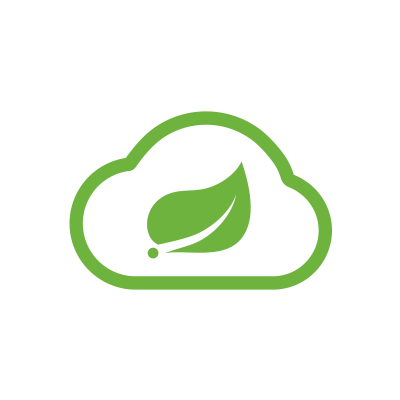
# Micro Services Spring Cloud

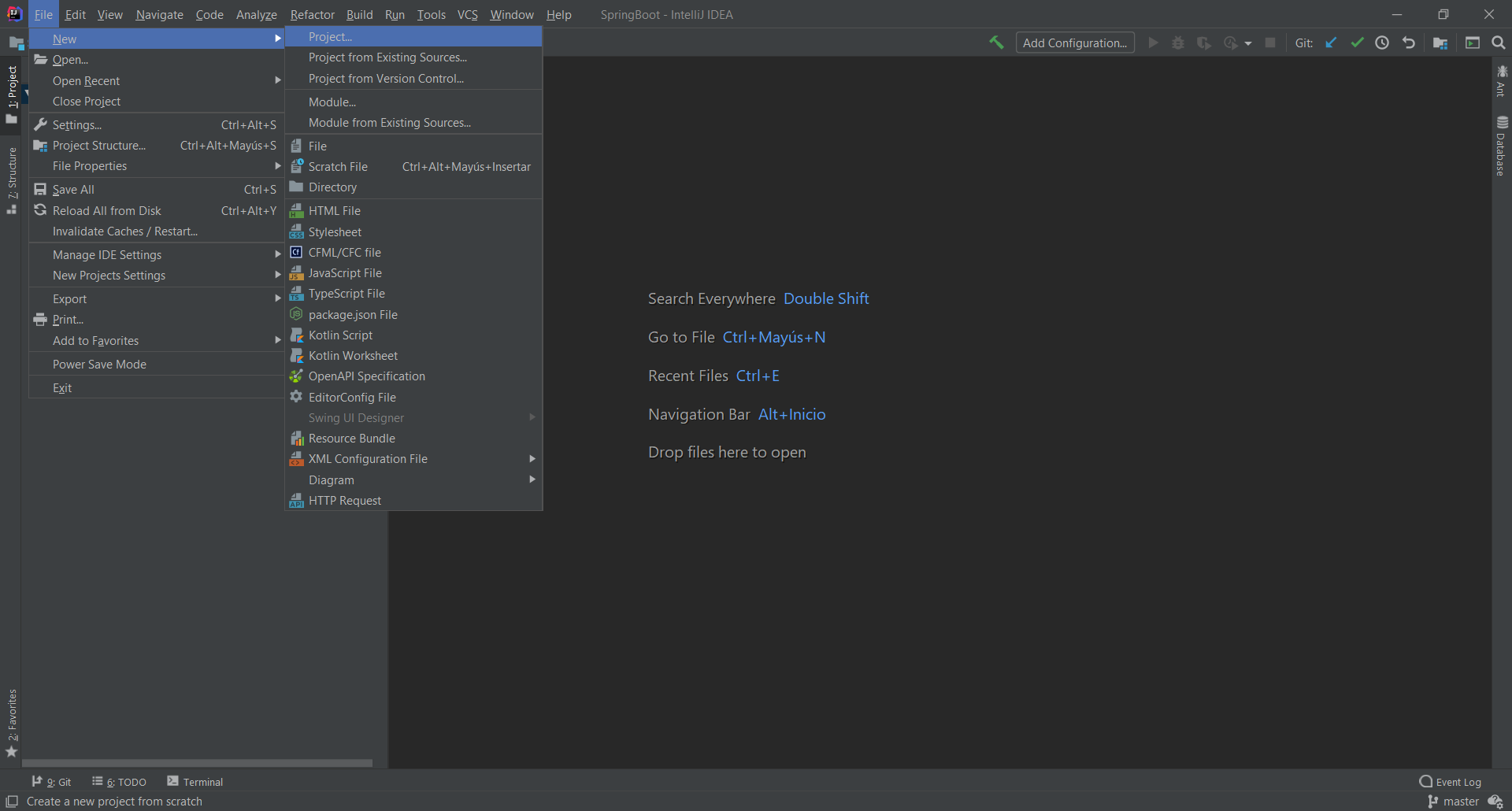


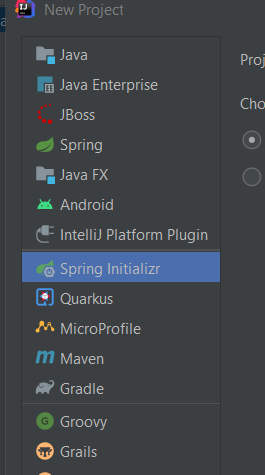
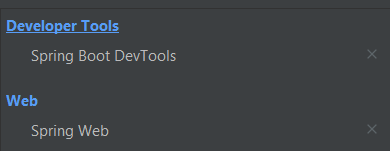
# Getting Started

There are many ways to start a project in spring, one is with initializr of <https://start.spring.io/>

Where you select your dependencies,

Another way its with the IDE, in this case Intellij IDE



 🡺Dependencies 

Then click on next 🡺 change name, click on next 🡺 select the dependencies. Click on next and finish🡺 Ready.

