**What is Swagger :**

[Swagger](https://swagger.io/) (Swagger 2) is a specification for describing and documenting a REST API. It specifies the format of the REST web services including URL, Resources, methods, etc. Swagger will generate documentation from the application code and handle the rendering part as well.

**Add dependency:**

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.6.1</version>

<scope>compile</scope>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.6.1</version>

<scope>compile</scope>

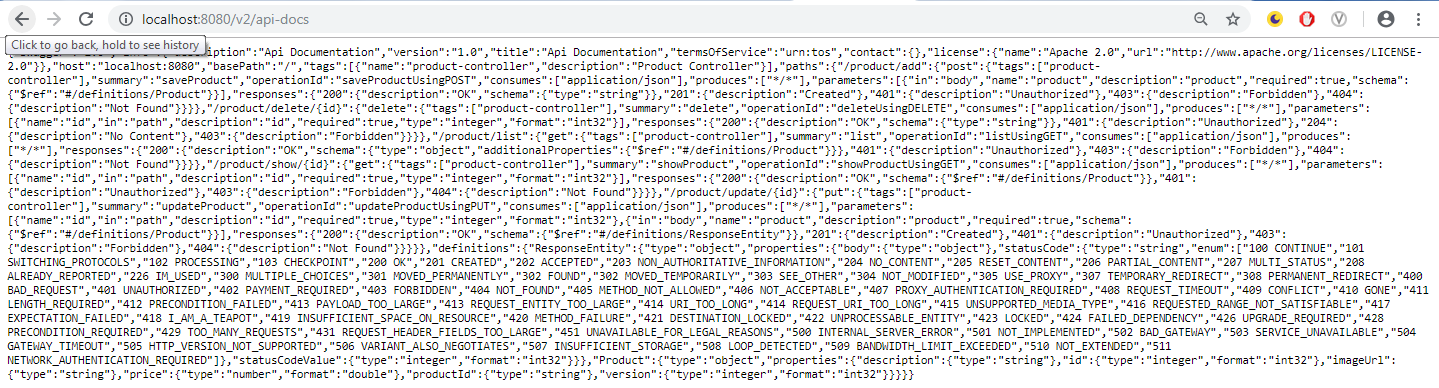
</dependency>

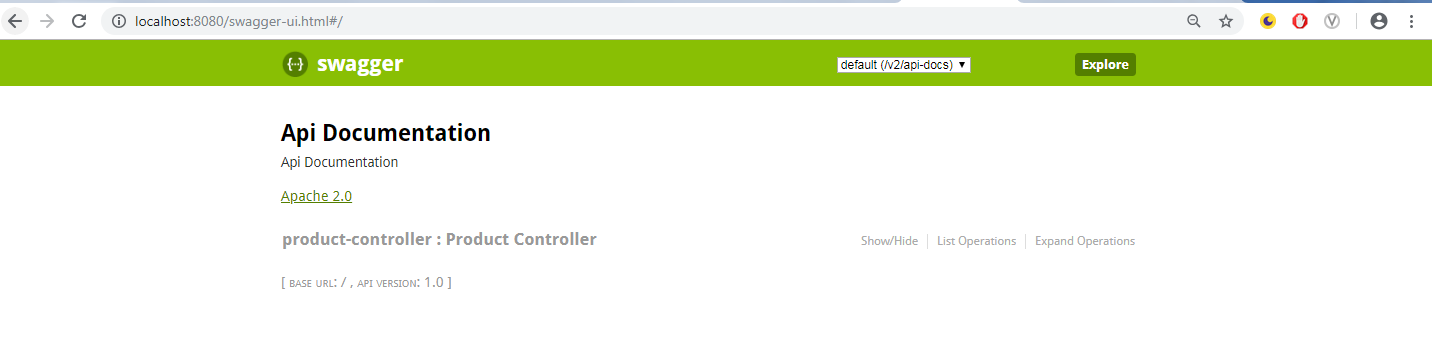
Example:

