Check for updates

WEB TOOL

BioJS InterMineTable Component: A BioJS component for displaying data from InterMine compatible webservice endpoints [version 1; peer review: 1 approved]

Alexis Kalderimis, Radek Stepan, Julie Sullivan, Rachel Lyne, Michael Lyne, Gos Micklem

Department of Genetics and Cambridge Systems Biology Centre, Cambridge University, Cambridge, CB2 3EH, UK

V1 First published: 13 Feb 2014, 3:46
<https://doi.org/10.12688/f1000research.3-46.v1>

Latest published: 13 Feb 2014, 3:46
<https://doi.org/10.12688/f1000research.3-46.v1>

Abstract

Summary: The InterMineTable component is a reusable JavaScript component as part of the BioJS project. It enables users to embed powerful table-based query facilities in their websites with access to genomic data-warehouses such as <http://www.flymine.org>, which allow users to perform flexible queries over a wide range of integrated data types.

Availability: <http://github.com/alexkalderimis/im-tables-biojs>;
<http://github.com/biojs/biojs>; <http://dx.doi.org/10.5281/zenodo.8301>.

Open Peer Review

Approval Status

1

version 1

13 Feb 2014

1. **Arnaud Ceol**, Istituto Italiano di Tecnologia (IIT@SEMM), Milan, Italy

Any reports and responses or comments on the article can be found at the end of the article.

BIOJS

This article is included in the **BioJS** collection.

Corresponding author: Gos Micklem (g.micklem@gen.cam.ac.uk)

Competing interests: No competing interests were disclosed.

Grant information: InterMine has been developed with the support of the following grants, awarded to Dr. G. Micklem: the Wellcome Trust (Grant number: 090297), and the National Human Genome Research Institute (Grant number: R01HG004834). The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding bodies.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Copyright: © 2014 Kalderimis A *et al.* This is an open access article distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. Data associated with the article are available under the terms of the [Creative Commons Zero "No rights reserved" data waiver](#) (CC0 1.0 Public domain dedication).

How to cite this article: Kalderimis A, Stepan R, Sullivan J *et al.* **BioJS InterMineTable Component: A BioJS component for displaying data from InterMine compatible webservice endpoints [version 1; peer review: 1 approved]** F1000Research 2014, 3:46
<https://doi.org/10.12688/f1000research.3-46.v1>

First published: 13 Feb 2014, 3:46 <https://doi.org/10.12688/f1000research.3-46.v1>

Introduction

There are currently a number of genomics data-warehouses available which are powered by the InterMine¹ platform. This set includes large curated services dedicated to the primary Model Organism database (MOD) communities as part of the InterMOD project², the collected data sets of research projects such as the modENCODE project³, as well as a range of other resources including metabolic-Mine⁴, TargetMine⁵, FlyTFMine⁶, and MitoMiner⁷. In addition to being accessible through web-interfaces these resources also provide web-service access (to be described in a forthcoming paper).

The InterMine system provides users with a number of benefits. A typical InterMine instance, such as FlyMine⁸ or YeastMine⁹, contains feature annotations, protein data, publications, biochemical pathways, orthology, Gene Ontology (GO), array expression results, and other kinds of data, all integrated into a single knowledge graph. This means end users are able to ask questions across different data types. InterMine's particular data integration strategy puts minimal limitations on the kinds of queries that can be performed: any arbitrary number of data-sets can be referred to in the same query (provided links exist between them) and a wide variety of logical constraints can be added. The InterMine platform thus provides a basis for very flexible, user-defined queries over linked data sets.

The BioJS¹⁰ project seeks to provide a suite of reusable JavaScript components that members of the bioinformatics community will find useful for producing analysis and visualisation tools. The InterMineTable BioJS component contributes towards this aim by adding a data query and exploration tool to the set of BioJS components which exposes the full flexibility and power of user defined queries over integrated linked data in a clear user interface.

Installation

As a visual BioJS component, the intended audience is web-developers aiming to provide extended functionality to web-based resources for life-scientists. It is expected to be deployed within modern browser environments with access to third-party resources. With this in mind, installation comprises of including the dependencies for the InterMine table component on the page (usually added in the head section of an HTML page), see [Supplementary materials A](#). Once the dependency on the InterMine tables library is loaded, the InterMineTable BioJS component may be included (see [code listing 1](#)).

Listing 1. Loading the BioJS InterMine Table library

```
<script
  src="Biojs.InterMine.Table.js"></script>
```

This last resource contains the definition of the InterMineTable BioJS component. As it is not available from a reliable third party source, it currently needs to be downloaded from the BioJS registry¹¹, and hosted locally.

