**58.2. Spring REST - Creating a Spring REST Controller**

**Spring REST Controller**:

Here we will develop a Spring REST Controller.

Spring RestController annotation is a convenience annotation that is itself annotated with @Controller and @ResponseBody. This annotation is applied to a class to mark it as a request handler.

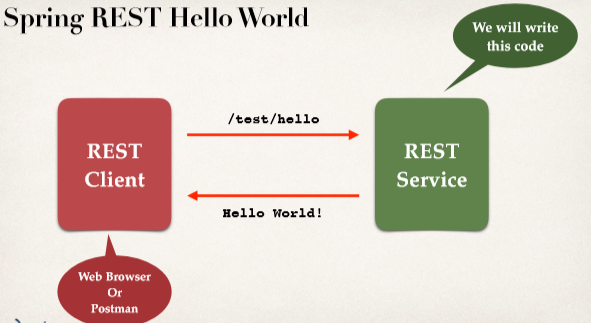
Spring RestController annotation is used to create RESTful web services using Spring MVC. Spring RestController takes care of mapping request data to the defined request handler method. Once response body is generated from the handler method, it converts it to JSON or XML response.

**Spring REST Support**:

* Spring Web MVC provides support for Spring REST
* New annotation **@RestController** 
  + Extension of **@Controller**
  + Handles REST requests and responses
* Spring REST will also automatically convert Java POJOs to JSON
  + As long as the Jackson project is on the class path or pom.xml

**Spring REST Hello World**:

Here we have "Rest Service" and "Rest Client". Here we will basically make a request for a "**/test/hello**". This REST service will response with "Hello World".





**Code**:

**package** com.ruhul.springdemo.rest;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/test")

**public** **class** DemoRestController {

// add code for the "/hello" endpoint

@GetMapping("/hello")

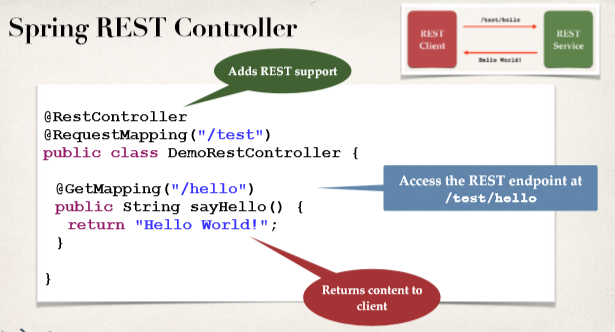
**public** String sayHello() {

**return** "Hello World";

}

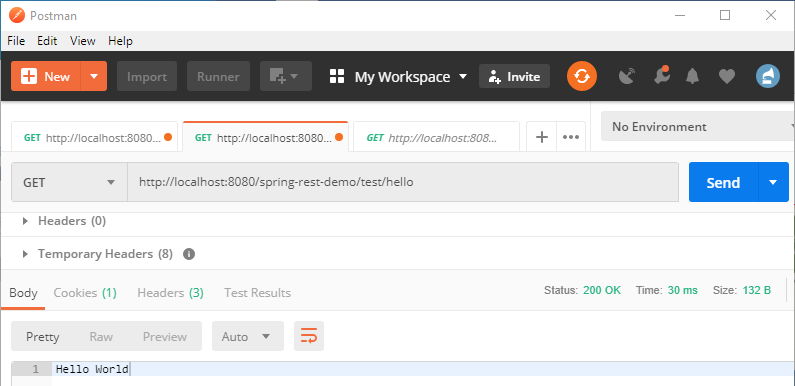
}

**Spring REST Controller**:



**Testing with rest client**:

URL: <http://localhost:8080/spring-rest-demo/test/hello>



**Development process (Step-by-Step)**:

1. Add Maven dependency for Spring MVC and Jackson project
2. Add code for All Java Config: @Configuration
3. Add code for All Java Config: Servlet Initializer
4. Create Spring REST Service using @RestController

**Step-1: Add Maven dependency for Spring MVC and Jackson project**:

<!-- Add Spring MVC and REST support -->

<!-- Spring MVC also have REST support -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.0.5.RELEASE</version>

</dependency>

<!-- Add Jackson for JSON converters -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.5</version>

</dependency>

<!-- Add Servlet support for Spring's AbstractAnnotationConfigDispatcherServletInitializer -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

</dependency>

**Step-2: Add code for All Java Config: @Configuration:**

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration

@EnableWebMvc

@ComponentScan("com.ruhul.springdemo")

**public** **class** DemoAppConfig {

}

**Web App Initializer:**

* Spring MVC provides support for web app initialization
* we have to makes sure that code is automatically detected
* Our code is used to initialize the servlet container

**AbstractAnnotationConfigDispatcherServletInitializer**

When we are building Spring MVC applications and also REST application, then we will make use this.

**Step-3: Add code for All Java Config: Servlet Initializer:**

**package** com.ruhul.springdemo.config;

**import** org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

**public** **class** MySpringMvcDispatcherServletInitializer **extends**

AbstractAnnotationConfigDispatcherServletInitializer {

@Override

**protected** Class<?>[] getRootConfigClasses() {

**return** **null**;

}

@Override

**protected** Class<?>[] getServletConfigClasses() {

**return** **new** Class[] { DemoAppConfig.**class** };

}

@Override

**protected** String[] getServletMappings() {

**return** **new** String[] { "/" };

}

}

**Step-4: Create Spring REST Service using @RestController:**

**package** com.ruhul.springdemo.rest;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

@RequestMapping("/test")

**public** **class** DemoRestController {

// add code for the "/hello" endpoint

@GetMapping("/hello")

**public** String sayHello() {

**return** "Hello World";

}

}

**App Enhancement with Default Page**:

Now if we run our application we will get "404" error. We have to solve it. But our REST still work. Link: <http://localhost:8080/spring-rest-demo/test/hello>

For solve this problem we have to add a web page in webapp root directory.

**File: index.jsp**:

**Location: src/main/webapp**:

<html>

<head>

<title>Spring Rest Demo</title>

</head>

<body>

<h1>Spring REST DEMO</h1>

<hr />

<a href=*"*${pageContext.request.contextPath}*/test/hello"*>Hello</a>

<br />

</body>

</html>

**Note**:

After adding the link (<a> tag) in the "**index.jsp**" file if we will get an error then we have to add the dependency in "**pom.xml" file.**

<!-- Add support for JSP ... get rid of Eclipse error -->

<dependency>

<groupId>javax.servlet.jsp</groupId>

<artifactId>javax.servlet.jsp-api</artifactId>

<version>2.3.1</version>

</dependency>

58.2. Spring REST - Creating a Spring REST Controller