TAP-Autocomplete: A Javascript Library for providing metadata and ADQL keyword suggestions

prepared by: S. Voutsinas, D. Morris and N.C. Hambly
affiliation : University of Edinburgh
approved by: GENIUS management board
reference: GAIA-C9-TN-IFA-STV-002-1
issue: 1
revision: 0
date: 2017-03-07
status: Issued

Abstract

We describe a Javascript library for TAP/ADQL Archive User Interfaces that provides an Autofill functionality for ADQL UI textboxes. This assists users while typing an ADQL query, providing them with a context–specific list of suggestions including ADQL keywords and metadata for the service that the library is initiated on.
This work has received funding from the European Community's Seventh Framework Programme (FP7–SPACE–2013–1) under grant agreement number 606740.

Document History

Issue	Revision	Date	Author	Comment
0	3	2017-03-01	NCH	Proof read and prepared for Livelink issue and final delivery
0	2	2017-02-28	NCH	Tweaked for Texlipse set up and imported into DPAC SVN
0	1	2017-02-27	STV	Creation of draft skeleton based on DQP ICD

Contents

1 Introduction

Based on a request for a potential enhancement to the GACS interface that would allow a user to trigger an auto–fill, ADQL keyword and metadata suggestion list for the ADQL query textbox (JIRA issue C9VALBT-41) we have built a Javascript library[1] that has the capability of being plugged in with multiple different back–end modes. In particular this has served as an improvement to our own GENIUS DQP demonstrator[2] and was then proposed and is in the process of being integrated into the Gaia Archive Core Systems interface.

1.1 Objectives

The objective of this document is to describe the library, its capabilities and run options. Furthermore we describe how we use it with the resources and data we have locally and it's use in the GACS integration interface and how it will enhance the usability of the GACS system.

1.2 Scope

This document describes the following resources:

- https://github.com/stvoutsin/tap-autocomplete

and along with that software forms part of Deliverable D3.4 for the GENIUS project (a European Community Seventh Framework Programme, FP7–SPACE–2013–1, under grant agreement number 606740).

1.3 Applicable Documents

WOM-033	Gaia Catalogue and Archive Software Requirements Specification
AB-026	Gaia data access scenarios summary
NCH-031	Science Archive Architecture and Development SRS

1.4 References

[AB-026], Brown, A., Arenou, F., Hambly, N., et al., 2012, *Gaia data access scenarios summary*,

[1] https://github.com/stvoutsin/tap-autocomplete
[2] http://genius.roe.ac.uk

GAIA-C9-TN-LEI-AB-026,
URL http://www.rssd.esa.int/cs/livelink/open/3125400 4

[NCH-031], Hambly, N.C., 2014, *CU9 Science Archive Architecture and Development Workpackage Software Requirements Specification (WP930)*,
Draft 3,
URL http://gaia.esac.esa.int/dpacsvn/DPAC/CU9/docs/WP930_Architecture/ECSS/SRS/GAIA-C9-SP-IFA-NCH-031/,
GAIA-C9-SP-IFA-NCH-031 4

[WOM-033], O'Mullane, W., 2009, *Gaia Catalogue and Archive Software Requirements and Specification*,
GAIA-C9-SD-ESAC-WOM-033,
URL http://www.rssd.esa.int/cs/livelink/open/2907710 4

2 Acronyms

The following table has been generated from the on-line Gaia acronym list:

Acronym	Description
ADQL	Astronomical Data Query Language
CSS	Cascading Style Sheet
DPAC	Data Processing and Analysis Consortium
DQP	Distributed Query Processing
ESAC	European Space Astronomy Centre
GACS	Gaia Archive Core Systems
GENIUS	Gaia European Network for Improved User Services
GWT	Google Web Tools
HTML	Hyper Text Markup Language
ICD	Interface Control Document
ID	Identifier
IDE	Integrated Development Environment
SVN	SubVersioN (revision control system)
TAP	Table Access Protocol
UI	(Graphical) User Interface

3 Project overview

This project was based on a library built by Grégory Mantelet, the Javascript ADQL editor[3], which is an extension of the native javascript editor CodeMirror[4], for ADQL that provides ADQL syntax highlighting for a text area. GACS is already using this library to provide syntax highlighting for the query textbox thus the extension we built was designed to replace this, with our own extension, which along with syntax highlighting also provides autocomplete suggestions.

