**@JsonIgnore**

* Fieldurile notate cu @JsonIgnore nu sunt transformate de Jackson in JSON

**@JsonProperty(“nume”)**

* Modifica numele la field caer va aparea in JSON

**@ResponseBody**

* Anotatia data se pune impreuna cu @Controller pentru a spune ca orice obiect returnat trebuie serializat in JSON
* Defapt, @RestController = @Controller + @ResponseBody

**Swagger**

* Swagger un proiect ce genereaza REST API documentation pentru RESTful web services. Ofera si un user interface pentru a accesa RESTful web services creat de noi in browser
* Cu Swaggeer putem vedea capacitatile la un Rest API fara a accesa codul.
* **Este doar pentru RestController!**

Pasi:

1. Adaugam dependenta:

<dependency>  
 <groupId>org.springdoc</groupId>  
 <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>  
 <version>2.0.4</version>  
</dependency>

1. Accesam linkul:

localhost:8080/swagger-ui/index.html

* Putem adauga un bean ce va contine informatii despre REST creat de noi:

@Bean  
public OpenAPI openAPI(){  
 Server server1 = new Server();  
 server1.setUrl("localhost:8080");  
 server1.setDescription("Local host server");  
  
 Server server2 = new Server();  
 server2.setUrl("google.com");  
 server2.setDescription("google server");  
  
 Contact contact = new Contact();  
 contact.setEmail("test@mail.ru");  
 contact.setName("test");  
 contact.setUrl("facebook.com");  
  
 Info info = new Info()  
 .title("Tutorial")  
 .version("1.0")  
 .contact(contact)  
 .description("Test API exposes")  
 .termsOfService("ok.ru");  
  
 return new OpenAPI().info(info).servers(List.*of*(server1,server2));  
  
}

Le putem vedea si aici:

[localhost:8080/v3/api-docs](http://localhost:8080/v3/api-docs)

* @Tag(name=”nume”,description=”descriere”) – o punem cu @RestController ca sa apara info despre el in interfata grafica
* @Operation(summary=””,description=””,tag= {“”,””,..., }) – o punem deasupra la metoda
* @RestController  
  @Tag(name = "Student Controller",description = "This is a Controller for students")  
  @RequestMapping("/students")  
  public class StudentController {  
   @GetMapping("/")  
   @Operation(summary = "retrieve a student",  
   description = "get a random create student",  
   tags = {"students","get"})  
   public Student getStudent(){  
   Student student = new Student();  
   student.setFirstName("Mititiuc");  
   student.setLastName("Eduard");  
    
   return student;  
   }  
  }

**RequestInterceptor**

**RestTemplate**

RestTemplate restTemplate = new RestTemplate();  
ResponseEntity<String> responseEntity = restTemplate.getForEntity("http://headers.jsontest.com/",String.class);  
System.*out*.println(responseEntity.toString());

* getForEntity(url,Clasa.class)
* postForEntity()

......

Toate cu ForEntity returneaza ResponseEntity

* getForObject() – returneaza object

...

* postForLocation() – returneaza un URI

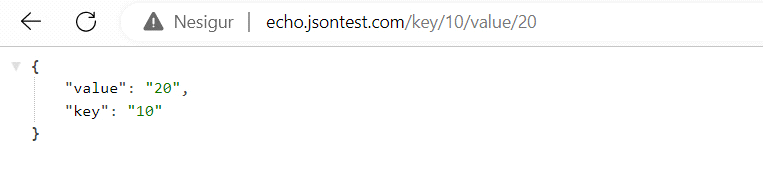
**WebClient**

* WebClient este o inlocuire pentru RestTemplate
* Acesta face parte din Spring WebFlux framework
* Dependenta:
* <dependency>  
   <groupId>org.springframework.boot</groupId>  
   <artifactId>spring-boot-starter-webflux</artifactId>  
  </dependency>
* Apoi cream un obiect ce va face un get request:

WebClient webClient = WebClient.*create*();  
WebClient.ResponseSpec responseSpec = webClient.get().uri("http://localhost:8080/api/20").retrieve();

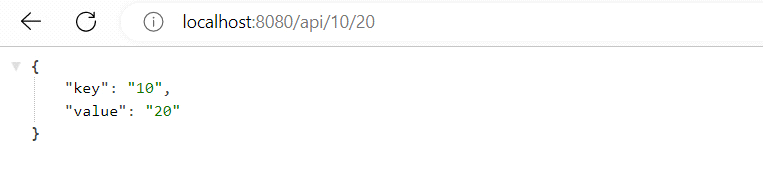
* Atentie! La etapa data requestul inca nu a fost facut, urmeaza inca a fi.
* Acum, daca vrem de ex sa luam niste date de alt site, am putea face asta liber in rest controller, dar o facem deocamdata tot cu localhost:
* @RestController  
  public class RestControllerApp {  
   @Autowired  
   private TeacherDAO teacherDAO;  
   @GetMapping("/api/{id}")  
   public Teacher teacher(@PathVariable Integer id){  
   System.*out*.println(teacherDAO.findById(id).get());  
   return teacherDAO.findById(id).get();  
   }  
   @GetMapping("/")  
   public Teacher teacher(){  
   WebClient webClient = WebClient.*create*();  
   WebClient.ResponseSpec responseSpec = webClient.get().uri("http://localhost:8080/api/{id}",21).retrieve();  
   Teacher teacher = responseSpec.bodyToMono(Teacher.class).block();  
   System.*out*.println(teacher);  
   return teacher;  
   }  
  }

Acum fie siteul:



El returneaza niste date in dependenta de numerele puse de noi.Noi putem prelua datele acestea de pe acest site si sa le returnam din rest controllerul nostru:

@GetMapping("/api/{key}/{value}")  
public KeyValue teacher(@PathVariable int key, @PathVariable int value){  
 WebClient webClient = WebClient.*create*();  
 WebClient.ResponseSpec responseSpec = webClient.get().uri("http://echo.jsontest.com/key/{key}/value/{value}",10,20).retrieve();  
 KeyValue keyValue = responseSpec.bodyToMono(KeyValue.class).block();  
 System.*out*.println(keyValue);  
 return keyValue;  
}



[Sending HTTP requests with Spring WebClient (reflectoring.io)](https://reflectoring.io/spring-webclient/)

[Mocking a WebClient in Spring | Baeldung](https://www.baeldung.com/spring-mocking-webclient)