3 heures

pour développer un micro-service avec les micro-frameworks Java

20/04/2016

Laurent Baresse

Developer & Geek
lbaresse@ekito.fr
@Baresse

ekito

Igor Laborie

Expert Web & Java
igor@monkeypatch.io
@ilaborie

Roadmap

Introduction to micro-XXX
MyCellar
Feign
Exercice 1
Coffee Time
SparkJava
Exercice 2

Practical notes

Requirement: JDK 8, Maven 3

Download the project

- https://ilaborie.org/DevoxxFR/FeignSparkJava_debut-exo-1.zip
- https://ilaborie.org/DevoxxFR/FeignSparkJava_debut-exo-2.zip

Or clone github project

- https://github.com/ilaborie/FeignSparkJava-exos


Introduction to micro-XXX

Microservices

Martin Fowler : Microservices (http://martinfowler.com/articles/microservices.html)

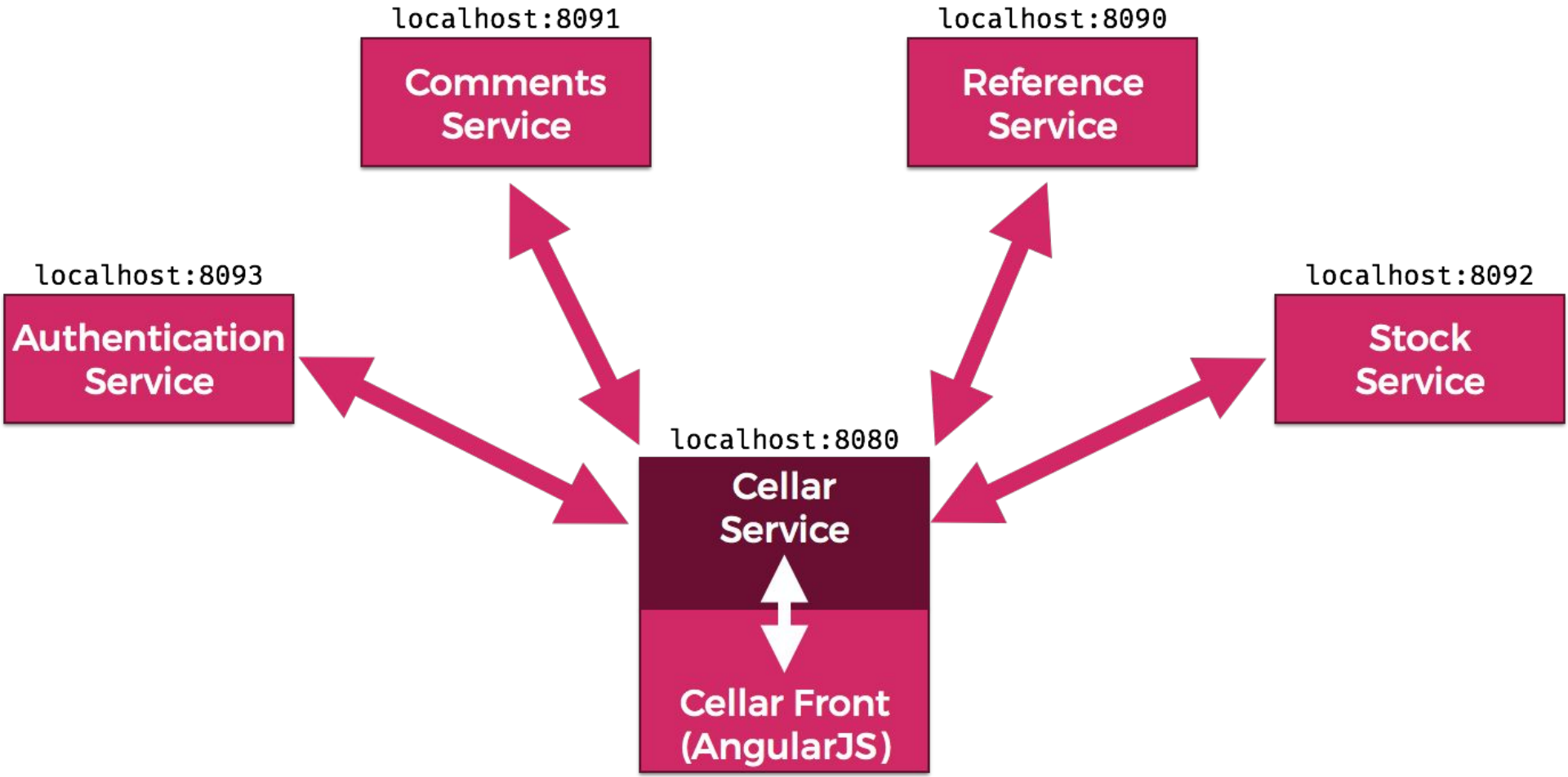
- Componentization via Services
- Decentralized governance
- Decentralized data management

