Keretrendszer alapú programozás

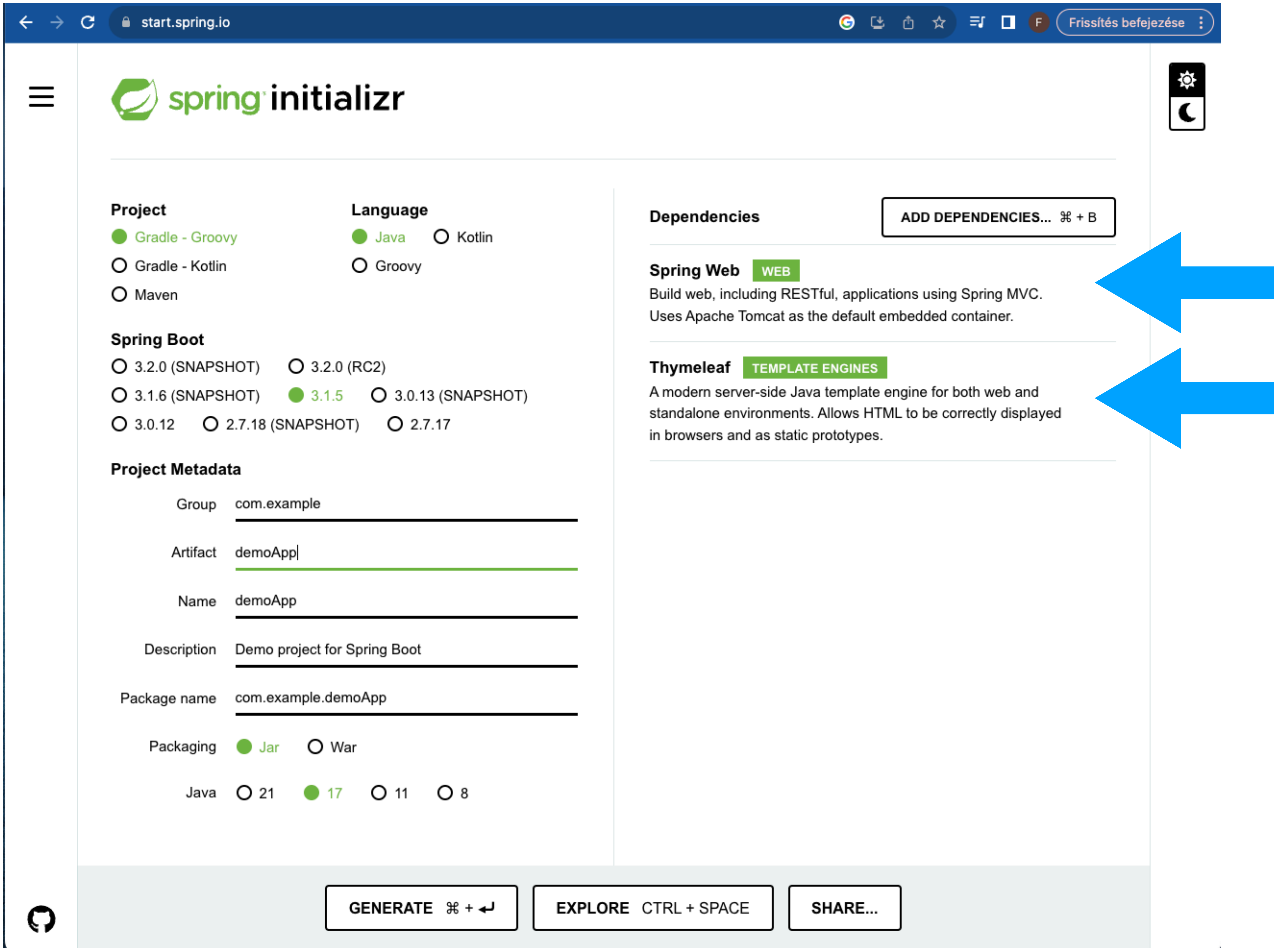
Fazekas Csaba

fazekas.csaba@uni-eszterhazy.hu

Spring Boot

spring initializr

https://start.spring.io/



Project
[✓ Gradle - Groovy] Gradle - Kotlin Maven

Language
[✓ Java] Kotlin Groovy

Spring Boot
3.2.0 (SNAPSHOT) 3.2.0 (RC2) 3.1.6 (SNAPSHOT) [✓ 3.1.5] 3.0.13 (SNAPSHOT) 3.0.12 2.7.18 (SNAPSHOT) 2.7.17

Project Metadata

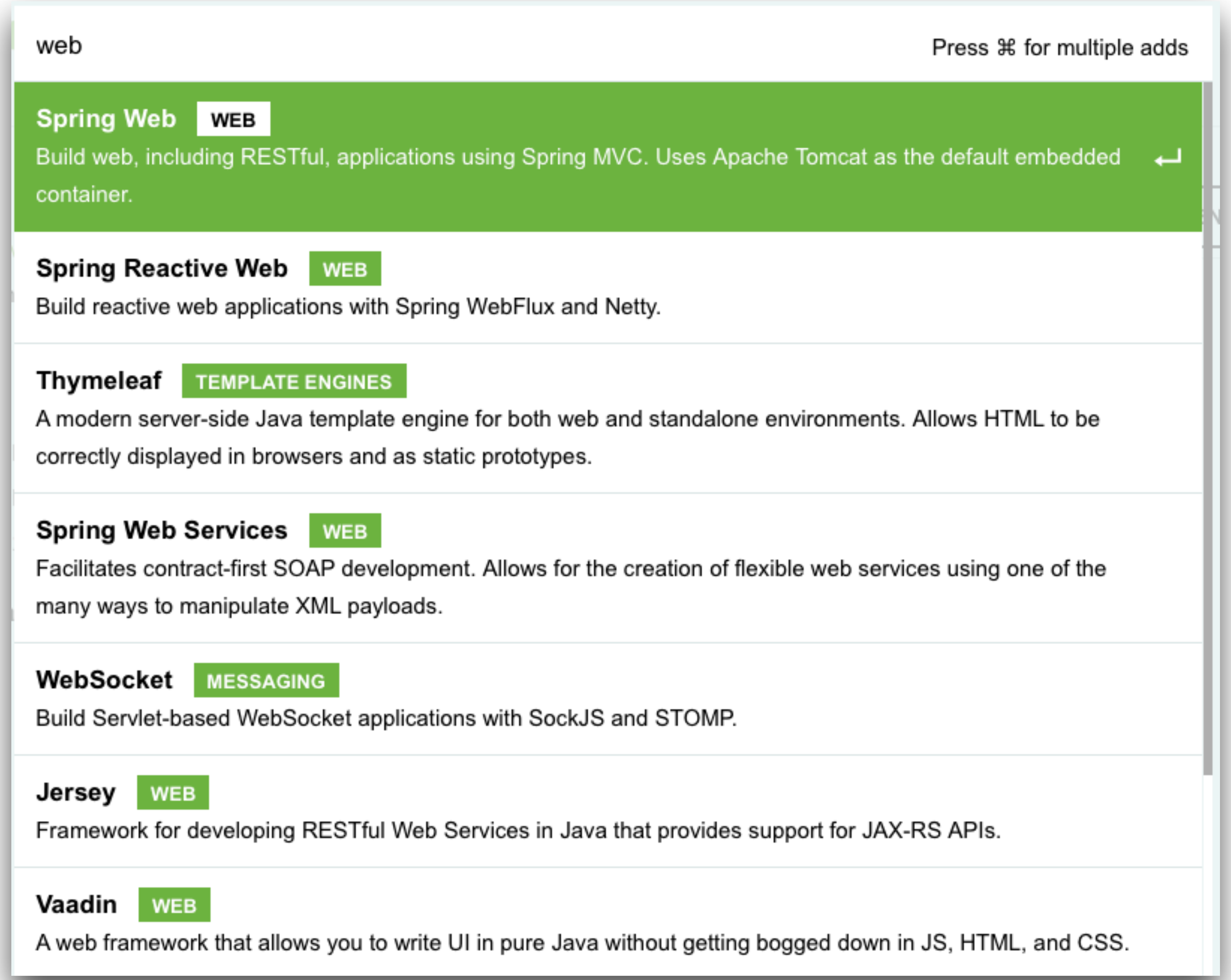
Group	com.example
Artifact	demoApp
Name	demoApp
Description	Demo project for Spring Boot
Package name	com.example.demoApp
Packaging	[✓ Jar] War
Java	21 [✓ 17] 11 8

