# Introduction to RESTful Web Services

* REST stands for REpresentational State Transfer. It is developed by Roy Thomas Fielding, who also developed HTTP.
* The main goal of RESTful web services is to make web services more effective. RESTful web services try to define services using the different concepts that are already present in HTTP. REST is an architectural approach, not a protocol.
* It does not define the standard message exchange format. We can build REST services with both XML and JSON.
* JSON is more popular format with REST(also default format) .

## The key abstraction is a resource in REST.

A resource can be anything. It can be accessed through a Uniform Resource Identifier (URI). For example:

The resource has representations like XML, HTML, and JSON. When we request a resource, we provide the representation of the resource.

## The important methods of HTTP are:

GET: It reads a resource.

PUT: It updates an existing resource.

POST: It creates a new resource.

DELETE: It deletes the resource.

For example, if we want to perform the following actions in the social media application, we get the corresponding results.

POST /users: It creates a user.

GET /users/{id}: It retrieves the detail of a user.

GET /users: It retrieves the detail of all users.

DELETE /users: It deletes all users.

DELETE /users/{id}: It deletes a user.

GET /users/{id}/posts/post\_id: It retrieve the detail of a specific post.

POST / users/{id}/ posts: It creates a post of the user.

HTTP also defines the following standard status code:

404: RESOURCE NOT FOUND

200: SUCCESS

201: CREATED

401: UNAUTHORIZED

500: SERVER ERROR

## RESTful Service Constraints :

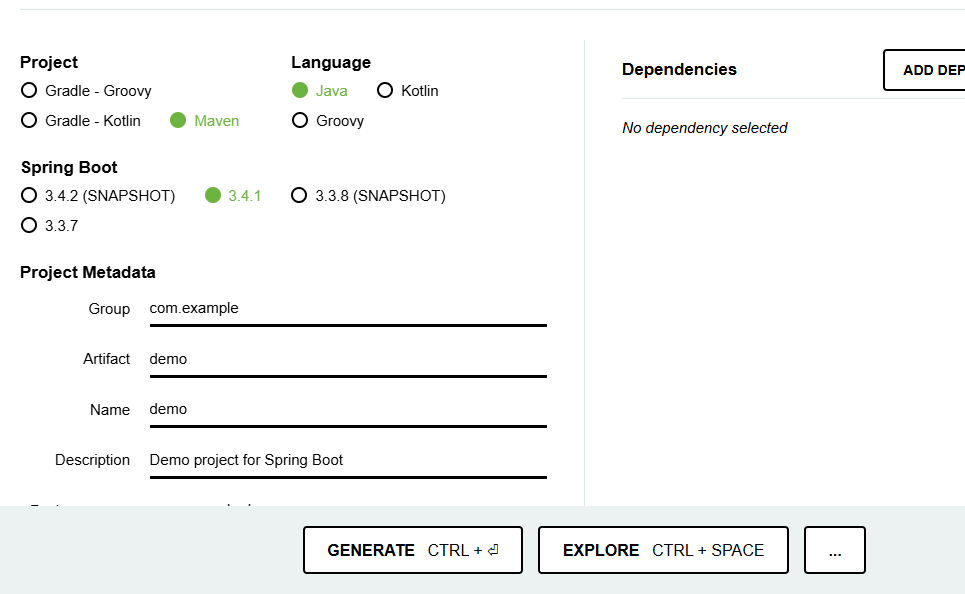
* There must be a service producer and service consumer.
* The service is stateless.
* The service result must be cacheable.
* The interface is uniform and exposing resources.
* The service should assume a layered architecture.

