YQL Open Data Tables

Developing Open Data Tables for YQL


```
<?xml version="1.0" encoding="UTF-8"?>
<table xmlns="http://query.yahooapis.com/v1/schema/table.xsd">
  <meta>
    <sampleQuery></sampleQuery>
  </meta>
  <bindings>
    <select itemPath="" produces="XML">
      <urls>
        <url>http://foo.com/{id}/</url>
      </urls>
      <inputs>
          <key id='id' type='xs:string' paramType='query' required="false" />
      </inputs>
      <execute><![CDATA[
       ... ]]>
      </execute>
    </select>
    <insert itemPath="" produces="XML">
       ...
    </insert>
  </bindings>
</table>
```

SELECT * FROM Internet

About YQL

Open Data Tables in YQL allow you to create and use your own table definitions, enabling YQL to bind to any data source through the SQL-like syntax and fetch data. Your addition contributes to an extensive list of tables already available to YQL. For more information on community Open Data Tables, visit:

http://datatables.org

Open Data Table Elements

Common Open Data Tables include:

meta — The meta element contains descriptive information about the Open Data Table, such as a sample query, author, or documentation link.

select / insert / update / delete — The `select`, `insert`, `update`, and `delete` elements describe the information needed for YQL to read data, add, update, and delete data using an API, respectively.

`INSERT`, `UPDATE`, and `DELETE` elements require the proper binding inputs, such as key, value, or map. Additions, updates, and deletions of data are performed within the `execute` element of an Open Data Table.

itemPath — A dot-path that points to where the repeating data elements occur in the response format. These are the "rows" of your table.

produces — The type of data coming back from the Web service.

url — The url element describes the URL that needs to be executed to get data for this table, given the keys in the key elements.

Open Data Table Elements (cont.)

key — The key element represents a named "key" that you provide in the WHERE or INTO clause of SELECT, INSERT, UPDATE, or DELETE statements in order to restrict the request.

value — Use the value element to assign a new "value" or update an existing one within an Open Data Table. The value element defines a field that can only be set as an input and therefore cannot be in YQL statements to satisfy the "where" clause. The value element only works with INSERT and UPDATE verbs.

id — Name of the key.

type — The type of data coming back from the Web service.

paramType — Determines how this key is represented and passed on to the Web service.

query: Add the id and its value as a id=value query string parameter to the URL.

matrix: Add the id and its value as a id=value matrix parameter to the URL path.

header: Add the id and its value as a id: value header to the URL request.

path: Substitute all occurrences of {id} in the url string with the value of the id.

variable: Use this key or field as a variable to be used within the execute sub-element instead of being used to format or form the URL.

required — Whether the key is required or not.

execute — Optional element containing server-side Javascript that YQL runs instead of fetching the <url>

Getting Started

1 Read the guide at **developer.yahoo.com/yql/guide**

2 Try the console at **developer.yahoo.com/yql/console**

YQL Execute: Executing JavaScript in Open Data Tables

The ability to execute JavaScript extends the functionality of Open Data Tables in many ways, including the following:

- **Access APIs that require authentication:** Netflix OAuth, FlickrAuth, Google AuthSub
- **Join data across services:** grab New York Times article tags and find associated flickr photos
- **Combine multiple searches into a single result:** twitter, Web, news, and images
- **Augment data:** convert ZIP codes to city/state via API
- **Create APIs from Web pages:** scrape celebrity birthdays from IMDB.com
- **Data transformation:** convert the result from xml to Google's kml format
- **Move business logic of your application to the cloud**

The ability to `execute` JavaScript is implemented through the `execute` sub-element within an Open Data Table definition.

YQL provides several objects you can use in your execute JavaScript code. Key objects include:

- **y.rest**: Make GET requests to remote Web services to pass parameters and headers in your request.

```
var myRequest = y.rest('http://example.com');
var data = myRequest.get().response;
```

- **y.query**: perform additional YQL queries within the execute element

```
var q = y.query('select * from html where url="http://
finance.yahoo.com/q?s=yhoo" and xpath="//div[@id=\'yfi_
headlines\']/div[2]/ul/li/a"');
var results = q.results;
```

Testing YQL Open Data Tables

YQL provides tools for testing and debugging your Open Data Tables.

NETWORK-LEVEL LOGGING

To enable network-level logging, you simply append debug=true to the YQL console URL or API query:

http://developer.yahoo.com/yql/console/?debug=true

When enabled, all network requests are uncached, so you can iteratively develop Open Data Tables more easily. Network logs display both the request headers and the response content for each network call between YQL and the remote service.

YQL responses provide an `id` key within the `diagnostics` element for each network call that occurs:

```
<diagnostics>
  <publiclyCallable>true</publiclyCallable>
  <url execution-time="11" id="5b81e4c4-11eb-43a5-866b-
   b1217498843e" proxy="DEFAULT"><![CDATA[http://
   datatables.org/alltables.env]]></url>
  <url execution-time="6" proxy="DEFAULT"><![CDATA[
   http://www.datatables.org/zillow/zillow.search.
   xml]]></url>
  <url execution-time="139" id="9dd99f2c-54e3-493f-
   a818-9950f0798d2a" proxy="DEFAULT"><![CDATA[http://
   www.zillow.com/webservice/GetSearchResults.htm?zws-
   id=X1-ZWz1cse68iatcb_13bwv&address=1835%2073rd%20
   Ave%20NE&citystatezip=98039]]></url>
  <user-time>183</user-time>
  <service-time>156</service-time>
  <build-version>2355</build-version>
</diagnostics>
```

You can use the id key to access network-level logs within 5 minutes of running a YQL call. Append the id key provided in diagnostics to the logging endpoint:

http://query.yahooapis.com/v1/logging/dump?id=5b81e4c4-11eb-43a5-866b-b1217498843e

JAVASCRIPT LOGGING AND DEBUGGING

To help debug server-side JavaScript issues such as syntax errors and uncaught exceptions, use `y.log` along with `y.getDiagnostics`.

The following example logs "hello" along with a variable:

```
y.log("hello");

y.log(somevariable);
```

The output of `y.log` goes into the YQL diagnostics element when the table is used in a statement.

You can also use the following JavaScript to get the diagnostics that have been created so far:

```
var e4xObject = y.getDiagnostics();
```

Resources

YAHOO! QUERY LANGUAGE
developer.yahoo.com/yql

YQL GUIDE
developer.yahoo.com/yql/guide/yql_guide.pdf

YQL BLOG
yqlblog.net

YQL CONSOLE
developer.yahoo.com/yql/console

YQL COMMUNITY OPEN TABLES
datatables.org

YQL EXECUTE SCREENCAST
bit.ly/yql-execute-screencast