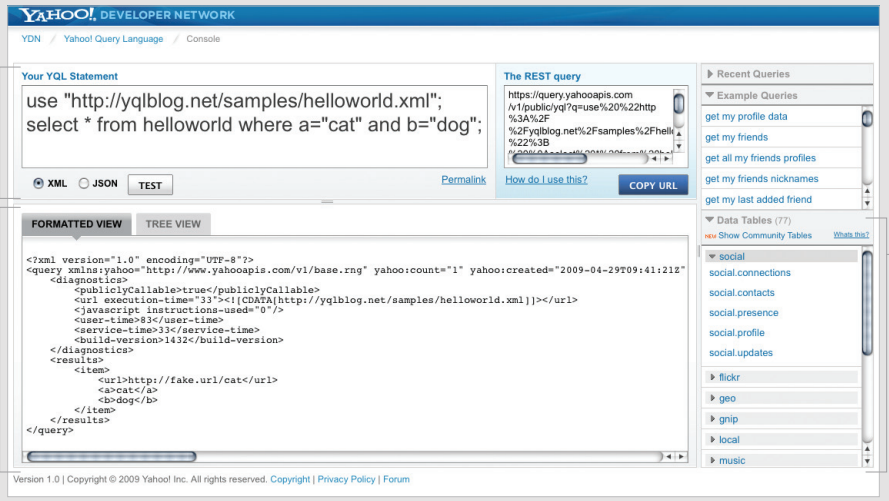
YQL
Yahoo! Query Language

developer.yahoo.com/yql

YQL Console

developer.yahoo.com/yql/console



SELECT * FROM Internet

About YQL

The Yahoo! Query Language (YQL) platform enables developers to query, filter, and combine data across the Web. YQL exposes a SQL-like syntax that is both familiar to developers and expressive enough for getting the right data. YQL supports four SQL-like verbs:

SELECT Fetch, combine, filter and project data.

DESC Describe the input fields for a table and other meta information.

SHOW Get a list of the tables/data sources.

USE Load an open data table.

The SELECT statement is the primary verb for YQL:

```
SELECT what FROM table WHERE filter
```

YQL also supports post-query functions like `sort` and `unique`.

Data Tables

A table in YQL is an external data source often containing very large collections of structured data. You can think of each item in a table as a "row" in a more traditional relational database. All data in YQL is treated as XML, and if the underlying table source does not provide XML, it converts into an XML-like representation.

YQL contains an extensive list of built-in tables for you to use that cover a wide range of Yahoo! Web services and access to off-network data. Create and define your own table definitions using YQL Open Data Tables.

Open Data Tables

Create and use your own table definitions with YQL Open Data Tables to bind to any data source through the SQL-like syntax and fetch data. Once you create a table, anyone can use the definitions in YQL.

Access any open data table with the YQL USE statement:

```
USE "http://myserver.com/mytables.xml" AS mytable;
SELECT * FROM mytable WHERE ...
```

Getting Started

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

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

YQL Web Service

YQL provides a Web service you can use to execute YQL statements from your application. There are two variations, one to access both private and public data using OAuth authorization:

http://query.yahooapis.com/v1/yql?q=[query]

Another simply accesses public data:

http://query.yahooapis.com/v1/public/yql?q=[query]

YQL returns XML or JSON/JSONP for all queries.

Dot Syntax

The SELECT statement's "what" and local filtering clauses use a "dot" style syntax to address parts in the table item data structure. Each dot-separated part in the field path refers to the name of the element (for JSON and XML documents), an attribute, or CDATA/text portion of the element (for XML documents). The more "dots" in a path, the deeper into the data structure the final element is found.

EXAMPLE QUERY

SELECT Title,Rating.AverageRating **FROM** local.search **WHERE** zip='94085' and query='pizza' **LIMIT** 1

RESULT

```
<?xml version="1.0" encoding="UTF-8"?>
<query xmlns:yahoo="http://www.yahooapis.com/v1/base.rng"
yahoo:count="3" yahoo:created="2009-04-27T04:42:57Z"
yahoo:lang="en-US" yahoo:updated="2009-04-27T04:42:57Z"
yahoo:uri="http://query.yahooapis.com/v1/yql?q=select+Title%2C
Rating.AverageRating+from+local.search+where+zip%3D%2794089
%27+and+query%3D%27pizza%27+limit+3">
    <diagnostics>
        <publiclyCallable>true</publiclyCallable>
        <url execution-time="338">
            <![CDATA[http://local.yahooapis.com/
            LocalSearchService/V3/localSearch?
            zip=94089&query=pizza&start=1&results=10]]>
        </url>
        <user-time>342</user-time>
        <service-time>338</service-time>
        <build-version>1390</build-version>
    </diagnostics>
    <results>
        <Result xmlns="urn:yahoo:lcl">
            <Rating>
                <AverageRating>4.5</AverageRating>
            </Rating>
            <Title>Giovannis Pizzeria</Title>
        </Result>
   </results>
</query>
```

Execute

Get full control over how YQL fetches data and presents it back to the user. In an open data table definition, developers add code to an `execute` sub-element. The YQL data engine runs this execute code when the YQL statement is processed. This enables:

Flexible Templating: Use JavaScript to add conditional logic and granularly format data.

Better data shaping and parsing: Use JavaScript to format and shape requests and responses.

Better support for calling external Web services: Use a variety of security and authentication mechanisms.

The following Open Data Table example searches for a fictional table in which "a" is the path and "b" is the term.

```
<?xml version="1.0" encoding="UTF-8"?>
<table xmlns="http://query.yahooapis.com/v1/schema/table.xsd">
  <meta>
    <sampleQuery>
      SELECT * FROM {table} where a='cat' and b='dog';
    </sampleQuery>
  </meta>
  <bindings>
    <select itemPath="" produces="XML">
      <urls>
        <url>http://fake.url/{a}</url>
      </urls>
      <inputs>
        <key id='a' type='xs:string' paramType='path'
        required="true" />
        <key id='b' type='xs:string' paramType='variable'
        required="true" />
      </inputs>
      <execute>
          <![CDATA[
              // Your javascript goes here.
              // We will run it on our servers.
              response.object = <item>
                  <url>{request.url}</url>
                  <a>{a}</a>
                  <b>{b}</b>
              </item>;
          ]]>
      </execute>
    </select>
  </bindings>
</table>
```

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