Object Model)

XML file this file contains information about the project and Configuration details used by Maven to build the project

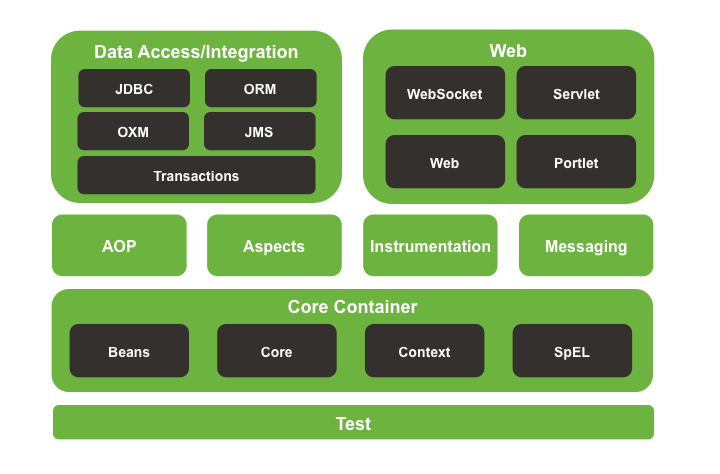
: Project such as :

Dependencies , Build directory , Source directory , Test directory , plugin



1. Spring Technologies

Spring Architecture



1. Spring – Core **(IOC – DI(contractor injection, setter injection)**

IOC : (Inversion Of Controller)

Giving control to the Container to get an instance of the Object is called IOC

Means instead of you are creating an object using new operator let the container do that for you.

DI (Dependency Injection)

Way of Injection properties to an object is called Dependency Injection

1. Constructor Injection

2. Setter Injection

1. Spring – DAO (Spring with DAO (JDBC program)
2. Spring – JdbcTemplate (Spring with JdbcTemplate)
3. Spring - MVC (Web :)

1. Spring Annotations :
2. SpringBoot
3. SpringBoot (@Sevice)

Spring Annotations:

@Autowired:

We can use the @Autowired to mark a dependency which spring is going to resolve and inject. We can use this annotatin with a constructor , setter injection

@Bean

Indicates that a method produces a bean to be managed by Spring container

@Bean annotation can be used with parameters like name, init mthod and destroy method

@Configuration

Used to @Bean methods : These classes are proceeded by the Spring container to generate bean definitions and service request for these beans at runtime

Spring Stereotype Annotations :

@Component

@Controller

@Service

@Repository

**@Component:**This annotation is responsible for converting a java class to the bean so that it can be recognized by Spring and used in the application context.

**Syntax:**

@Component

public class StudentDAOImpl implements StudentDAO {

    ...

}

@Component :

1. @Component annotation marks the Java class as a bean

@Component is an annotation that allows Spring to automatically detect our custom beans

@Repository :

1. Spring @Repository annotation is used to indicate that class provides the mechanism for Storage, retrieval, search , update and delete operation on Objects.

@SpringBootApplication is a convenience annotation that adds all of the following:

* @Configuration: Tags the class as a source of bean definitions for the application context.
* @EnableAutoConfiguration: Tells Spring Boot to start adding beans based on classpath settings, other beans, and various property settings.
* @EnableWebMvc: Flags the application as a web application and activates key behaviors, such as setting up a DispatcherServlet. Spring Boot adds it automatically when it sees spring-webmvc on the classpath.
* @ComponentScan: Tells Spring to look for other components, configurations, and services in the the com.example.testingrestdocs package, letting it find the HelloController class.

@SpringBootApplication :

is a convenience annotation that adds all of the:

@Configuration tags the class as a source of bean definitions for the application context.

@EnableAutoConfiguration tells Spring Boot to start adding beans based on classpath settings, other beans, and various property settings.

Normally you would add @EnableWebMvc for a Spring MVC app, but Spring Boot adds it automatically when it sees spring-webmvc on the classpath. This flags the application as a web application and activates key behaviors such as setting up a DispatcherServlet.

@ComponentScan tells Spring to look for other components, configurations, and services in the the hello package, allowing it to find the HelloController.

The main() method uses Spring Boot’s SpringApplication.run() method to launch an application. Did you notice that there wasn’t a single line of XML? No web.xml file either. This web application is 100% pure Java and you didn’t have to deal with configuring any plumbing or infrastructure.

**Difference between @Requestparam and @PathVariable annotation?**

**Answer:**These annotations are used for fetching information and data from an URL.

But, the main difference between them is, @Requestparam is only used to fetch query parameters and @PathVariable is used to fetch the complete URL.

