# Open PHACTS Explorer

## Bringing the web to the semantic web

Ian Dunlop[1], Rishi Ramgolam[2], Stephen Pettifer[1], Alasdair JG Gray[3], James Eales[4], Carole Goble[1], Jan Velterop[2]

[1] University of Manchester (ian.dunlop, steve.pettifer, carole.goble@manchester.ac.uk)
[2] AQnowledge Ltd. (ramgolam, velterop@aqnowledge.com)
[3] Heriot-Watt University (A.J.G.Gray@hw.ac.uk)
[4] University of Leicester (jme25@leicester.ac.uk)

**Abstract.** The Open PHACTS Explorer is a web application that supports drug discovery via the Open PHACTS API without requiring knowledge of SPARQL or the RDF data being searched. It provides a UI layer on top of the Open PHACTS linked data cache and also provides a javascript library to facilitate easy access to the Open PHACTS API.

## 1.1 Introduction

A common criticism of the semantic web and linked data is the lack of end-user tools available [1]. The Open PHACTS [2] project provides a linked data cache over a number of pharmacology datasets with access via an API rather than the more traditional SPARQL endpoint. This enables novice linked data users to focus on finding data rather writing queries. The Explorer is a web application which serves both as a demonstrator of the API and a pharmacology discovery tool. It was influenced by the original Open PHACTS demonstrator system [3].

The Open PHACTS Explorer[1] has the look and feel of a modern web application and is driven by the familiar web paradigm of searching for results and clicking on hyperlinks. The Explorer uses the Ember JS[2] Model-View-Controller (MVC) framework in conjunction with the Handlebars[3] web page template language. Handlebars is built on top of HTML which allows CSS to be used for styling. This reduces the learning curve for developers since they already have the core skills required. An EmberJS web application runs entirely within the users browser with a full MVC stack that reduces latency and the need for client/server round trips to fetch pages and data. Ruby on Rails[4] (RoR) is used to start the application and serve the static elements like

---

[1] https://github.com/openphacts/explorer2 & http://jupiter.cs.man.ac.uk
[2] http://emberjs.com/
[3] http://handlebarsjs.com/
[4] http://rubyonrails.org/

Javascript & CSS. RoR is also used to handle long running calls like the TSV download since experience with other applications revealed that this was not something currently possible in most browsers.

## 1.2    Test driven development

When developing the Explorer we took the opportunity to move the API access code into a separate library, OPS.js[5], which not only allows its reuse in a number of projects but also allows us to use test-driven development[6]. In prototype versions the API access code had been an integral part of the views. The Open PHACTS API documentation[7] specifies what response elements are expected/optional and the ops.js integration tests use this as a contract. When the API changes we can run the tests to see which parts of the OPS.js library no longer work as expected. Jasmine[8] handles the test cases and phantomjs[9] allows us to run the tests in a headless browser giving the possibility of using Continuous Integration[10].
OPS.js takes advantage of the backend APIs use of Cross Origin Resource Sharing[11] (CORS) that enables the browser to communicate directly with the API rather than use a technique like JSONP[12] that has caused issues with caching responses in the Open PHACTS system. To use the API requires a user and application specific key. It is these keys that enable the CORS mechanism since providing a set of keys tells the system that a Cross Origin Request from this browser is allowed and that the XHR network call from the users computer to the API is acceptable.

## 1.3    Consistency with the web – path of least resistance

The Explorer uses as many of the artifacts of a web application that users will be familiar with as possible. Navigation through pages is via hyperlinks starting from the search results page. Consistent URLs are used to navigate through its pages. If a user has an identifier for a compound eg abcd then they know that a URL of the form /compounds/abcd and /compounds/abcd/pharmacology will also successfully navigate to a page. Identifiers are currently UUIDs minted by the concept wiki[13] system. The Explorer has a modern look and feel using bootstrap[14] and adopts modern web para-

5    https://github.com/openphacts/ops.js
6    http://en.wikipedia.org/wiki/Test-driven_development
7    https://dev.openphacts.org/docs
8    http://pivotal.github.io/jasmine/
9    http://phantomjs.org/
10   http://en.wikipedia.org/wiki/Continuous_integration
11   http://en.wikipedia.org/wiki/Cross-origin_resource_sharing
12   http://en.wikipedia.org/wiki/JSONP
13   http://ops.conceptwiki.org/
14   http://getbootstrap.com/

digms like 'infinite' scrolling where instead of a user clicking on a next or page number button the act of scrolling to the bottom of the current list loads in the next set of results.

