# Traitement fichiers GTFS en Java

## I, Les bibliotheques utiles

Les bibliotheques pour lire et traiter les fichiers GTFS en Java:

1. **OneBusAway GTFS Modules**

* **Description** : La bibliothèque **OneBusAway GTFS Modules** est un ensemble de modules qui facilitent la lecture, l'analyse, et la manipulation des fichiers GTFS en Java.
* **URL** : <https://github.com/OneBusAway/onebusaway-gtfs-modules>
* **Fonctionnalités** :
  + Lecture des fichiers GTFS.
  + Chargement des données GTFS dans des structures de données Java.
  + Validation des données GTFS.
  + Requêtes sur les trajets, arrêts, routes, etc.
* **Uilisation**

Importer ce librairie dans fichier pom maven, puis importer diretement sur le code Java.

<dependency>

<groupId>org.onebusaway</groupId>

<artifactId>onebusaway-gtfs</artifactId>

<version>**version plus récen**t</version>

</dependency>

1. **GTFS-lib**

* **GTFS-lib** est une autre bibliothèque Java qui facilite la lecture et l'analyse des fichiers GTFS. Elle permet de convertir les fichiers GTFS en objets Java et de travailler avec ces objets.
* **URL** : <https://github.com/conveyal/gtfs-lib>
* **Fonctionnalités** :
  + Lecture et écriture des fichiers GTFS.
  + Recherche des trajets, arrêts, et horaires dans les fichiers GTFS.
  + Validation des fichiers GTFS pour assurer la conformité avec le format.
* **Utilisation**

Importer ce librairie dans fichier pom maven, puis importer diretement sur le code Java.

<dependency>

<groupId>com.conveyal</groupId>

<artifactId>gtfs-lib</artifactId>

<version>3.0.0</version>

</dependency>

1. **opentripplanner/otp**

**OpenTripPlanner** est une solution de planification de trajets qui peut lire et analyser les fichiers GTFS. Bien qu'il s'agisse d'un système de planification de trajets complet, il peut également être utilisé pour lire des fichiers GTFS dans Java.

* **Description** : **OpenTripPlanner** est une solution de planification de trajets qui peut lire et analyser les fichiers GTFS. Bien qu'il s'agisse d'un système de planification de trajets complet, il peut également être utilisé pour lire des fichiers GTFS dans Java.
* **URL** : <https://github.com/opentripplanner/OpenTripPlanner>
* **Fonctionnalités** :
* Lecture des fichiers GTFS.
* Génération de graphiques de transport à partir des données GTFS.
* Recherche d'itinéraires en utilisant les données de transport.

## II, Le projet Java pour travailler avec des fichiers GTFS de Hanoi

Nous allons construire un exemple complet d'un projet Maven en Java sur Eclipse pour lire, analyser et visualiser les données de transport public de Hanoi à partir des fichiers GTFS de HanoiS. Nous utiliserons la bibliothèque **OneBusAway GTFS Modules** pour manipuler les fichiers GTFS. Cet exemple comprend :

1. **Structure de projet Maven**.
2. **Code Java pour lire et analyser les fichiers GTFS**.
3. **Visualisation simple des données**.
4. **Fichier pom.xml pour les dépendances Maven**.

### Structure du Projet Maven

Ce sera une structure classique de projet Maven avec un fichier *pom.xml* pour gérer les dépendances que nous utiliserons dans le projet. Nous aurons un fichier *GTFSAnalyse* pour y placer le code Java, ainsi qu'un répertoire *resources* pour les fichiers GTFS.

### Code Java pour lire et analyser les fichiers GTFS.

Ce projet indépendant a pour but de lire et traiter les fichiers GTFS. Il s'agit d'un projet Maven qui gère les dépendances. Dans ce projet, nous utilisons la bibliothèque OneBusAway.

