**SPRINGBOOT TUTORIALS :-**

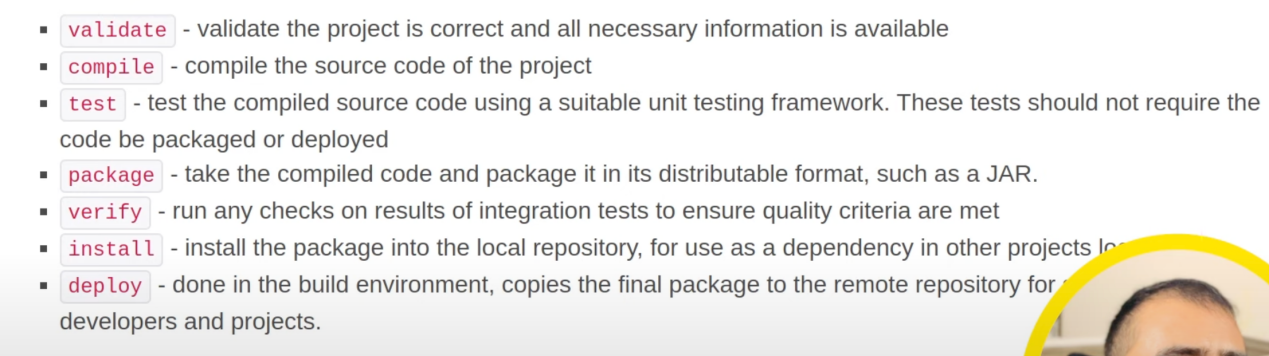
**MAVEN LANGUAGE USE IN SPRINGBOOT :-**

Singleton Maven is building tool whch will manage the dependencies which are required,

Maven is built around the concept of a build lifecycle, which defines the order of execution for goals.

The default Maven lifecycle consists of **8 major phases: Validate, Compile, Test, Package, Integration Test, Verify, Install, and Deploy.**

Each phase represents a specific step in the build process and has its own set of goals to be executed.



**STRUCTURE OF SPRINGBOOT PROJECT :-**

Singleton scope and prototype scope in springboot - By default spring framework will provide an object without or with creating. even if we need or no need of object.

we have an embedded TOMCAT SERVER in spring frameword by default.

**ANNOTATIONS in Spring boot :-**

@**Component** - as soon as we gave this annotation to the class , the spring container will be able to give the object for that particular class  
**Definition**: @Component It indicates that an annotated class is a Spring bean. Bean is nothing but a object in terms of Spring boot

**@AutoWired -** access By using this we will be able to connect different classes and only the object will be instantiated.

**Definition**: @Autowired is used for automatic dependency injection. It allows Spring to resolve and inject collaborating beans into your bean.

@Autowired

private Laptop laptop; //Laptop is a class name , laptop is the object we created using autowired.

**@Scope(value= “prototype”)** - By using this we wont get anobject by default( spring framework will not create an object by default) , so only if we create object by our own we will be able to access it.

**@Service :-**

**@Repository :-**

**@Controller :-**

### **@RestController :-**

**@Configuration :-**

**@SpringBootApplication :-**

**Definition**: @SpringBootApplication is a convenience annotation that combines @Configuration, @EnableAutoConfiguration, and @ComponentScan. It is used to mark a configuration class that declares one or more @Bean methods and triggers auto-configuration and component scanning.

This will read for only the classes of similar package, for other packages it wont work.