## 1.4    OPS.js

The Open PHACTS API abstracts the data layer and returns formatted results as specified in the API documents in JSON/XML/TSV etc format. There are two aspects, the network call and the parse method. The network method determines whether it is a success or failure based on the HTTP response code and returns the raw result if it is a success. The response can contain deep nesting and OPS.js flattens the results to as high a level as possible so that the application receives a representation of the model rather than the data. Each method call follows a standard pattern so that developers get used to the code and find its integration as easy as possible. jQuery[15] is used to handle the cross platform browser issues.

## 1.5    Using the Open PHACTS Explorer

A user begins by searching for 'something', eg aspirin. The search box autocompletes and suggests possibilities, the user selects one from the list if they want or continues with their own text and hits enter to search. The search uses the free text search call in the API to return all compounds and targets that the text might match. The results are formatted with an info box for each with the name of the entity and an image. The name is a hyperlink and clicking it navigates to the page for that result. Scrolling down the page of results will automatically load more if available.



---

15        http://jquery.com/
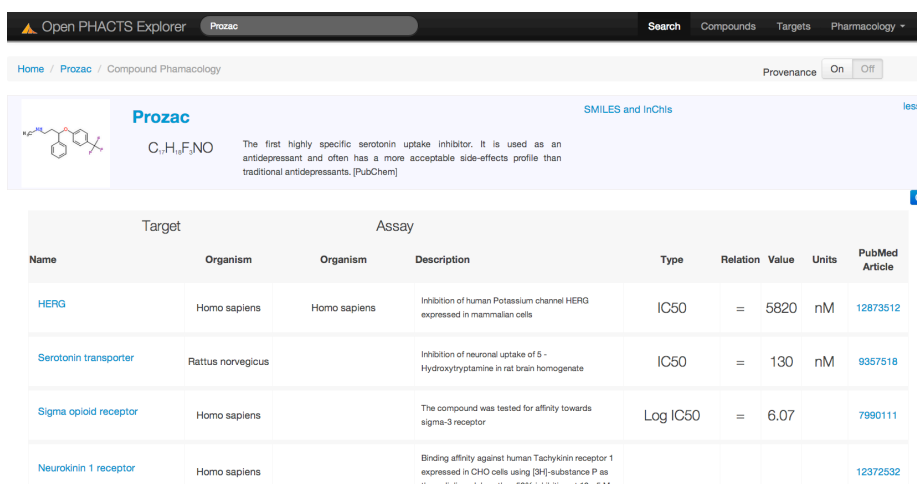
**Fig. 1.** Search results for Prozac

All pages can be linked to and bookmarked so you know that going to the same page tomorrow will give the same result. The layout tries to be as user friendly as possible to make readability easy and fast. Maximise/minimise is used wherever it makes sense to so that screen real estate can be used to its best extent. The bootstrap CSS framework was chosen due to its optimisation for mobile which allows the same site to be tailored for desktop or mobile use since tablets are increasingly being used to conduct science. The page for a search result contains further links to the data sources for the information that allows provenance to be quickly assessed. A hyperlink from the individual result page takes you to pharmacology where the first 50 results will be automatically loaded. As you scroll down the pharmacology results the banner at the top will be minimised so that screen usage is maximised and clutter reduced.



**Fig. 2.** Pharmacology data for Prozac

As you get to the bottom of the page the next 50 results are loaded in seamlessly using the infinite scrolling technique. No back/next results buttons are required.

### References

1. David Karger: The Semantic Web for End Users. ESWC 2013
2. Antony J. Williams, Lee Harland, Paul Groth, Stephen Pettifer, Christine Chichester,Egon L. Willighagen, Chris T. Evelo, Niklas Blomberg, Gerhard Ecker, Carole Goble, Barend Mons: Open PHACTS: semantic interoperability for drug discovery. Drug Discovery Today, Volume 17, Issues 21-22: http://dx.doi.org/10.1016/j.drudis.2012.05.016
3. Gray, Alasdair JG, et al. "The Pharmacology Workspace: A Platform for Drug Discovery." *ICBO*. 2012.