Usage

This component is used by instantiating the InterMine table component in a user-included JavaScript file, passing in the appropriate configuration for the desired source of data as well as the query over those data.

Once instantiated, the results of the query against the specified integrated data-warehouse are loaded into a component where they can

be browsed and manipulated. This means that the two critical concepts for using this component are 1) the location of the data-store, defined as the uniform resource locator (URL) pointing at the root of a set of web-services, and 2) the query to be run on the data in the store, defined in a configuration object. For example, to load a table of data from FlyMine the user would want the URL to point to FlyMine's webservices:

Listing 2. Specifying the Data-Store

```
var url = "http://www.flymine.org/query";
```

The query can be broadly defined as a list of fields, identified by paths from a root, constrained by a (possibly empty) set of filters. There are some refinements to this (such as sort-order, optional element definition, and constraint composition) for which more detailed documentation¹² exists. The concept of a path is important to the idea of a graph of linked data, as it enables chains of relationships between entities to be followed, with minimal syntactic overhead. For example the chain of relationships *the names of the protein domains of the proteins encoded by the genes belonging to a biochemical pathway* can be referred to as `Pathway.genes.proteins.proteinDomains.name`.

A query is defined as a plain JavaScript object which can be simple, such as the following query, which requests the common name, scientific name and taxon ID for all organisms in the data-store:

Listing 3. A simple Query, see Figure 1

```
// Available organisms
var query = {
  name: "An optional name",
  from: "Organism",
  select: [ "commonName", "name", "taxonId"],
};
```

or arbitrarily complex, such as the following query which combines information from multiple data sources (OMIM¹³, PANTHER¹⁴, Treefam¹⁵, KEGG¹⁶, Reactome¹⁷, FlyBase¹⁸) and across different organisms to find the *Drosophila melanogaster* genes in the pathways of genes which are orthologous to human genes implicated in Alzheimer's disease:

Listing 4. A Complex Query, see Figure 2

```
var disease = {
  from: "Disease",
  select: [
    "genes.homologues.homologue.pathways.genes.*"
  ],
  where: {
    "name": "Alzheimer*",
    "genes.organism.name": "Homo sapiens",
    "genes.homologues.homologue.organism.name":
      "Drosophila melanogaster"
  }
};
```

An element also needs to be present on the page where the table should be loaded. This can be any element (although a DIV element is conventional), and should be uniquely identifiable (through its ID for instance).

Listing 5. Defining the Target Element

```
var target = "#table-container";
```

These values are then passed to the component constructor, which builds a new table in the page, and loads the relevant data from the configured service:

Listing 6. Instantiation

```
var table = new Biojs.InterMine.Table({
  target: target,
```

```
url: url,
query: query
});
```

Once instantiated, a table will be loaded into the page displaying rows of data as specified by the query (see [Figure 1](#), [Figure 2](#)).

Interaction

The table, as well as providing a number of common dynamic features such as resorting, pagination and column rearrangement, also

Common Name	Name	Taxon Id
African malaria mosquito	Anopheles gambiae	7165
Denisova hominin	Homo sapiens ssp. Denisova	741158
Norway rat	Rattus norvegicus	10116
baker's yeast	Saccharomyces cerevisiae	4932
fruit fly	Drosophila melanogaster	7227
house mouse	Mus musculus	10090
human	Homo sapiens	9606
nematode	Caenorhabditis elegans	6239

Figure 1. A table of data loaded from FlyMine displaying the output of [Code Listing 3](#).

Genes » Secondary Identifier	Genes » Symbol	Genes » DB Identifier	Genes » Organism . Name
CG10000	CG10000	FBgn0039596	Drosophila melanogaster
CG10033	for	FBgn0000721	Drosophila melanogaster
CG10072	sgl	FBgn0261445	Drosophila melanogaster
CG10078	Prat2	FBgn0041194	Drosophila melanogaster
CG10079	Egfr	FBgn0003731	Drosophila melanogaster
CG10096	CG10096	FBgn0038032	Drosophila melanogaster
CG10097	CG10097	FBgn0038033	Drosophila melanogaster
CG10105	Sin1	FBgn0033935	Drosophila melanogaster

Figure 2. A table of data loaded from FlyMine displaying the output of [Code Listing 4](#).

permits much deeper interaction than other comparable table libraries. The table allows the underlying query to be changed: the constraints of the underlying query can be edited (Figure 3); columns (including ones referring to data types not in the original query) can be added; existing columns can be removed; changes made can be

undone; the data can be exported in a number of formats or sent to another application, such as a local Galaxy^{19–21} instance, or to a remote application such as GenomeSpace²² (Figure 4); the results can be saved as a reusable set (a *list*) within the originating service; individual items can also be previewed (Figure 5).

