OR 2013 EPrints User Group Session: Re-rendering of Deposit History

Bin Han
Digital Repository Developer
bin.han@concordia.ca

Tomasz Neugebauer
Digital Projects & Systems Development Librarian
tomasz.neugebauer@concordia.ca

Concordia University Libraries

Re-rendering of Deposit History

Abstract

The EPrints history module is a useful feature for digital preservation; it displays the revision files that are in XML format to visualize metadata change history. However, when an object is heavily revised, the readability of the information presented by the history module is a challenge. In 2012, we proposed a solution[1] to present only the changed fields related to embargo information. In this current proposal, a new method is implemented to keep track of changes to all metadata fields (eprint attributes) in a summary view by re-rendering the information in a concise and readable format.

Introduction

Digital preservation is defined as "the series of management policies and activities necessary to ensure the enduring usability, authenticity, discoverability and accessibility of content over the very long term."[2] The EPrints software platform has introduced a number of features that help support the preservation of digital objects, including the history module which keeps track of changes to metadata in an audit trail. The history module contributes to authenticity by tracking metadata changes.

Problem Definition

The repository editorial staff found it difficult to understand the colour coded XML display of the history module due to its format and length. In 2012, we deployed a function that tracks the changes on an embargo field, and displays only the information relevant to the modifications of that field, in a tabular format. The solution was implemented by creating a compound and multiple field that contains the critical information, reflecting which field was changed (from what to what), at which time, and by whom. The code in *eprint_fields_automatic.pl* compared related fields (security type, embargo date, document identification), then recorded information in the newly created history field. This approach works well for embargo information; however, it does not scale up well as a solution to track all the

[1] Bin Han, Tomasz Neugebauer, *Summary View for EPrint History.* 7th International Conference on Open Repositories, Edinburgh, Scotland, UK. July 9th – 13th, 2012.
[2] Guenther, R. & Kirchhoff A. NISO Webinar: Metadata for Preservation: A Digital Object's Best Friend. February 13, 2013.

metadata fields. Specifically, each metadata field is a column or table in the database; creating a separate field to store previous states would introduce too much data redundancy.

The problem of the default history module is that it provides too much redundant information visually, which increases the user's cognitive workload; however, it does contain all the required raw data. The question becomes: how to re-render the metadata history information, which is preserved in the revision files, by using the principle that only relevant modified fields will be displayed.

To address this, a three-step solution is offered:

1. Syntactically, compare XML revision files to address text differences
2. Semantically, parse the text difference to identify the alternated fields based on EPrints APIs
3. Visually, render the history log within EPrints Screen Plugin framework

Design and Implementation

Three sets of tools are applied to carry out the solution: the XML::SemanticDiff module, EPrints APIs, and the Screen Plugin framework. The architecture diagram is show as Figure 1.

1. The XML::SemanticDiff module

Note that XML::SemanticDiff is a Perl extension; it provides a way to compare the contents and structure of two XML documents. By default, it returns a list of hash references where each hash reference describes a single difference between the two documents. The main reason for choosing this module is that it offers event handlers, which can be overridden, so that the user can customize what to do with the differences. A custom handler class *Render_History* is created with the event handlers:

```
sub rogue_element {}     #captures the newly added elements
sub missing_element {}  #captures the removed elements
sub element_value()      #captures the modified elements
```

By passing in the revision files, the *SemanticDiff* helps us identify the changes in the particular revision.

```
my $render_history = Render_History->new();
my $diff =  XML::SemanticDiff->new(diffhandler => $render_history,  keepdata => 1);
my @results = $diff->compare( $pre_path, $cur_path );
```

2. EPrints APIs

By default, *SemanticDiff* module returns pairs of information; the XPath of the difference, and the value, the metafield information can be extracted from XPath. For example, */eprint[1]/documents[1]/document[3]/security[1]* reflects that the *security* field of *document* object has been touched. By using EPrints APIs, we can display the phrase name of the field.

```
my $metafield = $dataset->field($field);

if (defined $metafield) {

        my $phrasename = $metafield->{confid}."_fieldname_".$metafield->{name};

        my $meta_name = $repo->phrase($phrasename);

}
```