Dependencies

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Boot



web

Press ⌘ for multiple adds

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Reactive Web WEB
Build reactive web applications with Spring WebFlux and Netty.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

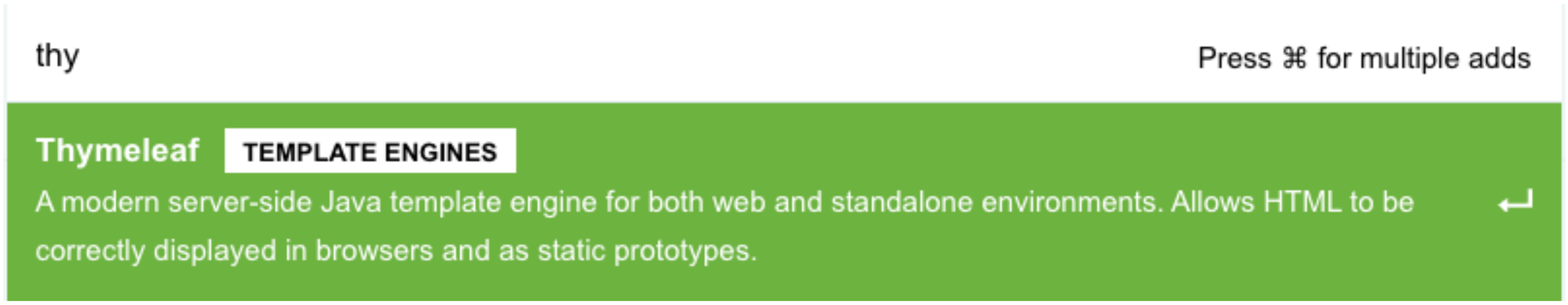
Spring Web Services WEB
Facilitates contract-first SOAP development. Allows for the creation of flexible web services using one of the many ways to manipulate XML payloads.

WebSocket MESSAGING
Build Servlet-based WebSocket applications with SockJS and STOMP.

Jersey WEB
Framework for developing RESTful Web Services in Java that provides support for JAX-RS APIs.

Vaadin WEB
A web framework that allows you to write UI in pure Java without getting bogged down in JS, HTML, and CSS.