Micro-frameworks

- Plenty of micro-framework availables

- Criteria
  - Simple and mastered in few hours
  - Community
  - Elegant & Java 8 oriented


MyCellar

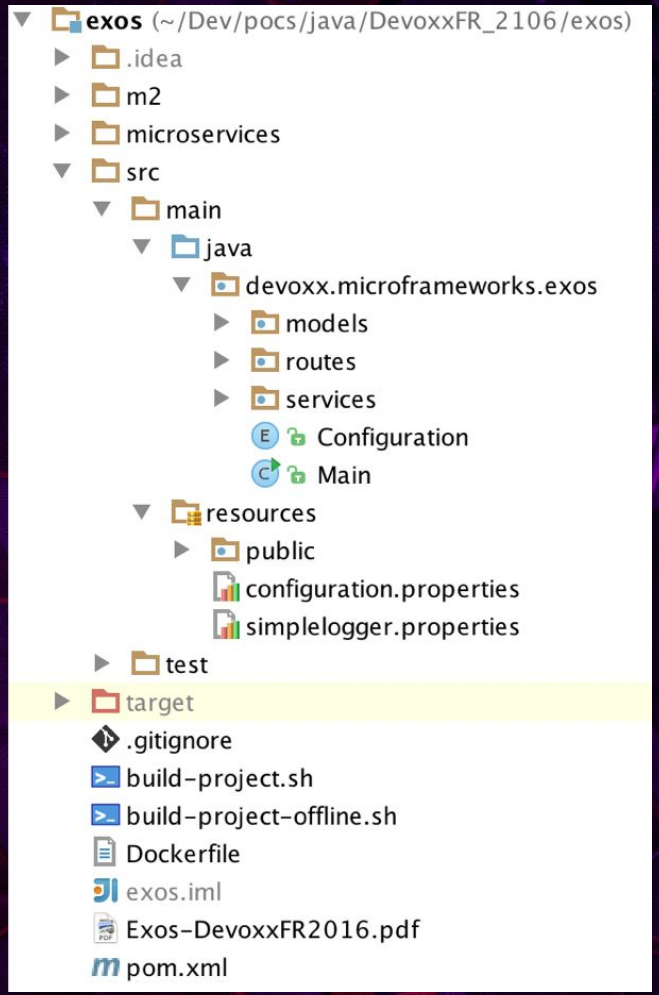

localhost:8091
Comments Service
localhost:8090
Reference Service
localhost:8093
Authentication Service
localhost:8092
Stock Service
localhost:8080
Cellar Service
Cellar Front (AngularJS)

Cellar Service

`microservices`: mockup services (authentication, reference, comment and stock)

`src/main/java` // `devoxx.microframeworks.exos`

- `models`: Pojo used in the application
- `routes`: Cellar service REST routes implementation **(Spark Java)**
- `services`: REST clients services implementation **(Feign)**
- `Main`: Cellar service Main class (embed cellar front-end)