This library was originally built and tested to operate on a TAP service, by making a request to the "/vosi"[5] endpoint of the service on initiation of the library, to gather the metadata of that service. The autocomplete service would then serve suggestions based on what was typed by the user, looking at both the ADQL keywords and all the metadata. This was then extended to use the TAP_SCHEMA tables of the TAP service it sits in front of to provide a context specific list of suggestions.

This means, for example, if a user has typed the name of a table, e.g. gaia_source and then the dot ('.') character to trigger the auto–complete event, the library will launch a TAP query that requests and returns all columns that belong to the gaia_source table. By adding a prefix, for example "gaia_source.astr.", this would search for columns that start with the "astr" keyword.

This works in a similar way to most code editors (such as in Eclipse IDE) allowing users that may not have detailed knowledge of the contents of all schemas and tables to formulate a query without having to go through the metadata tree (assuming the names are self-explanatory). It also makes it easier for advanced users to type out a query as the autocomplete allows them to use shortcuts while typing.

During integration with the ESAC team responsible for the GACS UI, we decided that it would be best to introduce another mode, which was a GWT (Google Web Toolkit[6]) back-end to the library. The reason for this was to avoid potential stress to the TAP service with the TAP_SCHEMA requests, and also because of the GACS design. This was already built in GWT thus the integration process would be much simpler and would fit with how other libraries are already being used.

[3] http://cdsportal.u-strasbg.fr/adqltuto/download.html
[4] https://codemirror.net/
[5] http://www.ivoa.net/documents/VOSI/
[6] http://www.gwtproject.org

4 Software components

The TAP-Autocomplete library was developed in Javascript and CSS and is designed to be initialised on an HTML "textarea" element. The library consists of the following components:

Dependencies:

- JQuery 1.7.2 or newer

Javasript files:

- tap-autocomplete.js
- tap-hint.js
- simple-hint.js
- codemirror.js

CSS files:

- simple-hint.css
- codemirror.css
- adql.css

GWT folder structure:

- src/ All Java GWT classes used by the example project
- war/ Generated application files for GWT

These files are required and need to be imported into the HTML content where the library is to be used.

The GWT github project linked above is built as a GWT prototype project, and the autocomplete library files can be located at https://github.com/stvoutsin/tap-autocomplete/tree/master/gwt-genius/public.

The GWT project has a Genius.html template file where example usage can be found. The src/ directory of that projects also contains the Java files used by the GWT prototype. This prototype has been used to assist in the integration process for the GACS interface.

5 How to use the library

5.1 Library Imports

Within the HTML page you are using to deploy the library import the following:

```
<script  type="text/javascript" src="adql_syntax/
jquery-1.7.2.min.js"></script>
<link rel="stylesheet" href="adql_syntax/codemirror.css">
<link rel="stylesheet" href="adql_syntax/adql.css">
<link rel="stylesheet" href="adql_syntax/simple-hint.css">
<script src="adql_syntax/codemirror.js"></script>
<script src="adql_syntax/adql.js"></script>
<script src="adql_syntax/tap-hint.js"></script>
```

5.2 Initialization

Initialise the Autocomplete class:

```
var params = {

    textfieldid: "textfield",
    web_service_path: 'genius/autocompleteAsync',
    tap_resource: 'https://gaia.esac.esa.int/tap-server/tap/',
    servicemode: "TAP"

}

var autocompleteInstance = new TapAutocomplete(params);
```

5.3 Additional Parameters

The Class can be initialised with the following parameters:

```
* @param {string} textfieldid - The textarea ID that the
TapAutocomplete class will be instantiated for.

* @param {string} textAreaElement - The textarea element
that the TapAutocomplete class will be instantiated for.

* @param {string} web_service_path - The resource path
that will run the TAP_SCHEMA requests and return a list
of keywords

* @param {string} tap_resource - The TAP service

* @param {string} servicemode - Mode: TAP/VOSI/jsontree/gwt

* @param {string} autocomplete_info_id - Id of loader element
to be toggled while keywords are being loaded

* @param {string} autocomplete_info_id - Id of loader element
to be toggled while keywords are being loaded

* @param {string} initial_catalogues - Array of strings,
For each (0 or more) catalogue, fetch list of children tables
on startup
```

5.4 Example HTML

An Example HTML textarea, on which the library can be initialised:

```
<table align="center">
  <tr>
    <td colspan="2" style="font-weight:bold;">Please enter your query:
    </td>
  </tr>
```

```
  <tr>
    <td id="nameFieldContainer"><textarea id="textfield"
    class="gwt-TextArea"></textarea></td>
    <td id="sendButtonContainer"></td>
  </tr>
  <tr>
    <td colspan="2" style="color:red;" id="errorLabelContainer">
    </td>
  </tr>
</table>
```

5.5 GWT/GACS Integration

The following describes the integration process with the existing GWT–based GACS interface.