Spring Boot

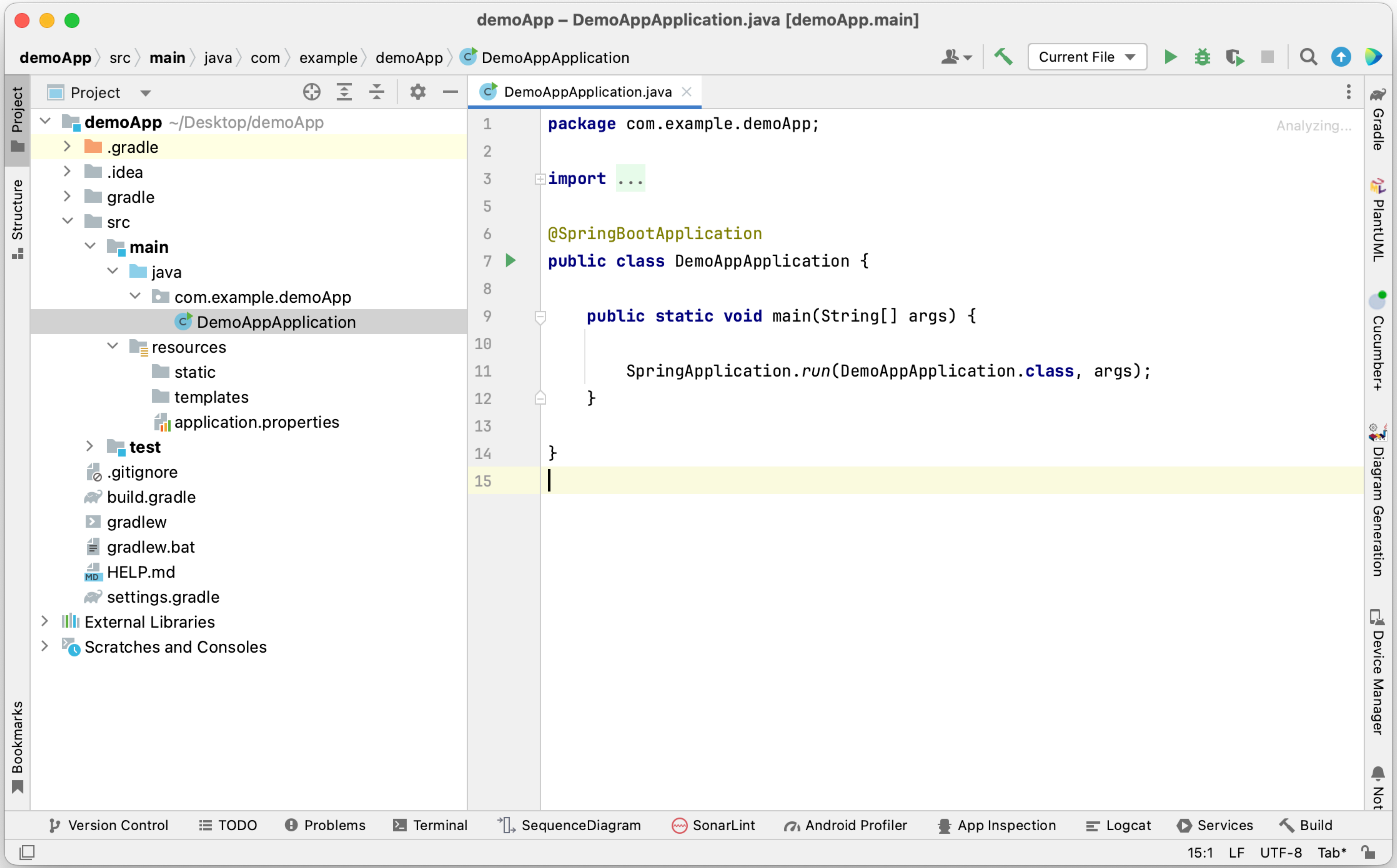

thy

Press ⌘ for multiple adds

Thymeleaf TEMPLATE ENGINES

A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

Spring Boot



```java
package com.example.demoApp;

import ...

@SpringBootApplication
public class DemoAppApplication {

    public static void main(String[] args) {

        SpringApplication.run(DemoAppApplication.class, args);
    }

}
```

Application Properties

Alkalmazás beállításai

- Parancssorban indításkor

```
java -jar demoApp-0.0.1-SNAPSHOT.jar —server.port=9000
```

- src/main/resources/**application.properties**

```
server.port=9000
spring.application.name=MyApp
```

- src/main/resources/**application.yaml**

```
server
   port: 9000
```

@Value

- application.properties tartalmát érhetjük el vele:

@Value(“${property_key_name}”)

```
@Value("${spring.application.name}")
```

- amennyiben nem elérhető akkor futásidőben: Illegal Argument exception. Emiatt default érték adható neki:

```
@Value(“${spring.application.name:MyApp}")
```

@Value

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoAppApplication {

   @Value(“${spring.application.name:MyApp}")
   String applicationName;

   public static void main(String[] args) {

      SpringApplication.run(DemoAppApplication.class, args);

   }

}
```

Kód futtatása az alkalmazás elindítása után

Application Runner
Command Line Runner

Kód futtatását teszik lehetővé ezek az interfészek a Spring Boot alkalmazás elindulása után.

Application Runner:

```
@SpringBootApplication
public class DemoAppApplication implements ApplicationRunner {

    @Value("${spring.application.name:MyApp}")
    String applicationName;

    public static void main(String[] args) {

        SpringApplication.run(DemoAppApplication.class, args);
    }

    @Override
    public void run(ApplicationArguments args) throws Exception {
        System.out.println(applicationName);
    }
}
```

Mi is a probléma?

Inversion of Control -> valahogy meg kell szereznünk vele szemben az irányítást a konténer felet!

Megoldás:

- Application Runner - ApplicationArguments paraméter
- Command Line Runner - String paraméter

Mindkettő kód futtatását teszik lehetővé a Spring Boot applikáció elindulása után.

Application Runner

```
@SpringBootApplication
public class DemoAppApplication implements ApplicationRunner
{

  @Value("${spring.application.name:MyApp}")
  String applicationName;

  public static void main(String[] args) {

    SpringApplication.run(DemoAppApplication.class, args);
  }

  @Override
  public void run(ApplicationArguments args) throws Exception
  {
    System.out.println(applicationName);
  }
}
```

Service komponens

@Service

- El lehet vele különíteni az üzleti logikát az @RestController osztályoktól.
- Két részre bontható a felépítése:
  - Egy interface-re
  - és ezt az interfészt megvalósító @Service osztályra.

Product.java osztály

```java
public class Product {

    private int id;
    private String name;
    private String price;

    public Product(int id, String name, String price) {
        this.id = id;
        this.name = name;
        this.price = price;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getPrice() {
        return price;
    }

    public void setPrice(String price) {
        this.price = price;
    }
}
```

Készítsünk egy ProductService interface-t!

```java
import java.util.List;

public interface ProductService {

    void createProduct(Product product);
    void deleteProduct(int id);
    void updateProduct(int id, Product newProduct);
    List<Product> listProducts();
}
```

Hozzuk létre a Service-t ami megvalósítja a ProductService interface-t!

```
public class ProductServiceImpl implements ProductService{

   static ArrayList<Product> productRepository = new ArrayList<>();
   static {
       productRepository.add( new Product( productRepository.size(), “Cola",
       "1.0") );
       productRepository.add(
          new Product(productRepository.size(),"Sandwich", "3.0") );
       productRepository.add( new Product(productRepository.size(),"Salad",
       "4.0") );
   }
}
```

Implementáljuk a hiányzó metódusokat!

```java
@Override
public void createProduct(Product product) {
    productRepository.add(product);
}

@Override
public void deleteProduct(int id) {
    productRepository.remove(id);
}

@Override
public void updateProduct(int oldProductId, Product newProduct) {
    newProduct.setId(oldProductId);
    productRepository.add(newProduct);
}

@Override
public List<Product> listProducts() {
    return productRepository;
}
```

Készítsünk egy RestAPI-t osztályt ami felhasználja a @Service-t!

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class RestApi {

    @Autowired
    ProductService productService;
}
```

GET end-point

```
@GetMapping(value = "/products")
List<Product> listAllProducts(){
    return productService.listProducts();
}
```

