TP 2 : Parse données de Restful web services

L'objectif de ce TP est de vous faire

— invoquer des Restful web services et examiner les réponses,

— analyser deux formats données échangées : XML et JSON.

Pour cela, nous allons utiliser les web services accessibles gratuitement sur les sites `http://www.dneonline.com/calculator.asmx`, `http://api.geonames.org`, et `https://labs.bible.org`.

Pour la suite, vous devez télécharger les supports de TP sur arche (il contient un fichier de code source `callWebService.java` et les bibliothèques `jar` pour xml et json parser). Le développement est fait en Java. Vous créez un projet Java/Java Application, puis importez les sources dans le projet crée.

Notes : Vous ne modifiez pas le contenu du `callWebService.java`, ni changez son nom, sauf si on vous demande le faire comme dans le cas des exercices 4 et 5. Vous pouvez seulement le déplacer dans le package correspondant.

Exercice 1 — JSONPlaceholder web service

Cette exercice consiste à réaliser une application java permettant d'invoquer le web service sur `http://jsonplaceholder.typicode.com`. La réponse du service est un document en format JSON. L'objectif de cette exercice est d'analyser (parser) les réponses reçues pour récupérer l'information concernée.

Pour cela, vous trouverez sur Arche le fichier **callWebService.java** qui permet d'invoquer le web service et récupérer la réponse. Dans ce fichier, deux fonctions sont mises à disposition :

1. **void initializeService(String url)** : permet de connecter au serveur du web service dont adresse est donné par **url**
2. **String callJSONPlaceholderService(String serviceName, int code)** : permet d'invoquer le web service avec le nom donné par **serviceName** et un code. La fonction retourne la réponse du web service provoqué ; les données en format JSON.

Vous allez utiliser ces deux fonctions pour invoquer et récupérer la réponses, puis analyser les données JSON dans la réponse reçue et afficher le contenue de chaque élément.

Votre programme produira les affichages comme dans l'exemple ci-dessous :

```
----------------------------
url : http://jsonplaceholder.typicode.com
serviceName : posts
code : 1

----------------------------
Response of callJSONPlaceholderService
{
```

```
  "userId": 1,
  "id": 1,
  "title": "sunt aut facere repellat provident ...",
  "body": "quia et suscipit\nsuscipit recusandae ..."
}

----------------------------
Parsing the JSON response
userID : 1
ID: 1
Titre : sunt aut facere repellat provident ...
Contenu : quia et suscipit
suscipit recusandae ...
```

Exercice 2 — Information d'un pays

Dans cette exercice, nous allons analyser les données XML dans la réponse du web service REST **countryInfo** – la recherche d'information d'un pays à l'aide d'un code ISO. Ce service est accessible via le lien `http://api.geonames.org/countryInfo?country=ISO_CODE&username=LOGIN`, ou `ISO_CODE` est le code ISO-639-1 et `LOGIN` est votre identifiant.

De même, dans le fichier **callWebService.java** les fonctions suivantes sont mises à disposition :

1. **void initializeService(String url)** : permet de connecter au serveur du web service dont adresse est donné par **url**
2. **String callCountryInfoService(String serviceName, String isoCode, String login)** : permet d'invoquer le web service dont le nom est donné par **serviceName** et deux autres paramètres le code ISO **isoCode** et l'identifiant **login**. La réponse est en format XML et contient les informations d'un pays (par example : le capitale, la population, la superficie, ...).

Vous allez utiliser ces deux fonctions pour invoquer et récupérer la réponse de web service **countryInfo**, puis analyser les données XML dans la réponse reçue et afficher le contenue de chaque élément XML.

Votre programme produira les affichages comme dans l'exemple ci-dessous :

```
----------------------------
url : http://api.geonames.org
serviceName : countryInfo
isoCode : FR
login : XXX
----------------------------
Response of callCountryInfoService
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<geonames>
    <country>
```

```
        <countryCode>FR</countryCode>
        <countryName>France</countryName>
        <isoNumeric>250</isoNumeric>
        <isoAlpha3>FRA</isoAlpha3>
        <fipsCode>FR</fipsCode>
        <continent>EU</continent>
        <continentName>Europe</continentName>
        <capital>Paris</capital>
        <areaInSqKm>547030.0</areaInSqKm>
        <population>64768389</population>
        <currencyCode>EUR</currencyCode>
        <languages>fr-FR,frp,br,co,ca,eu,oc</languages>
        <geonameId>3017382</geonameId>
        <west>-5.14127657354623</west>
        <north>51.0889894407743</north>
        <east>9.56009360694225</east>
        <south>41.3645589826522</south>
        <postalCodeFormat>#####</postalCodeFormat>
    </country>
</geonames>
---------------------------
Parsing the JSON response
Nom du pays : France
Continent : Europe
Capital : Paris
Monnaie : EUR
```

Exercice 3 — NET Bible Web Service

Dans cette exercice, nous allons analyser les données en XML et JSON dans la réponse vers le service web de `https://labs.bible.org` (voir TP1). De même, dans le fichier **callWebService.java** les fonctions suivantes sont mises à disposition :

1. **void initializeService(String url)** : permet de connecter au serveur du web service dont adresse est donné par **url**
2. **String callBibleTagService(String serviceName, String titre, int chapter, int verse, String type)** : permet d'invoquer le web service dont le nom est donné par **serviceName** et les paramètre **titre**, **chapter** et **verse** correspondent respectivement au titre de l'évangile, le chapitre et la phrase dans le bible, et le format de la réponse est donnée dans le paramètre **type**.
3. **String callBibleMutiTagService(String serviceName, String[] titre, int[] chapter, int[] verse, String type)** : similaire que la fonction précédente mais avec une demande multiple sur **titre**, **chapter** et **verse**.

