DSpace Summer of Code Ideas 2009

This is an archived listing of project ideas for Google Summer of Code in 2009. To see which projects actually took part in GSoC 2009, please visit the Past DSpace Summer of Code Projects page

Contents

Ideas for Potential Google Summer of Code 2009 projects

Feel free to review previous ideas in the DSpace Summer of Code Ideas 2008 page, if anything seems appropriate, please feel free to bring it forward.

DSpace 2.0 Initiative

After some discussion within the DSpace 2.0 group, we've concluded that using the GSoC as a source for moving forward the DSpace 2.0 initiative is an excellent idea. If you have an idea for DSpace 2.0 development, please feel free to add it here. We will be organizing a list of ideas within the DSpace 2.0 Architecture group and presenting that list here as well. If you would like to discuss DSpace 2.0 projects explicitly, please feel free to join the dspace-architecture list as well and drop us a note there.

https://lists.sourceforge.net/lists/listinfo/dspace-architecture

- General note on code from the GSoC developers
  - Code should be maintainble and adhere to the code style of the DS2 project
  - Code must not use any packages with licenses which are incompatible with free open source (commercial, GPL, LGPL)

Authentication/Authorization providers

Port LDAPAuthenticator, X509Authenticator, create new Authenticators for open id/open auth etc.

- The task would be to create DS2 providers for authentication and principal extractors:
  - LDAP
  - OpenId
  - X509
  - Basic Auth
  - Kerberos
  - Active Directory
  - Shibboleth
  - CAS
- The task would also be to create DS2 providers for authorization:
  - LDAP
  - OpenAuth
  - IP Based

    - Kerberos

Storage Service implementations

DSpace 2 has a generalized storage service API which allows a DS2 reposoitory to use many possible systems to store repository data (DBMS, JCR, etc.) and even other repositories (Fedora, Eprints, etc.)

- The task would be to implement additional Storage Service providers for DS2 (some examples listed)
    - JCR (Alfresco/Xythos/jackrabbit??) REST
    - JCR (Alfresco/Xythos/jackrabbit??) RMI
    - Fedora REST
    - Filesystem + Sesame (or other TripleStore)
    - Filesystem + DBMS
    - IRoDS/SRB
    - s3/SimpleDB
    - Other?

Distributed repository searching system

Cloud style search system which allows repositories to register and indicate they are searchable and what search interfaces and parameters they support from a standard set

- The task would be to develop an application (probably python based in appengine since it is free but any free scalable alternative is good) which repositories could register themselves with to indicate they want to participate in wider/global search
    - Defining the standard set of parameters and search options would be the first task (DC? Author/title/publish date?)
    - Defining registration information would be the second task (geolocation, name, associations, type of repo, etc.)
    - Defining the interface to register and the interface to query for searchable repositories
    - Defining the method for getting back search results (realtime, email, etc.)
    - Deciding how much caching of results and information the central system should do (should the repo be allowed to specify this)

This would be mostly be a proof of concept which could be used to build a more reobust and production oriented method of inter-repository communication later on.

Core, Logging, Event Management

Administration and Acces Control

User Management - Enhancement

- User self management
    - delete ones account
    - reminder of account
    - reminder of unfinished tasks and submissions
- User contact (for alerts, feedback, ads)
    - all active users
    - all registered users
    - users of a group
    - users with special rights (i.e. all submitters)
- manage non-active users
  based on set of rules
    - reminders sent
    - deletion
- check for invalid accounts
- validate unsendable emails (registration, alert and so on)

Metadata Management

Extend Item templates to support creating Collection default ResourcePolicies for Items, Bundles and /or Bitstreams

Item templates currently support metadata fields one may want in an Item by default, but do not support anything else like default bitstreams or default permissions on bitstreams, It would be good to extend Item template to have more features.

Proposed by --Mark Diggory 14:58, 24 March 2009 (GMT)

LinkOut

The metadata of a document can be very useful to propose services to the user around the document.
For instance:

- an author name can be used to send an e-mail (if it is a DSpace user) or to make Google Scholar Search
- an ISSN can be used to access publisher home page

- the title can be used for a citation search
- a CAS can be used to search a chemical database (or any ID of a gene, plant, etc. can serve to make a database search).
- an institutional Id. can link to applications around this Id providing some service
- etc.

_ I would propose to be able to parameterize a link from any given metadata field to a "linked services" page (services provided by external applications OR by DSpace itself, for instance documents of the same author, on the same subject, etc.).
This is a generalisation of existing features proposed by different patches. Christophe.Dupriez 14:50, 17 March 2007 (EDT) _

Typed Metadata Fields

DSpace is organized a little bit the same way than "old fashion" card catalogue: a database maintains a strict and coherent storage of data and indexes are dynamically built with Lucene to rapidly find something in the database.

