

TD2 – Service Web Tomcat RESTcountries

Prérequis

Téléchargez et décompressez à racine du répertoire `${WORKSPACE_DIR}` :

- le fichier **TD2 – Service Web Tomcat RESTcountries** :
 - **td2.tgz** sur Moodle

À l'issue de ces prérequis, votre répertoire de travail devra ressembler à :

```
${WORKSPACE_DIR}
  apache-tomcat-10.1.19
  javafx-sdk-21.02.2
  td0
  td1
  td2
  21-RESTcountries (contenant un script deploy.*)
```

L'objectif de ce TD est d'implémenter sur un serveur **Tomcat** un **service web simplifié** en Java qui consommera RESTcountries.

Votre serveur **Tomcat** sera a priori sur <http://localhost:8080>

Votre **service web simplifié** sera a priori sur <http://localhost:8080/21-RESTcountries>.

Pour l'API **HttpURLConnection**, vous pouvez ajouter le proxy :

Au milieu des importations :

```
import java.net.Proxy ;
```

Au milieu de votre appel à **HttpURLConnection** :

```
Proxy proxy = new Proxy(Proxy.Type.HTTP, new InetSocketAddress("129.20.239.9", 3128)) ;
HttpURLConnection connection = (HttpURLConnection) url.openConnection(proxy) ;
```

Avant de démarrer le serveur Tomcat, il faut le configurer en ligne de commande pour utiliser le proxy :

```
export JAVA_OPTS="-Dhttp.proxyHost=129.20.239.9 -Dhttp.proxyPort=3128"
```


Exercice 1 : Lister les pays

Développez une servlet **MyServletAll** qui fait la même chose que l'exercice 3 du TP1 sur les services web. Elle devra être accessible via l'URL dans un navigateur :

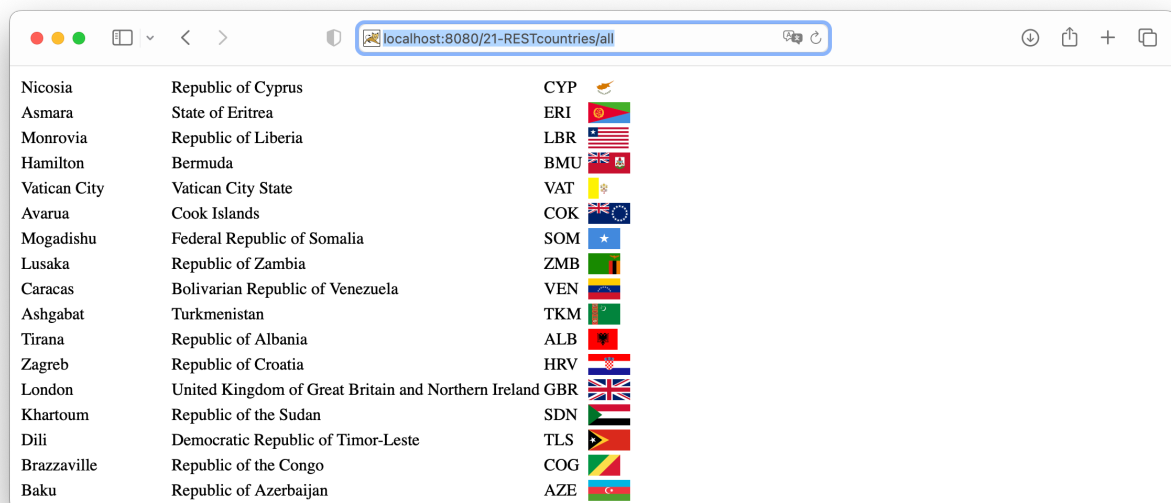
<http://localhost:8080/21-RESTcountries/all>

La servlet retournera en **GET** une page HTML contenant un tableau.
Le tableau affichera les colonnes **capital**, **name**, **cca3** et **flag** (png).
Le tableau contiendra un nombre de lignes qui peut être limité par **count** un paramètre qui peut être passé à l'URL.

