

(<http://baeldung.com>)

Configuración de Swagger 2 con una API REST de Spring

Última modificación: 18 de abril de 2018

por baeldung (<http://www.baeldung.com/author/baeldung/>)
(<http://www.baeldung.com/author/baeldung/>)

DESCANSO (<http://www.baeldung.com/category/rest/>)

Seguridad (<http://www.baeldung.com/category/security-2/>)

Primavera (<http://www.baeldung.com/category/spring/>) +

Otras lecturas:

Generar Spring Boot REST Client con Swagger (<http://www.baeldung.com/rest-spring-boot-rest-client-swagger-codegen>)

Aprenda cómo puede generar un cliente Spring Boot REST usando el generador de código Swagger.

Leer más

(<http://www.baeldung.com/spring-boot-rest-client-swagger-codegen>) →

Introducción a Spring REST Docs

(<http://www.baeldung.com/spring-rest-docs>)

(<http://www.baeldung.com/spring-rest-docs>)

Este artículo presenta Spring REST Docs, un mecanismo basado en pruebas para generar documentación para servicios RESTful que sea precisa y legible.

Leer más

(<http://www.baeldung.com/spring-rest-docs>) →

Introducción a AsciiDoctor en Java

(<http://www.baeldung.com/ascii-doctor>)

Aprenda a generar documentos utilizando AsciiDoctor.

Leer más

(<http://www.baeldung.com/ascii-doctor>)

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

Acabo de anunciar los nuevos módulos de *Spring 5* en REST With Spring:

>> **COMPRUEBA EL CURSO** (</rest-with-spring-course#new-modules>)

1. Información general

Al crear una API REST, la buena documentación es instrumental.

Además, cada cambio en la API debe describirse simultáneamente en la documentación de referencia. Lograrlo manualmente es un ejercicio tedioso, por lo que la automatización del proceso era inevitable.

En este tutorial, veremos **Swagger 2 para un servicio web Spring REST**. Para este artículo, usaremos la implementación de **Springfox** de la especificación Swagger 2.

Si no está familiarizado con Swagger, debe visitar su página web (<http://swagger.io/>) para obtener más información antes de continuar con este artículo.

2. Proyecto objetivo

La creación del servicio REST que utilizaremos en nuestros ejemplos no está dentro del alcance de este artículo. Si ya tiene un proyecto adecuado, úselo. Si no, los siguientes enlaces son un buen lugar para comenzar:

- Cree una API REST con el artículo [Spring 4 y Java Config \(/building-a-restful-web-service-with-spring-and-java-based-configuration\)](/building-a-restful-web-service-with-spring-and-java-based-configuration)
- Construyendo un servicio web RESTful (<https://spring.io/guides/gs/rest-service/>)

3. Agregar la dependencia de Maven

Como se mencionó anteriormente, usaremos la implementación de Springfox de la especificación Swagger.

Para agregarlo a nuestro proyecto Maven, necesitamos una dependencia en el archivo *pom.xml*.

```
1 <dependency>
2   <groupId>io.springfox</groupId>
3   <artifactId>springfox-swagger2</artifactId>
4   <version>2.7.0</version>
5 </dependency>
```

4. Integrando Swagger 2 en el proyecto

4.1. Configuración de Java

La configuración de Swagger se centra principalmente en **Docket** Bean.

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

```

1  @Configuration
2  @EnableSwagger2
3  public class SwaggerConfig {
4      @Bean
5      public Docket api() {
6          return new Docket(DocumentationType.SWAGGER_2)
7              .select()
8              .apis(RequestHandlerSelectors.any())
9              .paths(PathSelectors.any())
10             .build();
11     }
12 }

```

Swagger 2 se habilita a través de la anotación **@EnableSwagger2**.

Después de que se define el *Docket* Bean, su método **select()** devuelve una instancia de **ApiSelectorBuilder**, que proporciona una forma de controlar los puntos finales expuestos por Swagger.

Los predicados para la selección de *RequestHandler* se pueden configurar con la ayuda de **RequestHandlerSelectors** y **PathSelectors**. Usar *any()* para ambos hará que la documentación para toda su API esté disponible a través de Swagger.