Start others microservices

Open 4 terminals / consoles

./start-authentication-services.sh

./start-reference-services.sh

./start-comments-services.sh

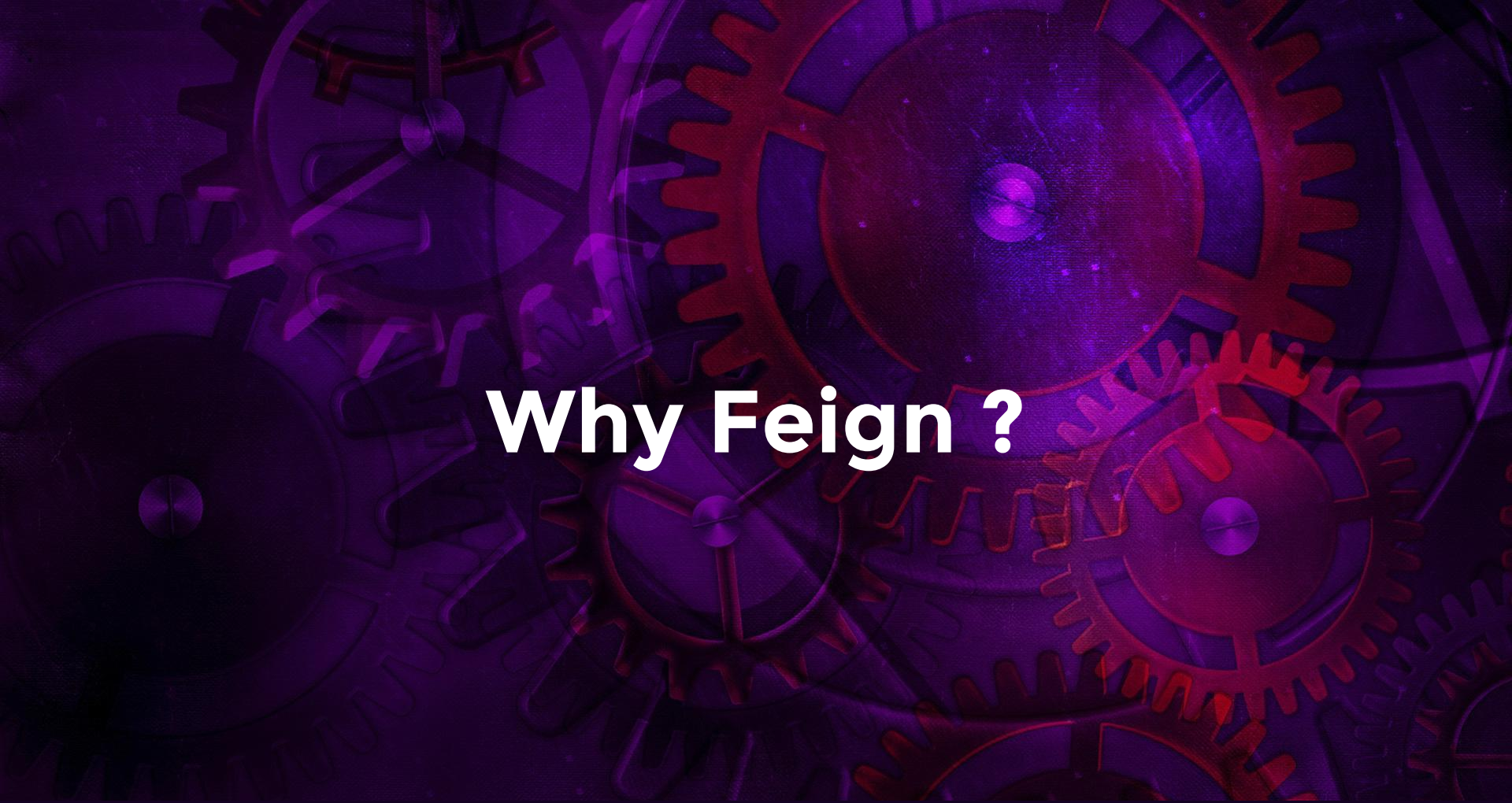
./start-stocks-services.sh

```
$ ./start-authentication-service.sh
[Thread-0] [INFO] log - Logging initialized @363ms
[Thread-0] [INFO] JettySparkServer - == Spark has ignited ...
[Thread-0] [INFO] JettySparkServer - >> Listening on 0.0.0.0:8093
[Thread-0] [INFO] Server - jetty-9.3.z-SNAPSHOT
[Thread-0] [INFO] ServerConnector - Started ServerConnector@50f244a7{HTTP/1.1,
[http/1.1]}{0.0.0.0:8093}
[Thread-0] [INFO] Server - Started @450ms
```

