Davrods enhancements as part of the Grassroots Infrastructure

Simon Tyrrell
The Earlham Institute
Norwich Research Park,
Norwich, NR4 7UZ, UK
simon.tyrrell@earlham.ac.uk

Xingdong Bian
The Earlham Institute
Norwich Research Park,
Norwich, NR4 7UZ, UK
xingdong.bian
@earlham.ac.uk

Robert P. Davey
The Earlham Institute
Norwich Research Park,
Norwich, NR4 7UZ, UK
robert.davey@earlham.ac.uk

ABSTRACT

This paper explains the enhancements we have made to the Davrods Apache module to expose a range of iRODS functionality that was previously unavailable, and configuration improvements to allow the default interface to be made more user-friendly. We have made changes to Davrods so that iRODS can be used as the storage mechanism for public facing websites without the need for users to be authenticated, making it easier to produce web-based data repository interfaces. We provide code to expose iRODS metadata as cross-referencing links between data objects and collections. We also describe a REST API that has been added for metadata functionality within iRODS to facilitate metadata manipulation by end users with the supplied client-side code from within their web browsers or from other web services.

Keywords

iRODS, Apache httpd web server, infrastructure, web client, RESTful web service

Introduction

The Grassroots[1] Infrastructure project aims to create an easily-deployable suite of computing middleware tools to help users and developers gain access to scientific data infrastructure that can easily be interconnected.

With the data-generative approaches that are increasingly common in modern life science research, it is vital that the data and metadata produced by these efforts can be shared and reused. The Grassroots Infrastructure project wraps up industry-standard software tools along with our own custom open-source software tools to give a consistent API that can be federated with others in terms of both data and services. This means institutions and groups can deploy a simple lightweight software suite, locally or as a virtual machine, to expose institutional data, connect up any existing data services, and federate their instance of Grassroots with other remote instances.

One of the major aims of the Grassroots Infrastructure is to allow users to share their wheat data, although it is by no means organism-specific, as easily and seamlessly as possible. For data storage, we use the iRODS[2] data grid system that gives users access to potentially differing file systems and data resources through a single data abstraction layer. Users are able to carry out typical filesystem actions as normal, such as creating files and directories and maintaining permissions, but with extra features such as distributed storage viewable across different institutions and the ability to add extensive metadata to files and directories.

iRODS ships with command-line clients to provide access to data storage managed by the platform, and many Application Programming Interfaces (APIs) exist in a variety of languages to support programmatic development such as PyRods[3] and Jargon[4]. An Apache httpd[5] module based on the WebDAV protocol, Davrods[6], exists to allow access to iRODS data stores using standard WebDAV commands. This project supports much of the basic functionality of a web-based data dissemination stack, but there were a set of missing features of Davrods that we have developed that can improve data searching, as well as the general user experience.

iRODS UGM 2017 June 13-15, 2017, Utrecht, Netherlands

Themed listings

The standard web pages produced by Davrods resemble the basic directory listings produced by Apache. Whilst simple and effective, the functionality of these pages is lacking, and there is little configuration ability to make their look and feel customisable. Therefore, using the concepts from the autoindex module[7], we have developed a mechanism to insert themes into Davrods listing pages using typical HTML and Cascading Style Sheets (CSS) elements. Three points in the web page were identified to allow the insertion of custom HTML chunks: the head section of the web page and the sections before and after the iRODS directory listings. These HTML chunks can either be set as strings in the Apache configuration file or point to separate files on disk to allow for easy modification. Any changes made to the files are instantly available for all subsequent requests whereas any changes to the string-based configuration require a restart of the Apache server. Additionally, the columns of the listings table have been marked up with consistent CSS classes to allow for easy customisation by server administrators when developing CSS for use on their own project or institution pages. For each of its generated pages, Davrods includes a link to the parent collection as a form of breadcrumbs. However if the iRODS instance has a multi-level hierarchy in its iRODS zone this can become unwieldy as a user can only travel up one level in the hierarchy at a time. We replaced this single level breadcrumb with a navigation element containing the full breadcrumb chain for the set of parent collections up to the root collection, thus resembling the navigation concept commonly used on many websites.

The autoindex module also includes the ability to specify default icons for various arbitrary data types, typically denoted by file extension. We have developed this within Davrods so that custom icons can be used for the iRODS data objects and collections. This can be further refined and default icons can be specified for data objects with unknown file types. For example, to use the icon stored within the image file at `/davrods_files/images/picture` for PNG, GIF and JPEG images, the directive would be:

```
DavRodsAddIcon /davrods_files/images/picture .png .gif .jpg .jpeg
```

An example screenshot of various parts of the theming functionality is shown in figure 1.

