Ingeniería Industrial
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

INTRODUCTION TO BIG DATA

Juan D. Velásquez
Felipe E. Vildoso

Ingeniería Industrial
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

PASO A PASO TAREA

PASO A PASO TAREA

Este documento busca orientarlos en la realización de la tarea, de manera de ordenar cómo los contenidos que han visto pueden ser aplicados.

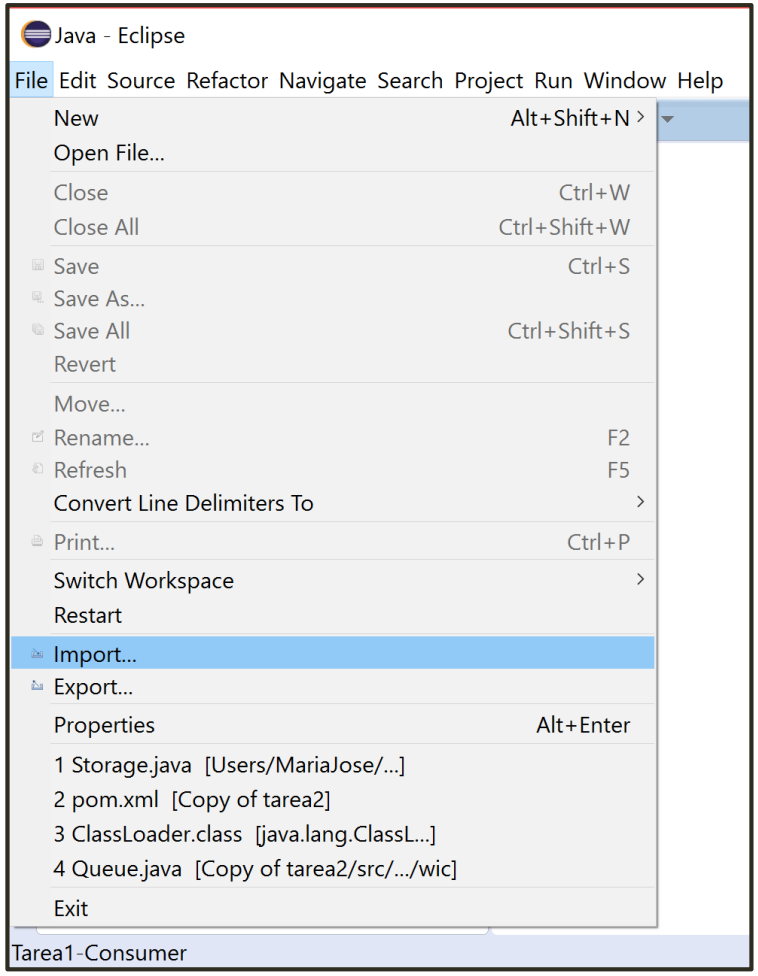
0 CORRER GET BOLETAS

- Felipe subió Tarea1-Consumer.rar, el cual contiene Get Boletas. Al correr esto verán en pantalla el JSON boleta.
- El documento Tarea1-Consumer.rar es proyecto Maven que pueden abrir con Eclipse o Netbeans (por lo que tienen que tener uno de estos instalados).