Or if you <3 Docker

```
$ docker-compose up
Creating sparkfeign-auth
Creating sparkfeign-reference
Creating sparkfeign-stocks
Creating sparkfeign-comments
Attaching to sparkfeign-auth, sparkfeign-reference, sparkfeign-stocks,
sparkfeign-comments
sparkfeign-auth | [Thread-0] [INFO] log - Logging initialized @578ms
sparkfeign-reference | [main] [INFO] WineDao - Parse wines database :323 wines
read.
sparkfeign-auth | [Thread-0] [INFO] JettySparkServer - == Spark has ignited ...
sparkfeign-reference | [Thread-0] [INFO] log - Logging initialized @565ms
sparkfeign-auth | [Thread-0] [INFO] JettySparkServer - >> Listening on 0.0.0.0:
8093
...
```


Feign
Consuming RESTful WebService


Why Feign ?

com.netflix.feign:feign-core:8.16.0

https://github.com/Netflix/feign

Apache License v2.0

Java 7+, Java 8 friendly

No dependencies*

* for feign-core, others extensions had few

Interface & Annotations [1/5]

```
public interface CatClient {

    List<Cat> findAll();


    Cat findById(String id);



    Cat create(Cat cat);
    // ...
}
```

Interface & Annotations [2/5]

```
public interface CatClient {
   @RequestLine("GET /cats")
   List<Cat> findAll();


   Cat findById(String id);



   Cat create(Cat cat);
   // ...
}
```

Interface & Annotations [3/5]

```
public interface CatClient {
   @RequestLine("GET /cats")
   List<Cat> findAll();

   @RequestLine("GET /cats/{id}")
   Cat findById(@Param("id") String id);



   Cat create(Cat cat);
   // ...
}
```

Interface & Annotations [4/5]

```
public interface CatClient {
   @RequestLine("GET /cats")
   List<Cat> findAll();

   @RequestLine("GET /cats/{id}")
   Cat findById(@Param("id") String id);


   @RequestLine("POST /cats")
   Cat create(Cat cat);
   // ...
}
```

Interface & Annotations [5/5]

```java
@Headers("Accept: application/json")
public interface CatClient {
   @RequestLine("GET /cats")
   List<Cat> findAll();

   @RequestLine("GET /cats/{id}")
   Cat findById(@Param("id") String id);

   @Headers("Content-Type: application/json")
   @RequestLine("POST /cats")
   Cat create(Cat cat);
   // ...
}
```

Build the instance

Use the `Feign.Builder()`

```
public static CatClient create(String url) {
   return Feign.builder()
           // extra configuration ...
           .target(CatClient.class, url);
}
```

Body Encoder & decoder

Available extensions: GSON, Jackson, JAXB, Sax

```
public static CatClient create(String url) {
   return Feign.builder()
           .encoder(new GsonEncoder())
           .decoder(new GsonDecoder())
           .target(CatClient.class, url);
}
```

It’s easy to create custom encoder/decoder.

Error Decoder

Feign throw a `FeignException` on NOK response

```
public static CatClient create(String url) {
   return Feign.builder()
           .errorDecoder((msg, response) ->
               new WTFException(response.status()))
           .target(CatClient.class, url);
}
```

RequestInterceptor customisation

`RequestInterceptor ~ Consumer<RequestTemplate>`

Example: add an HTTP header

```
public static CatClient create(String url) {
   // TODO get user JSON Web Tokens
   String jwt = "";
   return Feign.builder()
           .requestInterceptor(template -> {
               template.header("Authorization", "Bearer " + jwt));
           }).target(CatClient.class, url);
}
```

Others extensions ...

