Advanced Customisation: Scripting EPrints

EPrints Training Course

Taking Control: the EPrints API

- EPrints configuration files offer many opportunities for customisation and control
  - branding, workflow, controlled vocabs, authority lists, deposit types, metadata...
- EPrints API offers many more opportunities
  - the more perl-intensive configuration files
    - e.g. eprint_render.pl
  - and beyond..
    - plugins
    - command-line tools

Roadmap

- Core API
  - manipulating your data
  - accessing data collections
  - searching your data
- Scripting techniques
  - essentials – putting it all together
  - writing export plugins
  - writing screen plugins
  - writing command-line tools
  - writing CGI scripts

Part 1: Core API

About This Part of the Talk

- Light on syntax
  - `object->function(arg1, arg2)`
- Incomplete
- Designed to
  - give you a feel for the EPrints data model
  - introduce you to the most significant (and useful!) objects
    - how they relate to one another
    - their most common methods
  - act as a jumping off point for exploring

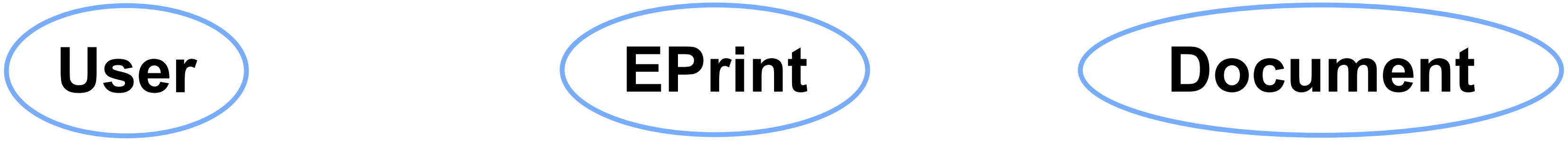
Finding Documentation

- EPrints modules have embedded documentation
- Extract it using perldoc
  - `perldoc perl_lib/EPrints/Search.pm`

Core API:
Manipulating Your Data

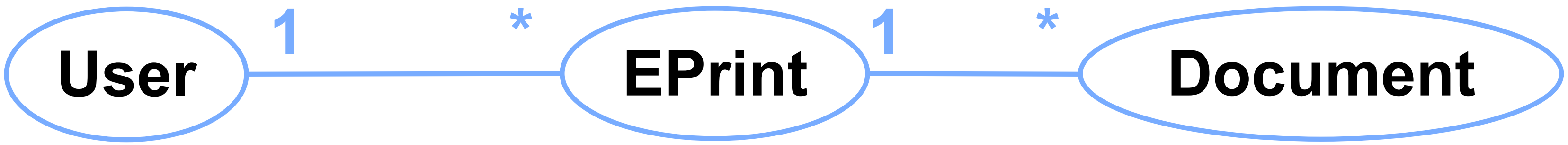
Data Model: 3 Core Objects

- EPrint
  - single deposit in the repository
- Document
  - single document attached to an EPrint
- User
  - single registered user



