Design Document - Java APIs for HCatalog DDL Commands

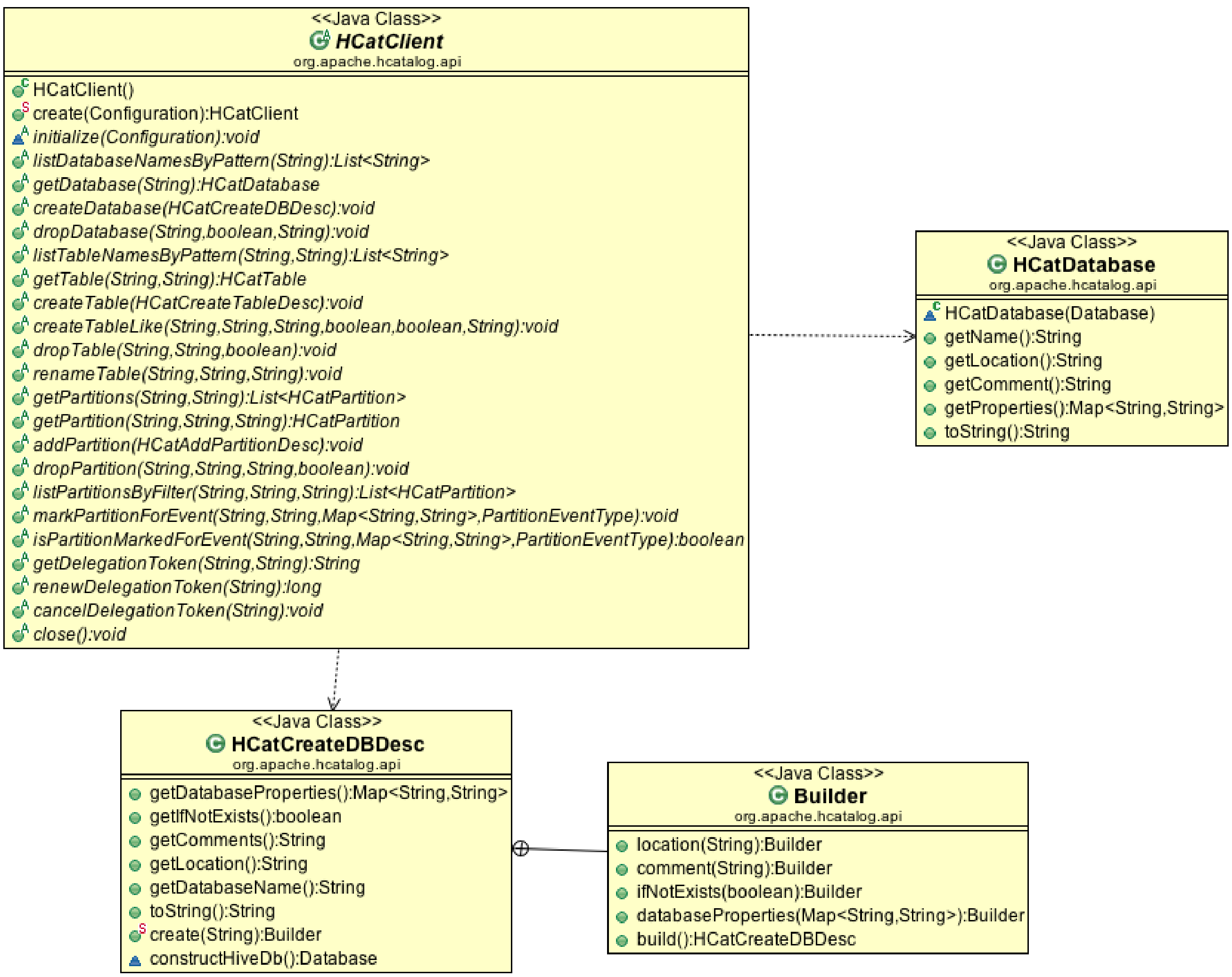
Overview

There are presently three ways to issue HCatalog DDL commands:

1. Command line interface
2. REST APIs (upcoming)
3. HiveMetaStore Client

Presently, java developers go through the Hive meta store (HMS) client interface to issue HCatalog DDl commands. Though the HMS client interface is public, it is not intended for public users. According to the hive user mailing list, the HMS client is not a public API and is subject to change in the future. So, it will be a good idea to have a java APIs in HCatalog which will provide a protect users from the changes made to the hive meta store client. Also, the under the covers either the Rest APIS or the hive metastore client can be used to provide end users with the required data.