Public access

The iRODS data workflow is based around the concept that users log in to see the data that they have permissions to access. Often for public websites serving open access data, there is a desire to give full read access to browse a list of files and directories without the need for dedicated login credentials. We have added the ability to have a default user to be specified within the Apache server configuration, that would be used to log into the iRODS system without the need for any user intervention. For example, to specify that the iRODS user called public_user, with a password of anonymous, is used, the following configuration would be used:

```
AuthType None
Require all granted
DavRodsDefaultUsername public_user
DavRodsDefaultPassword anonymous
```

In effect, this creates a public-facing website using the data stored in an iRODS storage system that is indistinguishable from a site generated from a regular directory on the local filesystem.

Metadata

One of the major benefits of iRODS is its ability to add metadata as attribute-value pairs to any data objects or collections stored within it. Previously with Davrods, any metadata held in an iRODS instance was not exposed. We therefore provide Davrods functionality to make metadata both viewable and editable, as well as a REST API interface to allow programmatic interrogation and modification of iRODS metadata. For giving read-only access to the metadata, the configuration directive `DavRodsHTMLMetadata` is used and it takes one of the following values:

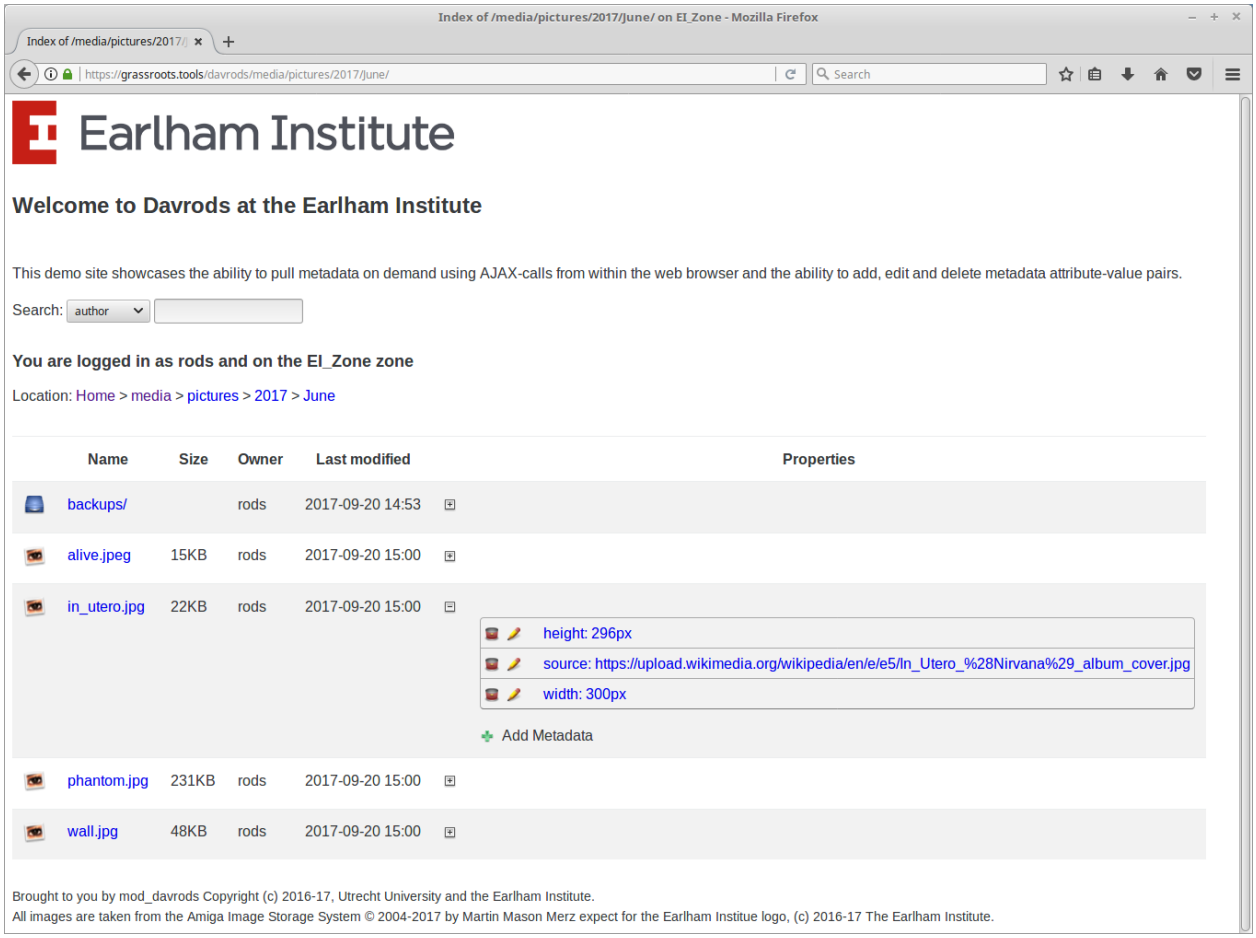

Figure 1. The themed interface showing the breadcrumbs, icons and HTML sections that can be configured

- **off**: The metadata display is disabled.
- **full**: All of the metadata is displayed for each data object and collection.
- **on_demand**: None of the metadata is initially included with the HTML pages sent by Davrods. Instead it can be delivered upon demand and inserted into the web page via AJAX requests from when the user clicks on the appropriate link associated with a given data object or collection.

