

Rapport TP3 EIP

Realisé par

BENAOUICHA Mohand-Said et BABAGHAYOU Khaled

8 mai 2017

Sommaire

Objectifs du projet	3
Apache Camel.....	3
Question 1.....	4
Question 2.....	4
Question 3.....	5
Question 4.....	6

Objectifs du projet

Le but de ce TP est de se familiariser avec les EIP (Enterprise Integration Patterns) les plus communément rencontrés. Le système des EIP repose sur l'échange de messages (constitués d'un corps « body » et d'entêtes « headers ») entre des points d'accès (endpoints) « producteurs » et « consommateurs » à travers des « routes ».

Les EIP sont une **tentative de formalisation pour les problèmes d'intégrations** qui est historiquement apparue avec un livre de Gregor Hohpe et Bobby Woolf.

Apache Camel

Le cœur du système CAMEL est un constructeur de moteur de routage, son objectif principal est de comment, à partir d'une source et de plusieurs destinations, le routage des messages est possible, sa philosophie repose sur le fait qu'il n'a pas d'a priori sur les données transportées. En d'autres termes, ceci implique que la conversion vers un format intermédiaire quelconque pour transporter les données n'est pas nécessaire.

C'est aussi un Framework d'intégration, implémentant les EIP (Enterprise Integration Pattern) définis par Gregor Hohpe & Bobby Woolf. Les EIP sont des patterns d'intégration qui nous permettent de résoudre ces problématiques d'intégration avec des solutions éprouvées.

On utilise Camel car il offre les caractéristiques suivantes :

- Routage et médiation
- EIP (Enterprise Integration patterns)
- Bibliothèques extensibles
- Architecture modulaire
- Conversion automatique de types

Il est Très léger, Testable, Facilement configurable, et la communauté qui l'utilise est très grande et active dans le web.

L'architecture d'un message Camel est définie ci-dessous



Question 1

On a utilisé la fonction

```
Scanner sc = new Scanner(System.in);
System.out.println("saisir le message :");
str = sc.nextLine();
if(str=="exit"){System.exit(0);}
```

Question 2

```
from("direct:consumer-1").to("log:affiche-1-log");
// On définit un consommateur 'consumer-2'
from("direct:consumer-2").to("file:messages");

from("direct:consumer-all")
    .choice()
    .when(header("destinataire").isEqualTo("ecrire"))
    .to("direct:consumer-2")
    .otherwise()
    .to("direct:consumer-1");

BeginWithW = "header";
if (str.startsWith("w")) {
    BeginWithW = "ecrire";
}
```

Question 3

On a implémenté tous les service basic effectuer dans les ancien TP dans notre serveur et tous les réponses seront reçues au format JSON

- Afficher tous les animaux
- Afficher un animal
- Ajouter un animal
- Supprimer un animal
- Changer un animal
- Afficher la position d'un l'animal en utilisant le service Geonames

3.A) La méthode pour rechercher un animal par son nom et affichez le résultat

```
from("direct:CitymanagerGetByName")
    .setHeader(Exchange.CONTENT_TYPE, constant("application/json"))
    .setHeader(Exchange.HTTP_METHOD, constant("GET"))
    .to("http://127.0.0.1:8084/animals/findOne/Canin")
    .log("reponse received : ${body}");
```

```
pt.sendBody("direct:CitymanagerGetByName", "");
```

```
3222 [main] INFO route5 - reponse received : {"name":"Canine","cage":"amazon","species":"Piranha","id":"9b2c907c-f553-4900-ae2f-521ef79f6ffd"}
```

3.B)

```
@RequestMapping(path = "/findOneAnimalPositionByName/{name}", method = GET,
    produces = APPLICATION_JSON_VALUE)
public String getCagePosition(@PathVariable String name) throws
    JAXBException, AnimalNotFoundException, CageNotFoundException {
    Cage cage = center.CageByName(center.findOneAnimalByName(name).getCage());
    try {
        String string = IOUtils.toString(new InputStreamReader(new
        URL("http://api.geonames.org/findNearbyJSON?lat=" +
        cage.getPosition().getLatitude() + "&lng=" +
        cage.getPosition().getLongitude() + "&username=mlgil").openStream()));
        return string;
    } catch (MalformedURLException e) {
        throw new HTTPException(404);
    } catch (IOException e) {
        throw new HTTPException(404);
    }
}
```

Question 4

Au premier lieu on a eu du mal à instancier le service **MyviewController** en même temps, donc on a dupliqué notre service en trois **MyviewController2** **MyviewController3**, et on a lancé chaque service dans port séparé

- MyviewController port 8084
- MyviewController2 port 8085
- MyviewController3 port 8086