Fusion Table Tutorial

ECEN 489 | Spring 2014 | Aguiar, Binder, Partal

Introduction

Fusion tables is a web application to gather visualize and share large data tables. It allows to filter and summarize several rows, and then create maps, charts, network graphs or custom layout. Can also be shared, allowing multiple people to work in the same table or set of tables and at the same time it can also select the part of data to be shared and what should be private.

Fusion Tables Online

A google account is needed in order to interact with fusion tables online. The fusion table must then be created in the drive as a google doc. If the fusion table was created in the app, this is where it can be found. If the data points include geographical coordinates, it is helpful to view them on a map. To do this, click on the visualize tab and select map. From here, the data points can visually be manipulated as desired. The different types of data points can be changed to different colors and the map can be manipulated exactly like looking for locations on google maps.

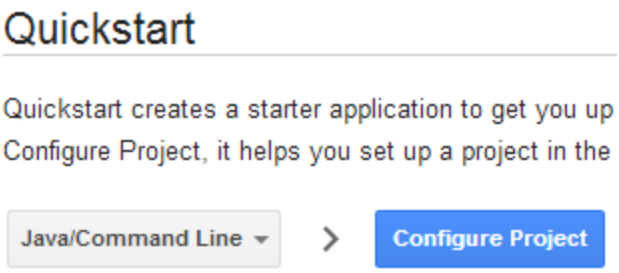
Fusion Tables in Java

Google has released client libraries for Fusion Tables in several languages. This tutorial focuses on using the Java client library and sample program which can be found at the following url:

https://developers.google.com/api-client-library/java/apis/fusiontables/v1

Step 1:

Use the quick start to setup a new project and generate the sample application and clients.json file which can be downloaded by following the directions on the page.



Step 2:

With the client library, sample application, and clients.json downloaded and loaded in Eclipse it is now possible to start interfacing with the fusion table api. Attached to this tutorial is example code for the following examples, create a table, insert data, show rows, list tables, and delete table.

Most likely there will be a need to insert large amounts of data into the Fusion Table quickly. Using the insert query from the attached example code will work well for a single insert SQL statement but API rate limiting put in place by Google will quickly stop attempts to execute queries in rapid succession. The insert API allows for sending up to 500 insert queries at once as long as they are sent in a single request. To do this it is necessary to build a request in such a way that the SQL is all in the body of the request. The following code shows how this is done:

```
String sql = <MULTIPLE SQL HERE>;
HttpContent content = ByteArrayContent.fromString(null, "sql=" + sql);
HttpRequest httpRequest = client.getRequestFactory().buildPostRequest(new
       GenericUrl("https://www.googleapis.com/fusiontables/v1/query"), content);
httpRequest.execute();
```

The multiple queries must be all in a single string where each query is separated from the other by a semi colon. An example of this would be as such:

```
String sql = “INSERT INTO table (col1, col2) VALUES (“val1”, “val2”); INSERT INTO table (col1, col2)
VALUES (“val3”, “val4”); INSERT INTO table (col1, col2) VALUES (“val5”, “val6”);”
```

Further information can be found from Google’s documentation on queries: https://developers.google.com/fusiontables/docs/v1/sql-reference.