## @Rest Controller:

Documentation: <https://spring.io/guides/gs/rest-service/>

In Spring’s approach to building RESTful web services, HTTP requests are handled by a controller. These components are identified by the [@RestController](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/RestController.html) annotation,

The Rest controller will be the one that handle your HTTP request and the class can support multiple annotations in order to create a CRUD.

The annotation [@RestController](https://docs.spring.io/spring/docs/current/javadoc-api/org/springframework/web/bind/annotation/RestController.html) will allow the class to receive HTTP request but has to have a annotations on a function to do a specific function:

@RequestMapping will be the one that put the endpoint to the controller http://localhost:8080/**users**

@GetMapping is the one that will handle the request GET (to obtain objects or a response)

@PostMapping is the one that will handle the request POST (to create a new object)

@PutMapping is the one that will handle the request PUT (to edit a object already created)

@DeleteMapping is the one that will handle the request Delete (to delete a object already created)

Example:

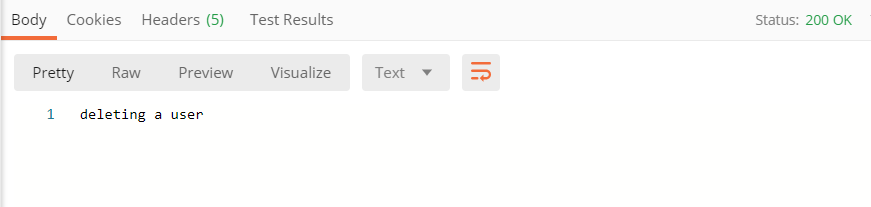
@RestController  
@RequestMapping("/users")  
public class Controller {  
  
 @GetMapping  
 public String getUser(){  
 return "getting user";  
 }  
  
 @PostMapping  
 public String postUser(){  
 return "user Created";  
 }  
  
 @PutMapping  
 public String editAUser(){  
 return "editing a user";  
 }  
  
 @DeleteMapping  
 public String deleteAUser(){  
 return "deleting a user";  
 }  
  
  
}

When we run the code, tomcat will deploy it to a local server by default the port is set in 8080

Using postman, we will send the request GET, PUT, POST, DELETE



Change the verb into , put, post ,delete



Status code:

<https://www.restapitutorial.com/httpstatuscodes.html>

Response of the http request

Rfttt

## @PathVariable

Path variable is an annotation that will allow take a value in the URL in order to user it in the function, for example when we want to obtain a specific user:

The annotation GetMapping will take the path (“/{**VARIABLE**}”) from the URL and will pass it to the function as an argument under the annotation @PathVariable dataType VARIABLE

@GetMapping(path = "/{VARIABLE}")  
public String getUser(@PathVariable String VARIABLE){  
 return "getting user: "+ VARIABLE;

@GetMapping(path = "/{userId}")  
public String getUser(@PathVariable String userId){  
 return "getting user: "+ userId;

PostMan:



## @RequestParam

Request params is an annotation that will take the parameters from the URL directly on the arguments of the function

Function(@RequestParams(value=”StringVariable”) dataType StringVariable, @RequestParams(value=”StringVariable2”) dataType StringVariable2)

@GetMapping()  
public String getUsers(@RequestParam(value = "page")int page,@RequestParam(value = "limit")int limit ){  
 return "getting users of page: "+page+" and limit of: "+ limit+" users";  
}



### Optional parameters:

To make optional parameters and void a error for Null Point Exception we will user **defaulValue=”value” AFTER value=”variable”**

@GetMapping()  
public String getUsers(@RequestParam(value = "page", defaultValue = "1")int page,@RequestParam(value = "limit", defaultValue = "50")int limit ){  
 return "getting users of page: "+page+" and limit of: "+ limit+" users";  
}



## Returning a Java Object as Returning value.

In this example, the controller will return a object in JSON

@GetMapping(path = "/{userId}")  
public UserRest getUserModel(@PathVariable String userId){  
 UserRest user1 = new UserRest();  
 user1.setFirstName("Jonathan");  
 user1.setLastName("Hernandez");  
 user1.setEmail("jonathan.hernandez.oliva@gmail.com");  
 return user1;  
}