Design



New Classes

HCatClient

The HCatClient is an abstract class containing all the APIs permitted HCatalog DDL commands. The implementation class will be provided as a configuration property, which will be used by the
"create" method. In this way, the implementation details will be masked to the users.

```
public abstract class HCatClient {
```

```
/**
 * Creates an instance of HCatClient.
 *
 * @param conf An instance of configuration.
 * @return An instance of HCatClient.
 * @throws IOException
 */
public static HCatClient create(Configuration conf) throws IOException{
  HCatClient client = HCatUtil.getHCatClient(conf);
    if(client != null){
        client.initialize(conf);
    }
    return client;
}

abstract void initialize(Configuration conf) throws HCatException;

/**
 * Get all existing databases that match the given
 * pattern. The matching occurs as per Java regular expressions
 *
 * @param databasePattern
 *          java re pattern
 * @return list of database names
 * @throws HCatException
 */
public abstract List<String> listDatabaseNamesByPattern(String pattern) throws HCatException;

/**
 * Gets the database.
 *
 * @param dbName The name of the database.
 * @return An instance of HCatDatabaseInfo.
 * @throws HCatException
 */
public abstract HCatDatabase getDatabase(String dbName) throws HCatException;

/**
 * Creates the database.
 *
 * @param dbInfo An instance of HCatCreateDBDesc.
 * @throws HCatException
 */
public abstract void createDatabase(HCatCreateDBDesc dbInfo)
        throws HCatException;

/**
 * Drops a database.
 *
 * @param dbName The name of the database to delete.
 * @param ifExists Hive returns an error if the database specified does not exist,
 *                 unless ifExists is set to true.
 * @param mode This is set to either "restrict" or "cascade". Restrict will
 *             remove the schema if all the tables are empty. Cascade removes
 *             everything including data and definitions.
 * @throws HCatException
 */
public abstract void dropDatabase(String dbName, boolean ifExists, String mode) throws HCatException;

/**
 * Returns all existing tables from the specified database which match the given
 * pattern. The matching occurs as per Java regular expressions.
 * @param dbName
 * @param tablePattern
 * @return list of table names
 * @throws HCatException
 */
public abstract List<String> listTableNamesByPattern(String dbName, String tablePattern)
        throws HCatException;

/**
```