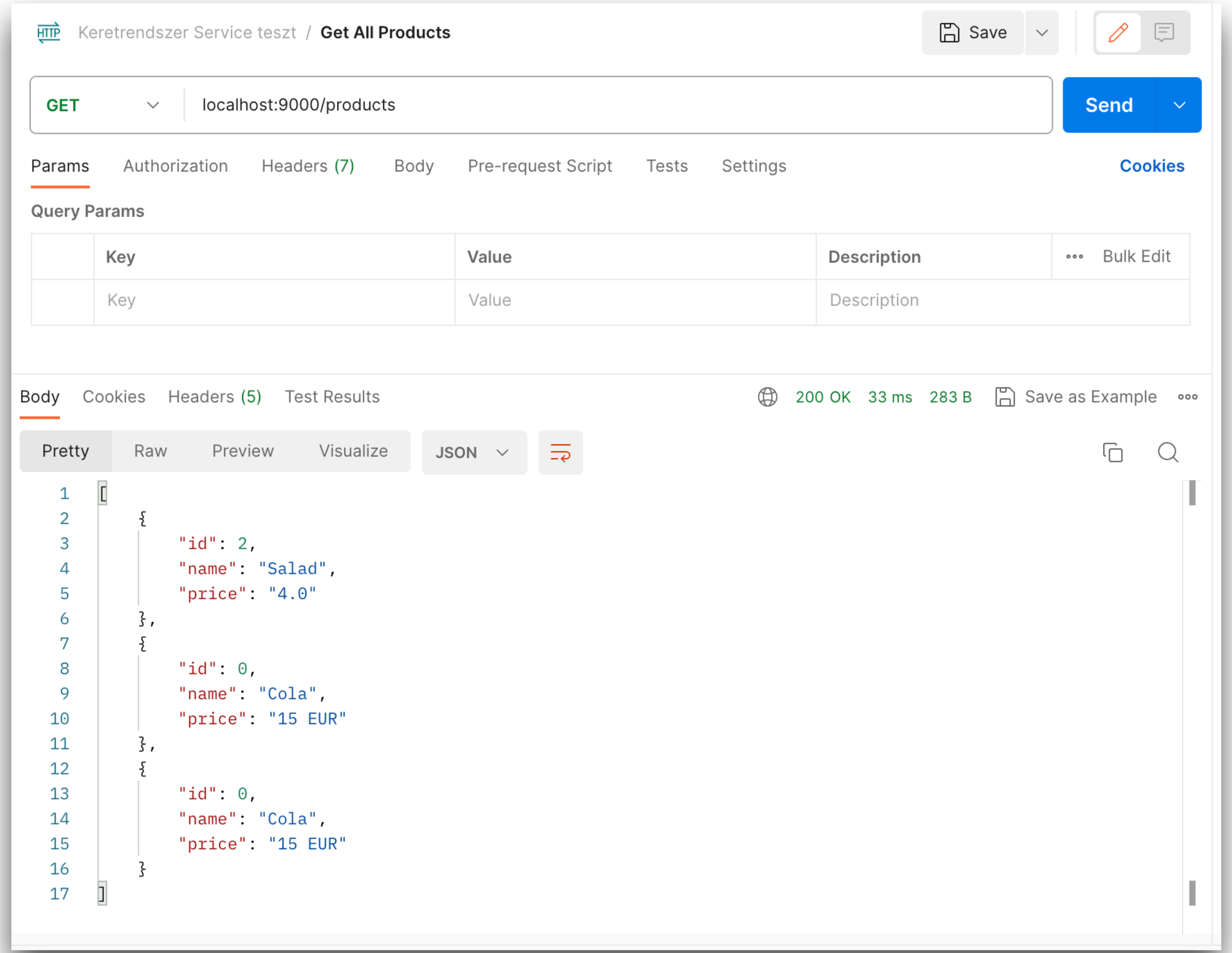
PUT end-point

```java
@RequestMapping(value = "/products/{id}", method = RequestMethod.PUT)
public ResponseEntity<String>
updateProduct(@PathVariable("id") int id, @RequestBody Product newProduct) {
    productService.updateProduct(id, newProduct);
    return new ResponseEntity<>("Product is updated", HttpStatus.OK);
}
```

POST end-point

```
@DeleteMapping(value = "/products/{id}")
public ResponseEntity<String> delete(@PathVariable("id") int id) {
    productService.deleteProduct(id);
    return new ResponseEntity<>("Product is deleted", HttpStatus.OK);
}
```

GET end-point próba Postman segítségével



localhost:9000/products

Query Params

Key	Value	Description
Key	Value	Description

200 OK 33 ms 283 B

```
[
    {
        "id": 2,
        "name": "Salad",
        "price": "4.0"
    },
    {
        "id": 0,
        "name": "Cola",
        "price": "15 EUR"
    },
    {
        "id": 0,
        "name": "Cola",
        "price": "15 EUR"
    }
]
```

Dokumentáljuk az API végpontokat!

OpenAPI specifikáció

- “machine-readable interface definition language”
- Története:
    - 2010-ben kezdte a Swagger fejlesztését Tony Tam.
    - 2015 márciusa: SmartBear megvásárolja a Swaggert.
    - 2015 novembere: SmartBear létrehozza az OpenAPI Initiative-ot. Amihez Google, Microsoft, PayPal stb csatlakoznak.
    - Swagger specifikáció átnevezésre kerül: OpenAPI

https://en.wikipedia.org/wiki/OpenAPI_Specification

Mi az az OpenAPI specifikáció?

The OpenAPI Specification (OAS) defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic.

When properly defined via OpenAPI, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. Similar to what interface descriptions have done for lower-level programming, the OpenAPI Specification removes guesswork in calling a service

https://spec.openapis.org/oas/latest.html

Hogy adható a projekthez?

