SWAGGER

Le code pour documentation

REST

REpresentational State Transfer

REST

REpresentational State Transfer

- Principe d'architecture
- Sans état
- Avantages premier
  - Respect du standard HTTP (Verbes GET, POST)
  - Plus simple & moins verbeux que SOAP

Documenter une API

Non standard

Non VERSIONNABLE

Non UTILISABLE

SWAGGER

C'est quoi ?

- Description d'une API REST
- Grammaire JSON / YAML
- Équivalent du WSDL du SOAP
- Standard de fait

SWAGGER

Alternatives ?

- Aucun standard ne s'impose
- WADL (Web Application Description Language)
  - Soumis au W3C par sun en 2009 mais non standardisé

SWAGGER

Version 2.0

- Initié en 2014
- Un seul fichier JSON
- Support du markdown

SWAGGER

écosystème

Outillage

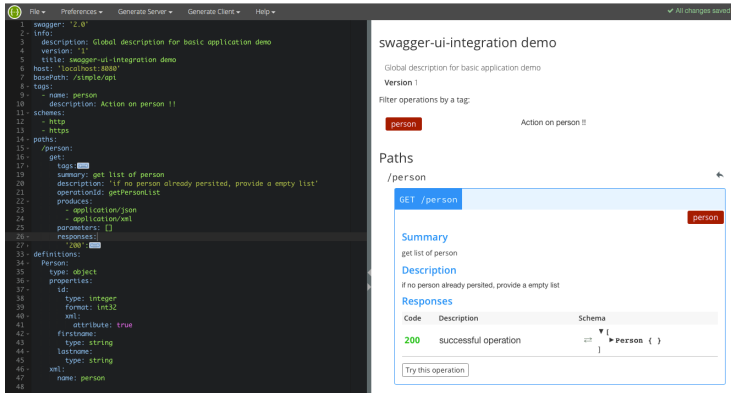
- Swagger codegen
- swagger editor
- Swagger UI
- Génération documents
- plugin(**s**) intelliJ
- container UI & editor