Appendix A

FusionTableSample.java

```java
/*
* Copyright (c) 2012 Google Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

package com.google.api.services.samples.fusiontables.cmdline;

import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp;
import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver;
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport;
import com.google.api.client.http.HttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.client.util.DateTime;
import com.google.api.client.util.store.DataStoreFactory;
import com.google.api.client.util.store.FileDataStoreFactory;
import com.google.api.services.fusiontables.Fusiontables;
import com.google.api.services.fusiontables.Fusiontables.Query.Sql;
import com.google.api.services.fusiontables.Fusiontables.Table.Delete;
import com.google.api.services.fusiontables.FusiontablesScopes;
import com.google.api.services.fusiontables.model.Column;
import com.google.api.services.fusiontables.model.Table;
import com.google.api.services.fusiontables.model.TableList;

import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Arrays;
import java.util.Collections;
import java.util.Date;
import java.util.UUID;


public class FusionTablesSample {

  /**
  * Be sure to specify the name of your application. If the application name is {@code null} or
  * blank, the application will log a warning. Suggested format is "MyCompany-ProductName/1.0".
  */
  private static final String APPLICATION_NAME = "ECEN489-FusionTableDemo/1.0";

  /** Directory to store user credentials. */
  private static final java.io.File DATA_STORE_DIR =
      new java.io.File(System.getProperty("user.home"), ".store/fusion_tables_sample");

  /**
  * Global instance of the {@link DataStoreFactory}. The best practice is to make it a single
  * globally shared instance across your application.
  */
  private static FileDataStoreFactory dataStoreFactory;
```

```
  /** Global instance of the HTTP transport. */
  private static HttpTransport httpTransport;

  /** Global instance of the JSON factory. */
  private static final JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();

  private static Fusiontables fusiontables;

  //---------------------------------------------------------------------------------

  /** Authorizes the installed application to access user's protected data. */
  private static Credential authorize() throws Exception {
    // load client secrets
    GoogleClientSecrets clientSecrets = GoogleClientSecrets.load(
        JSON_FACTORY, new InputStreamReader(
            FusionTablesSample.class.getResourceAsStream("/client_secrets.json")));
    if (clientSecrets.getDetails().getClientId().startsWith("Enter")
        || clientSecrets.getDetails().getClientSecret().startsWith("Enter ")) {
      System.out.println(
          "Enter Client ID and Secret from https://code.google.com/apis/console/?api=fusiontables "
          + "into fusiontables-cmdline-sample/src/main/resources/client_secrets.json");
      System.exit(1);
    }
    // set up authorization code flow
    GoogleAuthorizationCodeFlow flow = new GoogleAuthorizationCodeFlow.Builder(
        httpTransport, JSON_FACTORY, clientSecrets,
        Collections.singleton(FusiontablesScopes.FUSIONTABLES)).setDataStoreFactory(
        dataStoreFactory).build();
    // authorize
    return new AuthorizationCodeInstalledApp(flow, new LocalServerReceiver()).authorize("user");
  }

  //---------------------------------------------------------------------------------
  //Main function
  public static void main(String[] args) {
    try {

      //facilities to interface with a website
      httpTransport = GoogleNetHttpTransport.newTrustedTransport();
      dataStoreFactory = new FileDataStoreFactory(DATA_STORE_DIR);

      // authorization
      Credential credential = authorize();

      // set up global FusionTables instance
      fusiontables = new Fusiontables.Builder(
          httpTransport, JSON_FACTORY, credential).setApplicationName(APPLICATION_NAME).build();

      // run commands
      listTables();

      String tableId = createTable();     //create a table
      insertData(tableId);                //insert data into table
      showRows(tableId);                  //not quite working yet
//      deleteTable(tableId);
      // success!
      return;
    } catch (IOException e) {
      System.err.println(e.getMessage());
    } catch (Throwable t) {
      t.printStackTrace();
    }
    System.exit(1);
  }
```

```
/**
* @param tableId
* @throws IOException
*/
private static void showRows(String tableId) throws IOException {
  View.header("Showing Rows From Table");

  Sql sql = fusiontables.query().sql("SELECT Text,Number,Location,Date FROM " + tableId);

  try {
    sql.execute();
  } catch (IllegalArgumentException e) {
    // For google-api-services-fusiontables-v1-rev1-1.7.2-beta this exception will always
    // been thrown.
    // Please see issue 545: JSON response could not be deserialized to Sqlresponse.class
    // http://code.google.com/p/google-api-java-client/issues/detail?id=545
  }
}

/** List tables for the authenticated user. */
private static void listTables() throws IOException {
  View.header("Listing My Tables");

  // Fetch the table list
  Fusiontables.Table.List listTables = fusiontables.table().list();
  TableList tablelist = listTables.execute();

  if (tablelist.getItems() == null || tablelist.getItems().isEmpty()) {
    System.out.println("No tables found!");
    return;
  }

  for (Table table : tablelist.getItems()) {
    View.show(table);
    View.separator();
  }
}

/** Create a table for the authenticated user. */
private static String createTable() throws IOException {
  View.header("Create Sample Table");

  // Create a new table
  Table table = new Table();
  table.setName(UUID.randomUUID().toString());
  table.setIsExportable(false);
  table.setDescription("Sample Table");

  // Set columns for new table
  table.setColumns(Arrays.asList(new Column().setName("Text").setType("STRING"),
      new Column().setName("Number").setType("NUMBER"),
      new Column().setName("Location").setType("LOCATION"),
      new Column().setName("Date").setType("DATETIME")));

  // Adds a new column to the table.
  Fusiontables.Table.Insert t = fusiontables.table().insert(table);
  Table r = t.execute();

  View.show(r);

  return r.getTableId();
}

/** Inserts a row in the newly created table for the authenticated user. */
private static void insertData(String tableId) throws IOException {
  Sql sql = fusiontables.query().sql("INSERT INTO " + tableId + " (Text,Number,Location,Date) "
```

```
        + "VALUES (" + "'Google Inc', " + "1, " + "'1600 Amphitheatre Parkway Mountain View, "
        + "CA 94043, USA','" + new DateTime(new Date()) + "')");

    try {
      sql.execute();
    } catch (IllegalArgumentException e) {
      // For google-api-services-fusiontables-v1-rev1-1.7.2-beta this exception will always
      // been thrown.
      // Please see issue 545: JSON response could not be deserialized to Sqlresponse.class
      // http://code.google.com/p/google-api-java-client/issues/detail?id=545
    }
  }

  /** Deletes a table for the authenticated user. */
  private static void deleteTable(String tableId) throws IOException {
    View.header("Delete Sample Table");
    // Deletes a table
    Delete delete = fusiontables.table().delete(tableId);
    delete.execute();
  }

}
```

View.java

```
/*
* Copyright (c) 2012 Google Inc.
*
* Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
* in compliance with the License. You may obtain a copy of the License at
*
* http://www.apache.org/licenses/LICENSE-2.0
*
* Unless required by applicable law or agreed to in writing, software distributed under the License
* is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express
* or implied. See the License for the specific language governing permissions and limitations under
* the License.
*/

package com.google.api.services.samples.fusiontables.cmdline;

import com.google.api.services.fusiontables.model.Table;

/**
* Utility methods to print to the command line.
*/

public class View {

  static void header(String name) {
    System.out.println();
    System.out.println("================== " + name + " ==================");
    System.out.println();
  }

  static void show(Table table) {
    System.out.println("id: " + table.getTableId());
    System.out.println("name: " + table.getName());
    System.out.println("description: " + table.getDescription());
    System.out.println("attribution: " + table.getAttribution());
    System.out.println("attribution link: " + table.getAttributionLink());
    System.out.println("kind: " + table.getKind());
```

```
  }

  static void separator() {
    System.out.println();
    System.out.println("-------------------------------------------------------");
    System.out.println();
  }
}
```