## Advantages of RESTful web services :

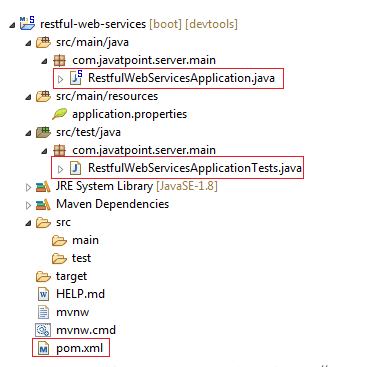
* RESTful web services are platform-independent.
* It can be written in any programming language and can be executed on any platform.
* It provides different data format like JSON, text, HTML, and XML.
* It is fast in comparison to SOAP because there is no strict specification like SOAP.
* These are reusable.
* They are language neutral.

# Initializing a RESTful Web Services Project with Spring Boot

## Creating project with : **start.spring.io**



## Project Structure :



**Add below dependencies and create above project structure :**

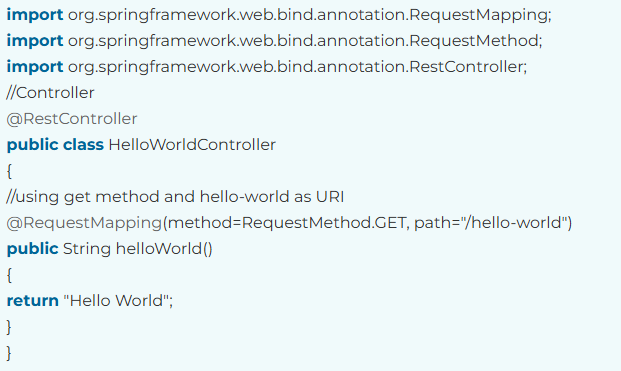
**Spring starter core , web , jpa ,devtools,h2**

## Creating a Hello World Service

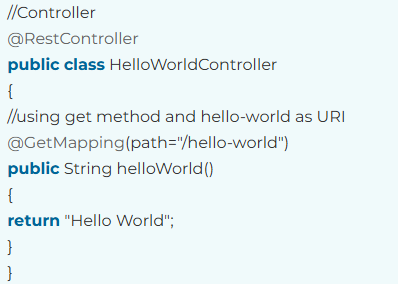
**Step 1:** Create a new class with the name HelloWorldController in the package

**Step 2:** Whenever we create a web service, we need to define two things Get method and the URI. Now create the helloWorld() method which returns the string "Hello World." If we want to tell the spring MVC that it is going to handle the REST request, we have to add @RestController annotation. Now it becomes a rest controller which can handle the Rest request.

The next thing we have to do is create a mapping for the method. Add @RequestMapping annotation just above the helloWorld() method. The HelloWorldController looks like the following:



We can also improve the above code by using the @GetMapping annotation instead of @RequestMapping. Here the method specification is not required.



Step 3: Run the RestfulWebServiceApplication. It displays the string Hello World on the browser.

## @RestController @RequestMapping @GetMapping Annotation

## Usage:

- ‘@RequestMapping’: This is a more general-purpose annotation that can be used to map methods to HTTP requests, specifying the HTTP method as a parameter. It can be used to handle various HTTP methods like GET, POST, PUT, DELETE, and more.

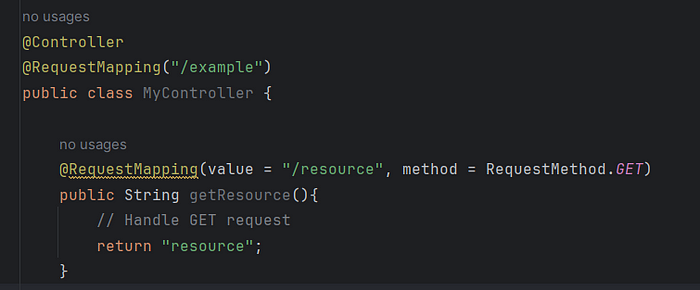
- ‘@GetMapping’ : This is a specialized version of ‘@RequestMapping’ that is designed specifically for mapping HTTP GET requests. It’s a shortcut for specifying that a method should handle only GET requests. It’s more concise and expressive for handling GET requests.