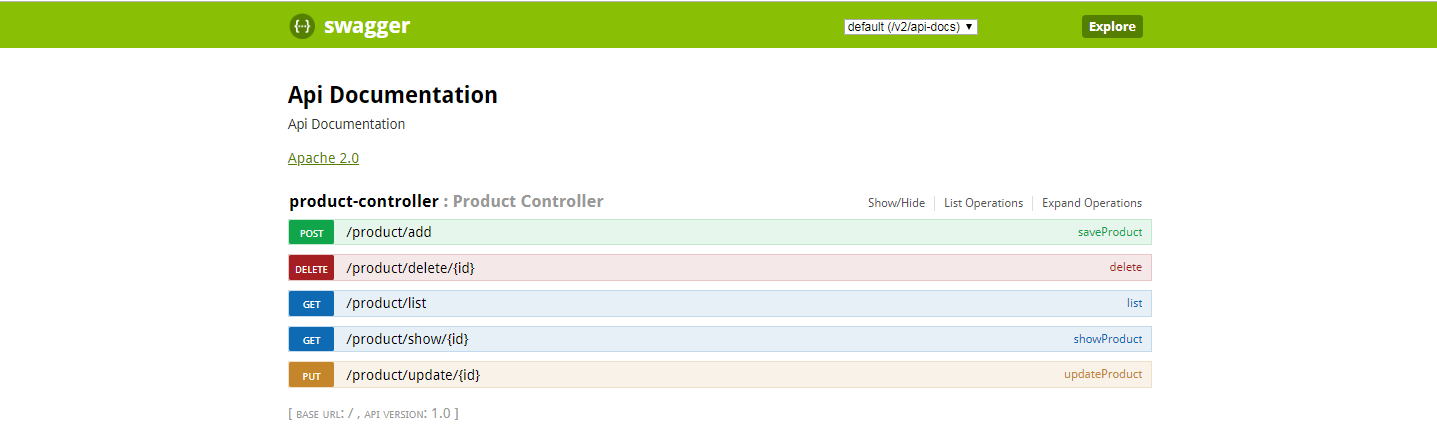
|  |
| --- |
| **Configuration swagger:**  @Configuration  @EnableSwagger2  **public** **class** SwaggerConfig {    @Bean  **public** Docket productApi() {  **return** **new** Docket(DocumentationType.***SWAGGER\_2***)  .select().apis(RequestHandlerSelectors.*basePackage*("com.mng.java.controller"))  .paths(*regex*("/product.\*"))  .build();  }  }  **Rest Controller:**  @RestController  @RequestMapping("/product")  **public** **class** ProductController {  @Autowired  **private** ProductService productService;  @Autowired  **public** **void** setProductService(ProductService productService) {  **this**.productService = productService;  }  //@RequestMapping(value = "/list", method= RequestMethod.GET)  @GetMapping(value = "/list")  **public** ResponseEntity<Map<Integer,Product>> list(Model model) {  Map<Integer, Product> productList = productService.listAllProducts();    **if** (productList == **null** || productList.isEmpty()){  System.***out***.println("no product found");  **return** **new** ResponseEntity<Map<Integer,Product>>(HttpStatus.***NO\_CONTENT***);  }    System.***out***.println("productService.listAllProducts() result: " + productList);  **return** **new** ResponseEntity<Map<Integer,Product>>(productList, HttpStatus.***OK***);  }  //@RequestMapping(value = "/show/{id}", method= RequestMethod.GET)  @GetMapping(value = "/show/{id}")  **public** ResponseEntity<Product> showProduct(@PathVariable Integer id, Model model) {  Product product = productService.getProductById(id);    **if** (product == **null**){  System.***out***.println("no product found");  **return** **new** ResponseEntity<Product>(HttpStatus.***NO\_CONTENT***);  }    System.***out***.println("productService.getProductById() result: " + product);  **return** **new** ResponseEntity<Product>(product, HttpStatus.***OK***);  }  //@RequestMapping(value = "/add", method = RequestMethod.POST)  @PostMapping(value = "/add")  **public** ResponseEntity<String> saveProduct(@RequestBody Product product) {  **if**(productService.exists(product.getId())) {  System.***out***.println("a product with name " + product.getId() + " already exists");  **return** **new** ResponseEntity<String>("alreday exixts", HttpStatus.***CONFLICT***);  }  productService.saveProduct(product.getId(),product);  System.***out***.println("productService.saveProduct() result: " + productService.listAllProducts());  **return** **new** ResponseEntity<String>("Product saved successfully", HttpStatus.***CREATED***);  }  //@RequestMapping(value = "/update/{id}", method = RequestMethod.PUT)  @PutMapping(value = "/update/{id}")  **public** ResponseEntity updateProduct(@PathVariable Integer id, @RequestBody Product product) {  Product productObj= productService.getProductById(id);    **if** (productObj == **null**){  System.***out***.println("product with id {} not found");  **return** **new** ResponseEntity<Product>(HttpStatus.***NOT\_FOUND***);  }  productObj.setProductId(product.getProductId());  productObj.setDescription(product.getDescription());  productObj.setImageUrl(product.getImageUrl());  productObj.setPrice(product.getPrice());  productService.saveProduct(product.getId(),productObj);    System.