Browsing and Searching SBC datasets
Updated 2009

1. Introduction and goals:
The unit of data designated by SBC for publication and sharing is a dataset which is composed of an EML metadata document and one or more associated data tables (documentation for EML and datasets is elsewhere). When delivering these via the website, there are 2 use cases to accommodate:
A. user knows very little about our data, and wants to browse.
B. user wants to search for datasets using specific terms or an individual's name.

1.1 Considerations:
1. As of 2009, we have ~130 EML datasets, all with table(s) in a local Metacat catalog, which can be queried to return a list of datasets. However, a list format often does not allow room for details which might be necessary to locate an appropriate dataset, and also does not indicate how datasets may be related to each other.
2. Many SBC datasets can be queried with our EML Data Query tool (EDQ, documentation elsewhere), which can accommodate both single data tables and groups of identically formatted tables. At least one data delivery mechanism should clearly indicate when the EDQ tool is available.

1.2 Plan:
We defined the concept of a data 'collection' to be a group of related datasets. The use of a data collection layer means that datasets can be described atomically and with very little consideration for the delivery method, but still can appear together – e.g., to facilitate browsing (use case A), and to indicate that the group shares an EDQ controller. We have not imposed any rules about what constitutes a collection; it can be one dataset or many, and collections have been rearranged. For example, during the early stages, individual monitoring datasets were each assigned to separate collections, but when we added similar data from a manipulation experiment, the monitoring datasets were grouped together into one collection, and the manipulation datasets into another. Currently, datasets are assigned to only one collection.

The data browse application described here started out as a proof-of-concept to demonstrate and test ways to group datasets so that they could appear together when necessary. It has now been in production for about 2 years, with a few modifications and additions. It has received some useful (and even complementary) feedback, and plans for further development are below.

We also wanted to accommodate use case B (simple searches). This became particularly necessary in 2008 to accommodate the LTER Executive Board (EB) request that sites make their data catalogs searchable by PI names and research topics. For this use case, a user may already know that a dataset exists, or simply wishes to know what datasets researcher "Doe" has contributed. Since simple lists are useful for returning results from simple searches using terms or names, we used the Metacat query mechanism for returning dataset searches (i.e., we do not search collections).

A library analogy (in which datasets are like books) is useful to distinguish between browsing and searching. When a user "browses" SBC data collections, it is similar to the library visitor who strolls down an aisle reading the spines of books which are grouped by topic. A user "searching" for datasets is analogous to the library visitor who uses the card-catalog to locate a specific item.

2. Browsing collections:

2.1 Overview

One Perl script drives the views for 3 classes, which are keyed to data collections with a controlled vocabulary of terms for habitats, measurement types, or LTER network core research topics. This script can accommodate multiple stages, which are subroutines, with one stage as the default. Views are created using the Perl template toolkit module, and a 'wrapper' with the SBC web page boilerplate is also added. There is also a static portal page for the application, which contains introductory material and is created along with the rest of the static pages in the website (documentation elsewhere). It has forms for the 3 browse categories, and calls the Perl script. Whenever a view is returned, the user may initiate a new view from the forms which appear in the left panel.

Only one form's request can be handled with this script – there are no intersections. This functionality was considered for a time so that a user could fine-tune a request, such as "show me biogeochemistry data in the watershed." See below for more information.

Page content (i.e., descriptions of data collections and keywords for the forms) is provided by XML files. These were the most straightforward for development; I could simulate using a web service to deliver data while developing the templates, and experiment with data models for a collection. So at present, the application is relatively unsophisticated. This is adequate for now while the number of collections is small, and it makes changes to the model easy. See below for future plans.