The indexing process has a big added value as it insures that the users, from what they know, find what they need. This process must be tailored to the exact needs of each application.

We would like to propose to make DSpace evolve toward a concept of highly parameterized "pluggable indexation" where multilingual, multialphabet issues could be solved by easily accepting contributions and where new datatypes (duration, Longitude/Latitude, etc.) coud also be perfectly supported.

This "openness" of indexation will also naturally bring the question of "openness" of editing of a given value type (or language) and display.

The proposal is therefore:

1. to define an architecture for "data types management"
2. to allow the definition of Java programmed Data Types Managers to handle a given type: the managers would provide validation, clean-up, tokenization, toString, toXML and toHTML methods on data values of a given type
3. to allow the parameterization of DSpace to associate a data type to each DC (or other) fields for all languages or for a specific language
4. in DSpace, to call the data type managers everywhere a value must be validated, indexed, displayed.

JCR

A possibly efficient and "future proof" way would be to integrate a JCR compliant repository for storage, indexing and retrieval of metadata and bitstreams?

- Introduction to JCR
- JackRabbit

Christophe.Dupriez

The DSpace 2.0 default content storage implementation is JCR/JackRabbit. A great project would be to look at alligning 2.0 and 1.x storage models further. --Mark Diggory 21:56, 11 March 2010 (UTC)

i18n of DSpace Objects

At the moment only static parts of the DSpace UI are i18n. These are kept in message catalogues.

Variable metainformation is not presented language dependant.

The metainformation of an item beeing stored as metadata is theoretically presentable language dependant, wheras the metadata of other DSpace objects (communities, collections, epersons, groups) is not stored as metadata in DSpace terms and it is not possible to have it i18n.

There should be a distinction between properties (e.g. for config purpose) and metainformation of DSpace objects. The metainformation should be expressed as DSpace metadata objects, even for metametadata.

Management of input-forms via db (DONE last year not yet integrated)

Move the input-forms from file based storage to the database and make them manageable via the UI.
This should include the management of templates and sanity checks and be flexible enough to enable the below mentioned "Item type based input".

Item type based metadata collection (DONE last year not yet integrated)

At present metadata collection is based on the input-forms.xml file. This file can be edited to collect different metadata for different collections. An alternative is proposed that would allow the administator to collect different metadata based on item type. There are no doubt many ways this could be achieved, for example:

- An alternative input-forms.xml file based on item type.

or..

- An admin page that would allow the administrator to select metadata terms for each item type. Sensibly this information would be held in the database rather than a file. If I understand Christophe's proposal above then this option would dovetail nicely in that the definition of data type would no longer be done in input-forms.xml.

Robin Taylor.
University of Edinburgh.

Add support to upload new metadata schemas as a file into the DSpace Metadata Registry. (DONE last year not yet integrated)

It requires a developer to add whole new metadata schemas to the dspace metadata registry, recommend creating an upload form for the registry that creates new namespaces/fields based on at least the existing metadata schema configuration file format.

Proposed by --Mark Diggory 16:38, 24 March 2009 (GMT)

Community Tools and Community Trends

Visualization of Author Association in DSpace

Institutional repositories gather an organization's scholarly content and facilitate knowledge sharing and dissemination of intellectual output. In an educational environment one of the major content contributors to Institutional repositories are students and associated academia represented as contributing primary and co-authors. Since content contribution is one of the major bottlenecks as far as institutional repositories are concerned, it would be a major initiative to satisfy stakeholder's interest by identifying these author associations. This may also have a positive impact on potential content contributors. In-addition to this another area of interest for repository stakeholders would be ranking the relative strength of association between the contributing primary and co-authors. Our aim in this project is to extract author associations from each contributed article and represent these associations in a semantic scale (i.e. 1 represent the highest co-authorship) to describe strong and weak co-authorship relations. This approach can be further extended by designing inference rules that would bridge association between authors interested in slightly disparate research domains (i.e. computer science and Information science) through a co-authorship relationship (i.e. A co-authoring with B; B co-authoring with C; there by inferring A's potential association with C through B). We plan to achieve this by using a Semantic Web framework making use of popular reasoners (e.g. Jena). To better analyse the strength and weakness of associations it would be interesting to visualize author associations using popular visualization software such as Pajek or UCINET. An application that would generate an "Author Association Report" (build upon GSOC2009 report generation project code) for a given author would be interesting for repository stakeholders and potential contributors to Institutional repositories.

Proposed by: Jayan C Kurian, Lecturer, RMIT International University, Vietnam. Email: jayan.kurian@rmit.edu.vn

Potential Student: In-discussion.