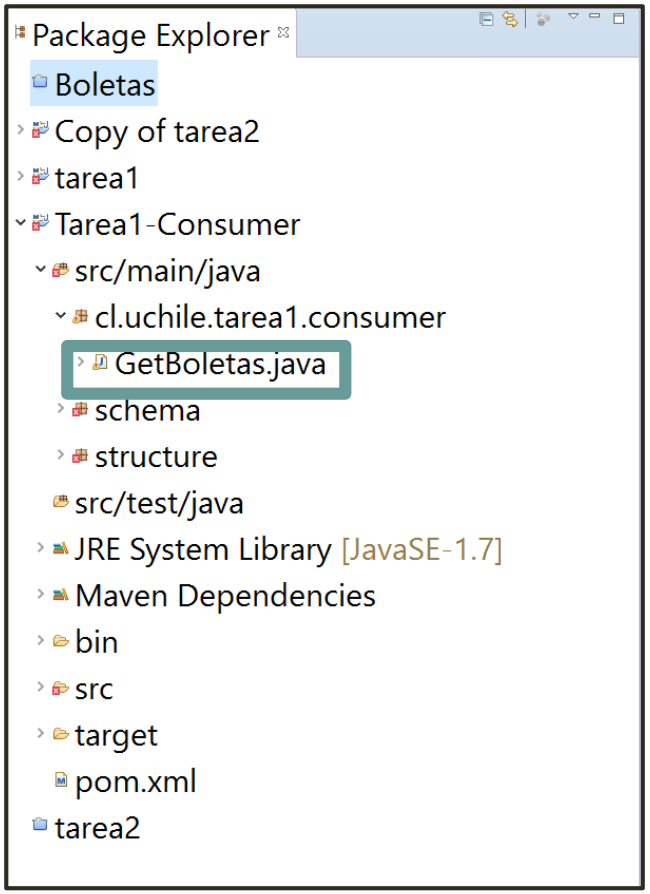
Ahora... ¿Cómo corro esto?

0.1 IMPORTAR EN ECLIPSE

- Descomprimir Tarea1-Consumer.rar.
- Abrir Eclipse
- Importar Tarea1-Consumer.rar con la opción Maven Proyect.



0.2 ¿CÓMO CORRER GET BOLETAS?



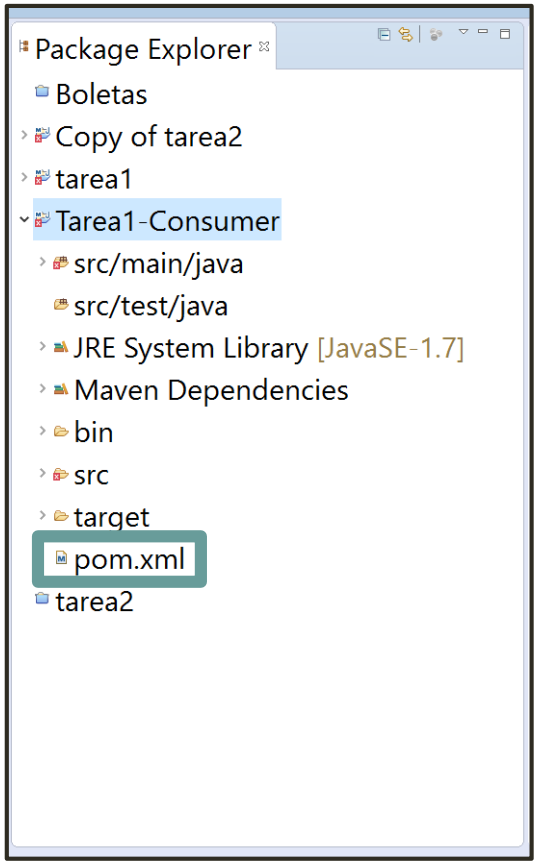
En la barra superior hay un símbolo de play que si ponen el cursor dice ***"Run GetBoletas"***

