William LIN | Ange PAGEL | Nelson PEREIRA | Jia YAN | Yin XU

TP OpenStreetMap (Partie Python)

`27 janvier 2022` `Ensimag 5MMGDDGE` `Systèmes d'Information Géographique`

Contexte

Cette seconde partie du TP OpenStreetMap se concentre sur le développement de l'opération `GetMap` d'un serveur WMS (Web Map Service) : on souhaite que le serveur soit capable de retourner une image (*tuile*) représentant un sous-ensemble (une couche) des éléments d'une base de données géographique dans une région donnée.

D'autre part, un client utilisant la bibliothèque Leaflet permet de requêter ce serveur WMS et de superposer les tuiles récupérées sur un fond de carte.

> **Note** : En complément de cette documentation, vous pouvez vous référer aux nombreux commentaires du code pour comprendre son fonctionnement en détails.

Architecture

Client

Le client consiste en deux fichiers, `index.html` et `map.js`. Il s'appuie sur la bibliothèque JavaScript Leaflet, dont le code de `map.js` se sert pour paramétrer les communications avec le serveur WMS et afficher les différentes couches de tuiles sur un fond de carte.

Serveur

- L'architecture de notre serveur WMS reprend l'architecture de base du serveur WMS servant de base à ce TP, situé dans le fichier `WMSserver.py`.
- Le code relatif à la génération des tuiles est détaché dans le fichier `tiler.py`, et le paramétrage des couleurs d'affichage des différents éléments est isolé dans le fichier `tiler_colors.py`.
- Le module `drawer.py` encapsule les fonctionnalités de génération d'image de la bibliothèque `Pycairo`, et expose des méthodes utiles à la représentation des données géographiques (dessin de lignes, de polygones).
- La connexion à la base de données PostgreSQL est établie à l'aide des fonctions du module `database.py` (basé sur les bibliothèques `psycopg2` et `postgis`) et du fichier de configuration `config.py` associé.

Descriptif des fonctionnalités

Affichage de plusieurs couches

En plus de la couche `highways` (routes) traitée au cours du TP, notre serveur WMS supporte également le rendu des couches `buildings` (bâtiments), `natural` (terrains naturels) et `waterways` (cours d'eau).

Côté client, le code de `map.js` a été adapté pour requêter ces nouvelles couches et configurer leur superposition sur le fond de carte. Sur la carte, un contrôle permet de les afficher/masquer dynamiquement.

Paramétrage des couleurs d'affichage

Toutes les routes et tous les bâtiments ne s'affichent pas de manière identique ; en effet, il est possible de paramétrer la couleur d'affichage des différents éléments d'une couche depuis le fichier `tiler_colors.py`. Il s'agira par exemple, pour les `buildings`, de représenter les `apartments` en rouge et les `house` en bleu.

Mécanisme de mise en cache des tuiles

Un mécanisme de mise en cache des tuiles permet d'améliorer le temps de réponse du serveur pour les requêtes ciblant une région et une couche pour lesquelles les tuiles ont déjà été générées. Son implémentation repose sur le nommage des fichiers d'après la couche et la région concernées ; à chaque requête, le programme est en mesure de vérifier rapidement si l'image existe déjà, auquel cas il la renvoie directement et économise beaucoup de calcul.

Le nom d'un fichier est construit ainsi, où `layer` correspond à la couche demandée et `x0`, `y0`, `x1`, `y1` aux coordonnées de la région à dessiner :

```
tiles/{layer}/{x0}-{y0}-{x1}-{y1}-{layer}.png
```

Choix de conception

Projection des points géographiques sur l'image

Nous manipulons d'un côté des coordonnées géographiques exprimées dans un système de référence spatiale, et de l'autre une image représentée comme une matrice de pixels. Pour former une jolie tuile et représenter de manière cohérente les données géographiques, nous devons effectuer quelques transformations (très bien commentées dans le code de `tiler.py`).

Plutôt que de solliciter le SGBD et de réaliser ces transformations directement dans la requête SQL à l'aide des fonctions de PostGIS, nous avons choisi d'effectuer les calculs directement dans le code de `tiler.py`. Ce choix est motivé par un allègement et un gain de lisibilité de la requête SQL (qui peut déjà s'avérer relativement longue à exécuter), et un meilleur respect du principe de séparation des préoccupations (*Separation of concerns*).

Tests effectués

Aucun test automatique n'est mis en place ; le test du serveur WMS et des fonctionnalités de génération de tuiles a été réalisé « à la main ».

Points d'amélioration

Obsolescence des tuiles en cache

En l'état, toute tuile générée est mise en cache pour une durée indéterminée (jusqu'à la suppression du fichier). On imagine assez naturellement qu'il est préférable de rafraîchir périodiquement ces tuiles, car une grande quantité de données géographiques évolue dans le temps : une nouvelle route est construite, un bâtiment est détruit, etc.

Il s'agira d'améliorer le mécanisme de cache existant pour vérifier, sur la base de la date de création du fichier ou d'un timestamp intégré à son nom, si la tuile en cache que l'on s'apprête à renvoyer a besoin d'être générée à nouveau. Il faut alors définir un intervalle déterminant la « durée de validité » d'une tuile (par exemple « 1 mois »).