https://mvnrepository.com/

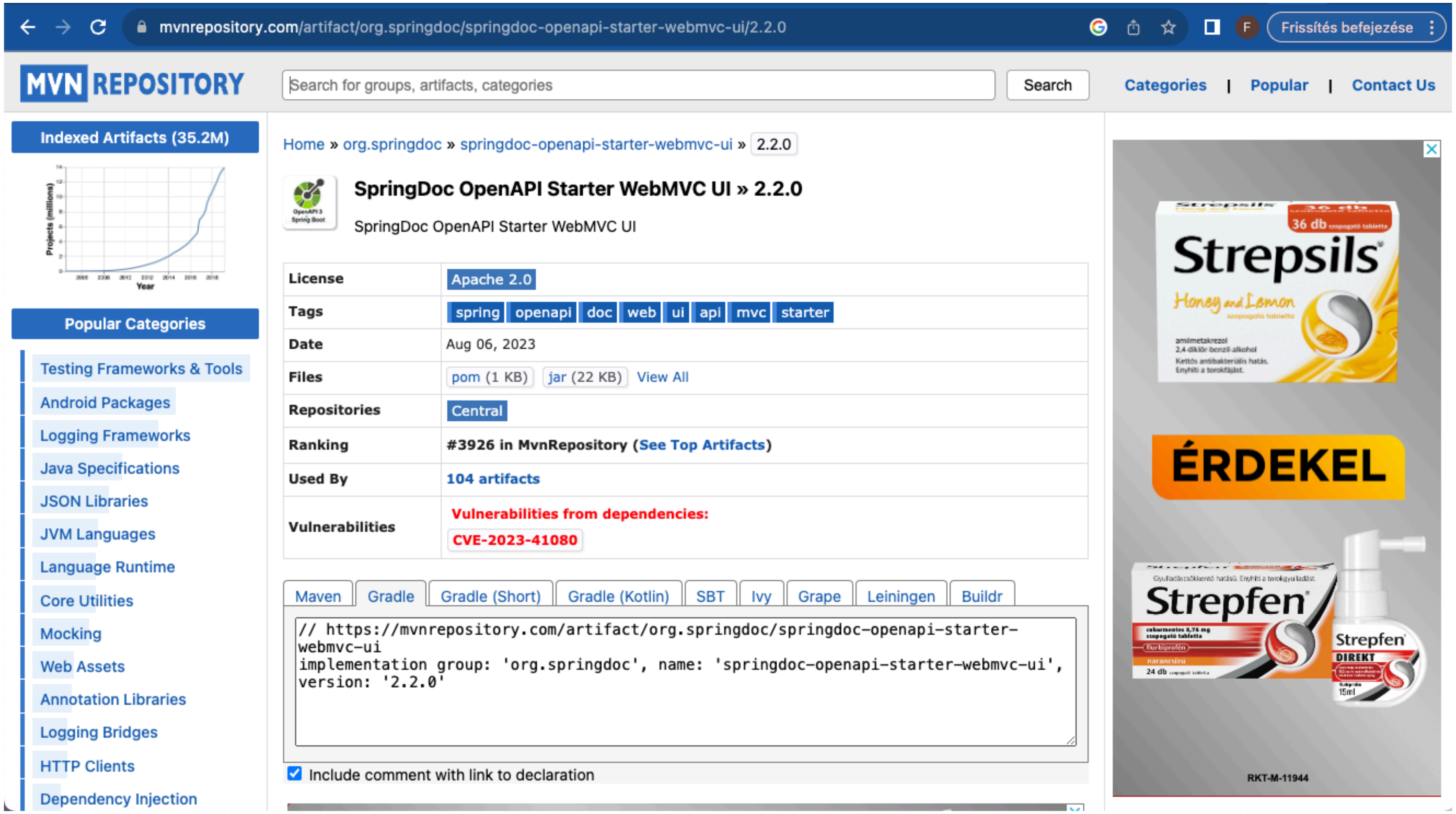
Keressünk arra, hogy: SpringDoc OpenAPI

Erre lesz szükségünk:
SpringDoc OpenAPI Starter WebMVC UI

2.2.0 verziót ha kiválasztjuk, akkor a Gradle dependency kimásolható:

implementation group: 'org.springdoc', name: 'springdoc-openapi-starter-webmvc-ui', version: ‘2.2.0'

Hogy adható a projekthez?


mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui/2.2.0

Frissítés befejezése

MVN REPOSITORY

Search for groups, artifacts, categories | Search | Categories | Popular | Contact Us

Indexed Artifacts (35.2M)

Popular Categories

- Testing Frameworks & Tools
- Android Packages
- Logging Frameworks
- Java Specifications
- JSON Libraries
- JVM Languages
- Language Runtime
- Core Utilities
- Mocking
- Web Assets
- Annotation Libraries
- Logging Bridges
- HTTP Clients
- Dependency Injection

Home » org.springdoc » springdoc-openapi-starter-webmvc-ui » 2.2.0

SpringDoc OpenAPI Starter WebMVC UI » 2.2.0

SpringDoc OpenAPI Starter WebMVC UI

License	Apache 2.0
Tags	spring openapi doc web ui api mvc starter
Date	Aug 06, 2023
Files	pom (1 KB) jar (22 KB) View All
Repositories	Central
Ranking	#3926 in MvnRepository (See Top Artifacts)
Used By	104 artifacts
Vulnerabilities	Vulnerabilities from dependencies: CVE-2023-41080

Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

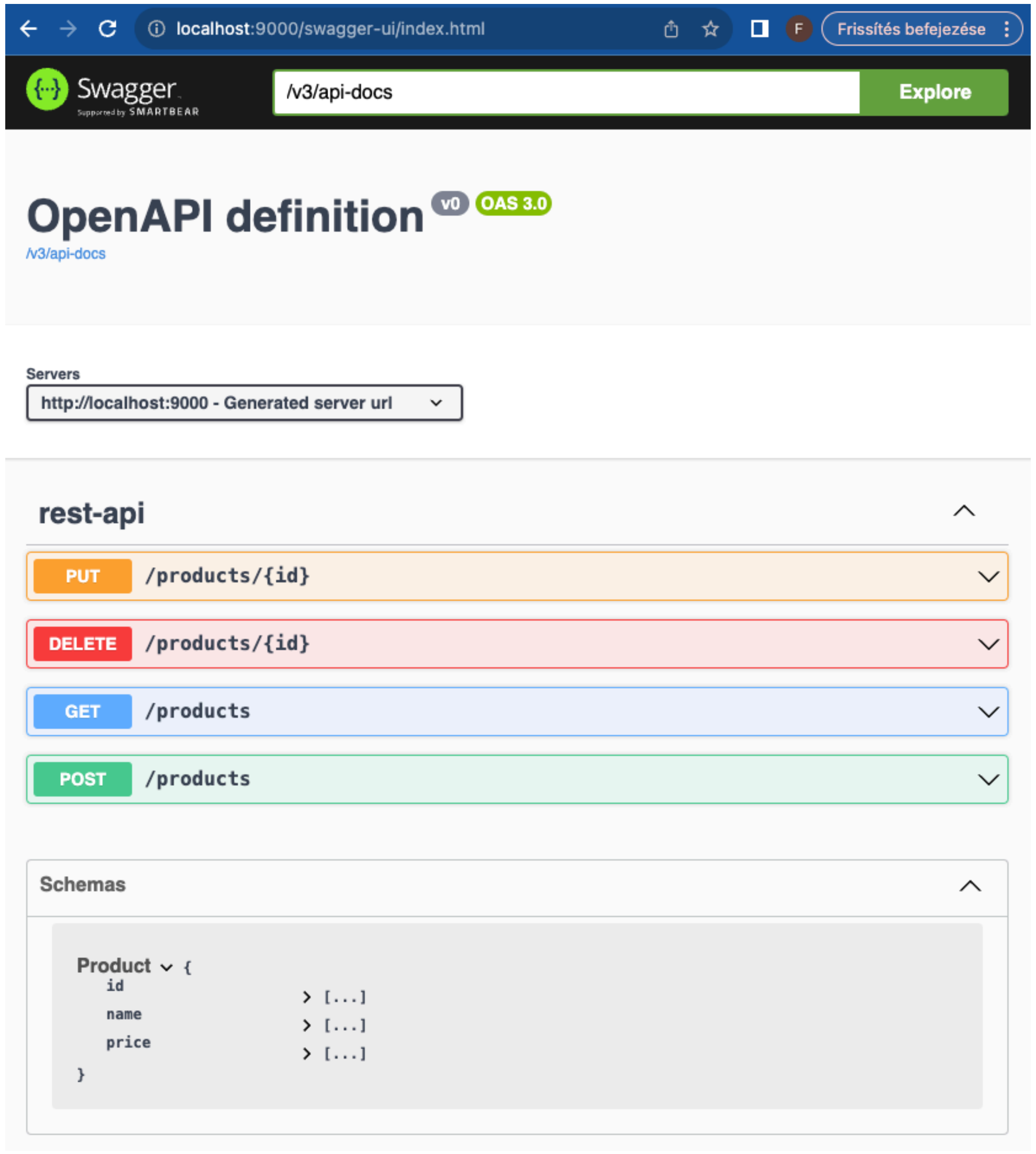
```
// https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-starter-webmvc-ui
implementation group: 'org.springdoc', name: 'springdoc-openapi-starter-webmvc-ui', version: '2.2.0'
```

