SPRING

Java provides 3 platforms

JSE : Java Standard Edition(Contains core language)

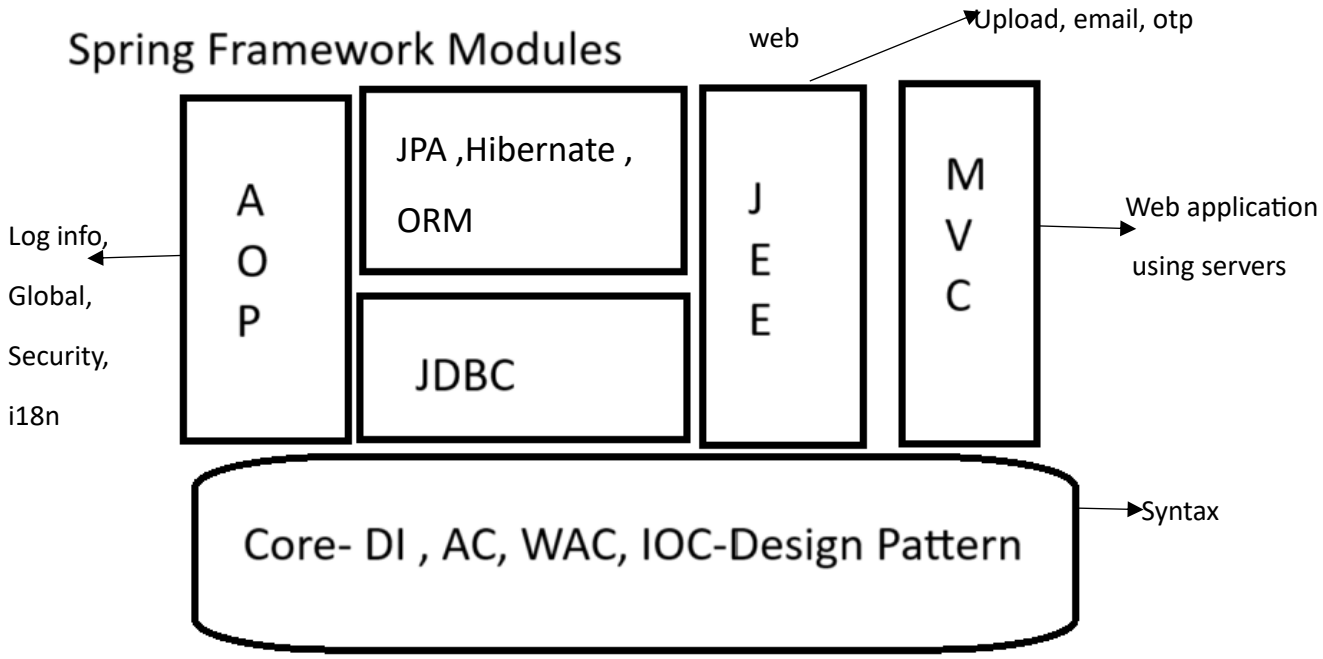
JEE : Java Enterprise Edition (Contains API's like servlet API, JSP API etc)

JME : Java Micro Edition

Spring is the base project from spring community(25projects)

Why Spring?

- Spring is an Enterprise application java framework.
- It is used to build enterprise application.
- It is an implementation of IOC (Inversion of Control-It is a design pattern)
- It is used to manage and configure the components (service, dao/repo, controller, servlets, beans etc) for application.



Spring 0RM module:- Helps us to integrate spring with ORM, JPA/hibernate servlets

AOP-Aspect Oriented Programming

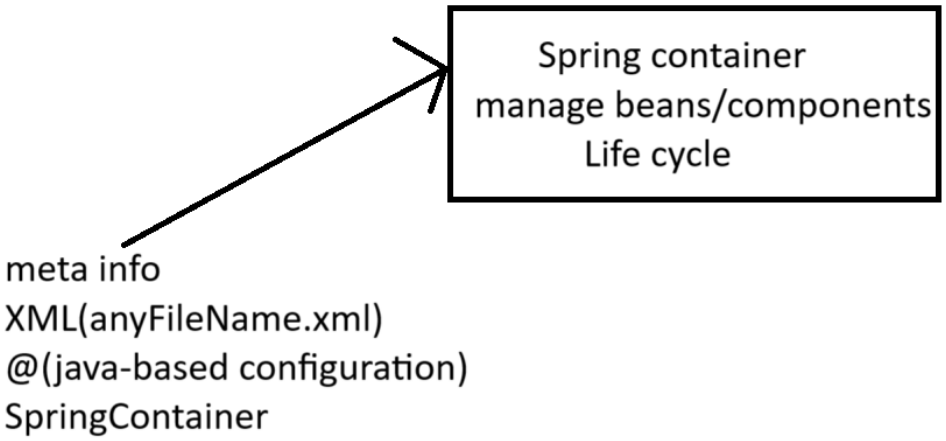
DI-Dependency Injection

AC-Application Context

WAC-Web Application Context

Spring Container is a object which manages other object for an application.