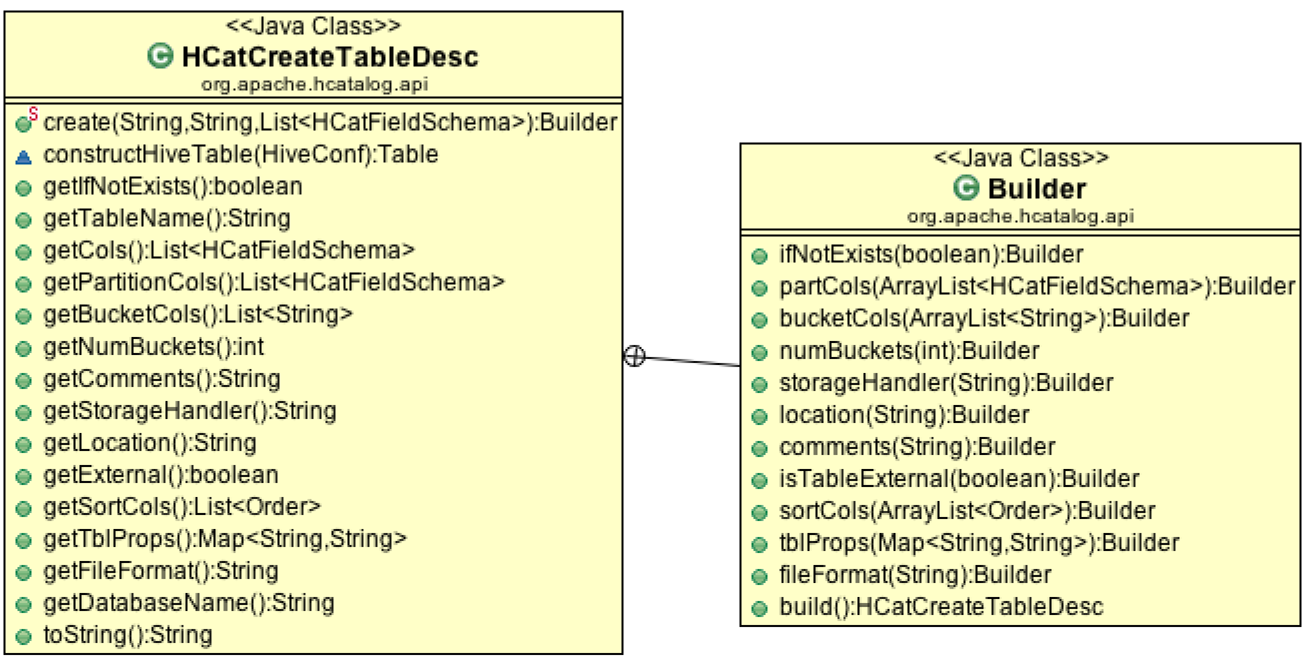
```
     * Gets the table.
     *
     * @param dbName The name of the database.
     * @param tableName The name of the table.
     * @return An instance of HCatTableInfo.
     * @throws HCatException
     */
    public abstract HCatTable getTable(String dbName, String tableName)
            throws HCatException;

    /**
     * Creates the table.
     *
     * @param createTableDesc An instance of HCatCreateTableDesc class.
     * @throws HCatException the h cat exception
     */
    public abstract void createTable(HCatCreateTableDesc createTableDesc)
            throws HCatException;

    /**
     * Creates the table like an existing table.
     *
     * @param dbName The name of the database.
     * @param existingTblName The name of the existing table.
     * @param newTableName The name of the new table.
     * @param ifNotExists If true, then error related to already table existing is skipped.
     * @param isExternal Set to "true", if table has be created at a different
     *                   location other than default.
     * @param location The location for the table.
     * @throws HCatException
     */
    public abstract void createTableLike(String dbName, String existingTblName,
            String newTableName, boolean ifNotExists, boolean isExternal,
            String location) throws HCatException;

    /**
     * Drop table.
     *
     * @param dbName The name of the database.
     * @param tableName The name of the table.
     * @param ifExists Hive returns an error if the database specified does not exist,
     *                 unless ifExists is set to true.
     * @throws HCatException
     */
    public abstract void dropTable(String dbName, String tableName,
            boolean ifExists) throws HCatException;

    /**
     * Renames a table.
     *
     * @param dbName The name of the database.
     * @param oldName The name of the table to be renamed.
     * @param newName The new name of the table.
     * @throws HCatException
     */
    public abstract void renameTable(String dbName, String oldName, String newName) throws HCatException;

    /**
     * Gets all the partitions.
     *
     * @param dbName The name of the database.
     * @param tblName The name of the table.
     * @return A list of partition names.
     * @throws HCatException the h cat exception
     */
    public abstract List<HCatPartition> getPartitions(String dbName, String tblName)
            throws HCatException;

    /**
     * Gets the partition.
     *
```

```
     * @param dbName The database name.
     * @param tableName The table name.
     * @param partitionName The partition name, Comma separated list of col_name='value'.
     * @return An instance of HCatPartitionInfo.
     * @throws HCatException
     */
    public abstract HCatPartition getPartition(String dbName, String tableName,
            String partitionName) throws HCatException;

    /**
     * Adds the partition.
     *
     * @param partInfo An instance of HCatAddPartitionDesc.
     * @throws HCatException the h cat exception
     */
    public abstract void addPartition(HCatAddPartitionDesc partInfo) throws HCatException;

    /**
     * Drops partition.
     *
     * @param dbName The database name.
     * @param tableName The table name.
     * @param partitionName The partition name, Comma separated list of col_name='value'.
     * @param ifExists Hive returns an error if the partition specified does not exist, unless ifExists is set
to true.
     * @throws HCatException
     */
    public abstract void dropPartition(String dbName, String tableName,
            String partitionName, boolean ifExists) throws HCatException;

    /**
     * List partitions by filter.
     *
     * @param dbName The database name.
     * @param tblName The table name.
     * @param filter The filter string,
     *    for example "part1 = \"p1_abc\" and part2 <= "\p2_test\"". Filtering can
     *    be done only on string partition keys.
     * @return list of partitions
     * @throws HCatException the h cat exception
     */
    public abstract List<HCatPartition> listPartitionsByFilter(String dbName, String tblName,
            String filter) throws HCatException;

    /**
     * Mark partition for event.
     *
     * @param dbName The database name.
     * @param tblName The table name.
     * @param partKVs the part k vs
     * @param eventType the event type
     * @throws HCatException the h cat exception
     */
    public abstract void markPartitionForEvent(String dbName, String tblName,
            Map<String, String> partKVs, PartitionEventType eventType)
            throws HCatException;

    /**
     * Checks if is partition marked for event.
     *
     * @param dbName the db name
     * @param tblName the tbl name
     * @param partKVs the part k vs
     * @param eventType the event type
     * @return true, if is partition marked for event
     * @throws HCatException the h cat exception
     */
    public abstract boolean isPartitionMarkedForEvent(String dbName, String tblName,
            Map<String, String> partKVs, PartitionEventType eventType)
            throws HCatException;
```

```
/**
 * Gets the delegation token.
 *
 * @param owner the owner
 * @param renewerKerberosPrincipalName the renewer kerberos principal name
 * @return the delegation token
 * @throws HCatException the h cat exception
 */
public abstract String getDelegationToken(String owner, String renewerKerberosPrincipalName) throws
    HCatException;

/**
 * Renew delegation token.
 *
 * @param tokenStrForm the token str form
 * @return the long
 * @throws HCatException the h cat exception
 */
public abstract long renewDelegationToken(String tokenStrForm) throws HCatException;

/**
 * Cancel delegation token.
 *
 * @param tokenStrForm the token str form
 * @throws HCatException the h cat exception
 */
public abstract void cancelDelegationToken(String tokenStrForm) throws HCatException;

/**
 * Close the hcatalog client.
 *
 * @throws HCatException the h cat exception
 */
public abstract void close() throws HCatException;
```