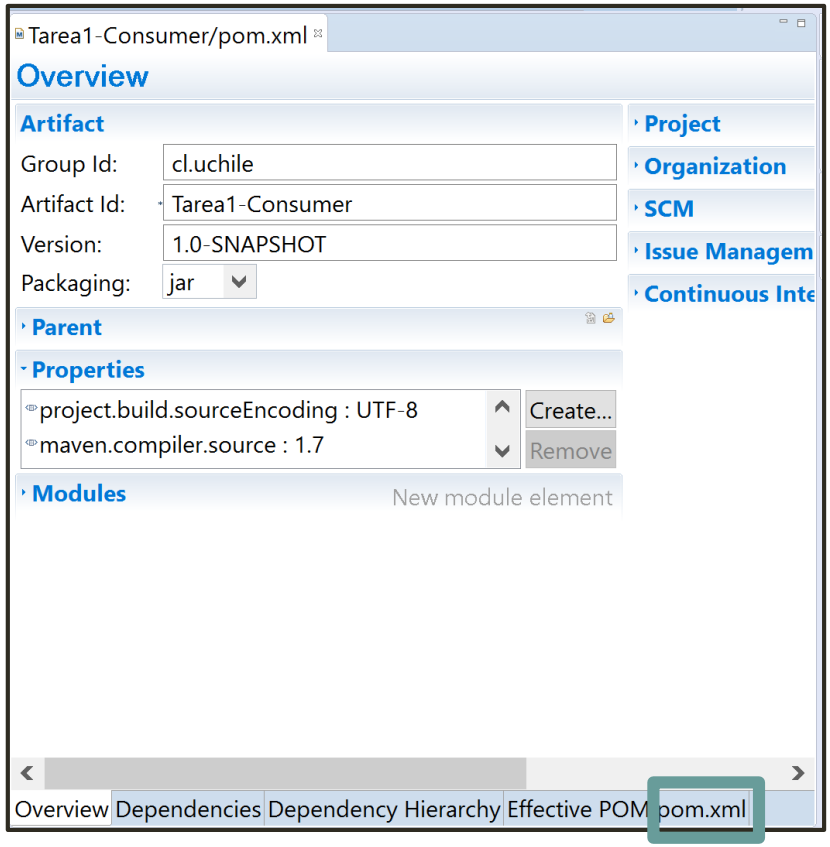
Verán el JSON en consola

UN PARÉNTESIS IMPORTANTE

POM

Como ya mencioné, Tarea1-Consumer es un proyecto Maven.
Maven es un gestor de dependecias, por lo que, basta con poner qué dependencias utilizaré en el pom para poder usar las funciones de esas librerías en el resto del proyecto.





POM

En esta parte, luego pueden agregar las dependencias que ocupen para pail y para parsear el JSON (búsquenlas en https://mvnrepository.com/)

- Verán lo siguiente (línea 45 aprox):

```
    <dependencies>
        <!-- https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients -->
        <dependency>
            <groupId>org.apache.kafka</groupId>
            <artifactId>kafka-clients</artifactId>
            <version>0.10.0.1</version>
        </dependency>
        <!-- https://mvnrepository.com/artifact/org.slf4j/slf4j-simple -->
        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-simple</artifactId>
            <version>1.7.21</version>
        </dependency>
    </dependencies>
</project>
```

- NO es necesario modificar esto en este paso, pero será útil tenerlo en cuenta para el resto de la tarea.

