ConalTuohy.com

V1.1

PROV IIIF Collection service, API Query Builder, and Photo Wall

This document covers the development of three related software components based on PROV's public Solr API and image server.

- Solr API Query Builder Form
- IIIF Collection Service Solr plugin
- IIIF Collection Photo Wall

Solr API Query Builder Form

This component implements an HTML5 form-based interface for users to construct a Solr query URI and submit it to PROV's public API.

This component is implemented in JavaScript, HTML5, and CSS.

Solr's web API is not directly usable with standard HTML forms. This component includes the JavaScript library `api-query-builder-form.js` which enables an ordinary HTML5 form to produce query URLs to invoke a Solr API.

The component provides an example of how to use the JavaScript, in the form of a web page called `api-query-builder-form.html`, which contains an HTML5 form, containing several different inputs, marked up according to certain conventions which allow the JavaScript code to correctly connect it to the PROV Solr API.

A second example, in the web page `iiif-collection-photo-wall.html`, shows a much simpler form being used with the same JavaScript library.

How to design a form for use with the JS library

Setting up the form

To use the library, a form designer should create an HTML5 form in an HTML page which includes a `script` element linking to the `api-query-builder-form.js` library.

The form's `class` attribute should contain the token `query-builder`, to signal to the JavaScript that the form is to be connected to the Solr API. Forms without this `class` value will be left untouched.

The form's `action` attribute should be set to the base URL of the Solr API; in production this should be set to "https://api.prov.vic.gov.au/search/query".

The form should then be populated with input elements as is normal for an HTML form. The JavaScript library will interpret each of the `input` elements in the form as corresponding to either:

- a metadata field defined in Solr's schema (e.g. `iiif-manifest`, `text`, or `series_id`), or
- a "control" parameter for Solr's API, to specify the desired output format, number of results, sort order, etc, e.g. `wt`, `rows`, or `sort`.

To distinguish metadata fields from control parameters, the `class` attribute of each input should include one or other of the tokens `metadata` or `control`.

Metadata fields

Elements within the form which have the class metadata will be treated as referring to Solr fields. These elements should have a name attribute that matches the name of a Solr field. When the form is submitted, the values of these elements will be serialized into the URL as portions of the q (query) parameter. The resulting Solr query will specify a logical conjunction of the various field values in the form; i.e. all the specified field values must match (they will be effectively combined with a boolean AND operator).

The form allows for multiple values to be specified for a given field, either by having multiple input elements with the same name attribute or by having a select element with the multiple attribute. When a field has multiple values, the resulting query will specify a logical disjunction of those values (i.e. they will be effectively combined with a boolean OR operator).

Normally the value of a field will be treated simply as a set of words, but it can be treated as a phrase (so that e.g. searching for “dog tag” will not return dogs), by including the token phrase in the input element’s class attribute.

The query builder form provided in api-query-builder-form.html includes a text box for free text search, and a selection of other fields including series number and creation dates, a method to select the desired output format, number of rows, etc.

The query builder form provided in iiif-collection-photo-wall.html includes a text box for free text search, and additional hidden fields to restrict results to records which are Images or Photographs, and for which digitised images are available, as well as hidden control fields to specify a number of rows to return, and that the output should be in IIIF Collection format.

Imposing additional constraints on the query

As is normal for HTML forms, hidden input elements will contribute to the query URL in just the same way as visible elements, except that the user will not be able to modify them.

In a form which contains input elements whose type is hidden, the hidden fields provide an additional set of constraints to the query which the user specifies. e.g. a form could be deliberately restricted to only records with a record_form of “Photograph or Image” by including a hidden field like this one:

```
<input type="hidden" name="record_form" class="metadata phrase" value="Photograph or
Image">
```

Retrieving options from Solr

Some Solr fields contain only a small number of distinct values (e.g. series_number, or record_form). For these fields, it may be useful to provide the user with a list of suggested values drawn from the actual data, and allow the user to select from the list.

To support this, the JavaScript code will create option elements for any select element or datalist element in the form which doesn't already contain option elements, or which contain just a single option whose value attribute is blank (since this means “any value”). e.g. to create a drop-down listbox for the series_id field, with a default option matching any series_id value, and an option for every series_id value actually used in Solr, you would use the following HTML5 markup:

```
<select name="series_id" class="metadata">
        <option value="" selected>(any series id)</option>
</select>
```

The values of these options will be retrieved from Solr by running a facet query when the form initializes. In a form with hidden fields, the facet query will take those hidden restrictions into account when retrieving options from Solr, so that the retrieved options include only values that, when combined with the hidden field constraints, can still yield at least one matching record.

