# GeotoolKit - Feature/Datastore

par Johann Sorel

Date de publication : 24 juin 2010

Dernière mise à jour :

Cet article explique comment manipuler les features et datastore avec la librairie GeotoolKit. Un zip contenant tous les exemples se trouve à la fin.

# I - le B.A-BA

L'organisation des données en SIG est normé, il existe plusieur normes tel que ISO-19115 ou ISO-19107 qui definisse la structure des données ou leurs metadonnées. Pour les images géoréférencés le terme exacte est "Coverage", pour les données vectorielles ou attributaires le terme est "Feature". C'est se dernier dont nous parlerons dans cette article.
On distingue deux catégories de features:
- feature simple
- feature complexe

# II - FeatureType

Le feature type défini la structure des features.

## II-A - Simple FeatureType

```
final FeatureTypeBuilder ftb = new FeatureTypeBuilder();
ftb.setName("test");
ftb.add("att_String", String.class);
ftb.add("att_Integer", Integer.class);
ftb.add("att_Double", Double.class);
ftb.add("att_Date", Date.class);
ftb.add("att_geom", LineString.class, DefaultGeographicCRS.WGS84);
ftb.setDefaultGeometry("att_geom");
final SimpleFeatureType sft = ftb.buildSimpleFeatureType();
```

Penser a la methode toString(), particulierement travailler pour simplifier le debuggage.
Voici ce que donne cette methode sur le feature type qu'on vient de créer :

```
DefaultSimpleFeatureType test identified extends Feature
+-----------------------------------------------------------------------+
¦ name        ¦ min ¦ max ¦ nillable ¦ type       ¦ CRS    ¦ UserData ¦
¦-------------+-----+-----+----------+------------+--------+----------¦
¦ att_String  ¦ 1   ¦ 1   ¦ true     ¦ String     ¦        ¦          ¦
¦ att_Integer ¦ 1   ¦ 1   ¦ true     ¦ Integer    ¦        ¦          ¦
¦ att_Double  ¦ 1   ¦ 1   ¦ true     ¦ Double     ¦        ¦          ¦
¦ att_Date    ¦ 1   ¦ 1   ¦ true     ¦ Date       ¦        ¦          ¦
¦ att_geom    ¦ 1   ¦ 1   ¦ true     ¦ LineString ¦ CRS:84 ¦          ¦
+-----------------------------------------------------------------------+
crs=WGS84(DD)
```

## II-B - Complex Feature Type

plus tard, blablabla ...

# III - Feature

## III-A - Simple Feature

```
final GeometryFactory gf = new GeometryFactory();
final SimpleFeatureBuilder sfb = new SimpleFeatureBuilder(sft);
sfb.set("att_String", "toto");
sfb.set("att_Integer", 43);
sfb.set("att_Double", 12.5d);
sfb.set("att_Date", new Date());
```

```
sfb.set("att_geom", gf.createLineString(new Coordinate[]{new Coordinate(10, 12), new
 Coordinate(5, 31)}));
final SimpleFeature f = sfb.buildFeature("unID");
```

Pareil un jolie toString qui a été fait :

```
org.geotoolkit.feature.simple.DefaultSimpleFeature
featureType:test
+-------------------------------------------+
¦ @id          ¦ unID                       ¦
¦ att_String   ¦ toto                       ¦
¦ att_Integer  ¦ 43                         ¦
¦ att_Double   ¦ 12.5                       ¦
¦ att_Date     ¦ Thu Jun 24 21:07:16 CEST 2010 ¦
¦ att_geom     ¦ LINESTRING (10 12, 5 31)   ¦
+-------------------------------------------+
```

## III-B - Complex Feature

blablabla

## IV - DataStore

Les datastore servent a stocker les features.
Le model (Schema a refaire en plus jolie et explicite)

## IV-A - Session

Les session peuvent contenir une serie de changement fait sur un datastore, sans pour autant que ses changements soit appliqué sur le Datastore.
Si vous avez utilisez geotools avant, vous avez sans doute experimenté la pagaille entre les DataSource, Transaction, FeatureStore ...etc...
Bref nous avons mi fin a tout ce m...ier.
-FeatureCollection étant java.util.Collection
-FeatureIterator étant java.util.Iterator
-Plus de transaction, on ouvre une session et on la commit quand on a besoin, simple, clair et efficace.
Une session peut etre créé en mode syncrhone ou asynchrone.
En mode synchrone, toute modification faite sur les feature collection se repercute sur le datastore. En mode asynchrone les changements sont gardés dans un "Diff" en mémoire et ne seront envoyé au datastore que lors du commit sur la session.
Attention a ne pas faire trop de changements en asynchrone sous peine de OutOfMemory. Chaque changement non commité sur la session ralenti les iterations sur les feature collection (logique vu que le resultat venant du datastore doit etre alteré en passant a travers le diff).

### IV-A-1 - Query

Afin d'obtenir une FeatureCollection a partir d'une session, Il faut avoir une requete.

```
// le nom "route" est correspong au nom d'un des schema present dans le datastore.
QueryBuilder qb = new QueryBuilder("route");
//qb.setFilter(monFiltre);
//qb.setStartIndex(46);
//qb.setEtc....

FeatureCollection col = session.getFeatureCollection(qb.buildQuery());
```

Faire un filtre spatial : (a mettre dans un autre tuto Filter/Expression)

```
FilterFactory2 ff = (FilterFactory2) FactoryFinder.getFilterFactory(null);
Filter bbox = ff.bbox("geom", minx, miny, maxx, maxy, "EPSG:XXXX");
Filter propEqual = ff.equal(ff.property("nom"), ff.literal("toto"), false);
Filter dual = ff.and(bbox, propEqual);
```

Et pour l'affichage : (a mettre dans un autre tuto : Map/Renderer)

```
MapContext pMapCntx = MapBuilder.createContext(DefaultGeographicCRS.WGS84);
URL pURL = *new* URL("file://D:\\CIE.shp");
DataStore pDS = DataStoreFinder.getDataStore("url",pURL);
Name pName = pDS.getNames().iterator().next();
FeatureCollection pFC = pDS.createSession(true).getFeatureCollection(QueryBuilder.all(pName));
MapLayer pLayer = MapBuilder.createFeatureLayer(pFC, RandomStyleFactory.createDefaultVectorStyle(pFC));
pMapCntx.layers().add(pLayer);
JMap2DFrame.show(pMapCntx);
```

## IV-B - FeatureCollection

## IV-C - Obtenir un DataStore

Pour un shapefile :

```
Map<String,Serializable> parametres = new HashMap<String,Serializable>();
parameters.put("url",new URL("cheminVersFichier"));
DataStore store = DataStoreFinder.getDataStore(parameters);
```

Pour une base postgis :

```
import static org.geotoolkit.jdbc.JDBCDataStoreFactory.*;

...

Map<String,Serializable> parametres = new HashMap<String,Serializable>();
params.put(DBTYPE.getName().toString(),          "postgisng");
params.put(HOST.getName().toString(),            "localhost);
params.put(PORT.getName().toString(),            5432);
params.put(SCHEMA.getName().toString(),          "public");
params.put(DATABASE.getName().toString(),        "mabase");
params.put(USER.getName().toString(),            "toto");
params.put(PASSWD.getName().toString(),          "mangeDuChocolat");
DataStore store = DataStoreFinder.getDataStore(parameters);
```

## V - Ressources

## VI - Remerciements

Martin Desruisseaux : pour sa JavaDoc toujours parfaite.