Documenting the REST Service

Antonio Goncalves
JAVA CHAMPION

@agoncal www.antoniogoncalves.org

Previous Module

- JAX-RS
- JSon representation
- REST APIs
- External components can interact
- HTTP
- Angular

Overview

REST APIs documentation

External systems can consume these APIs

Contract

Open API

Standard

Describe REST services

What Is API Documentation?

Share data through a REST API

Valuable only if it's accessible

Needs clear documentation

Human readable

Machine readable

What Is Open API Initiative?

Standard vocabulary

Defining and documenting an API

OpenAPI Specification (OAS)

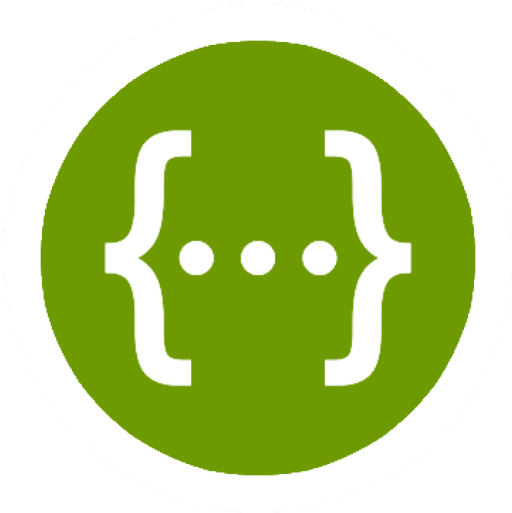
JSon or Yaml document

- Available operations
- Parameters
- Response returned
- Error response code

Open API Contract in JSon

```
{
  "info" : { "description" : "BookStore" },
  "host" : "localhost:8080",
  "basePath" : "/bookstore-back/api", "schemes" : [ "http", "https" ],
  "paths" : {
    "/books" : {
      "get" : {
        "summary" : "Returns all the books",
        "produces" : [ "application/json" ],
        "responses" : {
          "200" : {
            "description" : "Books found",
            "schema" : {
              "type" : "array",
              "items" : {
                "$ref" : "#/definitions/Book"
```

Swagger



Open source tools

Design, build, document, and consume

Open API Specification

Produce a JSon contract for Java EE

Generate TypeScript for Angular front-end

Documenting a REST Service

Top-down

Design our API
before writing code

Bottom-up

Write code that generates
documentation

```java
@SwaggerDefinition(
  info = @Info(description = "BookStore"),
  host = "localhost:8080",
  basePath = "/bookstore-back/api",
  schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

  @GET @Path("/{id : \\d+}")
  @Produces(APPLICATION_JSON)

  public Response getBook(@PathParam("id")
                          Long id) {
  }
```

```json
{
 "swagger": "2.0",
 "info": {"description":"BookStore"},
 "host": "localhost:8080",
 "basePath": "/bookstore-back/api",
 "schemes": [ "http", "https"],
```

```java
@SwaggerDefinition(
  info = @Info(description = "BookStore"),
  host = "localhost:8080",
  basePath = "/bookstore-back/api",
  schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

  @GET @Path("/{id : \\d+}")
  @Produces(APPLICATION_JSON)

  public Response getBook(@PathParam("id")
                          Long id) {
  }
```

```json
{
 "swagger": "2.0",
 "info": {"description":"BookStore"},
 "host": "localhost:8080",
 "basePath": "/bookstore-back/api",
 "schemes": [ "http", "https"],
 "paths": {
  "/books": {
```

```java
@SwaggerDefinition(
  info = @Info(description = "BookStore"),
  host = "localhost:8080",
  basePath = "/bookstore-back/api",
  schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

  @GET @Path("/{id : \\d+}")
  @Produces(APPLICATION_JSON)

  public Response getBook(@PathParam("id")
                          Long id) {
  }
```

```json
{
 "swagger": "2.0",
 "info": {"description":"BookStore"},
 "host": "localhost:8080",
 "basePath": "/bookstore-back/api",
 "schemes": [ "http", "https"],
 "paths": {
  "/books/{id}": {
   "get": {
```

```java
@SwaggerDefinition(
  info = @Info(description = "BookStore"),
  host = "localhost:8080",
  basePath = "/bookstore-back/api",
  schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

  @GET @Path("/{id : \\d+}")
  @Produces(APPLICATION_JSON)

  public Response getBook(@PathParam("id")
                          Long id) {
  }
```

```json
{
 "swagger": "2.0",
 "info": {"description":"BookStore"},
 "host": "localhost:8080",
 "basePath": "/bookstore-back/api",
 "schemes": [ "http", "https"],
 "paths": {
  "/books/{id}": {
   "get": {
    "produces": ["application/json"],
```

```java
@SwaggerDefinition(
  info = @Info(description = "BookStore"),
  host = "localhost:8080",
  basePath = "/bookstore-back/api",
  schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

  @GET @Path("/{id : \\d+}")
  @Produces(APPLICATION_JSON)

  public Response getBook(@PathParam("id")
                          Long id) {
  }
```

```json
{
 "swagger": "2.0",
 "info": {"description":"BookStore"},
 "host": "localhost:8080",
 "basePath": "/bookstore-back/api",
 "schemes": [ "http", "https"],
 "paths": {
  "/books/{id}": {
   "get": {
    "produces": ["application/json"],
    "operationId": "getBook",
```

```java
@SwaggerDefinition(
  info = @Info(description = "BookStore"),
  host = "localhost:8080",
  basePath = "/bookstore-back/api",
  schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

  @GET @Path("/{id : \\d+}")
  @Produces(APPLICATION_JSON)

  public Response getBook(@PathParam("id")
                          Long id) {
  }
```

```json
{
 "swagger": "2.0",
 "info": {"description":"BookStore"},
 "host": "localhost:8080",
 "basePath": "/bookstore-back/api",
 "schemes": [ "http", "https"],
 "paths": {
  "/books/{id}": {
   "get": {
    "produces": ["application/json"],
    "operationId": "getBook",
    "parameters": [{"name": "id",
                    "type": "integer",
                    "pattern": "\\d+"}],
```