***out***.println("productService.updateProduct() result: " + productService.listAllProducts());  **return** **new** ResponseEntity("Product updated successfully", HttpStatus.***OK***);  }  //@RequestMapping(value="/delete/{id}", method = RequestMethod.DELETE)  @DeleteMapping(value = "/delete/{id}")  **public** ResponseEntity<String> delete(@PathVariable Integer id) {  System.***out***.println("Before productService.delete() result: " + productService.listAllProducts());  System.***out***.println("Products list size: " + productService.listAllProducts().size());    Product productObj= productService.getProductById(id);  **if**(productObj == **null**) {  System.***out***.println("Unable to delete with Id " + id + " not found");  **return** **new** ResponseEntity<String>("Unable to delete with Id",HttpStatus.***NOT\_FOUND***);  }  productService.deleteProduct(id);  System.***out***.println("after productService.delete() result: " + productService.listAllProducts());  System.***out***.println("Products list size: " + productService.listAllProducts().size());  **return** **new** ResponseEntity<String>("Product deleted successfully", HttpStatus.***OK***);  }  }  **Service:**  **public** **interface** ProductService {  Map<Integer,Product> listAllProducts();  Product getProductById(Integer id);  **void** saveProduct(Integer id,Product product);  **void** deleteProduct(Integer id);  **boolean** exists(Integer id);  }  Service Impl:  @Repository  **public** **class** ProductServiceImpl **implements** ProductService {    Map<Integer,Product> products = **null**;    @PostConstruct  **public** **void** generateProducts() {  products = **new** HashMap<>();    Product p1 = **new** Product();  p1.setId(101);  p1.setProductId("dell laptop");  p1.setDescription("Dell product configuration info");  p1.setImageUrl("http://localhost:8080/delImage");  p1.setPrice(50000);  p1.setVersion(5);    products.put(101, p1);    Product p2 = **new** Product();  p2.setId(102);  p2.setProductId("lenovo laptop");  p2.setDescription("lenovo product configuration info");  p2.setImageUrl("http://localhost:8080/lenovoImage");  p2.setPrice(40000);  p2.setVersion(4);    products.put(102, p2);    Product p3 = **new** Product();  p3.setId(103);  p3.setProductId("HP laptop");  p3.setDescription("HP product configuration info");  p3.setImageUrl("http://localhost:8080/hpImage");  p3.setPrice(60000);  p3.setVersion(2);    products.put(103, p3);    Product p4 = **new** Product();  p4.setId(102);  p4.setProductId("Acer laptop");  p4.setDescription("Acer product configuration info");  p4.setImageUrl("http://localhost:8080/AcerImage");  p4.setPrice(35000);  p4.setVersion(6);    products.put(104, p4);    }  @Override  **public** Map<Integer,Product> listAllProducts() {  **return** products;  }  @Override  **public** Product getProductById(Integer id) {  **return** products.get(id);  }  @Override  **public** **void** saveProduct(Integer id, Product product) {  products.put(id,product);  }  @Override  **public** **void** deleteProduct(Integer id) {  products.remove(id);  }  @Override  **public** **boolean** exists(Integer id) {  // **TODO** Auto-generated method stub  **return** products.containsKey(id);  }  } |