Voici la structure du projet :

Une image contenant texte, capture d’écran, Police

Description générée automatiquement

* Les fichiers GTFS sont enregistrés dans src/main/resources
* Le code Java est situé dans src/main/java, dans le package gtfs

#### Le détail du code

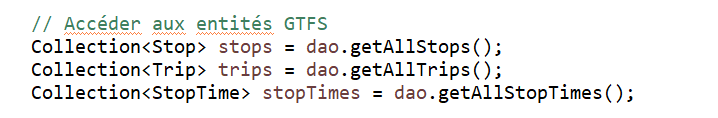
1. **Chargement des données GTFS :**

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

* Cette section du code indique le chemin vers le répertoire contenant les fichiers GTFS.
* Elle configure le lecteur de fichiers GTFS (GtfsReader) et lit toutes les entités (arrêts, trajets, etc.) en les stockant dans GtfsDaoImpl.
* L'appel à reader.run() lit effectivement tous les fichiers dans le répertoire spécifié.

1. **Récupération des données GTFS :**

****

**Description :**

* **dao.getAllStops()** : Récupère tous les arrêts (contenus dans le fichier stops.txt).
* **dao.getAllTrips()** : Récupère tous les trajets (contenus dans le fichier trips.txt).
* **dao.getAllStopTimes()** : Récupère tous les horaires des trajets, c'est-à-dire les moments où les bus/trains s'arrêtent aux différents arrêts (contenus dans le fichier stop\_times.txt).

1. **Affichage des coordonnées (latitude et longitude) de chaque arrêt :**

**Une image contenant texte, Police, ligne, capture d’écran

Description générée automatiquement**

* **Colonnes utilisées :**
  + **stop\_name** : Nom de l'arrêt (affiché dans stop.getName()).
  + **stop\_lat** : Latitude de l'arrêt (utilisée pour stop.getLat()).
  + **stop\_lon** : Longitude de l'arrêt (utilisée pour stop.getLon()).
* **Description :**
  + Cette fonction récupère les informations de latitude et longitude pour chaque arrêt et les affiche avec le nom de l'arrêt.

1. **Affichage du planning des trajets (timetable) par ligne**

* **Fichiers :**
* **trips.txt :** Pour chaque trajet (trip), on récupère l'ID du trajet et l'ID de la ligne associée.
* **stop\_times.txt :** Pour chaque trajet, on récupère les horaires d'arrivée et de départ aux différents arrêts.
* **stops.txt :** Pour chaque arrêt, on récupère son nom et ses coordonnées.
* **Colonnes utilisées :**
* **Dans trips.txt :**
  + - **trip\_id :** Identifiant du trajet (affiché avec trip.getId()).
    - **route\_id :** Identifiant de la ligne à laquelle appartient le trajet (affiché avec trip.getRoute().getId()).
  + **Dans stop\_times.txt :**
    - **arrival\_time :** Heure d'arrivée à l'arrêt pour ce trajet (récupéré avec stopTime.getArrivalTime()).
    - **departure\_time :** Heure de départ de l'arrêt (récupéré avec stopTime.getDepartureTime()).
    - **stop\_id :** Identifiant de l'arrêt pour ce trajet (lié au fichier stops.txt pour récupérer les noms d'arrêts).
  + **Dans stops.txt :**
    - **stop\_name :** Nom de l'arrêt (affiché avec stop.getName()).
* **Description :**
* Cette fonction affiche le planning (timetable) de chaque trajet. Pour chaque trajet, elle récupère les arrêts associés (via le fichier stop\_times.txt), puis affiche les heures d'arrivée et de départ de chaque arrêt.

**Une image contenant texte, capture d’écran, Police, nombre

Description générée automatiquement**

1. **Conversion des secondes en format hh:mm:ss**

**Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement**

* **Description :**
  + Cette fonction convertit les temps d'arrivée et de départ, qui sont en **secondes depuis minuit**, en un format lisible **hh:mm:ss**. Ce format est couramment utilisé dans les fichiers GTFS pour représenter les heures.

1. **Filtrage des StopTimes pour un trajet spécifique**

**Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement**

* **Fichier : stop\_times.txt**
* **Colonnes utilisées :**
* **trip\_id :** Identifiant du trajet (utilisé pour filtrer les StopTime associés à ce trajet particulier).
* **Description :**
* Cette fonction filtre les horaires (StopTime) pour ne conserver que ceux qui correspondent à un trajet spécifique (défini par son trip\_id). Cela permet d'afficher les arrêts et les horaires pour ce trajet uniquement.