Implement `Contract` for custom annotations (e.g. JAX-RS)

Use another HTTP client implements `Client`

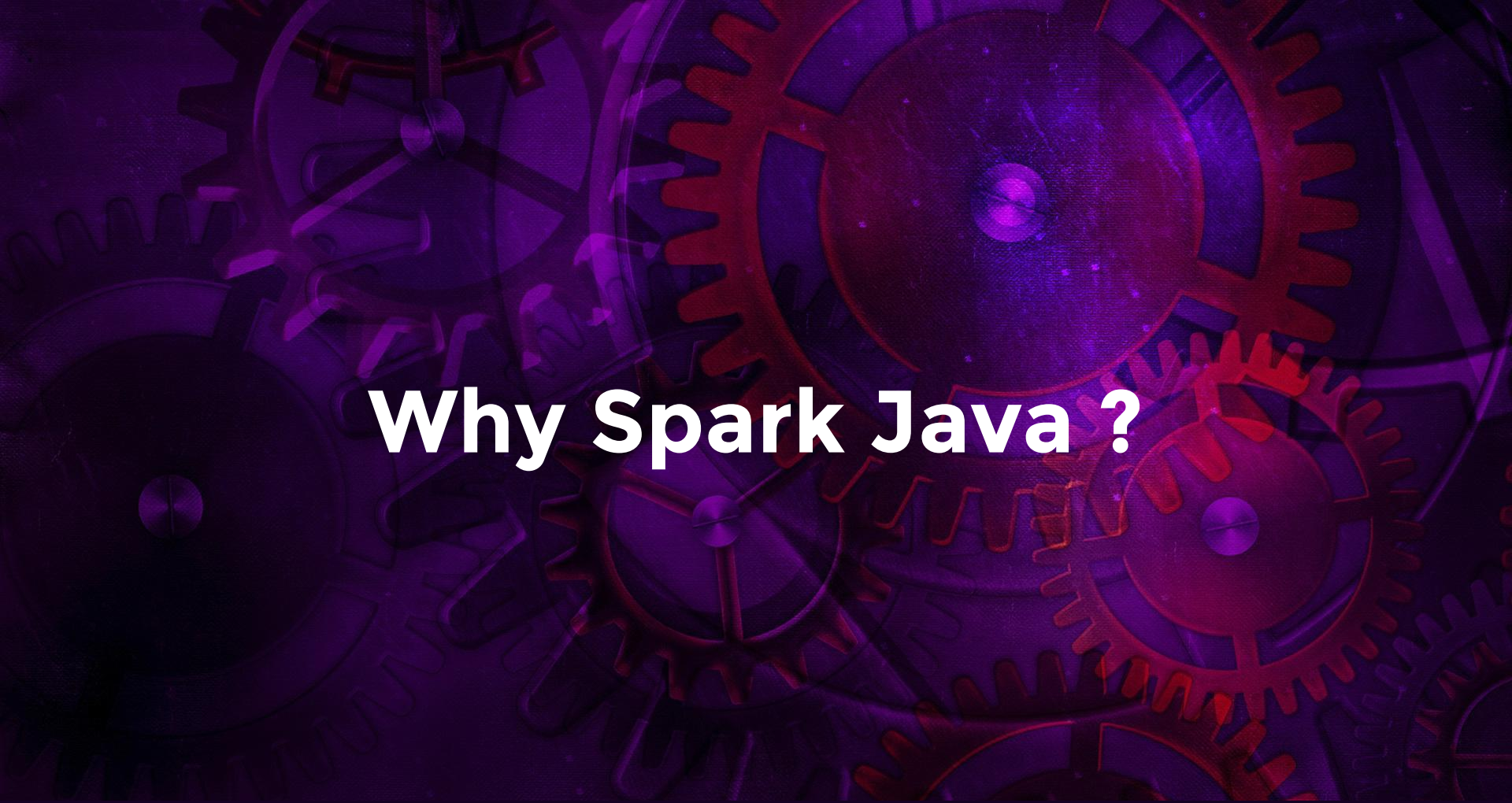
- Apache HttpClient: `feign-httpclient`
- OkHttp: `feign-okhttp`

Also have integration for

- SLF4J logging: `feign-slf4j`
- Ribbon: `feign-ribbon`
- Histryx circuit breaker: `feign-histryx`
- ...