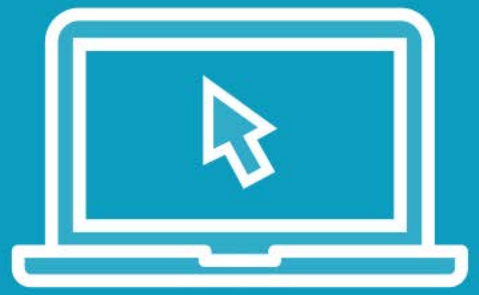
```java
@SwaggerDefinition(
  info = @Info(description = "BookStore"),
  host = "localhost:8080",
  basePath = "/bookstore-back/api",
  schemes = {HTTP, HTTPS}
)
@Path("/books")
public class BookEndpoint {

  @GET @Path("/{id : \\d+}")
  @Produces(APPLICATION_JSON)
  @ApiOperation("Returns a book",
                response = Book.class)
  @ApiResponses({
    @ApiResponse(code = 404,
                 message = "Not found"),
    @ApiResponse(code = 200,
                 message = "Book found"),
  })
  public Response getBook(@PathParam("id")
                          Long id) {
  }
```

```json
{
 "swagger": "2.0",
 "info": {"description":"BookStore"},
 "host": "localhost:8080",
 "basePath": "/bookstore-back/api",
 "schemes": [ "http", "https"],
 "paths": {
  "/books/{id}": {
   "get": {
    "produces": ["application/json"],
    "operationId": "getBook",
    "parameters": [{"name": "id",
                    "type": "integer",
                    "pattern": "\\d+"}],
    "summary": "Returns a book",
    "responses" : {
      "404" : { "description" : "Not found"}
      "200" : {
        "description" : "Book found",
        "schema":{
          "$ref" : "#/definitions/Book"}},
```

```java
@ApiModel("Book representation" )
@Entity
public class Book {

  @Id
  @ApiModelProperty("Identifier")
  private Long id;

  @ApiModelProperty("Title of the book")
  private String title;

  @ApiModelProperty("Describes the book")
  private String description;

  @ApiModelProperty("Unit cost")
  private Float unitCost;
```