Vous utiliserez l'API **Servlet** de **Jakarta** et l'API **URLConnection**.

Vous implémenterez le fichier :

- `src/MyServletAll.java` (contenant la servlet).



Exercice 2 : Trouver un pays

Développez une servlet **MyServletFind** qui permet de trouver un pays à partir de la capitale dont le nom est passé en paramètre avec **capital**. Elle devra être accessible via l'URL dans un navigateur :

<http://localhost:8080/21-RESTcountries/find?capital=>

La servlet retournera en **GET** une page HTML contenant un tableau.

Le tableau affichera les colonnes **capital**, **name**, **cca3** et **flag** (png) affiché avec une image de 20px de haut.

Le tableau sera vide si le résultat n'est pas unique ou s'il n'y a pas de résultat.

La servlet retournera en **POST** un objet JSON.

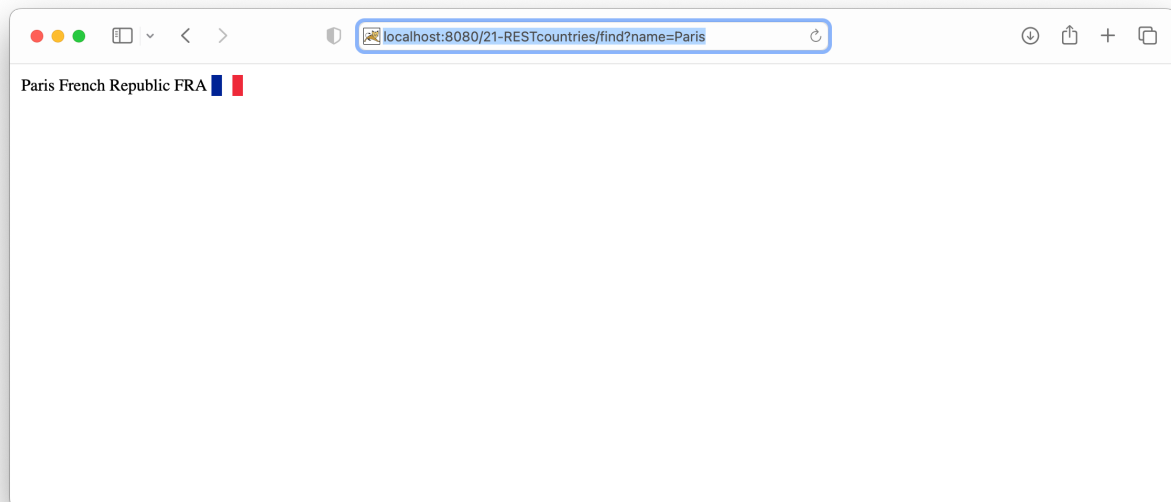
L'objet contiendra les champs **capital**, **name**, **cca3** et **flag** (png).

L'objet sera vide si le résultat n'est pas unique ou s'il n'y a pas de résultat.

Vous utiliserez les API **Json** et **Servlet** de **Jakarta** et l'API **HttpURLConnection**.

Vous implémenterez le fichier :

- `src/MyServletFind.java` (contenant la servlet).



```
{
  "capital" : "Paris",
  "name" : "French Republic",
  "cca3" : "FRA",
  "flag" : " https://flagcdn.com/w320/fr.png"
}
```

Exercice 3 : Proposer des capitales

Développez une servlet **MyServletComplete** qui permet de lister les capitales à partir de quelques lettres passées en paramètre avec **pattern**. Les autres informations seront le début de la liste et la taille de la liste passés en paramètres avec **from** et **count**. Elle devra être accessible via l'URL dans un navigateur :

<http://localhost:8080/21-RESTcountries/complete?pattern=&from=&count=>

La servlet retournera en **GET** une page HTML contenant un tableau.
Le tableau affichera une seule colonne contenant la liste des capitales.

La servlet retournera en **POST** un tableau JSON.
Le tableau contiendra la liste des capitales.