**@Bean :-**

**@GetMapping :-**

**@PostMapping:-**

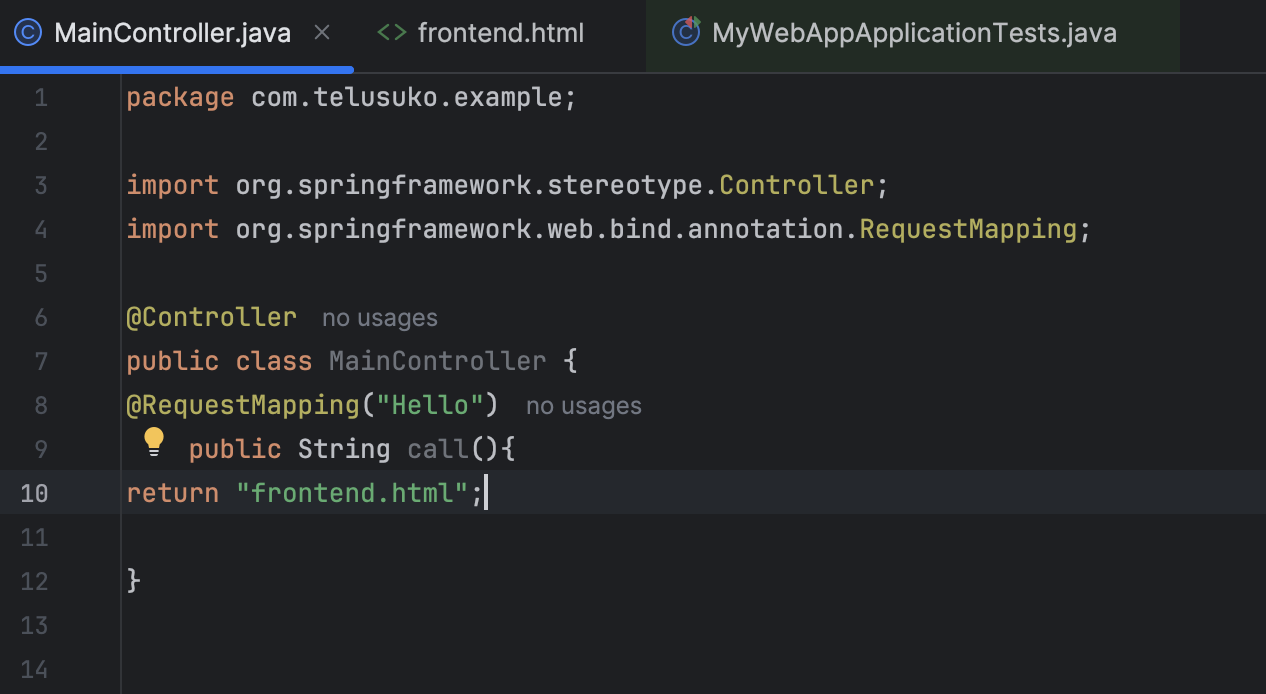
**@RequestMapping:-**

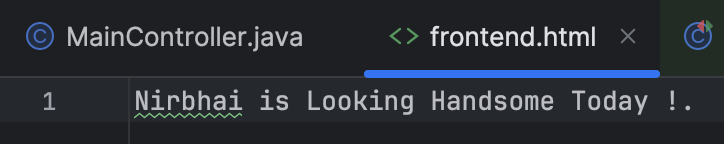
**@ResponseBody:-**

Spring container

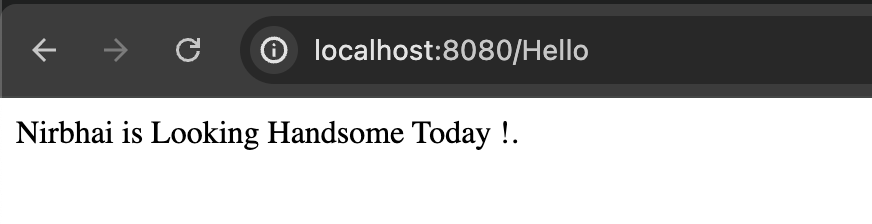
Spring bean -

Application.properties - Configuration prefix and sufix

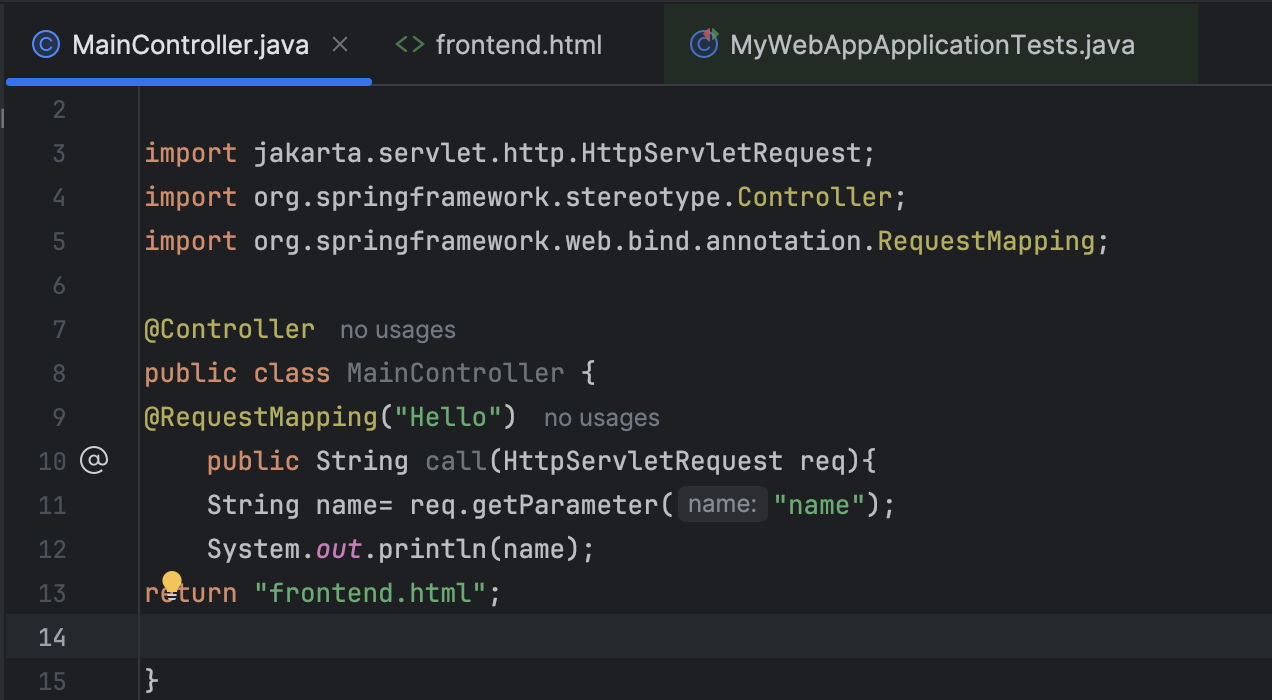




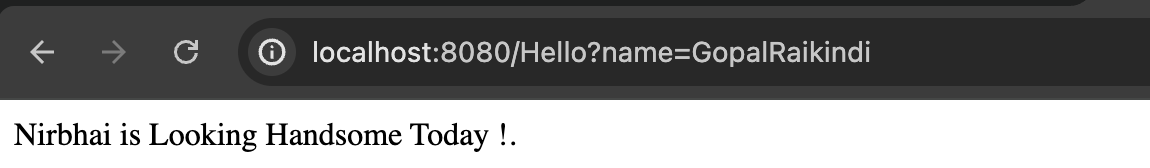
OUTPUT :-



SENDING DATA THROUGH REQUEST PARAMETER :-



INPUT :-



OUTPUT :- (We wrote in print statement so it will print in system console of IDE)



Now we need the req parameter of which the user is giving to get printed in the Page (through page which we are returning).

So in order to achieve this we will achieve through session.

**What is CLIENT SIDE CODE AND SERVER SIDE CODES IN WEB DEVELOPMENT :-**

### What is Client-Side and Server-Side?

#### Client-Side

* **Definition**: This is the part of the web application that runs in the user's web browser. It includes everything that the user interacts with directly.
* **Technologies Used**: HTML, CSS, JavaScript.
* **Example**: When you click a button on a web page, the code that handles that click is client-side code.

#### Server-Side

* **Definition**: This is the part of the web application that runs on a web server. It handles the logic, database interactions, and processes data before sending the result to the client.
* **Technologies Used**: Java, Python, Node.js, PHP, Ruby, etc.
* **Example**: When you submit a form on a web page, the data is sent to the server, which processes it (e.g., saves it to a database) and then sends a response back to the client.

### How Code Flows in a Web Application

1. **Client-Side Interaction**:

A user opens a web browser and enters a URL (e.g., [www.example.com](http://www.example.com" \t "/Users/dq-mac-air-m1/Documents/All FILES Github/aem-files/x/_new)).  
The browser sends a request to the server for that web page.

1. **Server-Side Processing**:  
   \* The server receives the request.  
   \* The server processes the request (e.g., fetches data from a database).  
   \* The server generates a response (e.g., an HTML page with data) and sends it back to the client's browser.  
     
   3. **Client-Side Rendering**:  
   The browser receives the response from the server.  
   The browser renders the HTML, CSS, and JavaScript to display the web page to the user.  
     
   **SUMMARY:-**

· **Client-Side**: Runs in the browser. Handles user interactions and displays data. Uses technologies like HTML, CSS, and JavaScript.

· **Server-Side**: Runs on the server. Processes requests, handles business logic, and interacts with databases. Uses technologies like Node.js, Python, Java, etc.

· **Code Flow**: The user interacts with the client-side code, which sends requests to the server-side code. The server processes these requests and sends responses back to the client.