2.2. Web Application Components:

2.2.1. cgi/
one perl script drives the views: showCollections.cgi

Stages and subroutines:
2.2.1.1 Return all the collections
```
  'default'          => \&displayAllDataCollections,
```

2.2.1.2 Return the collections keyed to a user-chosen habitat
```
 'by_habitat'        => \&displayByHabitat,
```

2.2.1.3 Return the collections keyed to a user-chosen measurementType
```
 'by_measurementtype' => \&displayByMeasurementType,
```

2.2.1.4 Return the collections keyed to a user-chosen core research area
```
 'by_coreresearcharea' => \&displayByCoreResearchArea,
```

The script is a little clunky – parts that could be parameterized have been copied to new subroutines. This is not good practice, but part of its nature as a 'proof-of-concept'. This should be cleaned up or replaced in the future.

2.2.2. lib/
This script needs only one template to display collections:
displayDataCollections.tmpl

Three other templates are in the website SVN checkout for creating forms on the data portal page (these are deprecated and should be removed)
data_formSearchByHabitat.tmpl
data_formSearchByMeasurementtype.tmpl
data_formSearchByCoreResearchArea

2.3. Content

The content comes from several XML files.

2.3.1 These 3 files provide content for the forms:
lterCoreResearchAreas.xml
habitats.xml
measurementTypes.xml
The main template called by the cgi (displayDataCollections.tmpl) pulls in these 3 XML files and cycles through them to generate a form for each. A node from one file (habitats) looks like:

```
…
  <habitat id="watershed">
    <label>Watershed</label>
    <description>riparian and stream habitats</description>
  </habitat>
…
```

So if you want to add a keyword to a group, all you have to do is add another node to the appropriate file and it will be automatically available for browsing. We made use of this flexibility when we added data from the beach habitat in 2007.

2.3.2 This file contains descriptions of all our data collections, and is what the cgi script actually searches:
dataCollections.xml
The structure of a data collection is still somewhat fluid. For example, when we published the cruise data their volume necessitated creating a custom index page. To link to this page from the cruise collections, I added an additionalInfo tag (which resembles the EML additionalInfo element). Again, see below for future plans. A sample collection node is included at the end of this document.

There is no schema for these XML files yet. However, the templates assume that certain nodes are present, so these can be assumed to be ‘required’:
<collection @id>
<abstract>
<title>
<dataPackages>
<habitats>
<measurementTypes>

<dataPackages>, <habitats>, and <measurementTypes> have repeatable children. <abstract> and <title> and @id are unique to a collection (not enforceable yet with XMLSchema).

These nodes are optional and repeatable, since not all collections have these:
<queryAppController>
<lterCoreResearchAreas>

2.4. Future plans

This application and the concept of a data collection was one of the driving forces behind SBC's involvement in the IMC "projectDB" project (see documentation elsewhere). Others in the IMC had expressed the need for a system to manage scientific projects, and an experiment or project is also a logical container for datasets and other resources. Work began on the projectDB in 2008, and continues. Eventually, the SBC data browse application's content could be fed from XML via the projectDB, where the datasets are listed as resources attached to that project. This would also facilitate linkages between the "research" and "data" areas of the website. It is also anticipated that a dataset could be attached to more than one project.

Intersecting queries: The application currently handles only one form's parameters per query, but it may be desirable to create queries that intersect, such as "show collections for biogeochemistry from the watershed." This was considered early in development, when it contained only 2 forms (habitats and measurement types). The small number of collections at that time would have meant returning quite a few empty result sets which seemed undesirable and so this functionality was postponed. Then in 2008, the EB asked for some additional browse terms to be enabled in each site's data catalog (i.e., for the network-defined "Core Research Areas"). In SBC's application, this was simple to accommodate as a 3rd menu. But it also meant that intersecting queries would become more complex. First, there are now some intersections that make no sense (e.g., between 'populations' and 'biogeochemistry') and second, there are more chances for empty results. So intersections will wait. Permissible intersections need to be defined and the system must be flexible enough to accommodate future key terms. The user interface must be redesigned so that it is clear what kinds of intersections are possible. And quite probably, some of the functionality originally requested by the EB will be accommodated by the network instead of by sites.