Exercice 1


SparkJava


Why Spark Java ?

TL;TR: The promise...

```xml
<dependency>
  <groupId>com.sparkjava</groupId>
  <artifactId>spark-core</artifactId>
  <version>2.3</version>
</dependency>
```

```java
import static spark.Spark.*;

public class HelloWorld {
    public static void main(String[] args) {
        get("/hello", (req, res) -> "Hello World");
    }
}
```

```
http://localhost:4567/hello
```

Spark Java

http://sparkjava.com

Apache License v2.0

Lightweight and embedded web server (Jetty)

Roadmap

Embedded Server Configuration

Routes

- Request
- Response

Filters

Exceptions

Embedded web server - Configuration

Configuration before any route / filter mapping

```
// If provided port = 0, an arbitrary available port will //
be used ; default 4567
port(int port)


// the folder in classpath
staticFileLocation(String folder)


// the external folder serving static files
externalStaticFileLocation(String externalFolder)
```

Routes

Routes are matched in the order they are defined. The first route that matches the request is invoked.

- Verb: `get`, `post`, `put`, `delete`, `head`, `trace`, `connect`, `options`
- Path: `"/path"`, `"/path/:param"`
- Callback: `(request, response) -> { }`

Optionally:

- `Accept-type = "application/json"`
- `ResponseTranformer`

Routes matching

```
// Show something
get("/", (request, response) -> { ... });

// Create something
post("/", (request, response) -> { ... });

// Update something
put("/", (request, response) -> { ... });

// Annihilate something
delete("/", (request, response) -> { ... });

// Appease something
options("/", (request, response) -> { ... });
```

Routes matching & URL parameters

NOTE : Query parameters are not used for URL matching

```
// matches "GET /hello/foo" and "GET /hello/bar"
// request.params("name") is 'foo' or 'bar'
get("/hello/:name", (request, response) -> {
   return "Hello: " + request.params("name");
});
```

Request

See `Spark.Request` for other methods and details

```
request.body();                  // request body sent by the client
request.bodyAsBytes();           // request body as bytes

request.headers();                 // the HTTP header list
request.headers("BAR");            // value of BAR header

request.params("foo");             // value of foo path parameter
request.params();                  // map with all parameters

request.queryParams();             // the query param list
request.queryParams("FOO");        // value of FOO query param
request.queryParamsValues("FOO") // all values of FOO query param
```

Response

See `Spark.Response` for other methods and details

```
response.body("Hello");          // sets content to Hello

response.header("FOO", "bar");   // sets header FOO with value bar

response.redirect("/example");   // browser redirect (HTTP 302)
response.redirect("/bar", 301); // moved permanently

response.status(401);            // set status code to 401

response.type("text/xml");       // set content type to text/xml
```

Filters

Filters can:

read the request

read/modify the response

```
before((request, response) -> {
    // ... check if authenticated
    halt(401, "You are not welcome here");
});

before("/protected/*", (request, response) -> {
    // ... check if authenticated
    halt(401, "Go Away!");
});

after((request, response) -> {
    response.header("foo", "set by after filter");
});
```