Figure 3. Editing the filters on a table.

Figure 4. Downloading the results of a query.

The screenshot shows a web application interface with a table of genes. The table has columns for 'Genes » Symbol', 'Genes » DB identifier', and others. A modal window titled 'Gene' is open, displaying details for a specific gene. The modal contains a table with the following data:

Cytological Location	98E3-98E3
DB identifier	FBgn0039596
Length	3278
Organism	<i>Drosophila melanogaster</i>
Secondary Identifier	CG10000
Symbol	CG10000

The background table shows rows with 'CG10000' and 'FBgn0039596'.

Figure 5. Previewing an item.

One particularly useful feature is the ability to view the contents of a single column, analysing it on aggregate and adding or editing filters. This facility is able to present summary charts for columns based on data type: binned histograms for numerical data (Figure 6), and column charts for categorical data (Figure 7, showing the user adding a filter by selecting items from the column).

Events

The standard mechanism for communication between components in JavaScript is event signalling. As per the BioJS specification, this component supports other objects registering event listeners so they may be notified when events of interest (such as user interactions) occur.

Once loaded, the table may emit a number of different events (as listed in the API documentation²³), and may be manipulated by calling methods on the instance, allowing the calling page to respond to user interactions. For example, if a developer wished to receive notifications when the user clicks on any of the cells in the table, they can register to listen for these events:

Listing 7. Adding an Event Listener

```
table.addListener("imo:click", function
  (type, id) {
    alert("User clicked on" + type + " " + id);
  });
```

This integration means that the table need not be an isolated part of an application, but can be fully integrated with other components. For example, instead of just notifying the user by using `alert`, the information about this object could be displayed in another component. If the user clicked on a protein, this could be detected, and other suitable components could be instantiated to display protein-specific analysis (see [code sample 8](#)).

Listing 8. Integrating with Other Components - Example 1

```
table.addListener("imo:click", function
  (type, id) {
    if ("Protein" === type) {
      table.service.findById(type, id)
        .then(loadProteinStructureDisplay);
    }
    function
      loadProteinStructureDisplay(protein) {
        // ... load other BioJS component here.
      }
  });
```

As well as responding to user interaction with the table, the table component exposes an API to change the state of the table by changing the query it represents. This allows communication in the other direction. For example if a linked component, such as a protein structure display, emits an event indicating the user has selected a given set of protein domains, the table could be modified by adding a filter for these domains to the current query (see [code sample 9](#)):

Listing 9. Integrating with Other Components - Example 2

```
/* Assuming a "displayer" component that
  emits the "domains:selected" event */
displayer.addListener("domains:selected",
  function (domains) {
    var currentQuery = table.getQuery();
    currentQuery.addConstraint({
      "Gene.proteins.proteinDomains.identifier":
        domains
    });
  });
```

Figure 6. Analysing and filtering a single column of numeric data within a table.

Figure 7. Analysing and filtering a single column of categorical data within a table.

In this way, the interoperability of these components makes them of increasing utility to developers, as more of them are published and integrated into third party applications.

Discussion

This tool addresses an important set of needs for bioinformatics developers. While there are some other libraries that aim to make creating dynamic tables easy to develop (such as DataTables²⁴), there are none that tightly integrate facilities such as query constraint editing, column summary analysis, data export, support for genomic data types, and list management without a great deal of development effort. Access to flexible and powerful data-warehousing tools on the InterMine platform makes these table components uniquely powerful in the life sciences sphere. Services that implement the InterMine public API are also able to make use of the features of these tables. The extremely high quality data sets curated by the principal Model Organism databases, which the InterMOD project members make available, makes this tool unique in its immediate access to integrated life sciences data.

Conclusions

It is hoped that this component will prove useful to those developing tools for researchers in the life sciences. Various publically funded groups, such as the MODs and other data-producing consortia, have put significant effort into creating, curating and composing high quality data sets. InterMine is an effective platform to add value to this work by integrating the data and exposing a flexible query API and user interface. The recent work in exposing these resources through web-services and producing reusable web-based components allows this investment to benefit not just visitors to the sites of InterMine applications, but any developer or user who wants to include complex query tools as part of their platform. With access to

a broad range of data sources meeting the needs of several diverse research communities, we expect that a great deal of duplicated effort can be avoided, saving significant amounts of time and money.

