Assignment – Course 2

Objective

In this project, you will exercise your knowledge of Scala from the course. The project is also a practice of join operation of multiple collections (data enrichment) In this project, you will achieve:

- ✓ Work with Scala collections
- ✓ Work with classes, traits, objects, and case classes
- ✓ Practice your knowledge in data enrichment (join)

Before you start

Data set

This time we use STM GTFS data set. In this project, we continue the practice of enrichment (the most common data transformation task to do). To download the dataset, visit http://www.stm.info/en/about/developers

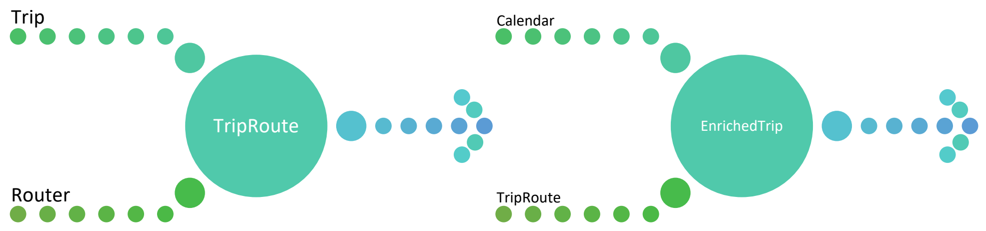
You have already analyzed the structure of the data. Note the path you have saved the dataset for further use in the application.

Project requirements

Enrich **trip** with **route** based on *route_id* and then enrich the result with **Calendar** based on *service_id*. At the end, you should write the final result in a CSV file with a proper header.

Note that here we create a new object that has information of trip, routes and calendar. It's possible to do both joins at the same time. But, for simplicity, do one at a time. Hence, you will have an intermediate class as well.

a. Trip
b. Calendar
c. Route
d. TripRoute (*intermediate class*)
e. EnrichedTrip (*final class*)



Enrichment

The act of enrichment is to lookup complementary information of a fact from another source of information. For example, in order to enrich a single sales order with the product information, you would take the product ID from the sales order and look it up in the product. The retrieved information (if found any) will be added to the sales order and make it an “enriched sales order”. This is translated to JOIN in SQL language.

There are different methods of implementing the enrichment (or join). The most obvious one is to define a lookup table. The best and very basic data structure for a lookup table is HashMap where each value is accessible using a key e.g. Product ID -> Product Details. Hence, you could read **route** into a Map first then read **trip** and for each trip lookup the **route** map with *route_id* and get the information of route and enrich the trip.

Here is the implementation of **RouteLookup**

```scala
package ca.mcit.bigdata.course2.project

import ca.mcit.bigdata.course2.project.model.Route

class RouteLookup(routes: List[Route]) {
  private val lookupTable: Map[Int, Route] =
        routes.map(route => route.route_id -> route).toMap

  def lookup(routeId: Int): Route = lookupTable.getOrElse(routeId, null)
}
```

and here is the implementation of **CalendarLookup**

```scala
package ca.mcit.bigdata.course2.project

import ca.mcit.bigdata.course2.project.model.Calendar

class CalendarLookup(calendars: List[Calendar]) {
  private val lookupTable: Map[String, Calendar] =
    calendars.map(calendar => calendar.service_id -> calendar).toMap

  def lookup(serviceId: String): Calendar = lookupTable.getOrElse(serviceId, null)
}
```