TP n°6

Toutes les ressources seront déclarées dans un package fr.tp.isima

Vous pouvez partir de vos propres sources ou de la correction. Toutefois la correction possède quelques éléments facilitant votre travail sur cette nouvelle version.

Exercice : Display Quote v4.0 !

Votre entreprise met en place l'intégration continue et a la ferme intention que toutes les équipes se plient à ces nouvelles règles. Pour pouvoir s'intégrer à cette plateforme, vous devez migrer votre projet display-quote à maven.

1°) Migrer le projet display quote vers maven

Suivez le doc d'installation de maven pour mettre en place l'outil sur votre machine.

Puis migrez votre tp vers Maven, suivez la documentation que vous trouverez sur github ici :

2°) Un exécutable générant des citations !

L'entreprise souhaite créer un exécutable qui affiche une citation aléatoire en lignes de commandes. Pour éviter la répétition d'informations, notre librairie va simplement appeler un service situé dans notre application Web.

a) **Créer le service findRandomQuote**

Créer une Servlet FindRandomQuoteServlet. Le service consultable en GET a un paramètre « name » correspondant au nom de l'utilisateur et rend sous forme Json la citation aléatoire.

La méthode findRandomQuote, à créer dans le classe Quote, peut-être comme suit :

```
public Quote findRandomQuote() {
	return quotes.get(ThreadLocalRandom.current().nextInt(quotes.size()));
}
```

Pour la sérialisation/désérialisation, l'utilisation de la bibliothèque Gson peut vous faciliter la vie. Par exemple une quote avec Gson :

```
final Quote q = session.findRandomQuote();
final Gson gson = new Gson();
resp.getWriter().write(gson.toJson(q));
```

Le pom est le suivant :

```
<dependency>
	<groupId>com.google.code.gson</groupId>
	<artifactId>gson</artifactId>
	<version>2.5</version>
</dependency>
```

b) **Créer le projet exécutable**

Consulter la documentation, qui détaille la façon dont on peut créer un projet exécutable avec Maven.

Ce projet, doit également être une dépendance de votre service web (ajoutez la référence dans votre pom.xml).

c) **Réaliser l'appel au service**

Il vous faut maintenant réaliser l'appel au service « findRandomQuote » depuis votre programme exécutable. Pour effectuer les tests vous pouvez écrire pour cet exercice les deux paramètres du programme en dur :

- L'url d'accès au service ;
- Le nom de l'utilisateur

Pour effectuer la requête, je vous conseille plutôt que d'utiliser l'API Url standard, une librairie Apache dite « fluent » pour effectuer vos requêtes http :

```
<dependency>
	<groupId>org.apache.httpcomponents</groupId>
	<artifactId>httpclient</artifactId>
	<version>4.5.1</version>
</dependency>
```

Voici un exemple d'appel à google avec cette librairie :

```
final HttpUriRequest uriReq =
RequestBuilder.get().setUri("http://www.google.fr/").addParameter("q",
"java").build();

final HttpClient client = HttpClientBuilder.create().build();
final HttpResponse resp = client.execute(uriReq);

try (InputStream is = resp.getEntity().getContent()) {
	final String out = IOUtils.toString(is);
	System.out.println("out " + out);
}
```

Au passage pour transformer l'inputStream en String, j'utilise l'indispensable librairie d'apache commons-io :

```
<dependency>
	<groupId>commons-io</groupId>
	<artifactId>commons-io</artifactId>
	<version>2.4</version>
</dependency>
```

d) **Finition**

Il nous reste à rendre notre programme paramétrable, afin de spécifier l'utilisateur et l'url du serveur.

Utilisons la librairie suivante spécialisée dans le parsing et la construction de paramètres :

```
<dependency>
	<groupId>commons-cli</groupId>
	<artifactId>commons-cli</artifactId>
	<version>1.2</version>
</dependency>
```

```
Voic quelques exemples d'utilisations
```

```
final Options opt = new Options();
opt.addOption("url", true, "The url of server where to find quotes");
```

Attention, le fait de mettre un argument à required n'a de conséquence que lors de l'utilisation du helpformatter:

```
final HelpFormatter hf = new HelpFormatter();
hf.printHelp("Main", opt);
```

Pour parser les arguments, faites comme suit :

```
 final CommandLineParser parser = new BasicParser();
 final CommandLine cmd = parser.parse(opt, args);
```

L'objet cmd permet avec la méthode hasOption de vérifier si une option est bien présente.