Esta configuración es suficiente para integrar Swagger 2 en el proyecto Spring Boot existente. Para otros proyectos de Spring, se requiere un ajuste adicional.

4.2. Configuración sin Spring Boot

Sin Spring Boot, no tiene el lujo de la configuración automática de sus manejadores de recursos. La IU de Swagger agrega un conjunto de recursos que debe configurar como parte de una clase que amplía *WebMvcConfigurerAdapter*, y se anota con **@EnableWebMvc**.

```

1  @Override
2  public void addResourceHandlers(ResourceHandlerRegistry registry) {
3      registry.addResourceHandler("swagger-ui.html")
4          .addResourceLocations("classpath:/META-INF/resources/");
5
6      registry.addResourceHandler("/webjars/**")
7          .addResourceLocations("classpath:/META-INF/resources/webjars/");
8  }

```

4.3. Verificación

To verify that Springfox is working, you can visit the following URL in your browser:

<http://localhost:8080/spring-security-rest/api/v2/api-docs>

The result is a JSON response with a large number of key-value pairs, which is not very readable. Fortunately, Swagger provides **Swagger UI** for this purpose.

5. Swagger UI

Swagger UI is a built-in solution which makes user interaction with the Swagger-generated API documentation much easier.

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

5.1. Enabling Springfox's Swagger UI

To use Swagger UI, one additional Maven dependency is required:

```
1 <dependency>
2   <groupId>io.springfox</groupId>
3   <artifactId>springfox-swagger-ui</artifactId>
4   <version>2.7.0</version>
5 </dependency>
```

Now you can test it in your browser by visiting `http://localhost:8080/your-app-root/swagger-ui.html`

In our case, by the way, the exact URL will be: `http://localhost:8080/spring-security-rest/api/swagger-ui.html`

El resultado debería verse más o menos así:



(/wp-content/uploads/2016/07/Screenshot_11.png)

5.2. Explorando la documentación de Swagger

Dentro de la respuesta de Swagger hay una **lista de todos los controladores** definidos en su aplicación. Al hacer clic en cualquiera de ellos, se enumerarán los métodos HTTP válidos (*BORRAR* , *OBTENER* , *CABEZA* , *OPCIONES* , *PATCH* , *POST* , *PUT*).

La ampliación de cada método proporciona datos útiles adicionales, como el estado de la respuesta, el tipo de contenido y una lista de parámetros. También es posible probar cada método usando la IU.

La habilidad de Swagger para sincronizarse con su código base es crucial. Para demostrar esto, puede agregar un nuevo controlador a su aplicación.

```
1 @RestController
2 public class CustomController {
3
4     @RequestMapping(value = "/custom", method = RequestMethod.POST)
5     public String custom() {
6         return "custom";
7     }
8 }
```

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

Ahora, si actualiza la documentación de Swagger, verá **un controlador personalizado** en la lista de controladores. Como saben, solo hay un método (*POST*) que se muestra en la respuesta de Swagger.

6. Configuración avanzada

El *Docket* Bean de su aplicación se puede configurar para darle más control sobre el proceso de generación de documentación API.

6.1. API de filtrado para la respuesta de Swagger

No siempre es deseable exponer la documentación para toda su API. Puede restringir la respuesta de Swagger pasando parámetros a los métodos ***apis()*** y ***paths()*** de la clase *Docket*.

Como se vio anteriormente, *RequestHandlerSelectors* permite usar los predicados *any* o *none*, pero también se puede usar para filtrar la API de acuerdo con el paquete base, la anotación de clase y las anotaciones de método.

PathSelectors proporciona filtrado adicional con predicados que escanean las rutas de solicitud de su aplicación. Puede usar *any()*, *none()*, *regex()* o *ant()*.