Adaptive Question Answering System based on the DSpace Mailing List Knowledge Base

Recent years have witnessed the tremendous usage of repository software since majority of scholarly content are published in digital form with no exception to the proliferation of DSpace instances. Popular software does manage user queries through mailing list supported by dedicated committers and contributors. A good number of questions asked in a mailing list would have been responded previously. In this case, a Question Answering (QA) system would help users by answering their questions, if it has been responded earlier or would suggest related answers encompassing the subject asked. For this, information available on the DSpace mailing list knowledge base can be extracted using template based extraction techniques or a rule based system. Once extracted, this can be classified according to a taxonomical structure (i.e. Functional Overview, Installation, Upgrading, Configuration, Customization, Architecture, and Versions) that represents the DSpace system architecture/documentation. The keywords automatically generated from the message text improve the adaptive retrieval of relevant information in this QA system. A test-bed for this QA system can be build using the DSpace platform and the taxonomical structure can be facilitated by the in-built controlled vocabulary feature.

Proposed by: Jayan C Kurian, Research staff, National University of Singapore, Singapore. email: Jayan@comp.nus.edu.sg, jayanntu@gmail.com

Potential Student: Ashly Markose, Post-graduate student, National University of Singapore, Singapore email : ashly@comp.nus.edu.sg, ashlymarkose@gmail.com

Research Trend Analysis using Institutional Repositories

Institutional repositories gather an organization's scholarly content and buttress knowledge sharing and dissemination of intellectual output. The communities and collections in a repository are designed according to an institution's distribution of research centers, schools and divisions. Each collection that represents a school, division or research centre holds erudite contents facilitated by respective academic projects mentored at those centers. By performing Co-Word analysis on each document collection, the individual research strength of that division or school can be found out. The same principle can be propagated to sub-collections as well as communities in a repository. This extrapolates the research strength of a particular division /school and in general can be extended to determine the profound research strength of an institution. In addition to this, a qualified variation or trend in the research strength of individual divisions/schools can also be found out by applying Co-Word analysis over a predetermined period of time. The above described feature can be extended as an add-on to the existing framework of DSpace and would facilitate the burgeoning representation of DSpace as a research platform.

Proposed by: Jayan C Kurian, Research staff, National University of Singapore, Singapore. email: Jayan@comp.nus.edu.sg, jayanntu@gmail.com

Potential Student: Ashly Markose, Post-graduate student, National University of Singapore, Singapore email : ashly@comp.nus.edu.sg, ashlymarkose@gmail.com

Search, Browse, Discovery and Semantic Web

Port DSpace Crosswalk API to be a SAX/XSLT pipeline or STAX rather than JDOM

History repeats itself... the Cocoon folks learned long ago that Pipelines would be both more efficient and more flexible if they were SAX driven rather than DOM driven. DSpace could learn a lesson from that play-book and re-implement the Crosswalk API to Be a suite of SAX XMLReaders That take DSpace Objects and serialize them to XML. This would leverage the existing work done in the Manakin DSpace Adapter API and bring it into a new Addon or dspace-api directly making it available as a much more efficient mechanism for getting DSpace Objects into XML.

Build, Installation, Testing and Running DSpace

Complete installation process (ant fresh_install, update, init_configs, etc) as Maven goals

We almost completed the process of porting the build system to Maven, however, that process stalled with several tasks that are critical for the installation /deployment of DSpace that releid heavily on Ant, we would like to explore 1.) Running ant tasks from Maven, 2.) creating a set of Maven plugins for DSpace installation update and deployment.

Sanity Checking Framework

- checking for prerequisites and rights
- are all components installed in running
- system diagnostics
- other dependencies
  like code and content
  metadata registry, input-forms and Messages.properties
- amount of bitstreams in db and assetstore

Build an automated testing system (unit tests, kinda)

- Create some automated tests to detect bugs in DSpace code, particularly in the org.dspace.content package. It does not need to be particularly fast. The test rig could load some test data into DSpace, check that it's there, perform various manipulations and check that the expected results appear. Authorisation would also be an important factor to test.

Unit tests that cover a lot of this have been written for 1.6 (though they aren't yet in trunk). What I would like to see is a mechanism for the construction of 'sandbox' environments for running such tests in. For example, proper testing would require a test database, test asset store, etc, which is all quite messy to do by hand, and tricky to automate. --James Rutherford

A useful precursor to unit tests, IMHO, would be an eclipse project file setting. This would allow developers to more easily build eclipse without them having to do all the IDE setup themselves. Then you could use the builtin JUnit support that eclipse has. Plus, of course, eclipse would help do development generally.