Data Model: Core Relationships

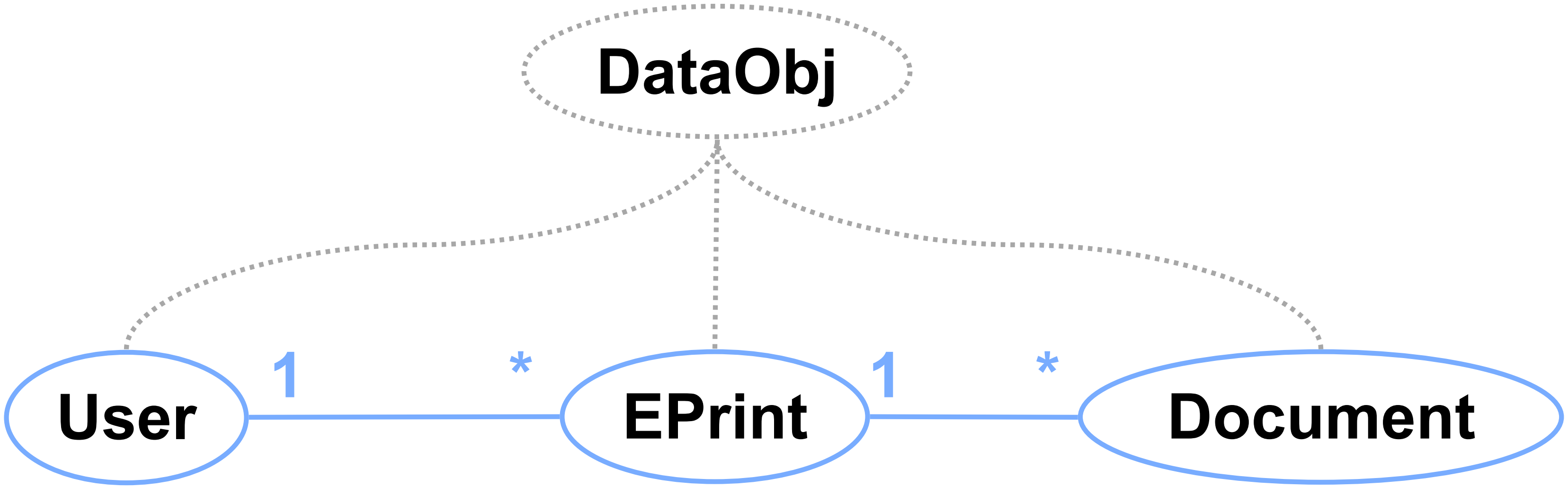
- 1 User owns (deposits) many EPrints
- 1 EPrint has many documents attached to it
- 1 Document may contain many files, but these are not part of the API
  - e.g. PDF = 1 file
  - e.g. HTML + images = many files



User 1 * EPrint 1 * Document

Data Model: DataObj

- All data objects inherit from DataObj
- Provides **common interface** to data



Accessing Data: DataObj interface

- `get_id()`
- `get_url()`
  - EPrint – abstract page
  - User – user summary page
  - Document – document download
- `get_type()`
  - EPrint – *article, book, thesis...*
  - User – *user, editor, admin*
  - Document – *pdf, html, word...*

Manipulating Data: DataObj Interface

- `get_value(fieldname)`
  - get the value of the named data field
  - `eprint->get_value( “title” )`
- `set_value(fieldname, value)`
  - set the value of the named field
  - `doc->set_value( “format”, “pdf” )`
- `is_set(fieldname)`
  - true if the named field has a value
  - `user->is_set( “email” )`

Manipulating Data: DataObj Interface (2)

- `commit()`
  - write any changes made to the object through to the database
  - e.g. after using `set_value`
- `remove()`
  - erase the object from the database
  - also removes any sub-objects and files
    - e.g. `eprint->remove`
    - removes EPrint and associated Documents from DB
    - removes Document files from filesystem

Getting Hold of Existing Data Objects

- `new(session, id)`
  - returns data object for an **existing** record
  - `EPrints::DataObj::EPrint->new(session, 1)`
  - `EPrints::DataObj::User->new(session, 1)`
  - `EPrints::DataObj::Document->new(session, 1)`
- User object has extra options
  - `user_with_email(session, email)`
  - `user_with_username(session, username)`

Creating New Data Objects

- Slightly different for each data object
- EPrint
  - `create(session, dataset, data)`
- User
  - `create(session, user_type)`
- Document
  - `create(session, eprint)`

Specific Methods

- Each data object also has specific methods for manipulating their data

EPrint Methods

- `get_user()`
  - get a **User** object representing the user to whom the EPrint belongs
- `get_all_documents()`
  - get a list of all the **Document** objects associated with the EPrint
- `generate_static()`
  - generate the static abstract page for the eprint
    - useful when you’ve modified the eprint values!
  - in a multi-language archive this will generate a page in each language

User Methods

- `get_eprints(dataset)`
  - get a list of EPrints owned by the user
- `mail(subject, message)`
  - send an email to the user

Document Methods

- `get_eprint()`
  - get the EPrint object the document is associated with
- `local_path()`
  - get the full path of the directory where the document is stored in the filesystem
- `files()`
  - get a list of (filename, file size) pairs

Document Methods: Main File

- `get_main()`
- `set_main(main_file)`
  - get/set the **main** file for the document
  - this is the file that gets linked to
  - in majority of cases, Document will have 1 file
    - e.g. PDF
  - but there may be some cases where a Document has many file
    - e.g. HTML document = .html files, images, stylesheets
      - set main to top level index.html

Document Methods: Adding Files

- `add_file(file, filename)`
- `upload(filehandle, filename)`
  - both add a file to the document
  - `add_file` uses full path to file
  - `upload` uses file handle
  - in both cases the document will be named `filename`