☑ Include comment with link to declaration

Ne felejtsük el szinkronizálni a Gradle függőségeket!

Hogyan használható?

http://localhost:9000/swagger-ui.html



OpenAPI definition v0 OAS 3.0

/v3/api-docs

Servers

http://localhost:9000 - Generated server url

rest-api

PUT /products/{id}

DELETE /products/{id}

GET /products

POST /products

Schemas

```
Product {
    id          > [...]
    name        > [...]
    price       > [...]
}
```

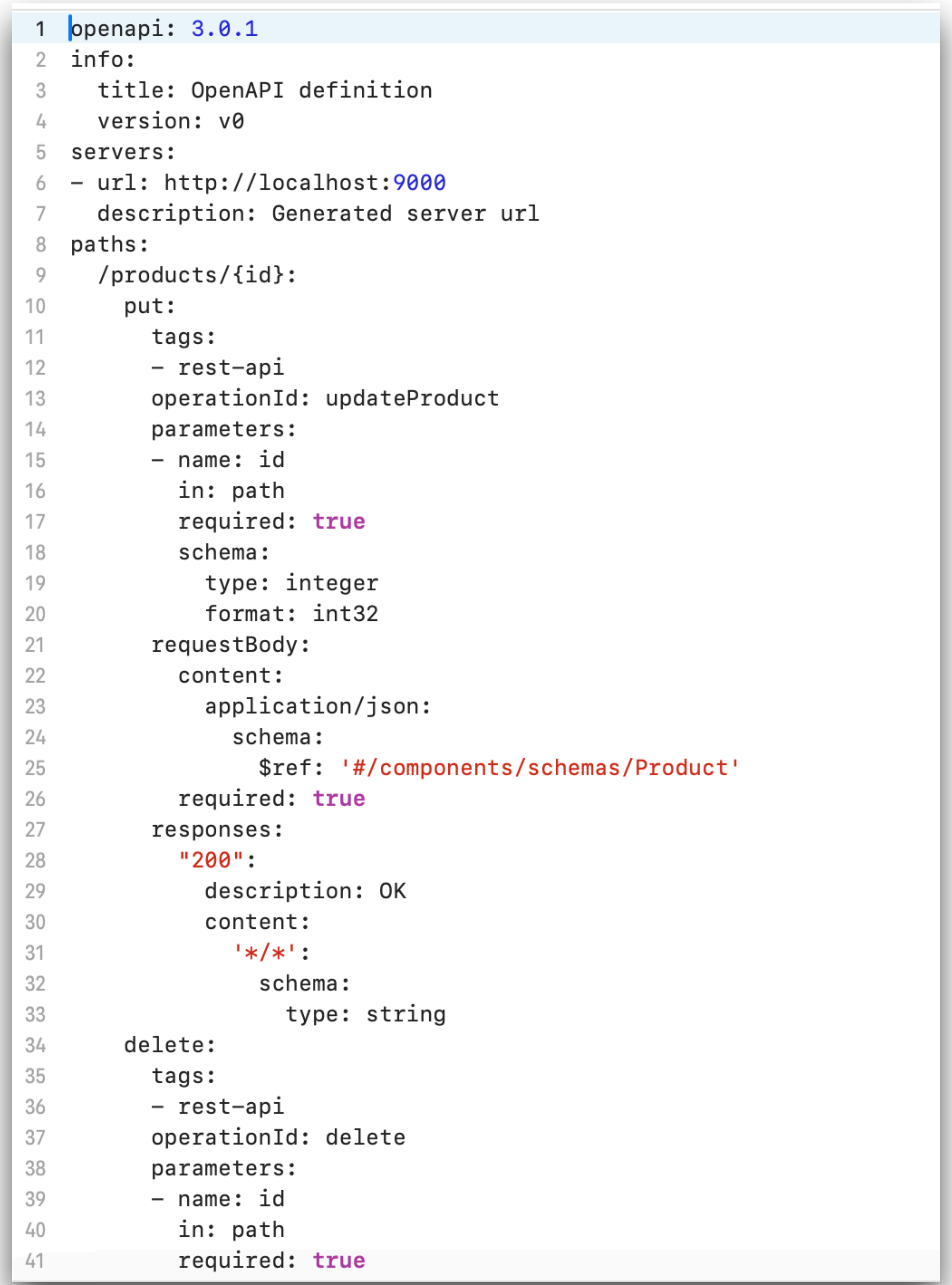
OpenAPI leírás is elkészül

http://localhost:9000/v3/api-docs

```
{"openapi":"3.0.1","info":{"title":"OpenAPI definition","version":"v0"},"servers":
[{"url":"http://localhost:9000","description":"Generated server url"}],"paths":{"/
products/{id}":{"put":{"tags":["rest-api"],"operationId":"updateProduct","parameters":
[{"name":"id","in":"path","required":true,"schema":
{"type":"integer","format":"int32"}}],"requestBody":{"content":{"application/json":
{"schema":{"$ref":"#/components/schemas/Product"}}},"required":true},"responses":{"200":
{"description":"OK","content":{"*/*":{"schema":{"type":"string"}}}}}},"delete":{"tags":
["rest-api"],"operationId":"delete","parameters":
[{"name":"id","in":"path","required":true,"schema":
{"type":"integer","format":"int32"}}],"responses":{"200":{"description":"OK","content":
{"*/*":{"schema":{"type":"string"}}}}}}},"/products":{"get":{"tags":["rest-
api"],"operationId":"listAllProducts","responses":{"200":{"description":"OK","content":
{"*/*":{"schema":{"type":"array","items":{"$ref":"#/components/schemas/
Product"}}}}}}},"post":{"tags":["rest-api"],"operationId":"createProduct","requestBody":
{"content":{"application/json":{"schema":{"$ref":"#/components/schemas/
Product"}}},"required":true},"responses":{"200":{"description":"OK","content":{"*/*":
{"schema":{"type":"string"}}}}}}}},"components":{"schemas":{"Product":
{"type":"object","properties":{"id":{"type":"integer","format":"int32"},"name":
{"type":"string"},"price":{"type":"string"}}}}}}
```