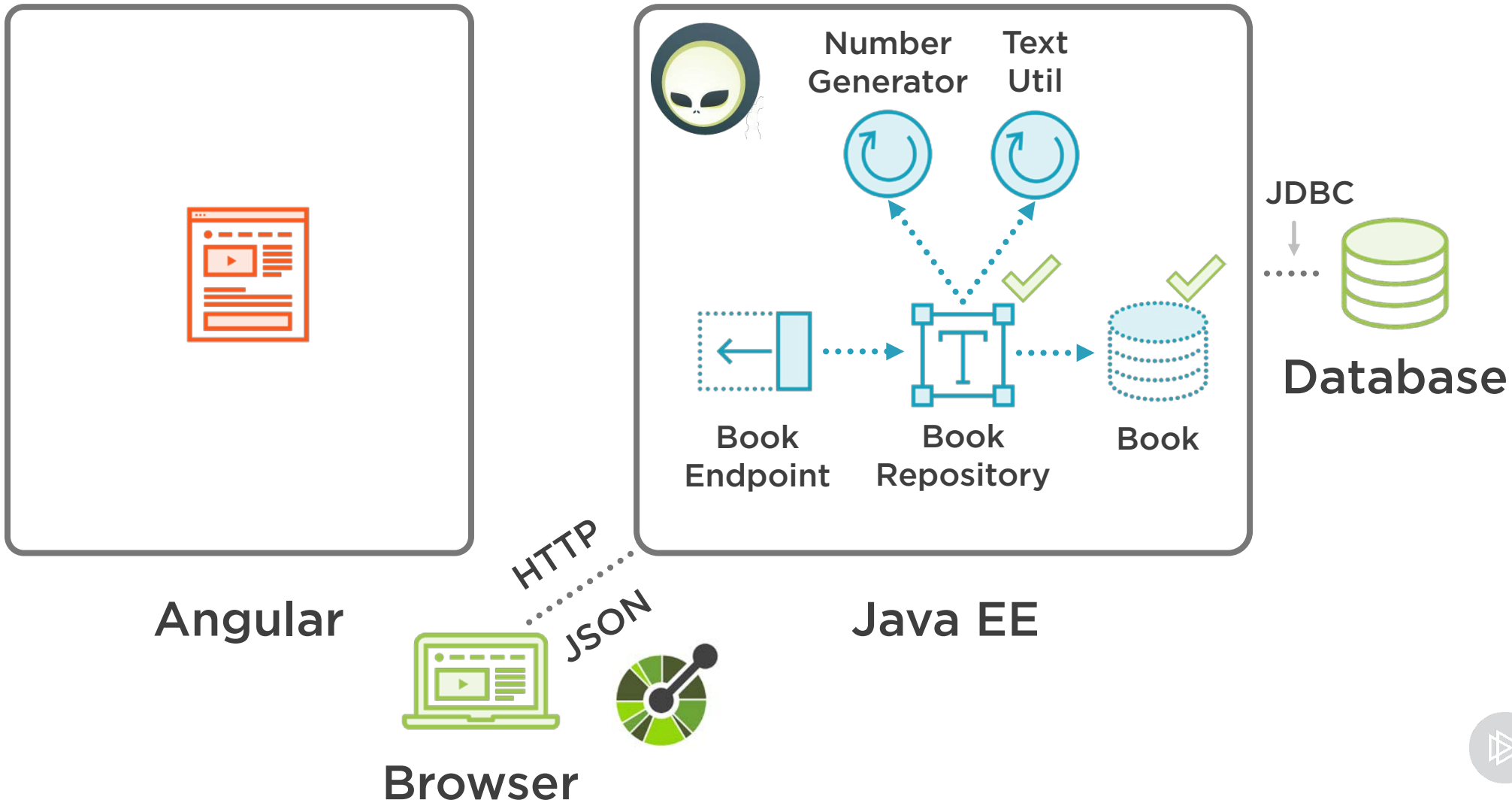
```json
"definitions" : {
 "Book" : {
  "description" : "Book representation",
  "type" : "object",
  "properties" : {
    "id" : {
      "type" : "integer",
      "description" : "Identifier”
    },
    "title" : {
      "type" : "string",
      "description" : "Title of the book”
    },
    "description" : {
      "type" : "string",
      "description" : "Describes the book”
    },
    "unitCost" : {
      "type" : "number",
      "format" : "float",
      "description" : "Unit cost”
    }
```

Demo



- Document our BookEndpoint
- Bottom-up approach
- Swagger annotations
- Generate JSon contract
- Maven plugin
- Swagger UI

Anatomy of the BookStore Application



Summary

- API documentation
- Inform external systems
- Adding Swagger annotations
- Generating JSon contract
- Automated with Maven
- Visualize contracts with Swagger UI

Next Module

Setup the Angular environment

Install other tools than the one used

Node JS

Browser

Building tool

Graphical components