For ex:-AC(I) , or bean Factory



Inverting the control of components(spring container) into external entity.

Component- by calling default constructor our component will create object.

Dependency Injection :- Its like as a relationship

inject through @autowired - it is used only for custom data not for inbuilt

It is a part of IOC pattern

when the class refer another class is called- dependency during compile time

same class converted to object

when the object refer another object is called-injection-during run time

the process of referring a dependent object is called DI.

2 ways to use to DI

1)Setter injection using @Autowired

2)Constructor injection

3 ways to using autowired

above custom data

above constructor

above setter method

First we have to make class as component and subclass as @autowired

Ex1:-

```
@Component
Class State{
        @autoWired
        District district;
}
Class District{
}
```

Ex2:-

```
@Component
public class State {
    @Autowired
    public City city;
    @Value("Karnataka")
    public String stateName;
    public State(City city){
        this.city = city;
    }
}
```

Spring configuration-it has meta info

java based configuration through @ annotation

@Bean-works on inbuilt /.class or custom also_(String,list,Patient)- it is used inside class above the method

@Component-works on custom java file only-(Patient) - it is used above the class.

Spring orm is used to make the integration for spring to the my sql database.

If we want to get single record of single column then use specific return type.

If we want to get single record multiple columns then use object array and use getsingleresult

If we want to get multiple record single or multiple column then use list of object array and use getresultlist

Spring mvc

spring supports-enterprise application and spring mvc - supports web application

It has 4 main components

1)Dispatcher servlet

2)handlermapping

3)controller

4)Viewresolver

Spring mvc will have standalone environment(which contains main method) and web environment

Container

beans - it is an object which is managed by container

which will have scope there is 6 scopes :-Singleton, prototype, request, session, global-session, websocket.

Container have 3

1)bean factory:-

-it is a core container

-it supports standalone application or environment

-it supports only functional requirements(Only crud) -- for any application there should need functional and non functional requirements (otp, security, languages as ex in google)

-it does not supports AOP

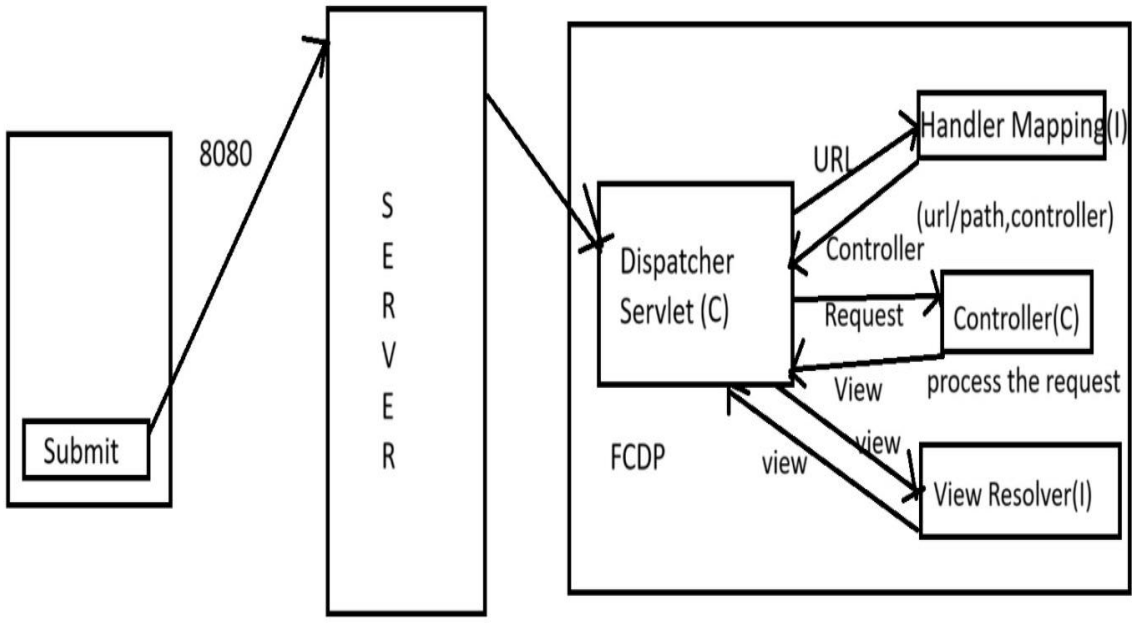
2)AC , 3)WAC

-it is a advanced container

-it supports standalone application or environment and web environment

-it supports both functional requirements(Only crud) -- for any application there should be need functional and non functional requirements(otp, security, languages as ex in google)

-it supports AOP



When the client makes the request by clicking submit button , first it will enter to tomcat server.