OpenAPI leírás is elkészül

http://localhost:9000/v3/api-docs.yaml



```yaml
openapi: 3.0.1
info:
  title: OpenAPI definition
  version: v0
servers:
- url: http://localhost:9000
  description: Generated server url
paths:
  /products/{id}:
    put:
      tags:
      - rest-api
      operationId: updateProduct
      parameters:
      - name: id
        in: path
        required: true
        schema:
          type: integer
          format: int32
      requestBody:
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Product'
        required: true
      responses:
        "200":
          description: OK
          content:
            '*/*':
              schema:
                type: string
    delete:
      tags:
      - rest-api
      operationId: delete
      parameters:
      - name: id
        in: path
        required: true
```

Egyedi elérési út is megadható

- application.properties-be kell beírni:

```
springdoc.swagger-ui.path=/swagger-ui.html
```

További információ

- https://springdoc.org/#Introduction

Generátorok

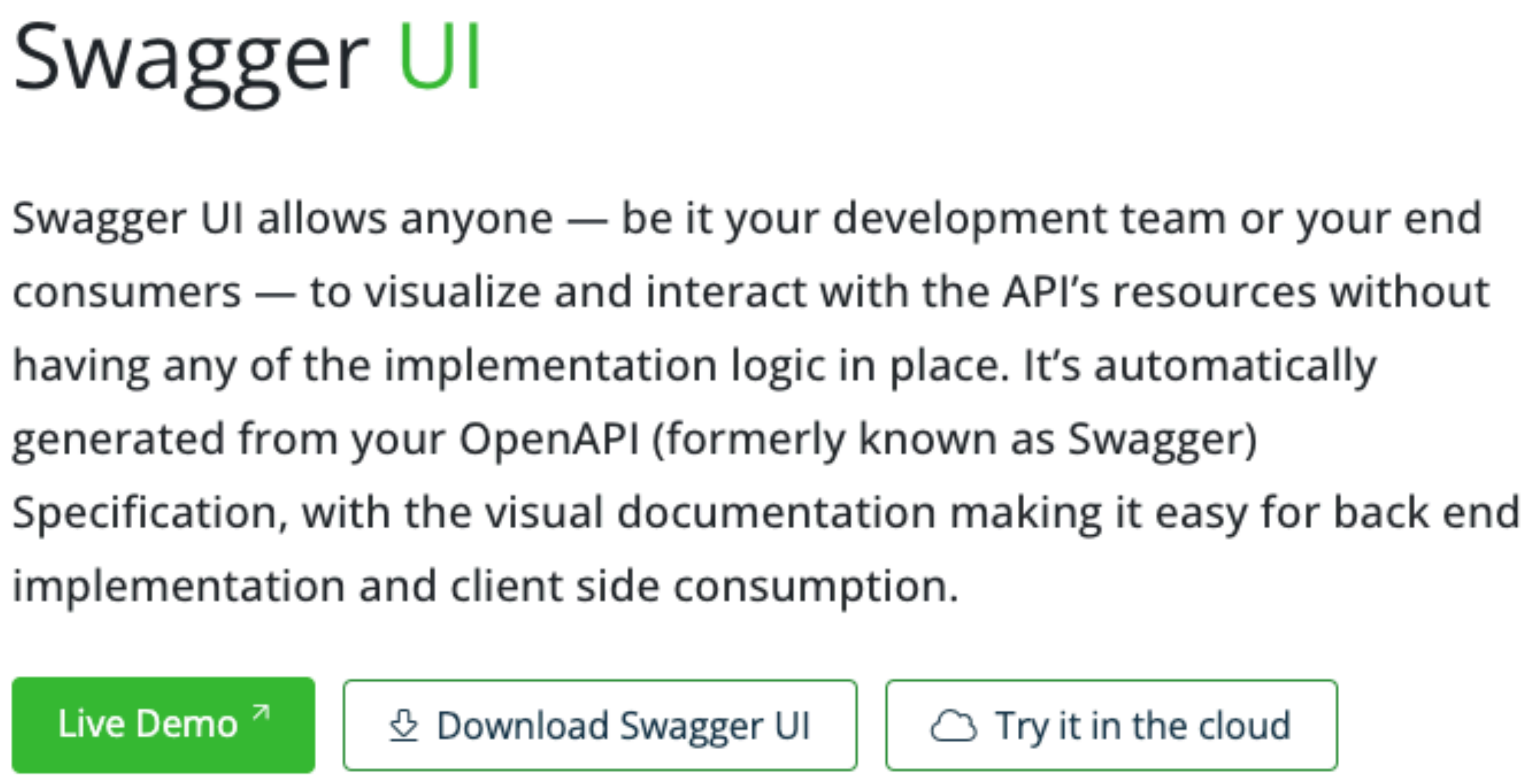
- OpenAPI leírás alapján készít klienst, vagy szervert.
- Különböző nyelvekhez léteznek.
- https://openapi-generator.tech/docs/generators
- Például az elkészült szerverhez a kliens kódokat le tudjuk generáltatni: https://openapi-generator.tech/#try

```
openapi-generator generate -i api-docs.yaml -g android -o /tmp/test/
```

```
[main] INFO  o.o.codegen.DefaultGenerator - Generating with dryRun=false
[main] INFO  o.o.c.ignore.CodegenIgnoreProcessor - No .openapi-generator-ignore file found.
[main] INFO  o.o.codegen.DefaultGenerator - OpenAPI Generator: android (client)
[main] INFO  o.o.codegen.DefaultGenerator - Generator 'android' is considered stable.
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/model/Product.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./docs/Product.md
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/api/RestApiApi.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./docs/RestApiApi.md
[main] INFO  o.o.codegen.TemplateManager - writing file ./README.md
[main] INFO  o.o.codegen.TemplateManager - writing file ./git_push.sh
[main] INFO  o.o.codegen.TemplateManager - writing file ./.gitignore
[main] INFO  o.o.codegen.TemplateManager - writing file ./pom.xml
[main] INFO  o.o.codegen.TemplateManager - writing file ./build.gradle
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/AndroidManifest.xml
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/ApiInvoker.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/JsonUtil.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/ApiException.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/Pair.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/request/
GetRequest.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/request/
PostRequest.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/request/
PutRequest.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/request/
DeleteRequest.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/request/
PatchRequest.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/auth/ApiKeyAuth.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/auth/
HttpBasicAuth.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./src/main/java/org/openapitools/client/auth/
Authentication.java
[main] INFO  o.o.codegen.TemplateManager - writing file ./gradlew
[main] INFO  o.o.codegen.TemplateManager - writing file ./gradlew.bat
[main] INFO  o.o.codegen.TemplateManager - writing file ./gradle/wrapper/gradle-wrapper.properties
[main] INFO  o.o.codegen.TemplateManager - writing file /Users/csabafazekas/Workspaces/workspace_EKE/Swagger/./
gradle/wrapper/gradle-wrapper.jar
[main] INFO  o.o.codegen.TemplateManager - writing file /Users/csabafazekas/Workspaces/workspace_EKE/
Swagger/./.openapi-generator-ignore
[main] INFO  o.o.codegen.TemplateManager - writing file ./.openapi-generator/VERSION
[main] INFO  o.o.codegen.TemplateManager - writing file ./.openapi-generator/FILES
################################################################################
# Thanks for using OpenAPI Generator.                                          #
# Please consider donation to help us maintain this project 🙏                 #
# https://opencollective.com/openapi_generator/donate                          #
################################################################################
csabafazekas@MacBook-Air-3 Swagger %
```