Intégration

- jax-rs
- Spring
- nodeJS
- go(lang)
- Python
- etc…

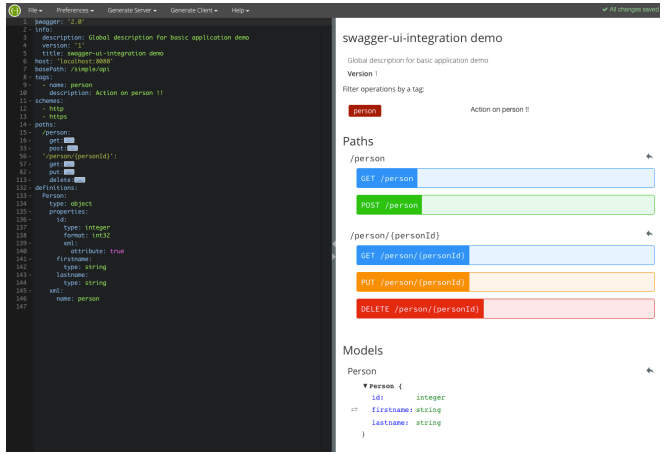
SWAGGER

swagger editor



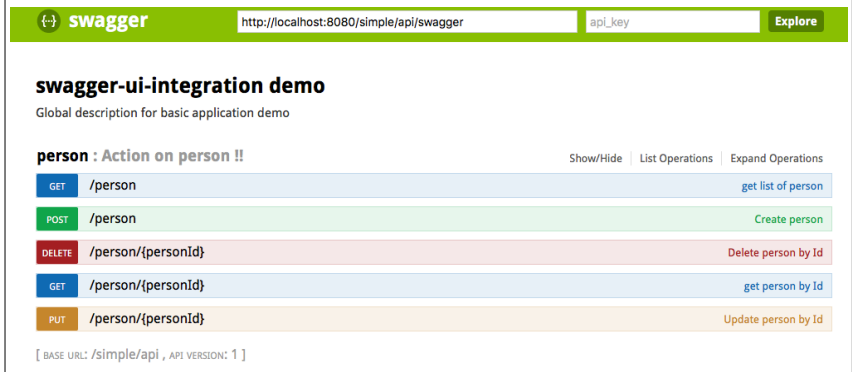
SWAGGER

swagger editor



SWAGGER

UI



SWAGGER

UI

2 types d'intégration

Serveur distinct

Pour
- Centralisation
- Annuaire

Contre
- Gestion CORS

Intégration à l'API

Pour
- Localisation
- Facilité

Contre
- Gestion sécurité

SWAGGER

intégration jax-rs et swagger-core

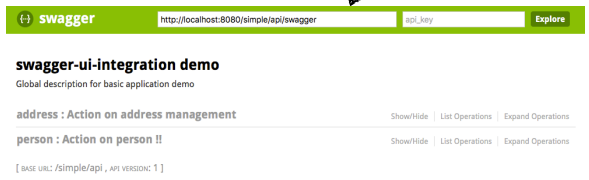
S'appui sur les annotations jax-rs (jsr 311)

Fourni son propre jeu d'annotation pour enrichir la description de l'API

SWAGGER

Description initiale de l'API

```
@SwaggerUIConfiguration
@ApplicationPath("api")
@SwaggerDefinition(
    info = @Info(title = "swagger-ui-integration demo", version = "1"
        , description = "Global description for basic application demo")
    , tags = {
    @Tag(name = "person", description = "Action on person !!")
    , @Tag(name = "address", description = "Action on address management")
  }
)
public class SimpleApplicationConfiguration extends Application {
}
```



SWAGGER

Description initiale de l'API

```
@SwaggerUIConfiguration
@ApplicationPath("api")
@SwaggerDefinition(
    info = @Info(title = "swagger-ui-integration demo", version = "1"
        , description = "Global description for basic application demo")
    , tags = {
    @Tag(name = "person", description = "Action on person !!")
    , @Tag(name = "address", description = "Action on address management")
  }
)
public class SimpleApplicationConfiguration extends Application {
}
```

`@SwaggerDefinition()` → Définition générale de l'API

`@Info()` → Information de l'API

`@Tag()` → Chapitre fonctionnel de l'API

SWAGGER

Description d'un endpoint global

```
@Path("person")
@Produces({
    "application/json", "application/xml"
})
@Api(tags = "person")
public class PersonEndpoint {
  …
}
```

`@Api()` → Description du endpoint

`@Tag()` → Rattachement au chapitre

SWAGGER

Description d'un endpoint GET liste

```
@GET
@Produces({
    "application/json", "application/xml"
})
@ApiOperation(
    value = "get list of person"
    , notes = "if no person already persited, provide a empty list"
    , response = Person.class
    , responseContainer = "List"
)
public List<Person> getPersonList() {
  return personService.getList(null);
}
```

`@ApiOpération()` → Opération lié au endpoint

SWAGGER

Description d'un endpoint GET unitaire

```
@GET
@Path("{personId:[0-9]*}")
@Produces({
    "application/json", "application/xml"
})
@ApiOperation(
    value = "get person by Id"
)
@ApiResponses({
    @ApiResponse(code = 200, message = "success", response = Person.class)
    , @ApiResponse(code = 404, message = "Person not found", response = String.class)
})
public Person getPerson(@PathParam("personId") Integer personId) throws PersonNotExistException {
  return personService.get(personId);
}
```

`@ApiOpération()` → Opération lié au endpoint

`@ApiResponses()` → Liste des réponses possible

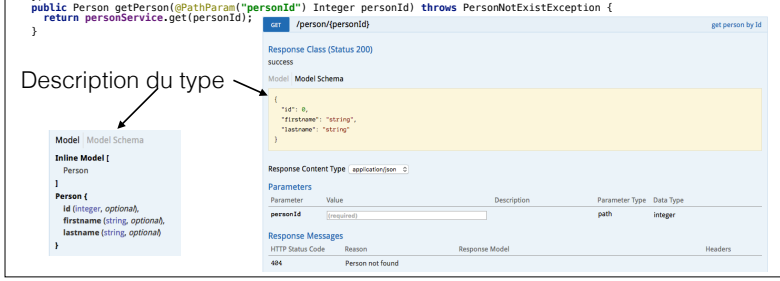
`@ApiResponse()` → Description d'une réponse

SWAGGER

Description d'un endpoint GET unitaire

```
@GET
@Path("{personId:[0-9]*}")
@Produces({
    "application/json", "application/xml"
})
@ApiOperation(
    value = "get person by Id"
)
@ApiResponses({
    @ApiResponse(code = 200, message = "success", response = Person.class)
    , @ApiResponse(code = 404, message = "Person not found", response = String.class)
})
public Person getPerson(@PathParam("personId") Integer personId) throws PersonNotExistException {
  return personService.get(personId);
}
```

