BCO-DMO
Biological and Chemical Oceanography Data Management Office

Legacy2Drupal

AGU 2011
December 7, 2011

Tobias T. Work[1], Andrew R. Maffei[3], Cynthia L. Chandler[2], M. Dickson Allison[1], Robert C. Groman[1], Steve L. Gegg[2]
1~Biology Department, 2~Marine Chemistry and Geochemistry Department, 3~Computer and Info Services, Woods Hole Oceanographic Institution, Woods Hole MA 02543

Abstract

# IN31B-1436

Content Management Systems (CMSs) such as Drupal provide powerful features that can be of use to oceanographic (and other geo-science) data managers. However, in many instances, geo-science data management offices have already designed and implemented customized schemas for their metadata. The NSF funded Biological Chemical and Biological Data Management Office (BCO-DMO) has ported an existing relational database containing oceanographic metadata, along with an existing interface coded in Cold Fusion middleware, to a Drupal 7 Content Management System. This is an update on an effort described as a proof-of-concept in poster IN21B-1051, presented at AGU2009. The BCO-DMO project has translated all the existing database tables, input forms, website reports, and other features present in the existing system into Drupal CMS features. The replacement features are made possible by the use of Drupal content types, CCK node-reference fields, a custom theme, and a number of other supporting modules.

This presentation describes the process used to migrate content in the original BCO-DMO metadata database to Drupal 7, some problems encountered during migration, and the modules used to migrate the content successfully. Strategic use of Drupal 7 CMS features that enable three separate but complementary interfaces to provide access to oceanographic research metadata will also be covered: 1) a Drupal 7-powered user front-end; 2) REST-ful JSON web services (providing a MapServer interface to the metadata and data; and 3) a SPARQL interface to a semantic representation of the repository metadata (this feeding a new faceted search capability currently under development). The existing BCO-DMO ontology, developed in collaboration with Rensselaer Polytechnic Institute's Tetherless World Constellation, makes strategic use of pre-existing ontologies and will be used to drive semantically-enabled faceted search capabilities planned for the site. At this point, the use of semantic technologies included in the Drupal 7 core is anticipated.

Using a public domain CMS as opposed to proprietary middleware, and taking advantage of the many features of Drupal 7 that are designed to support semantically-enabled interfaces will help prepare the BCO-DMO and other science data repositories for interoperability between systems that serve ecosystem research data.

Motivations for Using Drupal

ColdFusion 8	Drupal 7
Proprietary	Open-source
Custom code	Contributed Modules
No Semantic Web	Semantic web support

Highlighted above are just three of the advantages that BCO-DMO would gain from using Drupal instead of ColdFusion. The first of which is the fact that Drupal is Open Source. This enables BCO-DMO to have much more control over the software that is serving our data to the web. The second advantage is that because Drupal is Open Source and built with a modular nature in mind, BCO-DMO can take advantage of code written by the Drupal community as well as write custom code. There are literally thousands of Drupal modules, many of which would be useful to BCO-DMO. The third advantage is that Drupal 7 has semantic web capabilities built into the core. This will emerge as a significant advantage as semantic technologies play a larger role in facilitating data discovery, data access, and data interoperability.

Migrating to Drupal 7

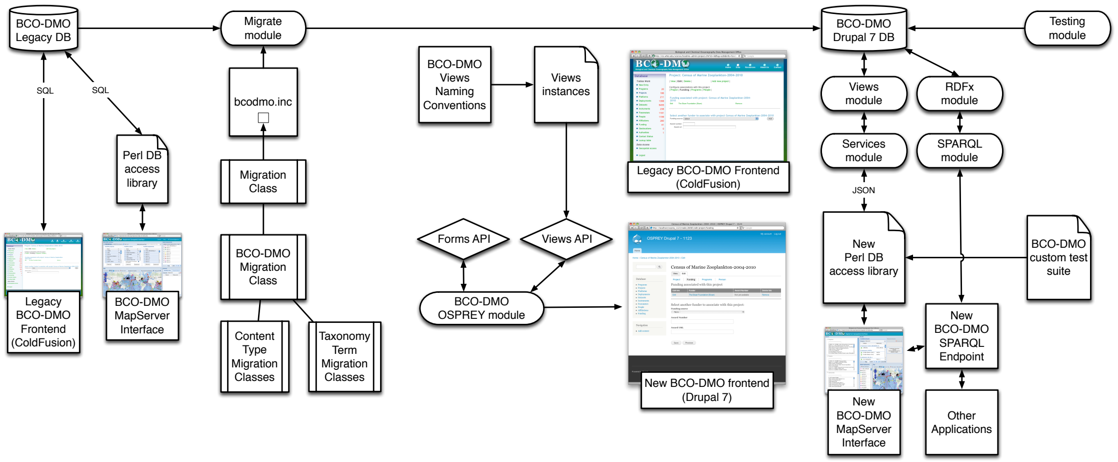
1 Inventory

2 Migrate

3 Customize UI

4 Integrate

5 Test


BCO-DMO Legacy DB
SQL
SQL
Perl DB access library
Legacy BCO-DMO Frontend (ColdFusion)
BCO-DMO MapServer Interface
Migrate module
bcodmo.inc
Migration Class
BCO-DMO Migration Class
Content Type Migration Classes
Taxonomy Term Migration Classes
BCO-DMO Views Naming Conventions
Views instances
Forms API
Views API
BCO-DMO OSPREY module
Legacy BCO-DMO Frontend (ColdFusion)
New BCO-DMO frontend (Drupal 7)
BCO-DMO Drupal 7 DB
Testing module
Views module
RDFx module
Services module
SPARQL module
JSON
New Perl DB access library
BCO-DMO custom test suite
New BCO-DMO SPARQL Endpoint
New BCO-DMO MapServer Interface
Other Applications

The first step is to examine the structure of existing data stores and interfaces. It is important to understand how data are organized and presented in the legacy system so that they can be represented accurately within Drupal.

In order for the legacy data to be imported into Drupal, one must create the necessary Drupal data structures to hold the legacy data. One might use the **CCK**, **References**, and **Corresponding Node References** modules to create data structures within Drupal. Using the **Migrate** module, a mapping between existing data and Drupal data structures can be established.

Depending on the legacy data entry method, building a user interface in Drupal to mimic the legacy system can require some creativity and a solid understanding of the Drupal core and contributed module APIs. In order to recreate the legacy data entry method in Drupal 7, BCO-DMO used a custom module to bring together the **Forms** and **Views** APIs and a series of Views that were assigned a name according to a strong naming convention (to make code simpler and easier to maintain). In addition, we took advantage of **Views**' display inheritance feature to make creation and maintenance of the Views easier. To achieve a tab-based data editing interface, we used a *hook_menu* of type *MENU_LOCAL_TASK* to create a tabbed interface similar to BCO-DMO's legacy input forms.

The legacy database access layer used SQL queries that became obsolete when we imported the data into Drupal. In place of the SQL queries, we will call authenticated Web Services built using the **Services** and **Views** modules to achieve integration with existing external applications.

To make sure that the Drupal interface works as designed, tests will be written using the **Testing** module. To assure that integration with external applications works and performs well, a custom test suite will be written. In addition, user testing will take place to assess usability.

Acknowledgments

This work has been funded by the National Science Foundation and the Woods Hole Oceanographic Institution.