Swagger UI Live Demo

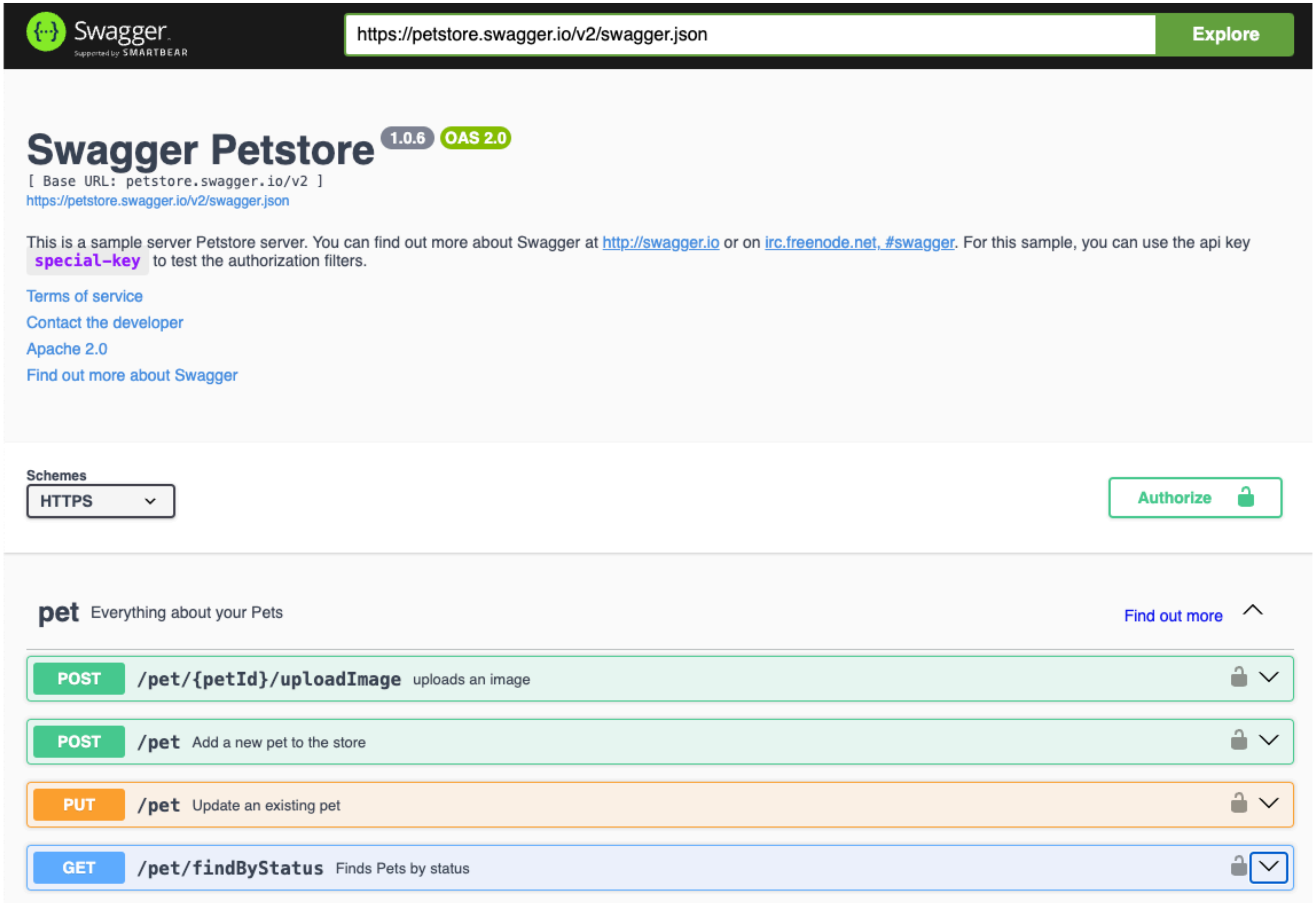
https://swagger.io/tools/swagger-ui/



Swagger UI

Swagger UI allows anyone — be it your development team or your end consumers — to visualize and interact with the API's resources without having any of the implementation logic in place. It's automatically generated from your OpenAPI (formerly known as Swagger) Specification, with the visual documentation making it easy for back end implementation and client side consumption.

Swagger UI Live Demo



Swagger Petstore 1.0.6 OAS 2.0

[Base URL: petstore.swagger.io/v2]
https://petstore.swagger.io/v2/swagger.json

This is a sample server Petstore server. You can find out more about Swagger at http://swagger.io or on irc.freenode.net, #swagger. For this sample, you can use the api key `special-key` to test the authorization filters.

Terms of service
Contact the developer
Apache 2.0
Find out more about Swagger

pet Everything about your Pets — Find out more

POST `/pet/{petId}/uploadImage` uploads an image
POST `/pet` Add a new pet to the store
PUT `/pet` Update an existing pet
GET `/pet/findByStatus` Finds Pets by status