En el siguiente ejemplo, le indicaremos a Swagger que incluya solo controladores de un paquete particular, con rutas específicas, usando el predicado *ant()*.

```
1 @Bean
2 public Docket api() {
3     return new Docket(DocumentationType.SWAGGER_2)
4         .select()
5         .apis(RequestHandlerSelectors.basePackage("org.baeldung.web.controller"))
6         .paths(PathSelectors.ant("/foos/*"))
7         .build();
8 }
```

6.2. Información personalizada

Swagger también proporciona algunos valores predeterminados en su respuesta que puede personalizar, como "Api Documentation", "Created by Contact Email", "Apache 2.0".

Para cambiar estos valores, puede usar el **método *apiInfo()*** (*ApiInfo apiInfo()*). La clase ***ApiInfo*** que contiene información personalizada sobre la API.

```
1 @Bean
2 public Docket api() {
3     return new Docket(DocumentationType.SWAGGER_2)
4         .select()
5         .apis(RequestHandlerSelectors.basePackage("com.example.controller"))
6         .paths(PathSelectors.ant("/foos/*"))
7         .build()
8         .apiInfo(apiInfo());
9 }
10
11 private ApiInfo apiInfo() {
12     return new ApiInfo(
13         "My REST API",
14         "Some custom description of API.",
15         "API TOS",
16         "Terms of service",
17         new Contact("John Doe", "www.example.com", "myeaddress@company.com"),
18         "License of API", "API license URL", Collections.emptyList());
19 }
```

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

6.3. Mensajes de respuesta de métodos personalizados

Swagger permite **globalmente anular mensajes de respuesta de métodos HTTP** a través de *expediente* 's **`globalResponseMessage()`** método. En primer lugar, debe indicar a Swagger que no use mensajes de respuesta predeterminados.

Supongamos que desea anular los mensajes de respuesta **500** y **403** para todos los métodos *GET*. Para lograr esto, un poco de código debe ser añadido a la *Expediente* bloque de inicialización 's (código original está excluido para mayor claridad):

```
1 .useDefaultResponseMessages(false)
2 .globalResponseMessage(RequestMethod.GET,
3     new ArrayList(new ResponseMessageBuilder()
4         .code(500)
5         .message("500 message")
6         .responseModel(new ModelRef("Error"))
7         .build(),
8     new ResponseMessageBuilder()
9         .code(403)
10        .message("Forbidden!")
11        .build()));
```

foo-controller : Foo Controller

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#)

GET /foos/{id} [findById](#)

Response Class (Status 200)

Model | Model Schema

```
{
  "id": 0,
  "name": "string"
}
```

Response Content Type */* ▼

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)	id	path	long

Response Messages

HTTP Status Code	Reason	Response Model	Headers
403	Forbidden!!!!		
500	500 message		

Try it out!

(/wp-content/uploads/2016/07/Screenshot_2.png)

7. Interfaz de usuario de Swagger con una API segura de OAuth

La interfaz de usuario de Swagger proporciona una serie de características muy útiles, que hemos cubierto hasta ahora aquí. Pero realmente no podemos usar la mayoría de estos si nuestra API está protegida y no es accesible.

Veamos cómo podemos permitir que Swagger acceda a una API segura de OAuth, utilizando el tipo de concesión de Código de autorización en este ejemplo.

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

Configuraremos Swagger para acceder a nuestra API segura utilizando el soporte de *SecurityScheme* y *SecurityContext*:

```

1  @Bean
2  public Docket api() {
3      return new Docket(DocumentationType.SWAGGER_2).select()
4          .apis(RequestHandlerSelectors.any())
5          .paths(PathSelectors.any())
6          .build()
7          .securitySchemes(Arrays.asList(securityScheme()))
8          .securityContexts(Arrays.asList(securityContext()));
9  }

```

7.1. Configuración de seguridad

Definiremos un bean *SecurityConfiguration* en nuestra configuración de Swagger y estableceremos algunos valores predeterminados:

```

1  @Bean
2  public SecurityConfiguration security() {
3      return SecurityConfigurationBuilder.builder()
4          .clientId(CLIENT_ID)
5          .clientSecret(CLIENT_SECRET)
6          .scopeSeparator(" ")
7          .useBasicAuthenticationWithAccessCodeGrant(true)
8          .build();
9  }

```

7.2. SecurityScheme

A continuación, definiremos nuestro *SecurityScheme*; esto se usa para describir cómo está asegurada nuestra API (Autenticación básica, OAuth2, ...).