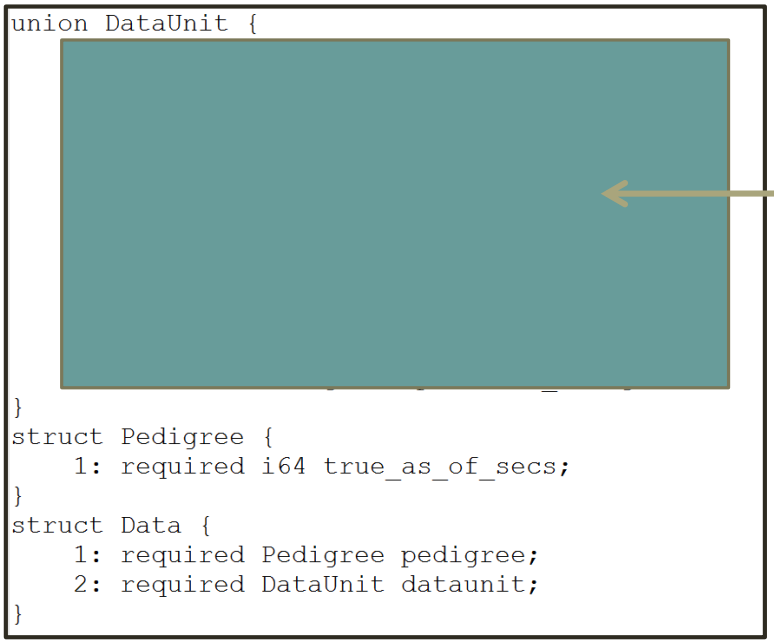
1 GRAPH SCEMA

- A partir del JSON de boletas, dibujen el graph schema que utilizarán.
- Tal como recomendó un compañero de ustedes, pueden copiar uno de estos en la página http://jsoneditoronline.org/ que les permitirá ver su estructura con mayor claridad.
- Pueden dejar de considerar ciertos “campos”, pero justifíquenlo.

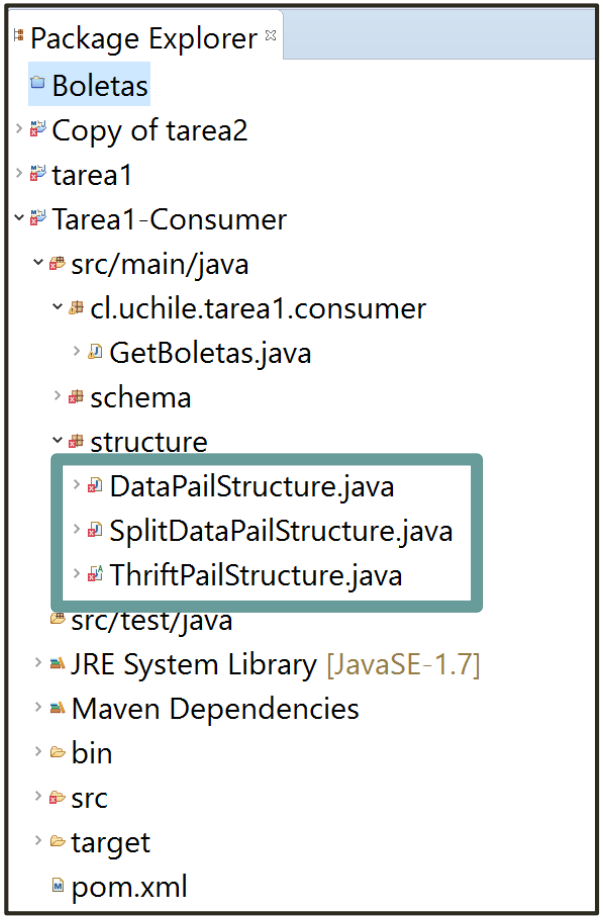
En caso de haber tenido problemas en el punto anterior, les adjunto también un ejemplo de JSON para que puedan avanzar: ejemplo.json

2 ESTRUCTURA THRIFT

- Guiandose con el graph schema que realizaron planteen la estructura thrift, tomando en consideración mantener lo siguiente:



Acá van las "opciones de datos" que definieron anteriormente como propiedades o relaciones



¿Por qué mantener "DataUnit", "Pedigree" y "Data"?

Como les dije en auxiliar, ustedes podrían usar el nombre que sea, PERO en este caso se hace referencia a ellos en los documentos de la carpeta structure de Tarea1-Consumer. Entonces, si utilizan otros nombres, les arrojará error al momento de almacenar los datos.

2 ESTRUCTURA THRIFT

- Cuando tengan su archivo schema.thrift (recuerden en la primera línea escribir namespace java schema), guárdenlo en la misma carpeta que el ejecutable de Thrift (en material docente).
- Llegar a esta carpeta mediante la línea de comando y ejecutar el siguiente código: thrift-0.9.3 --gen java schema.thrift
- Este generará una carpeta llamada gen-java.
- Muevan TODOS los archivos contenidos en esta a la carpeta **schema** de Tarea1-Consumer.

IMPORTANTE

En los pasos anteriores, se estará usando thrift en el proyecto, por lo que deberían incluirlo a dependencias (ver el paréntesis en diapos anteriores) o tendrán un error que les dirá que no reconoce algo.