Searching and linking

Each of the metadata attribute-value pairs are exposed as hyperlinks giving users a straightforward and standard method to find data objects or collections also containing the same pair. Additionally there is a general search mechanism provided to search across the entire metadata collection and this is available as a form within each page that Davrods delivers.

REST API

As well as adding a read-only view of the metadata, the ability to add, edit or delete metadata from within in a web browser for users with the appropriate permissions is also provided. For this feature to be active, the following

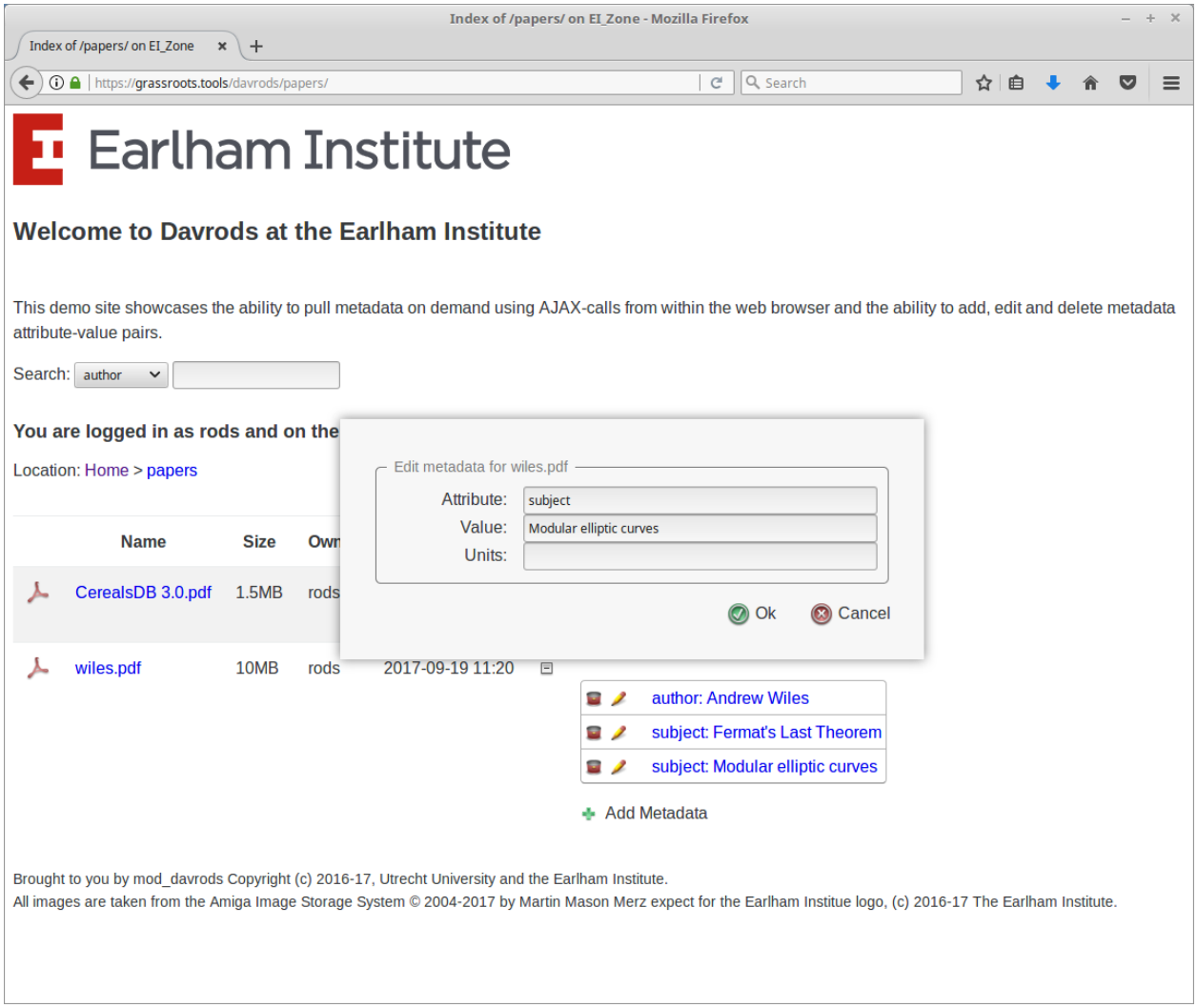

Figure 2. The metadata editor

configuration directive needs to be set:

```
DavRodsHTMLMetadataEditable true
```

