Accessing Data In EPrints

- EPrints Object Model Basics
- Export Plugin Basics
- Getting Sensible Data
- Your Virtual Machine

EPrints as a word

- eprints are electronic publications (papers, lectures, videos, etc)
- EPrints is a piece of software for managing eprints.
- EPrint objects are a subclass of a DataObj that represents an eprint.
- An EPrints system contains EPrint objects which contain Document objects which contain File objects which are an interface to filesytem files (or cloud buckets) which can be downloaded and opened by applications such as Acrobat to allow a person to read a publication (eprint)!!!1!!

EPrints Object Model Basics

- All first class objects inherit from the DataObj superclass.
- A DataObj is a collection of metadata fields.
- A DataSet is a collection of DataObjs of the same class.
- EPrint Objects contain Document Objects.
- Document Objects contain File Objects (but you don't need to worry about those).

Export Plugins

- Gives you a handle on a single item or list of items to be processed.
- Use the `output_dataobj` function for concatenating an output for each object.
- Use `output_list` for processing the whole list of objects.

Plugin Housekeeping

```
package EPrints::Plugin::Export::Text;

use EPrints::Plugin::Export::TextFile;
@ISA = ( "EPrints::Plugin::Export::TextFile" );

sub new
{
	my( $class, %opts ) = @_;
	my $self = $class->SUPER::new( %opts );

	$self->{name} = "ASCII Citation";
	$self->{accept} = [ 'dataobj/eprint', 'list/eprint' ];
	$self->{visible} = "all";

	return $self;
}
```

- Set package name and superclass.
- Register with EPrints
  - Name of plugin
  - What we can export
  - Who can use this

output_dataobj

```
sub output_dataobj
{
    my( $plugin, $dataobj ) = @_;
    my $cite = $dataobj->render_citation;
    return EPrints::Utils::tree_to_utf8( $cite )."\n\n";
}
```

- Return the output for this dataobj.
- render_citation is useful, but returns a DOM object.
- EPrints::Utils::tree_to_utf8 converts a DOM object to text.

output_list example

```
sub output_list
{
        my( $plugin, %opts ) = @_;

        my $part = csv( $plugin->header_row( %opts ) );

        my $r = [];

        push @{$r}, $part;

        # list of things
        $opts{list}->map( sub {
                my( $session, $dataset, $item ) = @_;

                my $part = $plugin->output_dataobj( $item, %opts );
                push @{$r}, $part;
        } );

        return join( '', @{$r} );
}
```

Getting Data from a dataobj

```
if ($dataobj->is_set($fieldname))
{
    $val = $dataobj->value($fieldname);
}
```

- Basic data access methods
- $val will contain a scalar, a hashref, an arrayref or a dataobj ref (more on that later)

Get all data from a dataobj

```
my $ds = $dataobj->dataset;
my @fields = $ds->fields;
my %vals;
foreach my $field (@fields)
{
    my $fieldname = $field->name;
    if ($dataobj->is_set($fieldname))
    {
        $vals{$fieldname} =
          $dataobj->value($fieldname);
    }
}
```

- Dataset contains list of all available metadata fields.
- Store in a hash for easy processing.
- Now you don't need to care (as much) about EPrints.

The Shape of Data

```
my $creators =
   $dataobj->value('creators');
use Data::Dumper;
print Dumper $creators;
```

- Metadata fields may not be simple (i.e. scalar)
- Data::Dumper is your friend.

```
$VAR1 = [
          {
            'name' => {
                        'lineage' => '',
                        'given' => 'Noura',
                        'honourific' => '',
                        'family' => 'Abbas'
                      },
            'id' => '10363'
          },
          {
            'name' => {
                        'lineage' => '',
                        'given' => 'Gary',
                        'honourific' => '',
                        'family' => 'Wills'
                      },
            'id' => '395'
          }
        ];
```

The Shape of Data

- Two types of field
  - simple
  - compound
- ...which may or may not be multiple
- The perl ref function is your friend.

Field Type	Data Returned
simple	scalar (sometimes)
compound	hashref
multiple simple	arrayref of scalars
multiple compound	arrayref of hashrefs

The Shape of Data

- The creators field is multiple and compound.
- Array because it's multiple.
- Hash contains a key for every subfield.
- The name subfield is of type 'name', which has structure!

```
$VAR1 = [
          {
            'name' => {
                        'lineage' => '',
                        'given' => 'Noura',
                        'honourific' => '',
                        'family' => 'Abbas'
                      },
            'id' => '10363'
          },
          {
            'name' => {
                        'lineage' => '',
                        'given' => 'Gary',
                        'honourific' => '',
                        'family' => 'Wills'
                      },
            'id' => '395'
          }
        ];
```

Render Methods

- EPrints contains render methods for every metadata field.
- These return DOM objects.
- You can render subfields of compound (and multiple compound) types.

```
my $text = EPrints::Utils::tree_to_utf8(
  $dataobj->render_value('creators_name')
);
```

Documents

- Documents are first class objects stored in the documents field.
- A number of images are associated with them.
- Don't use get_value for these.

```
my @docs = $dataobj->get_all_documents;
my $size = 'medium'; #could also be 'small' or 'preview'
my $icon_url = $doc->icon_url(size => $size);
my $download_url = $doc->get_url;
my $path_on_disk = $doc->local_path
```

Documentation

- See...
  - http://wiki.eprints.org
- Especially...
  - http://wiki.eprints.org/w/Accessing_Metdata_Fields

Your VM

- Everyone gets a personal VM
- EPrints installed, containing the last 3.5 years of eprints.ecs.soton.ac.uk publication records.
- EPrints is installed at:
  - /usr/share/eprints3
- Export plugins are at:
  - /usr/share/eprints3/perl_lib/EPrints/Plugin/Export
- Lab sessions Monday and Tuesday at 4pm next week.
- Check your email.