**REST API** – Representational State Transfer

* Rest API provides only data either in JSON / XML format.
* Representational state transfer means instead of sending java object to client we will be sending state of the object
* JSON/XML because as data should be in some format.
* Rest API acts as a bridge between Client User Interface and Data Base.
* REST calls can be made over HTTP
* REST is language independent
* We can call it as REST API, REST Web Services, REST Services, Restful API, RESTful Web Services, and RESTful Services. Generally all mean the same thing.

**JSON**

* **JSON =** JavaScript Object Notation.
* JSON is the data format for storing and exchanging data i.e. It is just a plain text data.

JSON:

{

“id” : 1,

“name” : “ABC”,

“email” : “abc@gmail.com”,

“marks” : [99 , 85 , 67 ,80],

“address” : {

“house-num” : “H/011”,

“pin” : 700789

},

“hobbies” : [{“title” : “playing” , “frequency” : 5} , {“title” : “Drawing” , “frequency” : 4}]

}

Java POJO:

public class UserDetails{

private int id;

private String name;

private String email;

private int[] marks;

private Address address;

private Hobbies[] hobbies;

//generate getters() and setters() for all

}

**Data Binding:-**

* Data binding is the process of Converting JSON data to Java POJO and vice versa.

**Spring REST** - **JSON Data Binding with Jackson**

* **Spring uses the Jackson project behind the scenes.**
* **Jackson handles data binding between JSON and Java POJO.**
* **By default, Jackson will call appropriate getters/setters method of Java POJO.**
* **JSON to Java POJO….call setter method on POJO.**

**Maven Dependency for Jackson:-**

<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->

<dependency>

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

<artifactId>jackson-databind</artifactId>

<version>2.9.0</version>

</dependency>

**Note:-**

* In rest web services we will receive input from client/ UI as JSON format but in order to process data in backend we need to convert / map that JSON data into Java POJO.
* That conversion / mapping is done by spring with the help of Jackson behind the scenes.

**HTTP Response – Status Code**

|  |  |  |
| --- | --- | --- |
| **Code Range** | **Description** | **Example** |
| 100 - 199 | Informational |  |
| 200 – 299 | Successful | i.e. status code = 200 means OK |
| 300 – 399 | Redirection |  |
| 400 – 499 | Client error | 404 = File not found ,401=Authentication required |
| 500 – 599 | Server error | 500 = Internal Server Error |

**Client Tool:-**

* For testing the REST API client tool are used.
* Most common Client tool is **POSTMAN**.
* [www.getpostman.com](http://www.getpostman.com)

**Spring REST:-**

**@RestController**

* Extension of @Controller
* Handles REST requests and response
* Spring REST automatically converts Java POJO to JSON as long as the Jackson project in pom.xml file.

**Spring REST Controller**

**@RestController** /\*Adds REST support\*/

**@RequestMapping ("/test")**

public class DemoRestController{

**@GetMapping("/hello")** /\* Access the rest end point at /test/hello\*/

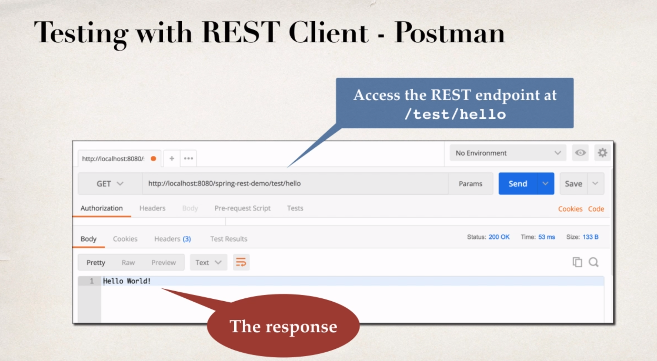
public String sayHello()

{

return "Hello world!"; /\* Returns content to client\*/

}

}



**Spring Rest – REST Controller**

Development Process:-

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

Step 2:- Add code for all java config : @Configuration

Step 3:- Add code for all java config : Servlet Initializer

Step 4:- Create Spring REST service using @RestController

**Step 1:-**

File :pom.xml

**<!--** [**https://mvnrepository.com/artifact/org.springframework/spring-webmvc Add**](https://mvnrepository.com/artifact/org.springframework/spring-webmvc%20Add) **Servlet Support-->**

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.2.1.RELEASE</version>

</dependency>

**<!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->**

<dependency>

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

<artifactId>jackson-databind</artifactId>

<version>2.9.5</version>

</dependency>

**<!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->**

<dependency>

<groupId>javax.servlet</groupId>

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

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

**Step 2:-**

@Configuration

@EnableWebMvc

@ComponentScan(basePackages = "com.abc.springDemo")

public class DemoAppConfig{

}

**Step 3:-**

* Used to initialize the servlet container
* Extend the abstract class **AbstractAnnotationConfigDispatcherServletInitializer**
* Override required methods
* Specify servlet mapping and location of app config

i.e. overridden methods are

@override

protected class<?>[] getServletConfigClasses(){

return new Class[] {DemoAppConfig.class};

}

@override

protected String[] getServletMappings(){

return new String[] {"/"};

}

**Step 4:-**

**@RestController** /\*Adds REST support\*/

**@RequestMapping ("/test")**

public class DemoRestController{

**@GetMapping("/hello")** /\* Access the rest end point at /test/hello\*/

public String sayHello()

{

return "Hello world!"; /\* Returns content to client\*/

}

}