The manipulation of metadata is facilitated by a REpresentational State Transfer (REST) API for querying and manipulating the iRODS metadata. The base URL for this API is at `/api/metadata` although the prefix can be changed by altering the `DavRodsAPIPath` parameter in the Apache configuration. The REST API contains the following functions:

- **get**: Retrieve all of the associated metadata for an iRODS item. It takes a single parameter, *id*, which is the iRODS id of a data object or collection to be queried. For example, to get the metadata for a data object with the id of 1.10021, the URL to call would be:

  ```
  api/metadata/get?id=1.10021
  ```

- **search**: Retrieve a list of all data objects and collections that have a given metadata attribute-value pair. It takes two parameters: *key*, the attribute to search for, and *value* which specifies the metadata value. There is a third optional parameter, *units*, for specifying the units that the metadata attribute-value pair must also have. So, to search for all of the data objects and collections that have an attribute called volume with a value of 11:

  ```
  api/metadata/search?key=volume&value=11
  ```

- **add**: Add a metadata attribute-value pair to a data object or collection. It takes three parameters: *id*, the iRODS id of a data object or collection, *key*, the attribute to add, and *value* which specifies the metadata value to be added. As with the `search` call listed above, there is a fourth optional parameter, *units*, for specifying the units that the metadata attribute-value pair will have. So, to add an attribute called volume with a value of 11 to a data object with the id of 1.10021:

  ```
  api/metadata/add?id=1.10021&key=volume&value=11
  ```

- **edit**: Edit a metadata attribute-value pair for a data object of collection and replacing one or more of its attributes, values, or units. It takes the following required parameters: *id*, the iRODS id of a data object or collection, *key*, the attribute to edit, and *value* which specifies the metadata value to edit. There is an optional parameter, *units* for specifying the units that the metadata attribute-value pair must also have to match. There must also be one or more of the following parameters to specify how the metadata will be altered: *new_key*, which is for specifying the new name for the attribute, *new_value*, for specifying the new metadata value and *new_units* for specifying the units that the metadata attribute-value pair will now have. So, to edit an attribute called *volume* with a value of 11 and units of decibels, for a data object with the id of 1.10021 and give it a new value of 8 and units of litres:

  ```
  api/metadata/edit?id=1.10021&key=volume&value=11&units=decibels&new_value=8&new_units=litres
  ```

- **delete**: Delete a metadata attribute-value pair from a data object of collection. It takes three parameters: *id*, which is the iRODS id of the data object or collection to delete the metadata from, *key*, which is the attribute to delete for and, *value*, which specifies the metadata value to delete. As before, there is an optional parameter, *units* for specifying the units that the metadata attribute-value pair must also have to be deleted. So to delete an attribute called volume with a value of 11 and units of decibels from a data object with the id of 1.10021:

  ```
  api/metadata/delete?id=1.10021&key=volume&value=11&units=decibels
  ```

We have included a set of JavaScript functions to allow a davrods administrator to easily give a user to access each of these API functions from within the browser. An example screenshot of the editor is shown in figure 2.

Future Work

Currently the REST API functions all return HTML fragments. However, in future, we would like to develop the possibility of specifying other datatypes such as JSON fragments to allow for integration and automation with other consuming web services.

Acknowledgements

The Grassroots project is strategically funded through the BBSRC cross-institute Designing Future Wheat programme grant, BB/P016855/1, and aims to develop a lightweight reusable set of open source software tools to allow researchers to share and federate life science datasets.

Availability

The source code is available at https://github.com/billyfish/davrods.

REFERENCES

[1] Grassroots Infrastructure, `https://grassroots.tools`, Visited last on 06.06.2017

[2] Hao Xu, Terrell Russell, Jason Coposky, Arcot Rajasekar, Reagan Moore, Antoine de Torcy, Michael Wan, Wayne Shroeder, Sheau-Yen Chen: iRODS Primer 2: Integrated Rule-Oriented Data System. Synthesis Lectures on Information Concepts, Retrieval, and Services, Morgan Claypool (2017)

[3] Python iRODS Client (PRC) `https://github.com/irods/python-irodsclient`, Visited last on 06.09.2017

[4] Jargon, `https://github.com/DICE-UNC/jargon`, Visited last on 06.09.2017

[5] The Apache HTTP Server Project, `http://httpd.apache.org/`, Visited last on 06.06.2017

[6] Ton Smeele, Chris Smeele: Davrods, An Apache WebDAV interface to iRODS. iRODS UGM 2016 proceedings, pp. 41-47 (2016)

[7] Apache Module mod_autoindex. `https://httpd.apache.org/docs/2.4/mod/mod_autoindex.html`, Visited last on 06.06.2017