3. Searching datasets

Data sets are searched by 2 forms which directly query the Metacat data catalog through the servlet at http://sbc.lternet.edu/catalog

Metacat has a unique query syntax (called 'pathquery') since it was developed before the acceptance of XQuery by W3C. The specification for pathquery can be found at http://knb.ecoinformatics.org/software/metacat. The search forms are written in javascript, and can be inserted as needed into HTML. Metacat indexes certain paths to increase query speed, and the indexing is configurable for each installation. Those writing pathqueries should tune their queries to a specific Metacat to take advantage of its indexing.

3.1 Files:
data/Metacat_searchboxes.html
contains the javascript for 2 forms:
1. to query using a surname, these paths are searched in the EML
/eml/dataset/creator/surname

2. to query using a term, these paths are searched in EML
/eml/dataset/title
/eml/dataset/abstract
/eml/dataset/keywordSet/keyword

For optimal speed, the path should exactly match the path indexed by Metacat (check with the Metacat maintainer).

Here is a chunk of javascript showing the pathquery syntax with the user input value:

```
// In a textstring, create a metacat query and add the data from the form
  var textstring =
  '<pathquery version=\"1.2\">' +
  '  <returndoctype>eml://ecoinformatics.org/eml-2.1.0</returndoctype>' +
  '  <returndoctype>eml://ecoinformatics.org/eml-2.0.1</returndoctype>' +
  '  <returndoctype>eml://ecoinformatics.org/eml-2.0.0</returndoctype>' +
  '  <returnfield>eml/dataset/title</returnfield>' +
  '  <returnfield>eml/dataset/creator/individualName/surName</returnfield>' +
  '  <returnfield>eml/dataset/creator/organizationName</returnfield>' +
  '  <returnfield>eml/dataset/dataTable/entityName</returnfield>' +
  '
<returnfield>eml/dataset/dataTable/physical/distribution/online/url</returnfield
>' +
  '  <querygroup operator=\"INTERSECT\">' +
  '    <queryterm casesensitive=\"false\" searchmode=\"contains\">' +
  '      <value>';

      textstring += box1.value ;

  textstring +=
  '      </value>' +
  '      <pathexpr>creator/individualName/surName</pathexpr>' +
  '    </queryterm>' +
      '    <queryterm casesensitive=\"false\" searchmode=\"starts-with\">' +
      '      <value>knb-lter-sbc</value>' +
      '      <pathexpr>@packageId</pathexpr>' +
      '    </queryterm>' +
      '  </querygroup>' +
      '</pathquery>';
```

4. Example of a collection node used by the browse application:

```
 <collection id="kelp_npp">
   <collectionName>SBC LTER: Net Primary Production of giant kelp</collectionName>
   <abstract>Ongoing time series of net primary production (NPP), growth, standing
      crop loss and rates from disturbance for M. pyrifera appropriate for examining
      seasonal and inter-annual patterns. Sampling is monthly at 3 reef sites, and began
      in 2002.</abstract>
   <dataPackages>
     <package>
       <docid>knb-lter-sbc.21</docid>
       <shortTitle>Giant Kelp NPP</shortTitle>
     </package>
     <package>
       <docid>knb-lter-sbc.24</docid>
       <shortTitle>Algal mass and CHN</shortTitle>
     </package>
   </dataPackages>
    <queryAppControllers>
     <controller>
       <controllerName>giant_kelp_net_primary_production</controllerName>
       <controllerLabel>Giant Kelp NPP</controllerLabel>
     </controller>
   </queryAppControllers>
   <queryAppControllers>
     <controller>
       <controllerName>kelp_chn</controllerName>
       <controllerLabel>Algal mass and CHN</controllerLabel>
     </controller>
   </queryAppControllers>
   <habitats>
     <habitat>reef</habitat>
   </habitats>
   <measurementTypes>
       <measurementType>ecosystem_processes</measurementType>
   </measurementTypes>
   <lterCoreResearchAreas>
     <lterCoreResearchArea>primary_production</lterCoreResearchArea>
     <lterCoreResearchArea>disturbance_patterns</lterCoreResearchArea>
   </lterCoreResearchAreas>
 </collection>
```