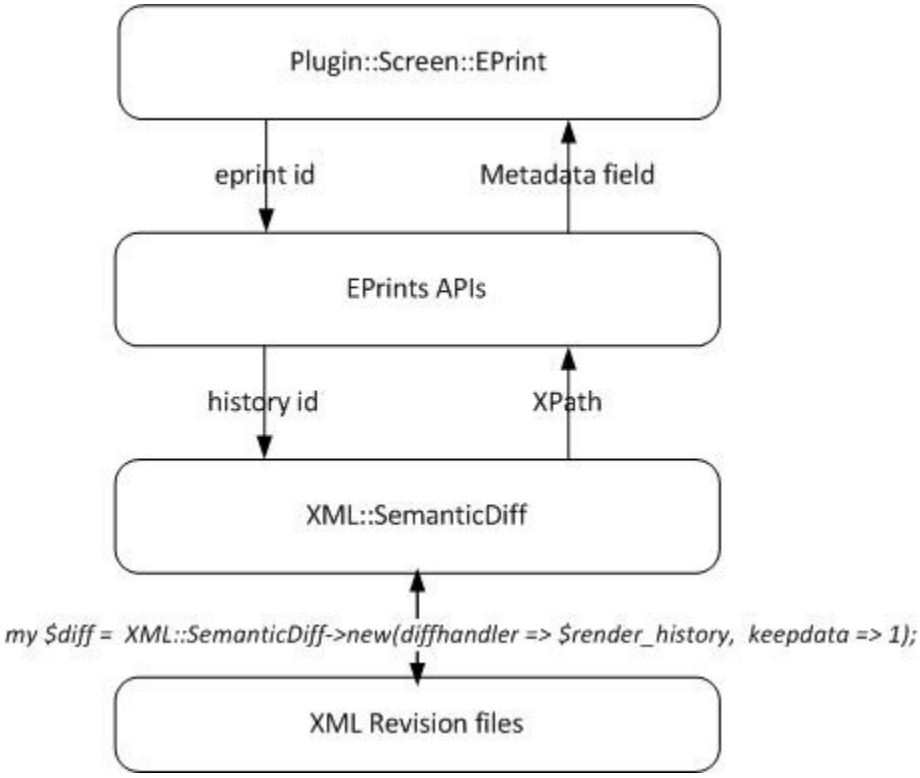

Figure 1 Implementation Architecture

3. Screen Plugin

The new function eventually fits into the EPrints Screen Plugin framework. The metadata module is located at: */opt/eprints3/perl_lib/EPrints/DataObj/MetaData.pm*, inherits *Screen::EPrints*.

```
package EPrints::Plugin::Screen::EPrint::Metadata;

our @ISA = ( 'EPrints::Plugin::Screen::EPrint' );
```

Figure shows how the EPrints default history module highlights changes introduced in revision 20 and 21; Figure 3 shows how the newly implemented audit trail renders the changing log for the same deposit. The later solution is much more compact and does not require users to do any comparison by themselves.

Future Work

By reusing the revision structure of EPrints, we have implemented a novel method of rendering the deposit history. Currently, it is deployed on EPrints 3.2.4, but after the upcoming system upgrade, it will be further tested on EPrints 3.3.11, and may then be packaged as an app in the Bazaar Store.

Conclusion

Metadata preservation is a key element for digital repositories. Maintaining metadata history is critical to preservation. We propose a more user-friendly method for displaying metadata record history.

Figure 3 Metadata Audit Trail

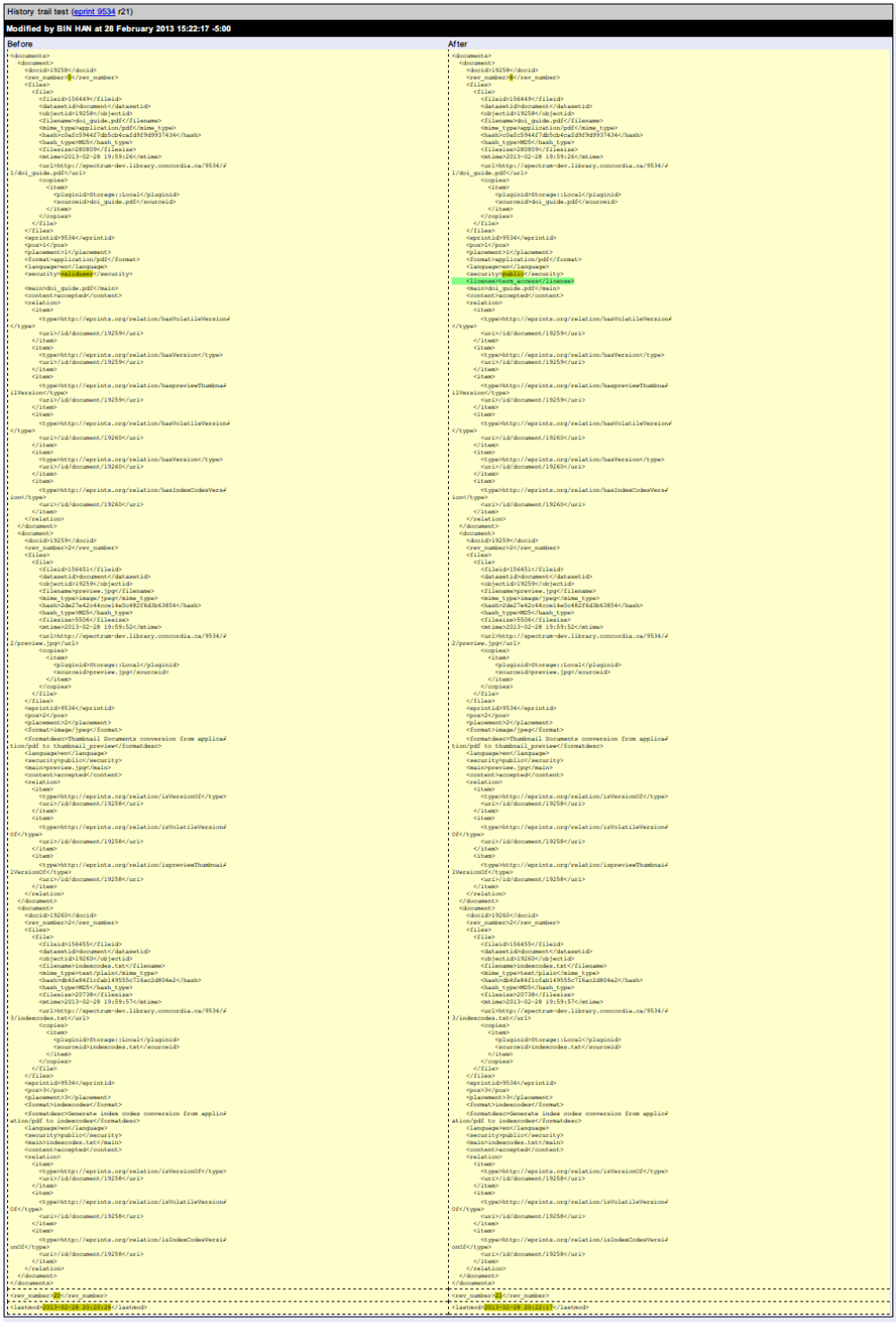

Figure 2 EPrints History Module