Les capitales proposées seront si possible triées par ordre alphabétique.

On ne tiendra pas compte de la casse.

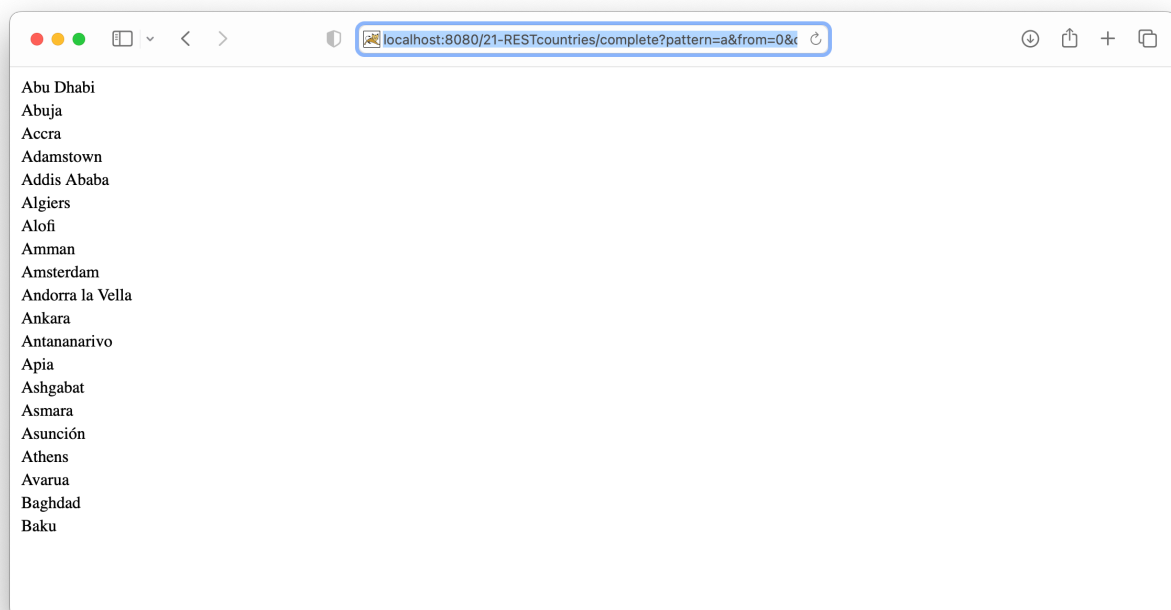
Si **pattern** est vide, on proposera la liste de toutes les capitales.

Par défaut, on prendra **from=0** et **count=10**.

Vous utiliserez les API **Json** et **Servlet** de **Jakarta** et l'API **HttpURLConnection**.

Vous implémenterez le fichier :

- **src/MyServletComplete.java** (contenant la servlet).