Query control parameters

Any element within the form whose class attribute includes the value control will directly contribute a parameter to Solr’s query URL. The URL parameter’s name and value will be the name attribute and the value of the element.

The control elements in the form can for example have name attributes such as rows, number, sort, etc. For full details of Solr's query parameters, see the Solr User Guide: https://solr.apache.org/guide/8_11/common-query-parameters.html

There should generally be an element named wt (writer type) to specify the desired Solr ResponseWriter, and hence the format which the search results are returned in. This parameter has a number of different valid values, of which the most useful are json, xml, and xslt. Solr's default output format is JSON, hence if the form doesn't include a wt control parameter, the resulting query URL will necessarily return data in JSON format.

When the wt parameter is set to xslt, Solr will first generate XML output and then transform it into something else using an XSLT stylesheet. When the wt parameter has the value xslt, Solr requires an additional parameter named tr (transform) which specifies the XSLT stylesheet to use. This tr parameter could be specified using a hidden input, or if multiple XSLT stylesheets are installed on the Solr server, then the user could be given a list of options using a select element.

Another software component in this project is an XSLT stylesheet which transforms Solr's output into IIIF Collection documents (which are in JSON format). To make use of this stylesheet, a form should include an element whose name attribute is wt and whose value is xslt, and an element whose name attribute is tr and whose value is solr-to-iiif.xsl.

The JavaScript library is aware of the relationship between wt and tr inputs, and will enable and disable the element whose name is tr depending on the value of the element whose name is wt, so that if wt is not set to xslt, the tr field will not be editable (and could even be hidden using CSS), and no tr parameter will be included in the query URL.

Similarly, any CSV-specific control parameters (whose names all begin with csv-) will be enabled only when the wt parameter is set to csv.

Form submission and processing

Normally when an HTML5 form is submitted, it will send a query directly to the URL specified in its action attribute. The api-query-builder-form.js library changes this default behaviour. The result of submitting a query-builder form is only that the form emits a urlChanged event. In order for something to actually happen, some other JavaScript will need to listen for this event, and take some action based on it.

When the user submits the query builder form (such as by clicking a button whose type is submit), the JavaScript library intercepts and suppresses the form's normal submit event processing, and instead it constructs a Solr query URL based on the values in the form, and causes the form to emit that URL as a urlChanged event. By adding a function to listen for this event (in the usual way for JavaScript event-handling), an HTML page can use the URL to actually make an HTTP request. The urlChanged event is a custom event, and includes a detail member which in turn includes three members:

- submitter which identifies the button which was used to submit the form.
- url which specifies the actual Solr query URL;
- downloadFilename which is a suggested download filename, such as e.g. "query-results.csv" or "query-results.json".

In the demonstration page api-query-builder-form.html, there is a urlChanged event listener which checks the event's submitter to see if the user clicked either a button named download or search, and either attempts to download from the URL or else sets it as the source of an iframe and a display hyperlink.

In the demonstration page iiif-collection-photo-wall.html, the urlChanged event triggers an event listener to update a grid of clickable image thumbnails. See the message-passing diagram in the Photo Wall section for full details.

IIIF Collection Service Solr plugin

This component is an XSLT 1.0 stylesheet intended to be used as a plugin for Apache Solr.

The stylesheet transforms query results from PROV's public Solr API into the form of a IIIF Collection. This will allow the public API to effectively function as a web service which implements the IIIF Presentation API.

The IIIF Collection documents returned by this service include links to image thumbnails and larger (though not full-size) images, along with rich metadata drawn from the Solr API.

The maximum number of IIIF Manifest items returned in each Collection can be specified by the `rows` parameter of the Solr query URL. If the number of results exceeds the specified maximum, the IIIF Collection will include a `seeAlso` link pointing to a second Collection containing more search results, and this Collection may include a link to yet another Collection, etc, until search results are returned.

Query URLs which specify this output format should also include a constraint that the `iiif-manifest` field is not blank. This is because records must provide IIIF data to be included in a IIIF Collection. When using the `api-query-builder-form.js` library to generate a query URL from a form, this can be achieved by including a hidden field with the name `iiif-manifest` and the value `*` (which means "any" in Solr's query language).