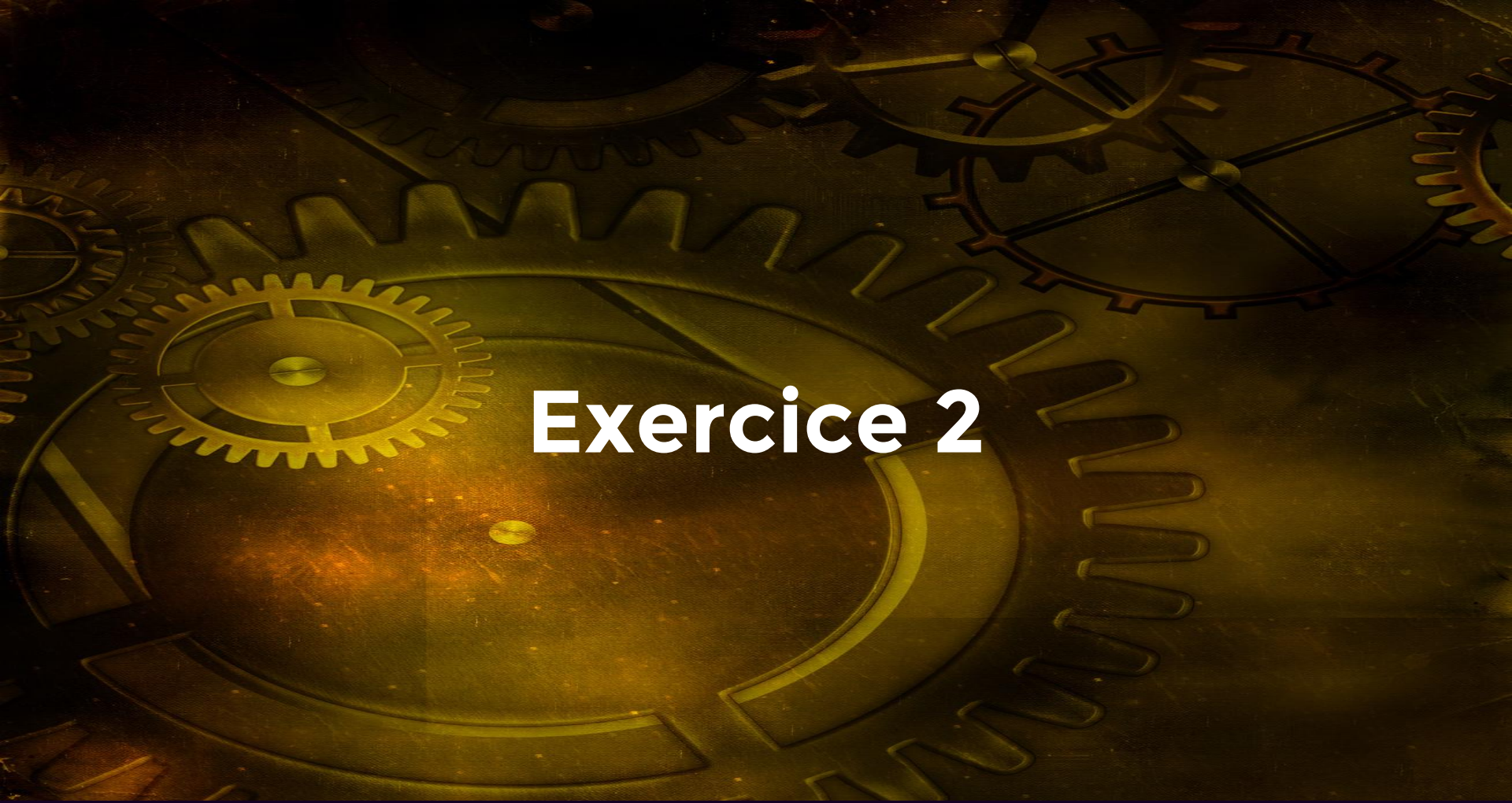
Exceptions

```
// Should an exception be thrown somewhere...
get("/throwexception", (request, response) -> {
    throw new NotFoundException();
});


// Exception handler
exception(NotFoundException.class, (e, request,
response) -> {
    response.status(404);
    response.body("Resource not found");
});
```

With Spark Java, you can also ...

- Query Maps
- Cookies
- Session
- WebSockets (only with embedded web server)
- Request/Response GZIP
- View and Templatings (mustache, thymeleaf, velocity, ...)


Exercice 2

Resources

Feign : https://github.com/Netflix/feign

SparkJava: http://sparkjava.com

Github (Exos)

https://github.com/ilaborie/FeignSparkJava-exos

Github (Microservices-Mocks)

https://github.com/ilaborie/FeignSparkJava-mocks


Q/A
Thanks !