JAX-RS:

it conatains definition for REST web services.

we can implement it by 2 ways:

1.jersey

2.RESTEasy

jersey:

It contains few important annotations:

@Path ----------It specifies the URI path which is placed over class or methods.

@PathParam--------It specifies the parameters from the URI path.

@produces---------It defines media type for the response such as XML, PLAIN, JSON etc. It defines the media type that the methods of a resource class or MessageBodyWriter can produce.

@consumes---------It defines the media type that the methods of a resource class or MessageBodyReader can produce.

@put:HTTP PUT request, used to update resource.

@post:HTTP POST request, used to create resource.

@Delete:HTTP DELETE request, used to delete resource.

@Get:HTTP Get request, used to retrive resource.

Spring MVC Components:(It is a module from Spring Framework.Meant for developing WebApplication,Distributed Application,Rest APi)

1.Front controller(Dispatcher Servlet).

2.Handler Mapping.

3.Controller

4.MOdel and View

5.Viewresolver.

6.View

This Spring Web Mvc module is developed based on 2 design patterns

a)MVC Design Pattern.

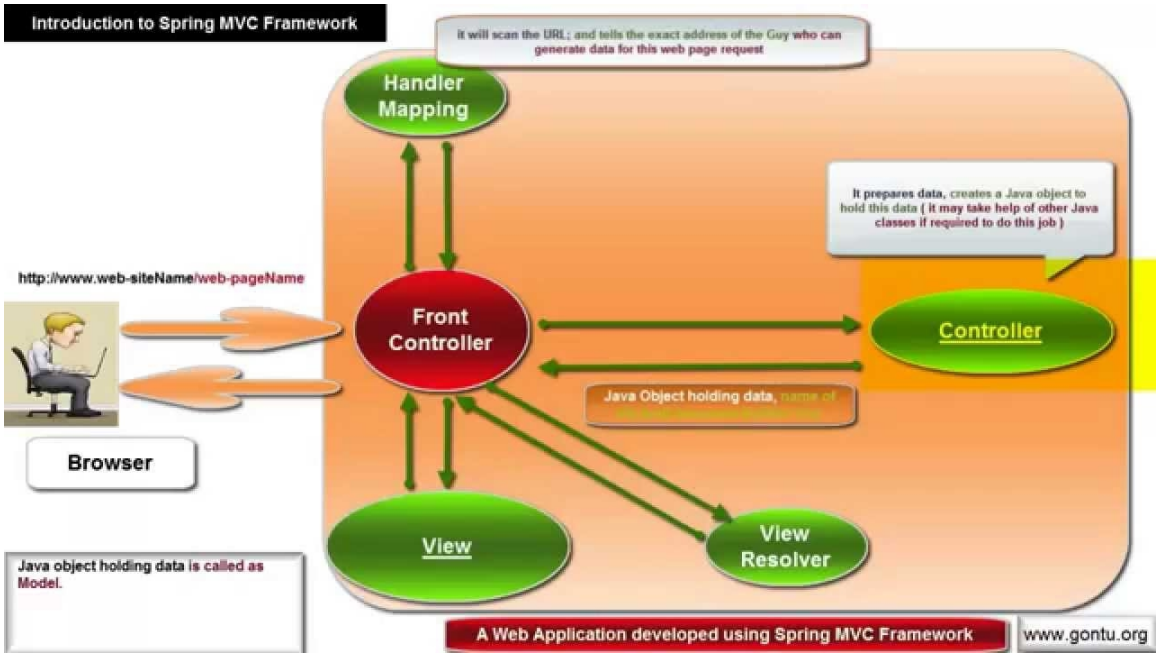
b)Front controller Design Pattern.

FrontController:-

-->It is responsible to perform pre-processing(ex:capturing the form data) and post-proccessing(Ex:sending response to the client in understandable form) of incoming requests.

-->In spring Web MVC application Dispatcher servlet as a front controller.