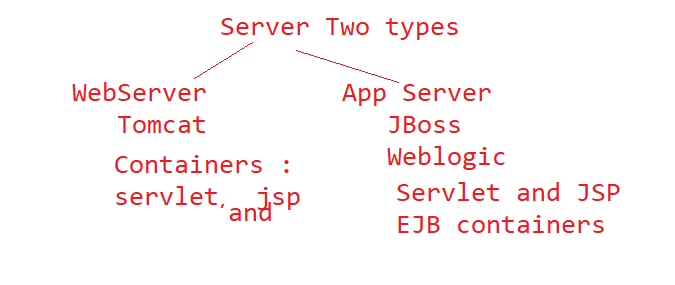
**@RequestBody?**

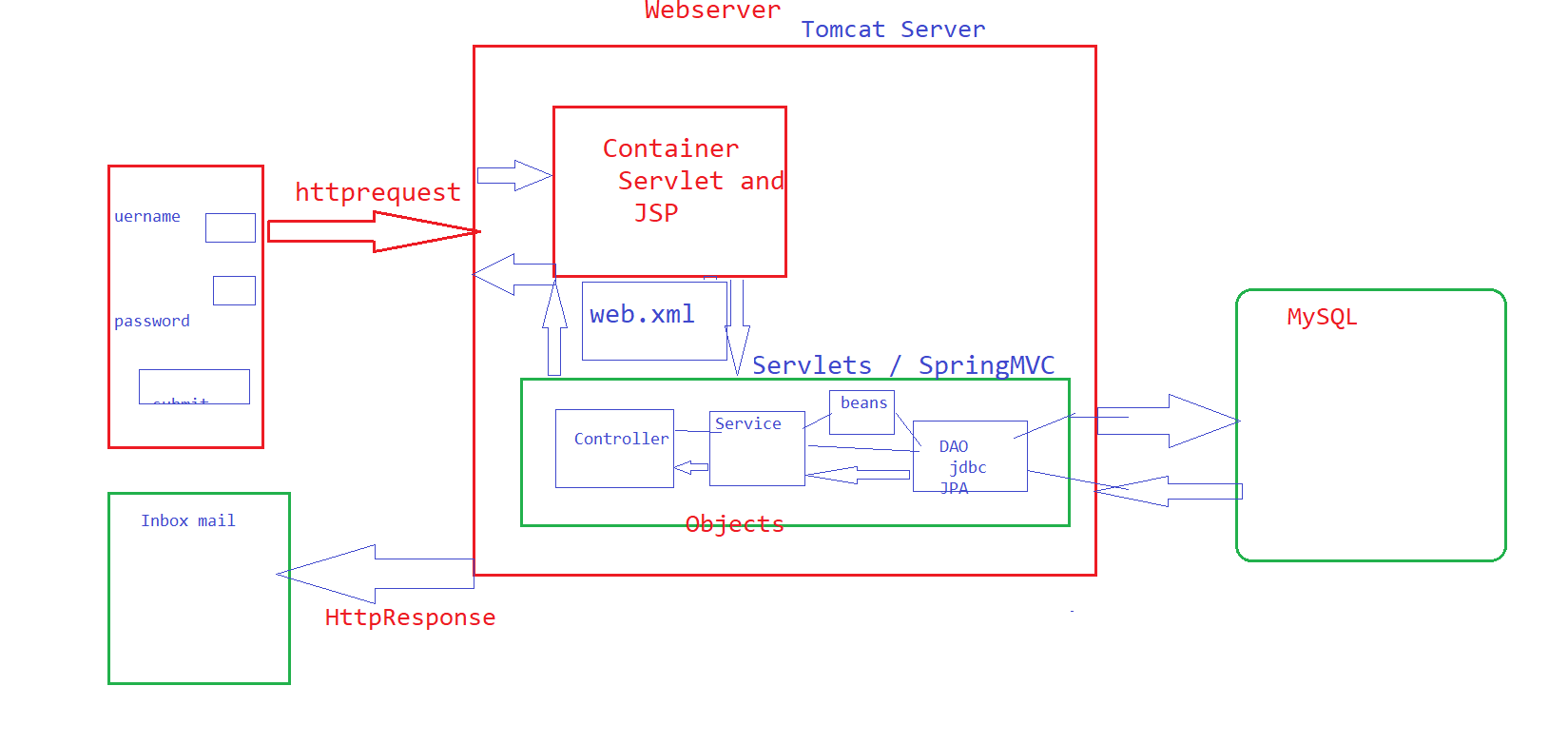
The @RequestBody annotation is responsible for binding the HTTP body request to a domain object. The incoming HTTP request is automatically de-serialized to the Java object by Spring with the

**@ResponseBody?**

When the @ResponseBody annotation is used in the MVC controller, it indicates that the developer needs to write a return type of declared method directly to the HTTP response body. Here, invoking of the model is not required and the view name is not interpreted by Spring.

1. Server :
2. Web Server
3. Tomcat
4. Application Server (Secure application like Bank)
5. JBOSS
6. Weblogic





1. Postman
2. get
3. post
4. put
5. delete