Description du type



SWAGGER

Description d'un endpoint POST

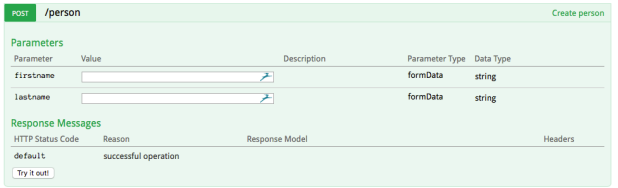
```
@POST
@Consumes("application/x-www-form-urlencoded")
@ApiOperation("Create person")
public Response createPerson(
    @FormParam("firstname") String firstName
    , @FormParam("lastname") String lastName
) throws PersonAlreadyPersisted {
  Person person = personService.persist(new Person(firstName, lastName));
  URI personUri = UriBuilder
      .fromResource(PersonEndpoint.class)
      .path(String.valueOf(person.getId()))
      .build();
  return Response.created(personUri).build();
}
```

`@ApiOpération()` → Opération lié au endpoint

SWAGGER

Description d'un endpoint POST

```
@POST
@Consumes("application/x-www-form-urlencoded")
@ApiOperation("Create person")
public Response createPerson(
    @FormParam("firstname") String firstName
    , @FormParam("lastname") String lastName
) throws PersonAlreadyPersisted {
  Person person = personService.persist(new Person(firstName, lastName));
  URI personUri = UriBuilder
      .fromResource(PersonEndpoint.class)
      .path(String.valueOf(person.getId()))
      .build();
  return Response.created(personUri).build();
}
```



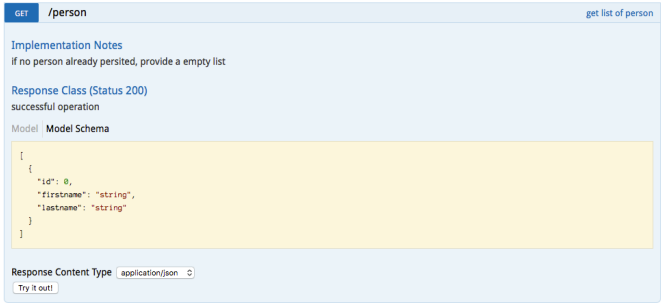
Utilisation des annotation jax-rs pour la documentation des paramètres

SWAGGER

Du code !!!!

SWAGGER

Les avantages de Swagger UI



Essayer l'API !!

SWAGGER

Les avantages de Swagger UI

Try it out! Hide Response

Curl ← Parce que la console, y'a que ça de vrai

```
curl -X GET --header 'Accept: application/json' 'http://localhost:8080/simple/api/person'
```

Request URL ← Pour essayer dans un navigateur ou REST tools

```
http://localhost:8080/simple/api/person
```

Response Body ← Résultat de la requête

```
[]
```

Response Code ← Code HTTP Résultat de la requête

```
200
```

Response Headers ← Et les headers !!!!!

```
{
  "server": "WildFly/10",
  "content-type": "application/json",
  "date": "Sun, 02 Oct 2016 22:23:07 GMT",
  "connection": "keep-alive",
  "x-powered-by": "Undertow/1",
  "content-length": "2"
}
```

PUB

SWAGGER

swagger-ui-integration

Intégration sans effort de swagger et swagger-ui

Un import maven et une annotation

3 niveaux de configuration

- Au niveau projet (annotation)
- Au niveau livrable (fichier de configuration ressources)
- Au niveau environnement (fichier de configuration local)

```
@SwaggerUIConfiguration()
```

PUB

SWAGGER

swagger-ui-integration

Intégration sans effort de swagger et swagger-ui

Un import maven et une annotation

3 niveaux de configuration

- Au niveau projet (annotation)
- Au niveau livrable (fichier de configuration ressources)
- Au niveau environnement (fichier de configuration local)

JAVA EE 6

LGPL

version 1.0

```
@SwaggerUIConfiguration(
    apiDocPath = "documentation"
    , apiDocIndex = "rest-index/index.html"
    , restApplicationPackageAsRoot = false
    , restApplicationPackage = "org.shipstone.swagger.demo.ws.rs"
)
```