Cassandra for Developers

Module 6

ORMs for Cassandra

Module plan

- Out-of-the-box ORM
- Hector
- Others

Out-of-the-box ORM

- `cassandra-driver-mapping.jar`
- `import com.datastax.driver.mapping.annotations.{Column,Table,PartitionKey};`
- @Table(keyspace, name)
- @Column(name)
  - for columns with different names in class and in C*
- @Transient
  - prevents next field from being mapped and stored
- @PartitionKey(0) (1) …
- @ClusteringColumn(0) (1) …
- @Enumerated(EnumType.(STRING|ORDINAL)

ORM Usage

```
Mapper<MyDAO> mapper = new
MappingManager(getSession()).mapper(MyDAO.class)

MyDAO dao = new MyDAO(…);

mapper.save(dao);

MyDAO d = mapper.get(…);

d.getXXX()

mapper.delete(d);
```

User-Defined Types in ORM

- @UDT(keyspace, name)
- @Field(name)
  - If different names in class and C*
- `UDTMapper<MyDAO2> mapper = new MappingManager(getSession()).udtMapper(MyDAO2);`
-
```
for (Row row: results) {
   Map<String, UDTValue> daomap = row.getMap(column, String.class, UDTValue.class);
  for (String key: daomap.keySet()) {
    MyDAO2 d = mapper.fromUDT(dao2.get(key));
  }
}
```

Accessor Interface

- Incapsulates custom queries for given Entity class
- @Accessor
  public Interface T1Accessor
- @Query(“SELECT * FROM T1 WHERE id=:id“)
  T1Type t1 = getRes(@Param(“id”) UUID id)
- @Query(“SELECT * FROM T1 WHERE id=?“)
  T1Type t1 = String getRes(UUID id)
- @Query(“UPDATE T1 SET V1=:v1 WHERE id=:id“)
  ResultSet updRes(@Param(“id”) UUID id, @Param(“name”) String name,
  @Param(address) Address address);

Accessor Interface (2)

- @Query(“SELECT * FROM T1 WHERE id=:id“)
  public Result<User> getAll();
- @Query(“SELECT * FROM T1 WHERE id=:id“)
  ListenableFuture<Result<User>> getAllAsync;))

Accessor-Anotated Interface

- T1Accessor acc = manager.CreateAccessor(UserAccessor.class);
- Result<User> accs = acc.getAll();

Hector

- http://hector-client.github.io
- High level object-oriented client
- Client-side failover
- Client-side connection pool

Hector

```
import me.prettyprint.hector.api.*;

Cluster cluster = HFactory.getOrCreateCluster(“cas-cluster”, ”localhost:9160”);

ColumnFamilyDefinition cfDef = HFactory.createColumnFamilyDefinition  (”MyKeyspace”,
”ColumnFamilyName”, ComparatorType.BYTESTYPE);


KeyspaceDefinition ksDef = HFactory.describeKeyspace(“MyKeyspace”);

If (ksDef == null) {

       ksDef = HFactory.createKeyspaceDefinition (”MyKeyspace”,
       ThriftKsDef.DEF_STRATEGY_CLASS, replicationFactor, Arrays.asList(cfDef));
       Cluster.addKeyspace(ksDef, true); // block until all nodes in sync

}

Keyspace ksp = HFactory.createKeyspace(“MyKeyspace”);
```

Hector Template

```
import
me.prettyprint.cassandra.service.template.ColumnFamilyTemplate;

…

ColumnFamilyTemplate<String, String> tpl = new
ThriftColumnFamilyTemplate<String, String> (ksp, columnFamily,
StringSerializer.get(), StringSerializer.get());
```

Hector: Update

```
ColumnFamilyUpdater<String, String> updater = tpl.createUpdater(“a key”); // key and column name

updater.setString(“domain”, ”www.datastax.com”);

updater.setLong(“time”, System.currentTimeMillis());

try {

      tpl.update(updater);

} catch (HectorException e) {

      // handle it

}
```

Hector: Read

```
try {
      ColumnFamilyResult<String, String> rs = tpl.queryColumns(“a key”);
      String value = rs.getString(“domain”);
} catch (HectorException e) {
      // handle it
}
```

Hector: Delete

```
try {
     tpl.deleteColumn(“key”, “column name”);
} catch (HectorException e) {
     // handle it
}
```

Hector: Column Integration

```
// Iterates over all columns for the row identified by key "a key"

SliceQuery<String, String, String> query = HFactory.createSliceQuery(ksp,
StringSerializer.get(), StringSerializer.get(), StringSerializer.get()).

      setKey("a key").setColumnFamily(columnFamily);

ColumnSliceIterator<String, String, String> iterator =

      new ColumnSliceIterator<String, String, String>(query, null,
"\uFFFF", false);

while (iterator.hasNext()) {

    // do something

}
```

Other frameworks and libraries

- High-level data access libraries
  - Hector
  - Astyanax
- ORM-like frameworks (NoSQL-generic)
  - Kundera
  - PlayORM
- Self-made wrappers/mappers/entity managers
  - https://github.com/valchkou/cassandra-driver-mapping
  - https://github.com/w3cloud/cassandra-jom
  - https://github.com/doanduyhai/Achilles

Thank you!

Questions?