Rapport - Projet de XML

> Alexandre Ly **71701881** - Thomas Bignon **71701742**

Conversion des données CSV en XML

Pour cette partie, nous avons fait le choix d'utiliser Python et la bibliothèque Pandas pour parser le CSV et traiter les données. C'est le fichier `csv_to_xml.ipynb` qui s'occupe de produire le fichier `base_ratp.xml` .

Les données conservées pour le XML sont les noms des stations, leurs descriptions (l'adresse), leurs identifiants et leurs coordonnés converties. Les coordonnées sont leurs positions pour le plan de l'extension, elles correspondent à leurs positions `x` et `y` entre `1000` et `9000` . Nous avons ensuite pour chaque ligne la liste des identifiants des stations et la liste des changements pour certaines stations.

Format du XML

Pour la racine, nous avons une balise `<root>` qui contient la balise `<data>` pour les données des stations et la balise `<lines>` pour la liste des lignes.

Données des stations

La balise `data` contient la liste des stations présentées comme ceci :

```
<station  id="2536"  x="4757"  y="6456">
    <name>Lamarck-Caulaincourt</name>
    <desc>Lamarck 53/53 bis rue - 75118</desc>
</station>
```

Données des lignes

La balise `<lines>` contient une liste de ligne présentée comme ceci :

```
<line  name="14"> ... </line>
```

Pour présenter la liste des stations, plusieurs cas se présentent :

- Quand la ligne n'a aucune bifurcation, il n'y a qu'une balise `<mpath>` .
- Quand la ligne a une bifurcation au milieu, il y a une balise `<lpath>` , une balise `<bifur>` contenant 2 balises `<subpath>` représentant la partie en haut pour la première puis la partie en bas pour la deuxième et enfin une balise `<rpath>` pour la fin de la ligne.
- Quand la ligne se sépare et fini par 2 terminus, il y a une balise `<fpath>` puis, comme précédemment, une balise `<bifur>` contenant 2 balises `<subpath>` .

Toutes ces balises “path” contiennent ensuite la liste des stations du segment qu’il représente. Les stations contiennent un ID et la liste des changements s’il y en a :

```
<station>
    <id>2224</id>
    <changes>
        <changeline>14</changeline>
        <changeline>11</changeline>
        <changeline>1</changeline>
        <changeline>7</changeline>
    </changes>
</station>
```

Schéma XML

Le schéma XML est dans le fichier `base_ratp.xml` et valide bien notre fichier XML :

```
❯ xmllint --schema base_ratp.xsd base_ratp.xml > out.txt
base_ratp.xml validates
```

Conversion des données XML en SVG

Nous avons implémenté tout ce qui était demandé. Pour générer le SVG d’une ligne, il faut exécuter la commande :

```
java -jar ./src/saxon-he-10.3.jar -s:src/base_ratp.xml -xsl:src/svg.xsl line=[code de la ligne] -
```

Pour générer toutes les stations, il faut exécuter le script `generate_all.sh` :

```
cd src && sh generate_all.sh
```

Nous avons aussi implémenté une extension. Nous avons créé un autre fichier `extension.xsl` qui génère une carte des lignes de métro parisiennes en fonction des coordonnées `x` et `y` de chaque station. Nous avons dû d’ailleurs modifier les coordonnées de la station Corentin-Celton dans `base_ratp.csv` car la carte générée nous a fait remarquer l’erreur.

Pour la générer, il faut exécuter la commande :

```
java -jar ./saxon-he-10.3.jar -s:base_ratp.xml -xsl:extension.xsl -o:extension.svg
```

Outputs

Tous les fichiers produits sont dans le répertoire `output/` .