Vous allez utiliser ces fonctions pour invoquer et récupérer la réponse de ce web service, puis analyser les données en JSON ou XML selon la réponse reçue et afficher les éléments dans la réponse.

Votre programme produira les affichages comme dans l'exemple ci-dessous :

```
---------------------------
url : https://labs.bible.org
serviceName : api
titre : John
chapter : 3
verse : 16
type : json

---------------------------
Response of callBibleTagService
[
 {
  "bookname":"John",
  "chapter":"3",
  "verse":"16",
  "text":"For this is the way God loved the world: He gave his one and
  only Son, so that everyone who believes in him will not perish but have
  eternal life. "
 }
]

---------------------------
Parsing the JSON response
bookname : John
chapter : 3
verse : 16
text : For this is the way God loved the world: He gave his one and only Son,
so that everyone who believes in him will not perish but have eternal life.

---------------------------
url : https://labs.bible.org
serviceName : api
titre : { John, Acts }
chapter : { 3, 1 }
verse : { 16, 12 }
type : xml

---------------------------
Response of callBibleMutiTagService
<?xml version="1.0" encoding="UTF-8"?>
<bible>
<title>John 3:16;Acts 1:12</title>
<item>
<bookname>John</bookname>
<chapter>3</chapter>
```

```
<verse>16</verse>
<text>
For this is the way God loved the world: He gave his one and only Son,
so that everyone who believes in him will not perish but have eternal life.
</text>
</item>
<item>
<bookname>Acts</bookname>
<chapter>1</chapter>
<verse>12</verse>
<text>
Then they returned to Jerusalem from the mountain called the Mount of Olives
(which is near Jerusalem, a Sabbath day's journey away).
</text>
<title>A Replacement for Judas is Chosen</title>
<titles>
<title>A Replacement for Judas is Chosen</title>
</titles>
</item>
<results>2</results>
</bible>

----------------------------
Parsing the XML response
bookname : John
chapter : 3
verse : 16
text : For this is the way God loved the world: He gave his one and only Son,
so that everyone who believes in him will not perish but have eternal life.
bookname : Acts
chapter : 1
verse : 12
text :Then they returned to Jerusalem from the mountain called the Mount of
Olives (which is near Jerusalem, a Sabbath day's journey away).
```

Exercice 4 — GeoNames web service

Dans cette exercice, nous allons analyser les données XML ou JSON dans la réponse du web service **Cities and Placenames Webservice** en fonction du choix dans le requête de l'utilisateur. Le service est documenté via le lien `http://www.geonames.org/export/JSON-webservices.html`.

Pour cela, créez une application Java avec **Swing GUI** ou **JavaFX** pour saisir les paramètres de la requête, ainsi que deux boutons de type radio pour choisir le type de données dans la réponse. Dans le formulaire de l'application, mettez par défaut les coordonnées : **north=44.1, south=-9.9, east=-22.4 et west=55.2**. Modifiez le fichier

fourni **callWebService.java** pour invoquer le service demandé avec ses paramètres puis affichez les noms des villes qui sont dans la réponse dans votre application. Inspirerez vous des programmes précédentes.

```
---------------------------
Cities and Placenames

Webservice Type : REST
Url for JSON data : api.geonames.org/citiesJSON
Url for XML data : api.geonames.org/cities

Parameters :
north,south,east,west : coordinates of bounding box

Result : returns a list of cities and placenames in the bounding box,
ordered by relevancy (capital/population). Placenames close together are
filterered out and only the larger name is included in the resulting list.

Exemples :
for JSON data :
http://api.geonames.org/citiesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2
&username=demo

for XML data :
http://api.geonames.org/cities?north=44.1&south=-9.9&east=-22.4&west=55.2
&username=demo
```

En cas de problème avec le **username=demo**, vous pouvez créer un compte sur le site : `http://www.geonames.org/login` puis activer l'utilisation des services web de votre compte en cliquant sur "**Click here to enable the free webservices**" sur le site : `http://www.geonames.org/manageaccount`

Exercice 5 — ContryInfo web service

Dans cette exercice, nous allons analyser les données XML, JSON ou CSV dans la réponse du web service **Country Info** (voir Exercice 2). Le service est documenté via le lien `https://www.geonames.org/export/web-services.html#countryInfo`.

Explorez ce service **Country Info** et créez trois exemples de requête de ce service pour récupérer les données dans la réponse en trois formats différents : XML, JSON et CSV. Testez les avec Postman.

Créez une application Java avec **Swing GUI** ou **JavaFX** pour saisir les paramètres de la requête, ainsi que trois boutons de type radio pour choisir le type de données dans la réponse. Modifiez le fichier fourni **callWebService.java** pour invoquer le service demandé avec ses paramètres puis affichez les noms des villes qui sont dans la réponse dans votre application. Inspirerez vous des programmes précédentes.