En nuestro caso aquí, definiremos un esquema OAuth utilizado para proteger nuestro servidor de recursos (<http://www.baeldung.com/rest-api-spring-oauth2-angularjs>):

```

1  private SecurityScheme securityScheme() {
2      GrantType grantType = new AuthorizationCodeGrantBuilder()
3          .tokenEndpoint(new TokenEndpoint(AUTH_SERVER + "/token", "oauthtoken"))
4          .tokenRequestEndpoint(
5              new TokenRequestEndpoint(AUTH_SERVER + "/authorize", CLIENT_ID, CLIENT_ID))
6          .build();
7
8      SecurityScheme oauth = new OAuthBuilder().name("spring_oauth")
9          .grantTypes(Arrays.asList(grantType))
10         .scopes(Arrays.asList(scopes()))
11         .build();
12     return oauth;
13 }

```

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente

Tenga en cuenta que usamos el tipo de concesión de Código de autorización, para lo cual debemos proporcionar un punto final de token y la URL de autorización de nuestro Servidor de autorización OAuth2.

Y aquí están los ámbitos que debemos definir:

```

1 | private AuthorizationScope[] scopes() {
2 |     AuthorizationScope[] scopes = {
3 |         new AuthorizationScope("read", "for read operations"),
4 |         new AuthorizationScope("write", "for write operations"),
5 |         new AuthorizationScope("foo", "Access foo API") };
6 |     return scopes;
7 | }

```

Estas se sincronizan con los ámbitos que realmente hemos definido en nuestra aplicación, para la API /foos.

7.3. Contexto de seguridad

Finalmente, necesitamos definir un contexto de seguridad para nuestra API de ejemplo:

```

1 | private SecurityContext securityContext() {
2 |     return SecurityContext.builder()
3 |         .securityReferences(Arrays.asList(new SecurityReference("spring_oauth", scopes())))
4 |         .forPaths(PathSelectors.regex("/foos.*"))
5 |         .build();
6 | }

```

Observe cómo el nombre que usamos aquí, en la referencia - *spring_oauth* - se sincroniza con el nombre que usamos anteriormente, en *SecurityScheme*.

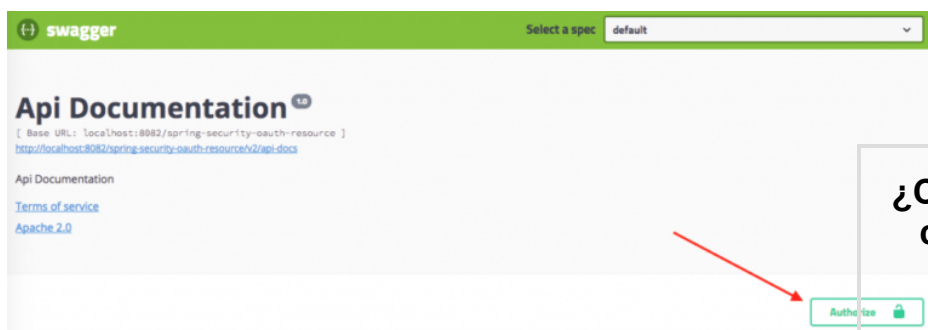
7.4. Prueba

De acuerdo, ahora que tenemos todo listo y listo para comenzar, echemos un vistazo a nuestra interfaz de usuario de Swagger e intentemos acceder a la API de Foo:

Podemos acceder a la UI de Swagger localmente:

```
1 | http://localhost:8082/spring-security-oauth-resource/swagger-ui.html
```

Como podemos ver, ahora existe un nuevo botón "Autorizar" debido a nuestras configuraciones de seguridad:



(http://www.baeldung.com/v/p-
**¿Cuál de estos es el más
 cercano a su trabajo /
 función actual?**