The main TAP Autocomplete Javascript class (tap-autocomplete.js), when run with the "gwt" mode, depends on a class named "gacsGetByKeyword" that needs to exist in the window element. This handles the communication between the GWT components and the Javascript classes. This function generates and returns a list of keywords, based on the parameters provided to it. If provided with a keyword such as "gaia_source.astr", the GWT component then fetches from the "TableData" Java class instance (which contains the table metadata), all metadata keywords that match "astr" and belong to the gaia_source table.

During the integration process, an additional requirement, which was to enable alias specific keyword matching was raised and later added to the library. This provides a user with specific suggestions if the keyword they are matching on is an alias. For example, if a user types a query such as

```
SELECT .. from gaia_source as gs where
gs.
```

and triggers the functionality on the last part, the library would provide suggestions for the gaia_source table columns.

An example of what this looks like in the GACS integration server is as follows:

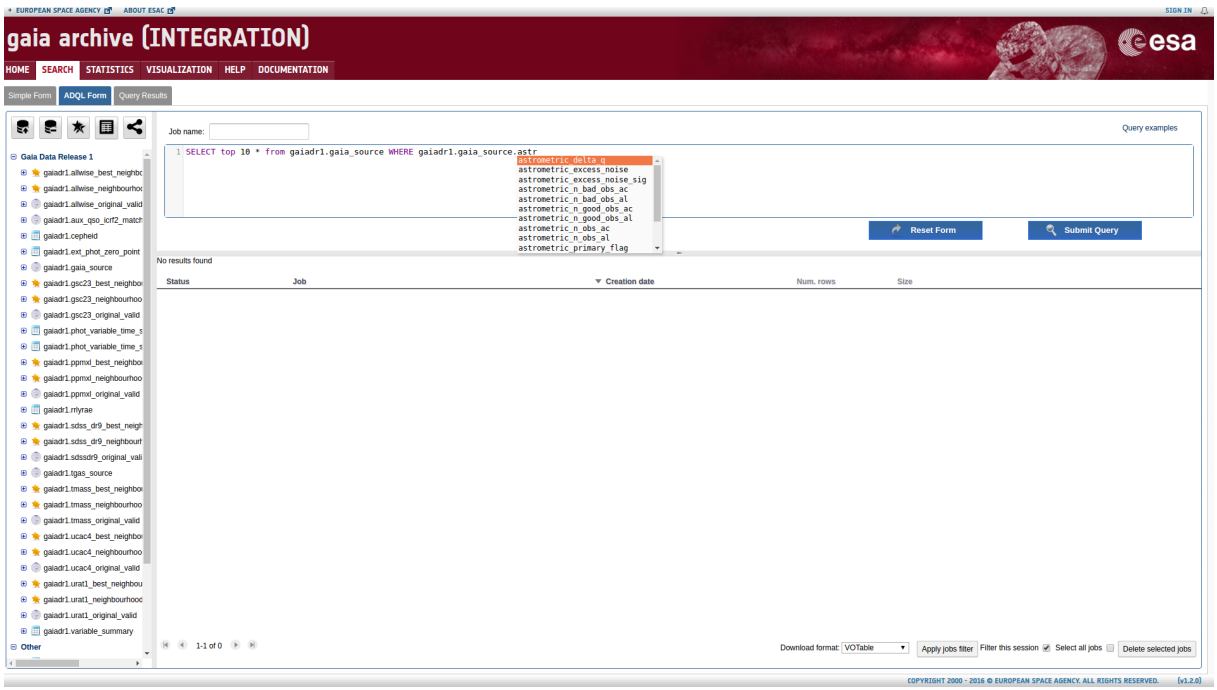

FIGURE 1: GACS Integration, TAP Autocomplete

Acknowledgements

This work has received funding from the European Community's Seventh Framework Programme (FP7–SPACE–2013–1) under grant agreement number 606740, a.k.a. the GENIUS project.