## Conciseness:

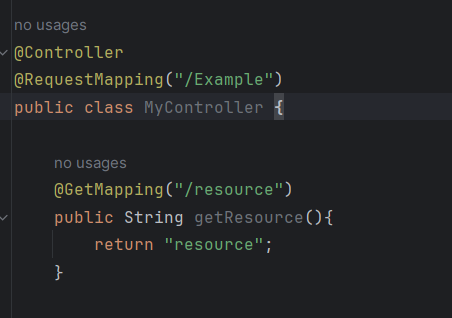
- ‘@GetMapping’ is more concise and readable when you only need to specify that a method handles GET requests. This can make your code easier to understand and maintain.

Here’s an example to illustrate the difference:

Using ‘@RequestMapping’:



Using ‘@GetMapping’:



In the second example using ‘GetMapping’, It’s clear that the ‘getResource’ method is specifically intended to handle GET requests, making the code more concise and self-explanatory.

In summary, ‘@RequestMapping’ is more general and versatile, allowing you to map methods to various HTTP methods, while ‘@GetMappling’ is a specific shortcut for handling GET requests, making the code more expressive and easier o understand when dealing with GET request. The choice between them depends on your specific use case and coding style preferences.

# Enhancing the Hello World Service to Return a Bean

Step 1: Create a helloWorldBean() method in HelloWordController.java file. Map the URI to "/hello-world-bean" and return HelloWorldBean.

//Controller

@RestController

public class HelloWorldController

{

//using get method and hello-world URI

@GetMapping(path="/hello-world")

public String helloWorld()

{

return "Hello World";

}

@GetMapping(path="/hello-world-bean")

public HelloWorldBean helloWorldBean()

{

return new HelloWorldBean("Hello World"); //constructor of HelloWorldBean

}

}

Step 2: Create a class HelloWorldBean.

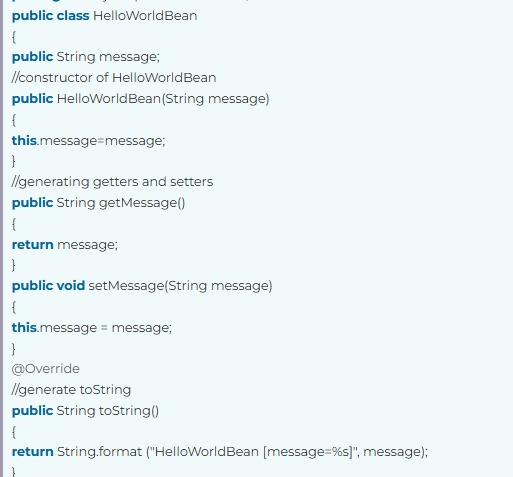
Step 3: Generate Getters and Setters.

Right-click -> Source -> Generate Getters and Setters -> check the box -> Ok

Step 4: Generate toString()

Right-click -> Source -> Generate toString().. -> Ok

**HelloWorldBean.java**



Step 5: Launch the HelloWorldController. The URL of the browser changes to localhost:8080/hello-world-bean.

It returns the message "Hello World" in JSON format.

{

message: "Hello World"

}

**24/12:**

@Controller v/s @RestController

| **@Controller** | **@RestController** |
| --- | --- |
| @Controller is used to mark classes as Spring MVC Controller. | @RestController annotation is a special controller used in RESTful Web services, and it’s the combination of @Controller and @ResponseBody annotation. |
| It is a specialized version of @Component annotation. | It is a specialized version of @Controller annotation. |
| In @Controller, we can return a view in Spring Web MVC. | In @RestController, we can not return a view. |
| @Controller annotation indicates that the class is a “controller” like a web controller. | @RestController annotation indicates that class is a controller where @RequestMapping methods assume @ResponseBody semantics by default. |
| In @Controller, we need to use @ResponseBody on every handler method. | In @RestController, we don’t need to use @ResponseBody on every handler method. |
| It was added to Spring 2.5 version. | It was added to Spring 4.0 version. |