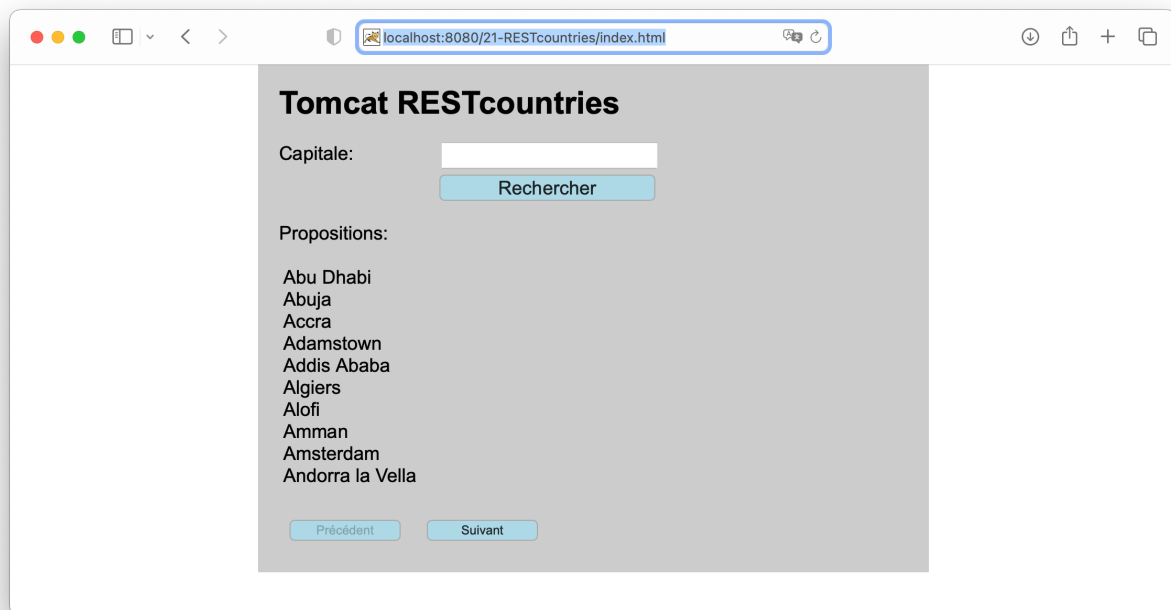
["Abu Dhabi", "Abuja", "Accra", ..., "Asuncion", "Athens", "Avarua", "Baghdad", "Baku"]

Exercice 4 : Compléter la recherche

Introduction :

Dans cet exercice, le fichier HTML de référence est fourni. Vous y accédez par l'URL :

<http://localhost:8080/21-RESTcountries/index.html>



Il est constitué :

- D'un formulaire de recherche de capitale constitué :
 - D'un champ de recherche **id="pattern"** ;
 - De deux champs cachés **id="from"** et **id="count"** ;
 - D'un bouton **[Rechercher]** dont l'appui déclenchera la fonction **handleFind()** qui permettra de lancer une recherche ;
- D'une **<div>** vide, dite de propositions, appelée **id="proposals"** qui affichera les propositions de recherche de capitale à partir des caractères entrés dans le formulaire ;
- De deux boutons **[Précédent]** et **[Suivant]** qui permettra de reculer en déclenchant la fonction **handlePreviousProposals()** ou d'avancer en déclenchant la fonction **handleNextProposals()** dans les propositions ;
- D'une **<div>** vide, dite de résultats, appelée **id="results"** qui affichera le résultat de la recherche.

Chaque modification du champ de recherche déclenchera la fonction **handleResetProposals()** qui met à jour les propositions.

Vous utiliserez les API **Json** et **Servlet** de **Jakarta** et l'API **HttpURLConnection**. Vous pourrez organiser les scripts JavaScript à votre guise. **N'oubliez d'insérer vos scripts dans le fichier HTML !**

Partie 1 :

Développez la fonction **handleFind()**. Elle devra envoyer une requête à la servlet **MyServletFind** en **POST** et devra remplir la **<div>** de résultats. **Attention, les échanges sont en JSON, il faudra interpréter le JSON !**

Pour le formatage des résultats, reprenez l'affichage HTML de l'exercice 2 mais générez-le en JavaScript à partir du JSON qu'il faudra donc interpréter !

En cas de souci, vous pourrez toujours faire des essais en GET pour éviter les difficultés du JSON.

Partie 2 :

Développez la fonction **handleResetProposals()**. Elle devra envoyer une requête à la servlet **MyServletComplete** en **POST** et devra remplir la **<div>** des propositions.

Pour le formatage des résultats, reprenez l'affichage HTML de l'exercice 3 mais générez-le en JavaScript à partir du JSON qu'il faudra donc interpréter !

En cas de souci, vous pourrez toujours faire des essais en GET pour éviter les difficultés du JSON.

Partie 3 :

Développez les fonctions **handlePreviousProposals()** et **handleNextProposals()** pour naviguer et mettre à jour la liste des propositions.

Partie 4 :

Rendez les propositions interactives de sorte que lorsqu'on clique sur l'une d'elle :

- On change le contenu du champ de recherche ;
- On effectue la recherche sur ladite proposition.