HCatCreateTableDesc

This class is a sub class of HCatCommandDesc and will be used by the users to create descriptor and validate it for the "create table" command.



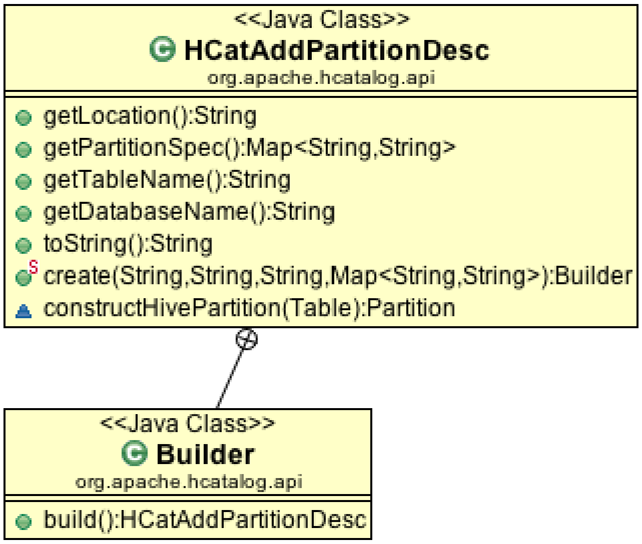
HCatCreateDBDesc

This class is a sub class of HCatCommandDesc and will be used by the users to create descriptors and validate it for the "create database" command.