Dispatcher servlet

-it is a class

-first component of spring mvc

-FCDP - front control design pattern

-it is provided by spring framework and manages on tomcat server

-it is only process but handled by handler mapping

Handler mapping

- It is a interface
- it take as key value pair , key is url or path and value is controller

controller

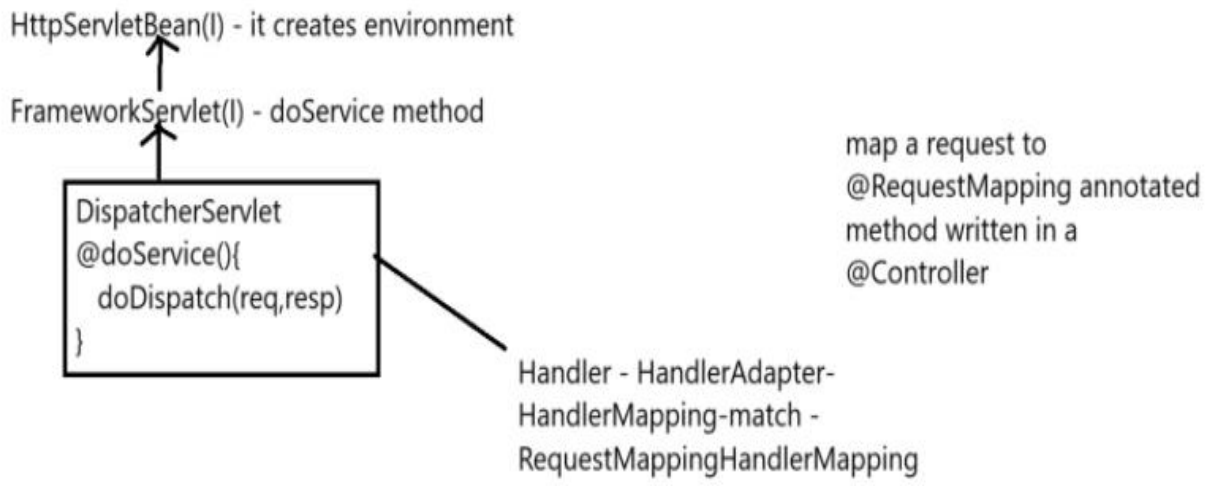
-It is a class

- It process the request from dispatcher servlet

-controller will return view to dispatcher servlet

View resolver

- It is a interface
- It will validate
- it will receive view and return view/ 404 error.



When we start putting annotation based configuration it will be looking for abstractAnnatotationConfigDispatchServletInitializer.

It will do many task

First task is registering dispatcher servlet

To register dispatcher servlet at the initialization stage abstractAnnatotationConfigDispatchServletInitializer it helps us to create WAC. After creating WAC it will looking for servlet config class(SpringConfiguration file).

It all happens on load on startup.

Once dispatcher servlet is initialized then it will looking for spring configuration object.

This is all set up, work will happen when client makes request.

How Dispatcher Servlet communicate with handler mapping -imp (4 components how communicated)

Request sent to interface servlet then to httpservlet then to httpservletbean which creates spring environment then to framework servlet – it is overriding doxxxx methods – doxxx method will call process request method – it internally call doService method (Its an abstract method in framework servlet)- framework servlet it has another subclass called dispatcherServlet – it overrides the doService method – it will call doDispatch method – doDispatch doesn't know how to communicate handler – so it will be looking for handler adapter- it will be looking for handlermapping- handlermapping its an interface – it will be looking for registered implementation – one of them is requestmappinghandlermapping – this class is responsible for map to a request to a @Requestmapping annotated method of @Controller class

-when it found particular handler – then only dispatcher servlet can communicate with that handler. – then after controller will

come. – it will return view to dispatcher servlet- it will forward it to view resolver – it will validate view name . – if its available it will give view or 404 error.

```
@GetMapping(value = "delete/{id}")
public RedirectView deleteUser(@PathVariable("id") int id , Model model ,
HttpServletRequest req){
   customerService.deleteCustomer(id);
   RedirectView redirectView=new RedirectView();
   redirectView.setUrl(req.getContextPath()+"/getallusers");
   return redirectView;
}

@GetMapping(value = "fetchUser")
public String fetchUser(@RequestParam int id, Model model){
   CustomerDto customer = customerService.getUserById(id);
   model.addAttribute("customer", customer);
   return "fetchuser.jsp";
}
```

pathvariable when we dont want key only value directly value in url

RequestParam