3 ALMACENAMIENTO CON PAIL

- En GetBoletas verán lo siguiente:

```
57              try {
58                  while (true) {
59                      ConsumerRecords<String, String> records = kafkaConsumer.poll(100);
60                      for (ConsumerRecord<String, String> record : records) {
61
62                          //aqui capturo lo que saco de la cola, es decir record.value() tiene
63                          System.out.println(record.value());
64                      }
65                  }
```

- **"record.value()"** es el JSON con el cual queremos trabajar. Esto es lo que almacenaremos, pero para esto necesitamos parsearlo. De esto SI hay documentación: si están perdidos pregúntenle a Google como parsear un JSON y SIEMPRE recuerden incluir la dependecia que ocupen en el pom.

3 ALMACENAMIENTO CON PAIL

- Entonces, en vez de solo imprimir **"record.value()"**, lo almacenaremos.
- Para usar Pail se requiere agregar la dependencia dfs-datastores al pom.
- En el archivo Storage que está en material docente verán un ejemplo de cómo usar Pail (lo abren desde Eclipse: File > Open File…).

Veamos algunos aspectos a considerar…

Acá solo estoy guardando 6 registros, nosotros debemos hacerlo para más casos: usen operadores para iterar.

Conservamos esto porque en nuestro caso se llama igual

```
public static void saveData() throws IOException {
    Pail<Data> pail = Pail.create("/test", new SplitDataPailStructure());
    TypedRecordOutputStream out = pail.openWrite();
    out.writeObject(makeProperty(1, "https://www.youtube.com/watch?v=HL1UzIK-flA", 12312314));
    out.writeObject(makeProperty(2, "https://www.youtube.com/watch?v=gqayCgvI_uI", 34512314));
    out.writeObject(makeProperty(3, "https://www.youtube.com/watch?v=rfjtpp90lu8", 89712648));
    out.writeObject(makeEdge(1, 234, 12312314));
    out.writeObject(makeEdge(2, 78, 34512314));
    out.writeObject(makeEdge(3, 5, 89712648));
    out.close();
}
```

¿Son makeProperty y makeEdge mágicos? NO, los definimos más abajo. Veamos eso …

Tendrán que definir uno de estos por cada dato que quieran guardar de un JSON

```
public static Data makeProperty(long id, String text, long time) {
    return new Data(
        new Pedigree(time),
        DataUnit.video_property(
            new VideoProperty(
                VideoID.video_id(id),
                VideoPropertyValue.url(text)
            )
        )
    );
}
```

Noten que se basa en la estructura thrift que ustedes dieron anteriormente. Indicando qué compone a Data: pedigree y DataUnit. ¿Qué DataUnit estoy considerando?

4 CORRER EL ALMACENAMIENTO

- Todos ya tienen las máquinas virtuales y saben cómo abrir el master y esclavos desde VMware.
- Utilizaremos esto para correr nuestro GetBoletas una vez que ya agregamos la parte de guardar con Pail.
- Es importante que compartamos la carpeta Tarea1-Consumer en nuestro espacio en VMware primero **¿Cómo?** Voy al Master>options>shared folders.

4 CORRER EL ALMACENAMIENTO

4.1. Desde eclipse hacer clic derecho al proyecto y poner Run As>Maven Clean y luego Maven Install

4.2. Luego de tener corriendo hadoop /opt/hadoop/hadoop/sbin/start-all.sh en el master

4.3. hay que ir a la carpeta donde este el código cd /mnt/hgfs/carpeta_compartida

4.4. Finalmente correr hadoop jar target/Tarea1-Consumer-1.0-SNAPSHOT-shaded.jar cl.uchile.tarea1.GetBoletas (si es que modificaron GetBoletas para hacer todo)

Ingeniería Industrial
FACULTAD DE CIENCIAS
FÍSICAS Y MATEMÁTICAS
UNIVERSIDAD DE CHILE

PASO A PASO TAREA