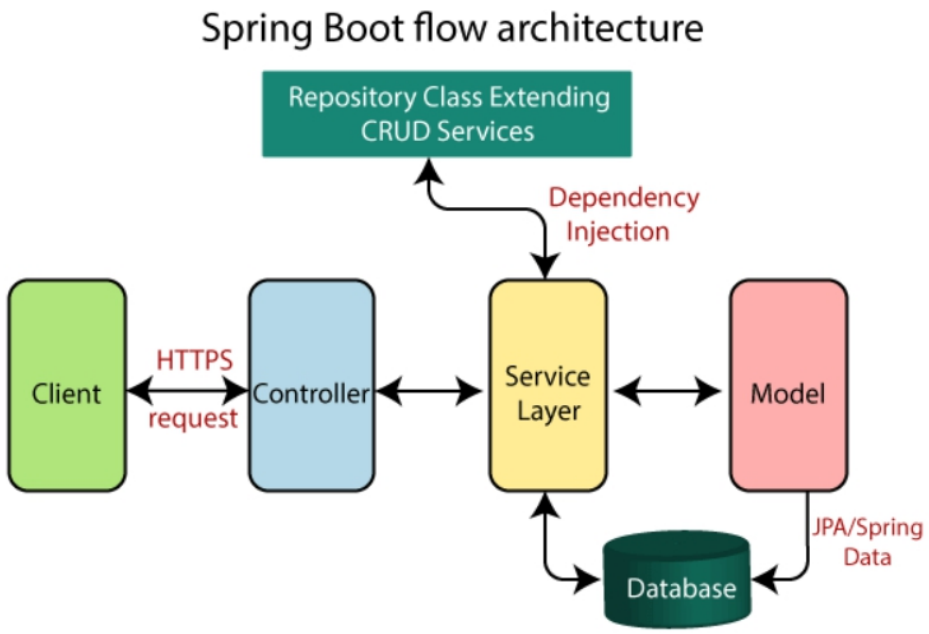
-->Dispatcher servlet is a predefined servlet provided by Spring MVC model.



When User Sends the Request that is received by Frontcontroller(Dispatcher Servlet) Than it secds that to the Handler mapper where it will scan URL and delivers the address to the Dispatcher servlet where the user trying to access.Now the Dispatcher Servlet have the address

so it goes to the controller It prepares data,creates a java object to hold this data(it may take help from other java classess if required to do).It sends that java object which is holding data along with the name of the component to retrive the data from the java object mixing it with html to prepare actual response to the dispatcher servlet.Now dispatcher servlet will have the java object which is to be consumed by some component whose name is available but not the address.So it sends the name to the another helper that is view Resolver which tells the exact address to the frontcontroller.

Now Dispatcher servlet will have java object which is holding the data as well as the exact address where the actual data is send to client.So Dispatcher servlet goes to the View(Final response) now this will put the java object in to the html and sends the final response to the dispatcher servelt in desired pattern.Now dispatcher servlet sends the response to the browser(user).



From the Run method,the main application context is kicked off which in turn seraches for the classes annotated with @configuration,initilaizes all the declared beans in those configuration classes,and based upon the scope of the beans it will be store in jvm called as IOC Container.After the creation of all the beans it automatically configure dispatcher servlet and registers the default handler mappings, message conversts and all other basic things.

SpringApplication.run method internal flow:

create application context.

check application type

REgister the annotated class beans with the context

create an instance of TomcatEmbeddedServletContainer and adds the context.used to auto deploy our jars.

Spring Boot uses all the modules of Spring-like Spring MVC, Spring Data, etc. The architecture of Spring Boot is the same as the architecture of Spring MVC, except one thing: there is no need for DAO and DAOImpl classes in Spring boot.

Creates a data access layer and performs CRUD operation.

The client makes the HTTP requests (PUT or GET).

The request goes to the controller, and the controller maps that request and handles it. After that, it calls the service logic if required.

In the service layer, all the business logic performs. It performs the logic on the data that is mapped to JPA with model classes.

A JSP page is returned to the user if no error occurred.