During development, the XSLT stylesheet is hosted externally to PROV, but upon completion of the development, PROV will be able to deploy the stylesheet directly within PROV's existing Solr engine. PROV's Solr server is already configured to include the XSLT ResponseWriter, so deploying the stylesheet as a plugin to Solr will simply require copying the stylesheet into the `xslt` folder of the Solr server.

In addition, to allow the photo wall and query builder to access the IIIF Collection service, the Solr web service will need to be configured to explicitly allow Cross Origin Resource Sharing (CORS). During development this setting isenabled using a web proxy on the development server, but the Solr API will need to have had CORS enabled before it can be called directly from JavaScript code in an arbitrary web page.

IIIF Collection photo wall

This component provides a front end on the IIIF Collection service.

This component is implemented as a JavaScript library named `iiif-collection-photo-wall.js`, which executes a Solr API request to retrieve a IIIF Collection document, and then converts that Collection into an HTML5 user interface displaying each image in the collection as a clickable thumbnail image.

This is intended to allow arbitrary subsets of PROV image collections to be displayed as a grid of images embedded in a web page. Clicking a thumbnail produces a larger image, along with metadata about the image, and a link to the corresponding catalogue entity page on PROV's collection website in which the record is presented in a IIIF viewer.

How to embed the photo wall in an HTML page

To embed a photo wall in an HTML page, the page should include a script element linking to the `iiif-collection-photo-wall.js` library. When the page loads, the JavaScript will scan the page looking for empty `div` elements whose `class` attribute includes the token `iiif-collection-photo-wall`, and will convert any such `div` into a photo wall.

The JavaScript library adds an event listener to observe the `data-iiif-collection-url` attribute of any photo wall `div` it finds. The `data-iiif-collection-url` attribute may be included directly in the HTML of the page, or it may be initially missing, and set later by some other JavaScript code. Whenever the `data-iiif-collection-url` value is changed, the JavaScript will load the IIIF Collection from that URL, and display its contents as a photo wall.

There are two demonstration web pages for the photo wall component.

The simpler page, iiif-collection-photo-wall-fixed-url.html, includes a link to the JavaScript, a link to the CSS stylesheet, and a div with a fixed data source:

```
<script src="iiif-collection-photo-wall.js"></script>
<link rel="stylesheet" href="iiif-collection-photo-wall.css">
<div id="photo-wall" data-iiif-collection-url="../sample-data/coffee.json" class="iiif-collection-photo-wall"></div>
```

The more complex page, iiif-collection-photo-wall.html, has a photo wall div with no data-iiif-collection-url attribute. Instead the page uses a search form in conjunction with the query-builder form JavaScript library to dynamically generate IIIF Collection URLs from the user's search query.

The page includes a small search form with an input named text (matching the name of a free text field in the Solr schema), hidden inputs wt and tr to specify that the result should be in IIIF Collection format, a hidden input named rows to restrict the number of images loaded at a time.

The page includes a link to the api-query-builder-form.js library so that it can use that form to generate Solr query URLs, and also includes an inline JavaScript event listener which listens to the form's urlChanged event, and uses it to update the photo wall div's data-iiif-collection-url attribute. See the commented source code of the page for full details

See the diagram below for a graphical depiction of the message passing between the software components.