Desarrollador

Desarrollador Senior

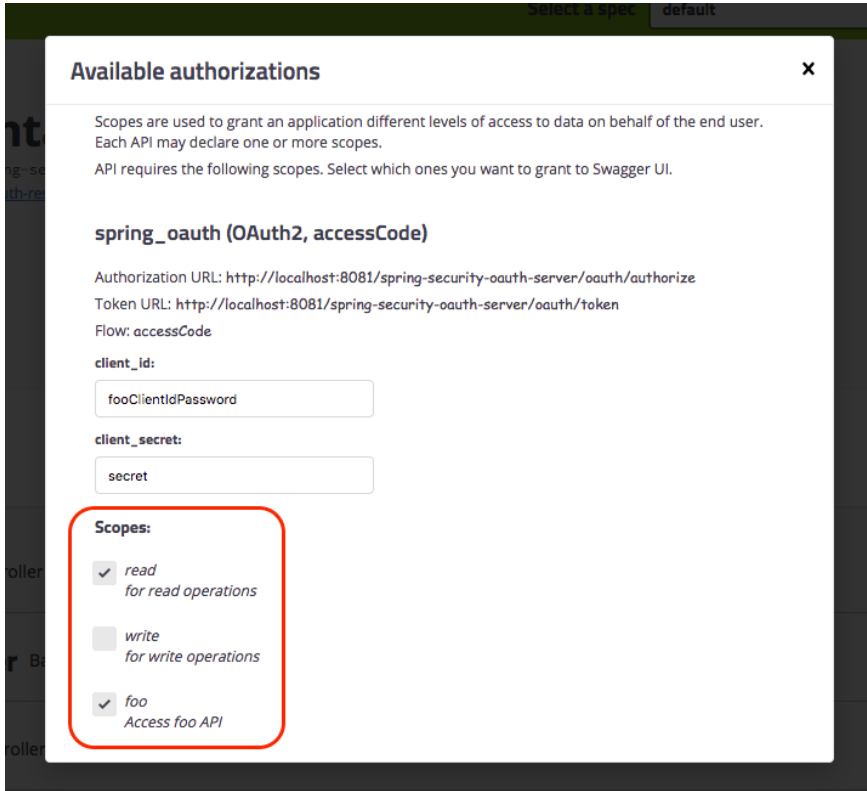
Desarrollador principal

Arquitecto

Gerente

content/uploads/2015/12/swagger_1-1024x362.png)

Cuando hacemos clic en el botón "Autorizar", podemos ver la siguiente ventana emergente para autorizar a nuestra interfaz de usuario de Swagger a acceder a la API de Foo:



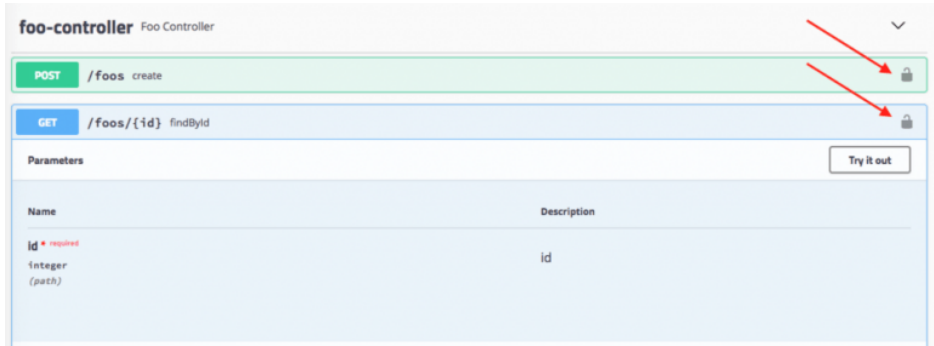
([http://www.baeldung.com/wp-](http://www.baeldung.com/wp-content/uploads/2015/12/swagger_2.png)

content/uploads/2015/12/swagger_2.png)

Tenga en cuenta que:

- Ya podemos ver CLIENT_ID y CLIENT_SECRET, ya que los hemos preconfigurado anteriormente (pero aún podemos cambiarlos)
- Ahora podemos seleccionar los ámbitos que necesitamos

Así es como están marcadas las API seguras:



([http://www.baeldung.com/wp-](http://www.baeldung.com/wp-content/uploads/2015/12/swagger_3-1024x378.png)

content/uploads/2015/12/swagger_3-1024x378.png)

¡Y ahora, finalmente, podemos alcanzar nuestra API!