!createdb.png|

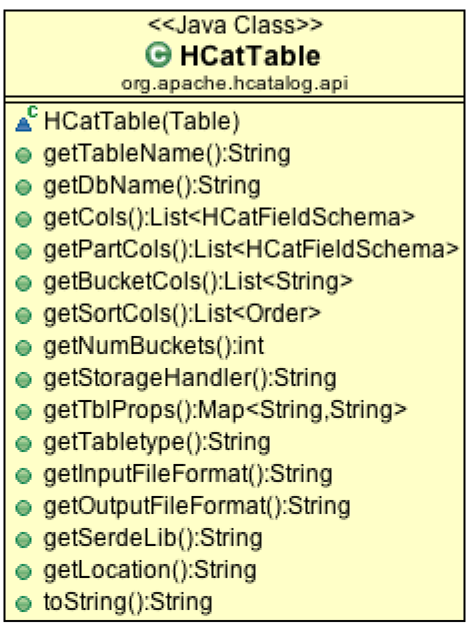
HCatAddPartitionDesc

This class is a sub class of HCatCommandDesc and will be used by the users to create descriptors and validate it for the "add partition" command.



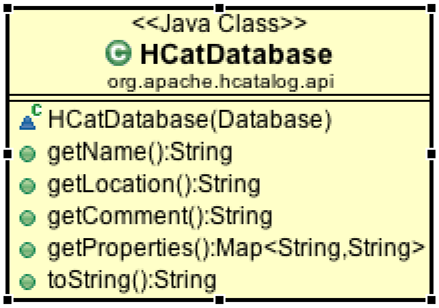
HCatTable

This class encapsulates the table information returned the HCatClient implementation class and provides a uniform view to the user.



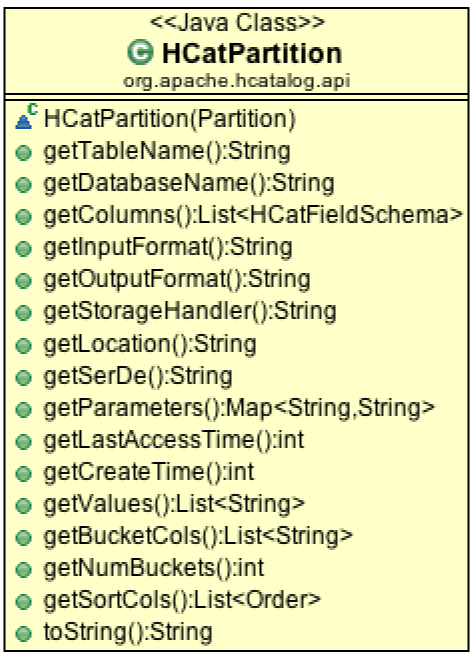
HCatDatabase

This class encapsulates the database information returned the HCatClient implementation class and provides a uniform view to the user.



HCatPartition

This class encapsulates the partition information returned the HCatClient implementation class and provides a uniform view to the user.



Usage

```
 Configuration config = new Configuration();
 config.add("hive-site.xml");
HCatClient client = HCatClient.create(config);
ArrayList<HCatFieldSchema> cols = new ArrayList<HCatFieldSchema>();
cols.add(new HCatFieldSchema("id", Type.INT, "id columns"));
cols.add(new HCatFieldSchema("value", Type.STRING, "id columns"));
HCatCreateTableDesc tableDesc = HCatCreateTableDesc.create(db, "testtable", cols).fileFormat("rcfile").build();
client.createTable(tableDesc);
```

Discussion Topics