Software availability

Zenodo: im-tables-biojs, doi: [10.5281/zenodo.830125](https://doi.org/10.5281/zenodo.830125)

GitHub: BioJS, <http://github.com/biojs/biojs>.

Author contributions

Alex Kalderimis wrote the manuscript and implemented the component, under the supervision of Gos Micklem, to a set of user specifications supplied by Julie Sullivan. Rachel Lyne, Radek Štěpán and Mike Lyne contributed to the component design and revised the manuscript. All authors have approved the manuscript.

Competing interests

No competing interests were disclosed.

Grant information

InterMine has been developed with the support of the following grants, awarded to Dr. G. Micklem: the Wellcome Trust (Grant number: 090297), and the National Human Genome Research Institute (Grant number: R01HG004834). The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding bodies.

The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Acknowledgements

The authors thank Manuel Corpas for useful feedback.

Supplementary materials

A Dependencies

```
<link
  rel="stylesheet"
  type="text/css"
  href="http://cdn.intermine.org/js/
intermine/im-tables/latest/iptables.css">
<script
  charset="UTF-8"
```

```
type="text/javascript"
src="http://cdn.intermine.org/js/
intermine/im-tables/latest/iptables-bundled.js">
</script>
```

Here we are referring to resources which are made publically available as part of a Content Delivery Network (CDN). These resources could just as well be hosted locally.

References

- Smith RN, Aleksic J, Butano D, *et al.*: **InterMine: a flexible data warehouse system for the integration and analysis of heterogeneous biological data.** *Bioinformatics*. 2012; **28**(23): 3163–3165. [PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Sullivan J, Karra K, Moxon SA, *et al.*: **InterMOD: integrated data and tools for the unification of model organism research.** *Sci Rep*. 2013; **3**: 1802. [PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Contrino S, Smith RN, Butano D, *et al.*: **modMine: flexible access to modENCODE data.** *Nucleic Acids Res*. 2012; **40**(Database issue): D1082–1088. [PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
- Lyne M, Smith RN, Lyne R, *et al.*: **metabolicMine: an integrated genomics, genetics and proteomics data warehouse for common metabolic disease research.** *Database (Oxford)*. 2013; **2013**: bat060. [PubMed Abstract](#) | [Publisher Full Text](#)
- Chen YA, Tripathi LP, Mizuguchi K: **TargetMine, an integrated data warehouse for candidate gene prioritisation and target discovery.** *PLoS ONE*. 2011; **6**(3):

- e17844.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
6. Adryan B, Teichmann SA: **FlyTF: a systematic review of site-specific transcription factors in the fruit fly *Drosophila melanogaster*.** *Bioinformatics*. 2006; **22**(12): 1532–1533.
[PubMed Abstract](#) | [Publisher Full Text](#)
 7. Smith AC, Robinson AJ: **MitoMiner, an integrated database for the storage and analysis of mitochondrial proteomics data.** *Mol. Cell Proteomics*. 2009; **8**(6): 1324–1337.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 8. Lyne R, Smith R, Rutherford K, *et al.*: **FlyMine: an integrated database for *Drosophila* and *Anopheles* genomics.** *Genome Biol*. 2007; **8**(7): R129.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 9. Balakrishnan R, Park J, Karra K, *et al.*: **YeastMine—an integrated data warehouse for *Saccharomyces cerevisiae* data as a multipurpose tool-kit.** *Database (Oxford)*. 2012; **2012**: bar062.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 10. Gómez J, García LJ, Salazar GA, *et al.*: **BioJS: an open source JavaScript framework for biological data visualization.** *Bioinformatics*. 2013; **29**(8): 1103–1104.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 11. **Biojs project registry.**
[Reference Source](#)
 12. **Pathquery documentation.**
[Reference Source](#)
 13. McKusick-Nathans Institute of Genetic Medicine, Johns Hopkins University (Baltimore, MD) Online mendelian inheritance in man, omim®. 2013.
[Reference Source](#)
 14. Mi H, Muruganujan A, Thomas PD: **PANTHER in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees.** *Nucleic Acids Res*. 2013; **41**(Database issue): D377–386.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 15. Li H, Coghlan A, Ruan J, *et al.*: **TreeFam: a curated database of phylogenetic trees of animal gene families.** *Nucleic Acids Res*. 2006; **34**(Database issue): D572–580.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 16. Ogata H, Goto S, Sato K, *et al.*: **KEGG: Kyoto Encyclopedia of Genes and Genomes.** *Nucleic Acids Res*. 1999; **27**(1): 29–34.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 17. Joshi-Tope G, Gillespie M, Vastrik I, *et al.*: **Reactome: a knowledgebase of biological pathways.** *Nucleic Acids Res*. 2005; **33**(Database issue): D428–432.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 18. Marygold SJ, Leyland PC, Seal RL, *et al.*: **Fly-Base: improvements to the bibliography.** *Nucleic Acids Res*. 2013; **41**(Database issue): D751–757.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 19. Goecks J, Nekrutenko A, Taylor J, *et al.*: **Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences.** *Genome Biol*. 2010; **11**(8): R86.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 20. Blankenberg D, Von Kuster G, Coraor N, *et al.*: **Galaxy: A web-based genome analysis tool for experimentalists.** *Curr Protoc Mol Biol*. 2010; Chapter 19: 19.10.1–19.10.21.
[PubMed Abstract](#) | [Publisher Full Text](#)
 21. Giardine B, Riemer C, Hardison RC, *et al.*: **Galaxy: a plat-form for interactive large-scale genome analysis.** *Genome Res*. 2005; **15**(10): 1451–1455.
[PubMed Abstract](#) | [Publisher Full Text](#) | [Free Full Text](#)
 22. **Genomespace.**
[Reference Source](#)
 23. **Biojs project api documentation.**
[Reference Source](#)
 24. **Datatables.**
[Reference Source](#)
 25. Kalderimis A, Micklem G: **im-tables-biojs.** *Zenodo*. 2014.
[Data Source](#)