Por supuesto, casi huelga decir que debemos tener cuidado de cómo exponer externamente la interfaz de usuario de Swagger, ahora que esta configuración de seguridad está activa.

8. Conclusión

En este tutorial, configuramos Swagger 2 para generar documentación de API para un REST API de Spring. También hemos explorado formas de visualizar y personalizar la producción de Swagger. Y finalmente, miramos una configuración simple de OAuth para Swagger.

¿Cuál de estos es el más cercano a su trabajo / función actual?

Desarrollador

Desarrollador Senior

Desarrollador Principal

Arquitecto

Gerente

La **implementación completa** de este tutorial se puede encontrar en el proyecto Github (<https://github.com/eugenp/tutorials/tree/master/spring-security-rest>) : este es un proyecto basado en Eclipse, por lo que debería ser fácil de importar y ejecutar tal cual.

Y, si usted es un estudiante de REST With Spring (<http://www.baeldung.com/rest-with-spring-course#master-class>) , vaya a la Lección 1 del Módulo 7 para profundizar en la configuración de Swagger con Spring y Spring Boot.

Acabo de anunciar los nuevos módulos de Spring 5 en REST With Spring:

>> VERIFIQUE LAS LECCIONES (/rest-with-spring-course#new-modules)

✉ Suscribir ▼

▲ el más nuevo ▲ más antiguo ▲ el más votado



Huésped

Muhammad Yousaf Sajjad



¿Puedes compartir el código en github?

+ 0 -

🕒 Hace 2 años ^



Huésped

Eugen Paraschiv (<http://www.baeldung.com/>)



Acabo de agregar el enlace al final del artículo. Saludos,
Eugen.

+ 3 -

🕒 Hace 2 años ^



Huésped

Daniel Twum



El tutorial también funcionó para mí.

Gracias por un buen comentario.

+ 0 -

🕒 Hace 2 años




Huésped

nelaturuk



Hola, no estoy usando el arranque de primavera. No puedo agregar el método `addResourceHandler`. Estoy usando Spring 4.0 y en el `websecurityadapter` he agregado `EnableWebSecurity`. Incluso si lo hago `EnableWebMvc` no parece funcionar para anular el método. Alguna sugerencia para esto?

+ 0 -

🕒 Hace 2 años 



Huésped

Eugen Paraschiv (<http://www.baeldung.com/>)

El método `addResourceHandler` no es Boot específico. En realidad, la sección 4.2 cubre la configuración adicional que necesitarás hacer cuando no uses Boot.

Y, en términos generales, no hay ninguna razón por la que necesite Boot para configurar Swagger. Espero que ayude. Saludos,
Eugen.

+ 0 -

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente



nelaturuk



Huésped

Gracias Eugen. ¡Agregué la configuración de Swagger en WebMVCSecurityAdapater como mencionó usted y funcionó!

+ 1 -

🕒 Hace 2 años ^

Eugen Paraschiv (<http://www.baeldung.com/>)

Huésped

Suena bien, me alegra que todo salga bien. Saludos, Eugen.

+ 1 -

🕒 Hace 2 años



Huésped

J Paredes



Empecé a trabajar en la integración de Swagger en mi proyecto actual. Este tutorial fue muy útil y bien organizado, pude trabajar fácilmente a través de la configuración inicial y luego ajustes más refinados usando esta guía. Bravo.

+ 1 -

🕒 Hace 2 años ^

Eugen Paraschiv (<http://www.baeldung.com/>)

Huésped

Me alegra que lo haya encontrado útil J. Cheers, Eugen.