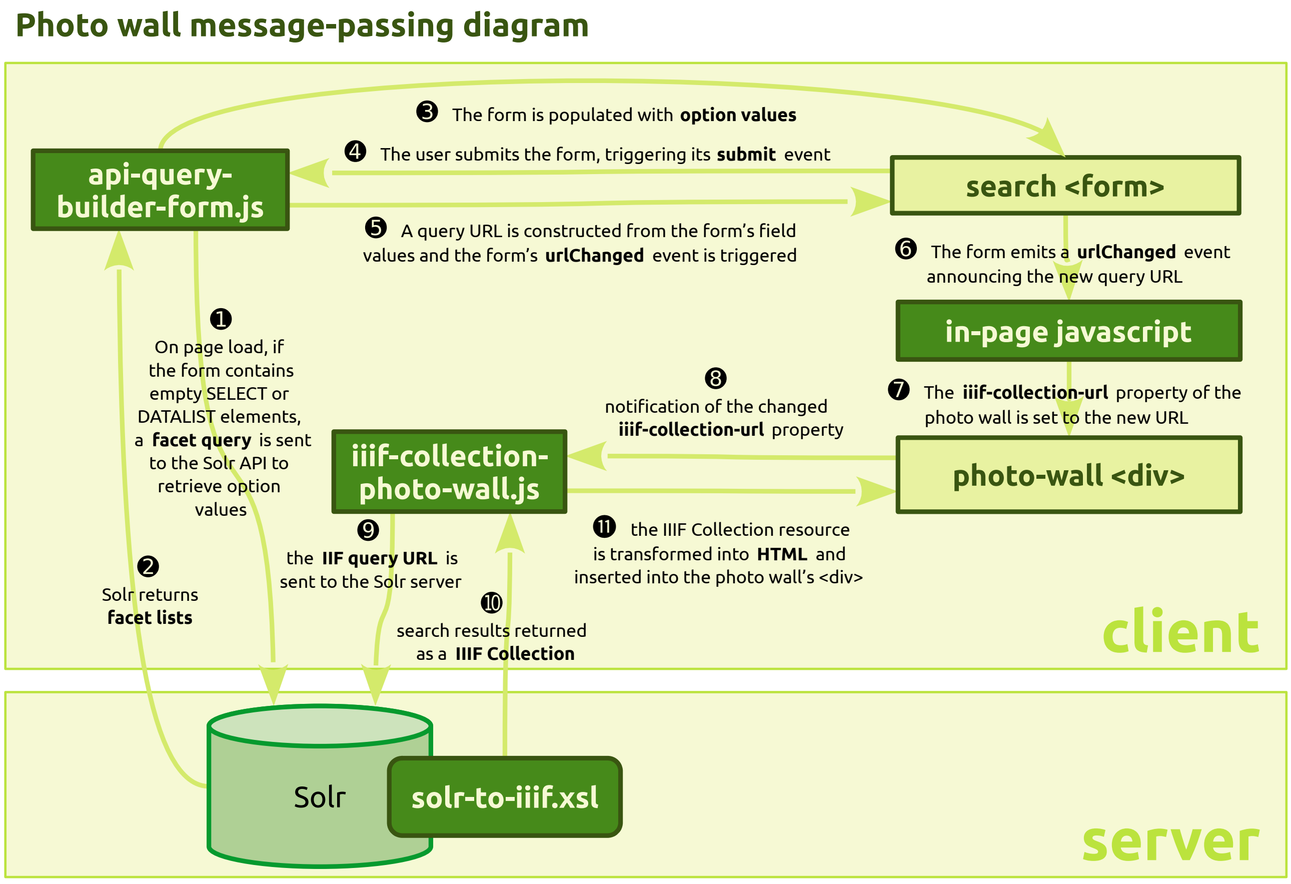

Photo wall message-passing diagram
❸ The form is populated with **option values**
❹ The user submits the form, triggering its **submit** event
api-query-builder-form.js
search <form>
❺ A query URL is constructed from the form's field values and the form's **urlChanged** event is triggered
❻ The form emits a **urlChanged** event announcing the new query URL
in-page javascript
❶ On page load, if the form contains empty SELECT or DATALIST elements, a **facet query** is sent to the Solr API to retrieve option values
❽ notification of the changed **iiif-collection-url** property
❼ The **iiif-collection-url** property of the photo wall is set to the new URL
iiif-collection-photo-wall.js
photo-wall <div>
❾ the **IIF query URL** is sent to the Solr server
⓫ the IIIF Collection resource is transformed into **HTML** and inserted into the photo wall's <div>
❷ Solr returns **facet lists**
❿ search results returned as a **IIIF Collection**
client
Solr
solr-to-iiif.xsl
server

Deployment

The Query Builder and Photo Wall components are simply JavaScript and CSS libraries which execute in the user's browser, and can be stored and hosted on any PROV web server.

The IIIF Collection service stylesheet can be deployed to PROV's Solr API by copying it into the Solr server's "xslt" folder. This will upgrade the Solr API to provide IIIF Collections as an additional format alongside the existing Solr formats of JSON, XML, and CSV.

Cross Origin Resource Sharing

In addition, in order that PROV's API can be invoked by web browser code such as the Query Builder, Photo Wall, or other IIIF client software, the API server needs a small configuration change to enable Cross Origin Resource Sharing, by returning the appropriate CORS HTTP headers. See <https://en.wikipedia.org/wiki/Cross-origin_resource_sharing> for details. This is something which PROV IT staff should be able to do easily in a matter of an hour or two, and should do as soon as possible.

Graphic Design

The Query Builder form and Photo Wall user interfaces are implemented in HTML, and their graphic design is controlled by CSS. A default CSS stylesheet is supplied for each component, with a basic graphic design.