Document Methods: Adding Files (2)

- `upload_url(url)`
  - grab file(s) from given URL
  - in the case of HTML, only relative links will be followed
- `add_archive(file, format)`
  - add files from a .zip or .tar.gz file
- `remove_file(filename)`
  - remove the named file

Other Data Objects

- Subject
  - a node in the subjects tree
- SavedSearch
  - a saved search associated with a User
- History
  - an event that took place on another data object
  - e.g. change to eprint metadata
- Access
  - a Web access to an object
  - e.g. document download
- Request
  - a request for a (restricted) document
- Explore these using `perldoc`

Core API:
Accessing Data Collections

Accessing Data Collections

- We’ve looked at **individual** data objects
  - but a repository holds many eprints and documents and has many registered users
- 2 key ways to manipulate data objects collectively:
  - built-in **datasets**
    - large fixed sets of data objects
  - by **searching** the repository
    - set of data objects matching specific criteria

Datasets

- All data objects in the repository are part of a collection called a **dataset**
- 3 core datasets:
  - `eprint`
    - all eprints
  - `user`
    - all registered users
  - `document`
    - all documents

Datasets (2)

- Also 4 subsets within `eprint` dataset which collect eprints in same **state**
  - `archive`
    - all eprints in live archive
  - `inbox`
    - all eprints which users are still working on
  - `buffer`
    - all eprints submitted for editorial review
  - `deletion`
    - all eprints retired from live archive

The DataSet Object

- Gives access to all the data objects in a particular dataset
- Also
  - tells us which data fields apply to that dataset
  - recall `get_value` and `set_value` methods
  - a repository’s metadata is configurable so this gives us a way to find out:
    - which fields are available in a particular repository
    - the **properties** of individual fields

Accessing DataSets

- `count(session)`
  - get the number of items in the dataset
- `get_item_ids(session)`
  - get the IDs of the objects in the dataset
- `map(function, args)`
  - apply function to each object in the dataset
  - function is called with args:
    - `(session, dataset, dataobj, args)`

Fields in a DataSet

- `has_field(fieldname)`
  - true if the dataset has a field of that name
- `get_field(fieldname)`
  - get a MetaField object describing the named field
- `get_fields()`
  - get list of MetaField objects describing all fields in the dataset

Datasets and MetaFields

- A MetaField
  - is a single field in a dataset
  - tells us properties of the field
    - `get_property(name)`
    - `set_property(name, value)`
    - e.g. name, type, input_rows, maxlength, multiple...
  - but not the field value
    - the value is specific to the individual data object
      - e.g. `eprint->get_value("title")`

Core API:
Searching the Repository

Searching the Repository

- The Search object allows us to search datasets for data objects matching specific criteria
- Provides access to the results

Starting a New Search

- `new(options)`
  - create a new search expression
  - must specify which dataset to search in

  -
    ```
    search = new Search(
     session => session,
     dataset => dataset,
     custom_order => “title” )
    ```
  - many other options can be specified
    - explore with `perldoc`

Adding Search Fields

- `add_field(metafield, value)`
  - add a new search field with the given value (search text) to the search expression
  - add as many fields as you like to the search criteria

Adding Search Fields: Example

- Example: full text search
-

```
search->add_field(
 dataset->get_field(“title”),
 “routing”,
 “IN”,
 “ALL” )
```

Adding Search Fields: Example (2)

- Example: full text search which matches word in title **or** abstract

```
search->add_field(
  [ dataset->get_field(“title”),
 dataset->get_field(“abstract”)
   ],
 “routing”,
 “IN”,
 “ALL” )
```

Adding Search Fields

- Example: date search
- `search->add_field(`

```
 dataset->get_field(“date”),
 “2000-2004”,
 “EQ”,
 “ALL” )
```

Processing Search Results

- Carry out a search using:
  - `list = search->perform_search()`
- Returns a List object which gives access to search results

The List Object

- Any ordered collection of data objects
  - usually the results of a search

Processing Lists

- `count()`
  - get the number of results
- `get_ids(offset, count)`
- `get_records(offset, count)`
  - get an array if data objects, or just their ids
  - optionally specify a range using count and offset
- `map(function, args)`
  - apply the function to each data object in the list

Manipulating Lists

- `newlist = list->reorder( neworder )`
- `newlist = list->union( list2 )`
- `newlist = list->intersect( list2 )`
- `newlist = list->remainder( list2 )`