* @RestController was introduced in Spring MVC 4 and is a convenience annotation that itself is annotated with @Controller and @ResponseBody annotation. This eliminates the need of annotating your controller methods with @ResponseBody individually.
* @RequestMapping annotation is used to map URLs such as /users onto an entire class or a particular handler method. A class-level annotation like /users maps a specific request path onto a controller, with additional method-level annotations further narrowing the mapping for a specific HTTP request method like “GET”, “PUT”, “POST” or “DELETE” etc.
* RequestMethod is an enumeration of possible HTTP request methods like GET, PUT, POST, DELETE, etc.
* @PathVariable annotation indicates that a method parameter should be bound to a URI template variable. To process the @PathVariable annotation, Spring MVC needs to find the matching URI template variable by name. You can specify it in the annotation or, if the URI template variable name matches the method argument name, you can omit that detail.
* @RequestBody annotation indicates a method parameter should be bound to the body of the HTTP web request. Behind the scene, a HttpMessageConverter is responsible for converting from the HTTP request message to an object and converting from an object to the HTTP response body.
* @ResponseBody annotation indicates a method return value should be bound to the HTTP response body. By annotating your class with @RestController, you no longer need to add this annotation to every method individually.
* ResponseEntity extends from HttpEntity which allows you to add a HttpStatus code directly to the response. The ResponseEntity represents the entire HTTP response. You can add header information, status codes and add content to the body.
* HttpHeaders represents HTTP request and response headers. This class has some convenience methods for setting popular header types like Content-Type, Access-Control-Allow-Headers, etc.