+ 3 -

🕒 Hace 2 años



Huésped

ladyNi



Hola Eugen,

¿Puede explicar cómo agregar un parámetro de encabezado para pasar con cada solicitud realizada? De hecho, tengo que enviar un token CSRF para cada solicitud que no sea GET. Intenté usar su APIKey, pero no pude hacerlo funcionar.

+ 1 -

🕒 Hace 2 años ^

Eugen Paraschiv (<http://www.baeldung.com/>)

Huésped

El token CSRF hace que algunos otros aspectos sean más difíciles de trabajar, y Swagger puede ser uno de esos aspectos. No he considerado la opción de integrar Swagger con Spring Security para que pueda tener el token CSRF disponible en Swagger. Es posible, simplemente no lo he investigado. Saludos, Eugen.

+ 1 -

🕒 Hace 2 años

¿Cuál de estos es el más cercano a su trabajo / función actual?



Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente



Huésped

jothi manickam

Hola, recibí el mensaje de respuesta para la solicitud de entrada del usuario pero no funcionó para el parámetro. Por favor, guíame para lograr esto ... necesito establecer cualquier anotación para el parámetro.

+ 1 -

🕒 Hace 2 años ^

Eugen Paraschiv (<http://www.baeldung.com/>)

Huésped

Bueno, Jothy - Tendría que mirar el código para saber qué está pasando allí. Adelante, abre un problema en Github con los detalles y estaré encantado de echarle un vistazo. Saludos, Eugen.

+ 0 -

Hace 2 años

Cargar más comentarios

CATEGORÍAS

- PRIMAVERA (HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
- DESCANSO (HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/)
- JAVA (HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
- SEGURIDAD (HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
- PERSISTENCIA (HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
- JACKSON (HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/)
- HTTPCLIENT (HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/)
- KOTLIN (HTTP://WWW.BAELDUNG.COM/CATEGORY/KOTLIN/)

SERIE

- TUTORIAL "VOLVER A LO BÁSICO" DE JAVA (HTTP://WWW.BAELDUNG.COM/JAVA-TUTORIAL)
- JACKSON JSON TUTORIAL (HTTP://WWW.BAELDUNG.COM/JACKSON)
- TUTORIAL DE HTTPCLIENT 4 (HTTP://WWW.BAELDUNG.COM/HTTPCLIENT-GUIDE)
- REST CON SPRING TUTORIAL (HTTP://WWW.BAELDUNG.COM/REST-WITH-SPRING-SERIES/)
- TUTORIAL DE SPRING PERSISTENCE (HTTP://WWW.BAELDUNG.COM/PERSISTENCE-WITH-SPRING-SERIES/)
- SEGURIDAD CON SPRING (HTTP://WWW.BAELDUNG.COM/SECURITY-SPRING)

ACERCA DE

- ACERCA DE BAELDUNG (HTTP://WWW.BAELDUNG.COM/ABOUT/)
- LOS CURSOS (HTTP://COURSES.BAELDUNG.COM)
- TRABAJO DE CONSULTORÍA (HTTP://WWW.BAELDUNG.COM/CONSULTING)
- META BAELDUNG (HTTP://META.BAELDUNG.COM/)
- EL ARCHIVO COMPLETO (HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE)
- ESCRIBIR PARA BAELDUNG (HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES)
- CONTACTO (HTTP://WWW.BAELDUNG.COM/CONTACT)
- INFORMACIÓN DE LA COMPAÑÍA (HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY/DESARROLLADOR)
- TÉRMINOS DE SERVICIO (HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
- POLÍTICA DE PRIVACIDAD (HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY)
- EDITORES (HTTP://WWW.BAELDUNG.COM/EDITORS)
- KIT DE MEDIOS (PDF) (HTTPS://S3.AMAZONAWS.COM/BAELDUNG.COM/BAELDUNG+-+MEDIA+KIT.PDF)

¿Cuál de estos es el más cercano a su trabajo / función actual?

- Desarrollador
- Desarrollador Senior
- Desarrollador principal
- Arquitecto
- Gerente

**¿Cuál de estos es el más
cercano a su trabajo /
función actual?**

Desarrollador

Desarrollador Senior

Desarrollador principal

Arquitecto

Gerente