Open Peer Review

Current Peer Review Status:

Version 1

Reviewer Report 18 February 2014

<https://doi.org/10.5256/f1000research.3700.r3676>

© 2014 Ceol A. This is an open access peer review report distributed under the terms of the [Creative Commons Attribution License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Arnaud Ceol

Computational Research Unit of the Center for Genomics, Istituto Italiano di Tecnologia (IIT@SEMM), Milan, Italy

Reusable libraries for building web applications are very welcome and allow interested groups to build modern, dynamic web tools without losing time to learning new technologies and having to study data access from various different resources.

Here, the authors propose a library to query all InterMine repositories. The result is a powerful tool: it is not only the web developers who can use this library to access in theory all the InterMine repositories, but users too can build and update queries as well as table directly from the web page by selecting graphically all fields and columns.

The manuscript provides some useful examples of possible queries. Starting from the example at the address <http://zenodo.org/record/8301> (selected from the Software Availability section), it is easy to have a running page that can be modified.

Nevertheless, starting a page from scratch is not as straight forward as it could be because the availability links and the installation description are confusing. It is not clear what the meaning of the data availability links provided are (there are three links in the abstract under the heading "availability" and two in the Software Availability section). The authors may want to add a short description to each so the user can redirect to the one he/she is interested in. Besides, in the Installation section of the manuscript it is written that the *"last resource contains the definition of the InterMine.Table BioJS component. As it is not available from a reliable third party source, it currently needs to be downloaded from the BioJS registry, and hosted locally"*. This is where the confusion is: what should be downloaded and from where? I didn't find references to the InterMine tables in Zenodo or from the GitHub repository. In the example (Zenodo), it looks like the code is actually inserted in the index.js file.

One of the advantages of the library proposed here is that it is based on the InterMine repositories. In the manuscript the authors provide the example of the FlyMine query, but the component should also be able to query all the others InterMine repositories (I was able to find

the updated code on GitHub reference to the worm repository). From what I understand from the author's comments is that a separate article should soon be available and will describe the web service implemented by the InterMine repositories and which are used by this BioJS library. It makes it difficult to understand what queries can be done (i.e. what are the tables and fields that can be queried). Nevertheless, this can be inferred by using the example provided (using the manage columns and adding filter buttons on the example page). The authors may want to provide a supplementary table with the address of the web services that can be queried (or a link to another registry if it exists).

In the Supplementary material A section, the last "<script/>" should be </script>.

To summarize, I believe that this library is useful and works well, and that after a few clarifications in the manuscript it will help the user integrate straightforward InterMine tables into their own webpages.

Competing Interests: No competing interests were disclosed.

I confirm that I have read this submission and believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

The benefits of publishing with F1000Research:

- Your article is published within days, with no editorial bias
- You can publish traditional articles, null/negative results, case reports, data notes and more
- The peer review process is transparent and collaborative
- Your article is indexed in PubMed after passing peer review
- Dedicated customer support at every stage

For pre-submission enquiries, contact research@f1000.com

F1000Research