On pointing your browser to http://localhost:8080/swagger-ui.html, you will see the generated documentation rendered by Swagger UI, like this:







**Customizing Swagger:**

Swagger documentation as it comes out of the box — but Swagger 2 has some great customization options.

Let’s start customizing Swagger by providing information about our API in the SwaggerConfig class

We can use the @Api annotation on our ProductController class to describe our API.

For each of our operation endpoints, we can use the @ApiOperation annotation to describe the endpoint and its response type, like this:

@ApiOperation(value = "View a list of available products", response = Iterable.class)

@RequestMapping(value = "/list", method= RequestMethod.GET,produces = "application/json")

public Iterable list(Model model){

Iterable productList = productService.listAllProducts();

return productList;

}

Swagger 2 also allows overriding the default response messages of HTTP methods. You can use the @ApiResponse annotation to document other responses, in addition to the regular HTTP 200 OK, like this.

## Swagger 2 Annotations for Model

You can use the @ApiModelProperty annotation to describe the properties of the Product model. With @ApiModelProperty, you can also document a property as required.

|  |
| --- |
| **public** **class** Product {  @ApiModelProperty(notes = "The database generated product ID")  **private** Integer id;  @ApiModelProperty(notes = "The auto-generated version of the product")  **private** Integer version;  @ApiModelProperty(notes = "The application-specific product ID")  **private** String productId;  @ApiModelProperty(notes = "The product description")  **private** String description;  @ApiModelProperty(notes = "The image URL of the product")  **private** String imageUrl;  @ApiModelProperty(notes = "The price of the product", required = **true**)  **private** **double** price;  }  **Controller:**  @RestController  @RequestMapping("/product")  @Api(value="online store", ~~description~~="Operations pertaining to products in Online Store")  **public** **class** ProductController {  @Autowired  **private** ProductService productService;  @Autowired  **public** **void** setProductService(ProductService productService) {  **this**.productService = productService;  }  //@RequestMapping(value = "/list", method= RequestMethod.GET)  @ApiResponses(value = {  @ApiResponse(code = 200, message = "Successfully retrieved list"),  @ApiResponse(code = 401, message = "You are not authorized to view the resource"),  @ApiResponse(code = 403, message = "Accessing the resource you were trying to reach is forbidden"),  @ApiResponse(code = 404, message = "The resource you were trying to reach is not found")  }  )  @ApiOperation(value="View a list of available products", response=List.**class**)  @GetMapping(value = "/list")  **public** ResponseEntity<Map<Integer,Product>> list(Model model) {  Map<Integer, Product> productList = productService.listAllProducts();    **if** (productList == **null** || productList.isEmpty()){  System.***out***.println("no product found");  **return** **new** ResponseEntity<Map<Integer,Product>>(HttpStatus.***NO\_CONTENT***);  }    System.***out***.println("productService.listAllProducts() result: " + productList);  **return** **new** ResponseEntity<Map<Integer,Product>>(productList, HttpStatus.***OK***);  }  //@RequestMapping(value = "/show/{id}", method= RequestMethod.GET)  @ApiOperation(value = "Search a product with an ID",response = Product.**class**)  @GetMapping(value = "/show/{id}")  **public** ResponseEntity<Product> showProduct(@PathVariable Integer id, Model model) {  Product product = productService.getProductById(id);    **if** (product == **null**){  System.***out***.println("no product found");  **return** **new** ResponseEntity<Product>(HttpStatus.***NO\_CONTENT***);  }    System.***out***.println("productService.getProductById() result: " + product);  **return** **new** ResponseEntity<Product>(product, HttpStatus.***OK***);  }  //@RequestMapping(value = "/add", method = RequestMethod.POST)  @ApiOperation(value = "Add a product")  @PostMapping(value = "/add")  **public** ResponseEntity<String> saveProduct(@RequestBody Product product) {  **if**(productService.exists(product.getId())) {  System.***out***.println("a product with name " + product.getId() + " already exists");  **return** **new** ResponseEntity<String>("alreday exixts", HttpStatus.***CONFLICT***);  }  productService.saveProduct(product.getId(),product);  System.***out***.println("productService.saveProduct() result: " + productService.listAllProducts());  **return** **new** ResponseEntity<String>("Product saved successfully", HttpStatus.***CREATED***);  }  //@RequestMapping(value = "/update/{id}", method = RequestMethod.PUT)  @ApiOperation(value = "Update a product")  @PutMapping(value = "/update/{id}")  **public** ResponseEntity updateProduct(@PathVariable Integer id, @RequestBody Product product) {  Product productObj= productService.getProductById(id);    **if** (productObj == **null**){  System.***out***.println("product with id {} not found");  **return** **new** ResponseEntity<Product>(HttpStatus.***NOT\_FOUND***);  }  productObj.setProductId(product.getProductId());  productObj.setDescription(product.getDescription());  productObj.setImageUrl(product.getImageUrl());  productObj.setPrice(product.getPrice());  productService.saveProduct(product.getId(),productObj);    System.***out***.println("productService.updateProduct() result: " + productService.listAllProducts());  **return** **new** ResponseEntity("Product updated successfully", HttpStatus.***OK***);  }  //@RequestMapping(value="/delete/{id}", method = RequestMethod.DELETE)  @ApiOperation(value = "Delete a product")  @DeleteMapping(value = "/delete/{id}")  **public** ResponseEntity<String> delete(@PathVariable Integer id) {  System.***out***.println("Before productService.delete() result: " + productService.listAllProducts());  System.***out***.println("Products list size: " + productService.listAllProducts().size());    Product productObj= productService.getProductById(id);  **if**(productObj == **null**) {  System.***out***.println("Unable to delete with Id " + id + " not found");  **return** **new** ResponseEntity<String>("Unable to delete with Id",HttpStatus.***NOT\_FOUND***);  }  productService.deleteProduct(id);  System.***out***.println("after productService.delete() result: " + productService.listAllProducts());  System.***out***.println("Products list size: " + productService.listAllProducts().size());  **return** **new** ResponseEntity<String>("Product deleted successfully", HttpStatus.***OK***);  }  }  **Swagger Config:**  @Configuration  @EnableSwagger2  **public** **class** SwaggerConfig {    @Bean  **public** Docket productApi() {  **return** **new** Docket(DocumentationType.***SWAGGER\_2***)  .select().apis(RequestHandlerSelectors.*basePackage*("com.mng.java.controller"))  .paths(*regex*("/product.\*"))  .build().apiInfo(metaData());  }    **private** ApiInfo metaData() {  ApiInfo apiInfo = **new** ApiInfo(  "Spring Boot Product REST API",  "Spring Boot Product REST API for Online Store",  "1.0",  "Terms of service",  **new** Contact("Nagendra Mekala", "https://springframework.guru/about/", "nagendra.kldm@gmail.com"),  "Apache License Version 2.0",  "https://www.apache.org/licenses/LICENSE-2.0");  **return** apiInfo;  }  } |

