DSpace 2.0 Comparing Existing Technologies

<?xml version="1.0" encoding="utf-8"?>
<html>

Data Models

Sakai Entity and Content Models

http://bugs.sakaiproject.org/confluence/display/MJN/Sakai+Entity+Model

Every Sakai Entity has the following features/capabilities:

- URL: A URL that can be used to access this entity.
- Reference: An internal reference.
- Id: A globally unique (UUID) identifier.
- Properties: A set of resource properties associated with the entity.
- XML Form: The entity in serialized XML form.

http://bugs.sakaiproject.org/confluence/display/MJN/Sakai+Content+Model

ContentResource object:

- Element: Description
- Content Length: Length in bytes of the content stream.
- Content Type: MIME Type of the content.
- Content: A byte array.
- Content: An input stream.

ContentCollection object:

- Element: Description
- Members: A list of reference strings.
- Member Resources: A list of Content Resource objects.
- Body Size: Aggregate size of the collection in 1024 units.

- *Sakai has migrated their Content Host Service to JCR, Sakai Wraps the JCR Factory API so that you are not exposed to how a JCR connection is retrieved, the JCR objects themselves are exposed to the user of the service so there is no need to learn another API. This presents a stronger case for use of JCR as an API for interacting with stores such as DSpace and Fedora, reuse of a Fedora JCR implementation may enable DSpace on Fedora, Sakai on Fedora, Sakai+DSpace on Fedora, etc.* --Mark Diggory 17:44, 26 September 2008 (EDT)

Here is a decent document backgrounder the reasons behind th Sakai move to JCR as its Content Hosting Service http://bugs.sakaiproject.org/confluence/display/CHS/JCR+rationale+and+implications

Fedora Object Model

http://www.fedora-commons.org/confluence/display/FCR30/Fedora+Digital+Object+Model

- PID: A persistent, unique identifier for the object.
- Object Properties: A set of system-defined descriptive properties that are necessary to manage and track the object in the repository.
- Datastream(s): The element in a Fedora digital object that represents a content item.

Java Content Repository API: JCR Nodes

The Java Content Repository API is a client side API for interacting with Content Repositories, it requires that the repository either be an implementation of a known driver (WEBDAV, CMIS, JDBC, XMLDB, Alfresco...)

The JCR API is an object tree comprised of Items, which can be Nodes, Properties or Relations. All data is added as Properties to Nodes so, Properties represent leaves in a tree of Nodes.

Metadata attached to JCR Nodes is namespace prefixed, much like XML/RDF so that XPath/SQL like queries can use prefixes and XML serialization can express properties in separate namespaces. JCR supports Content Models and typing with the addition of "mixin" properties.

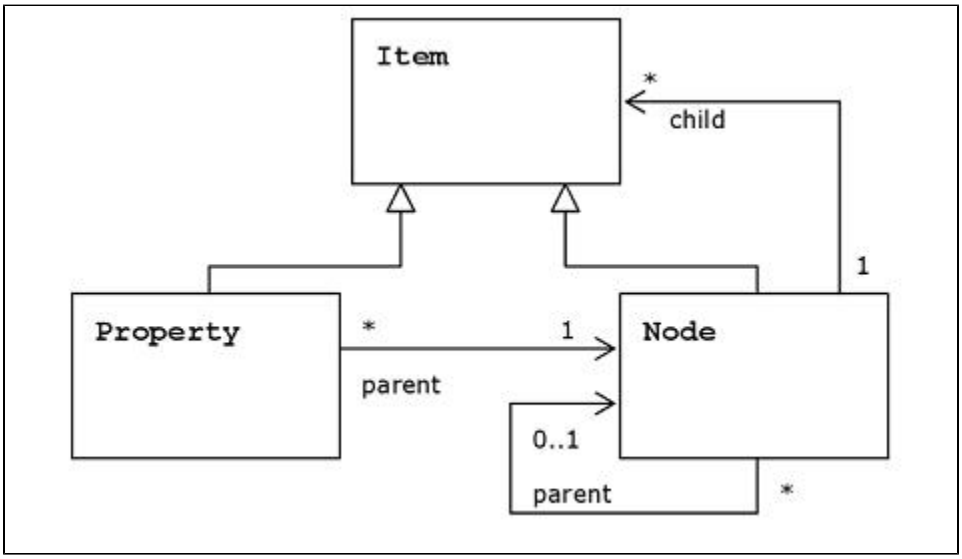
Examples of mixin properties:

*__mix:versionable__: allows a node to support versioning
*__mix:lockable__: enables locking capabilities for a node
*__mix:referenceable__: provides an auto-created jcr:uuid property that gives the node a unique, referenceable identifier

Every repository must support the primary node type, nt:base. There are a number of other common node types that a repository may support:

*__nt:unstructured__ is the most flexible node type. It allows any number of child nodes or properties, which can have any names. This node type represents JCRWiki entries.
*__nt:file__ represents files. It requires a single child node, called jcr:content. This node type represents images and other binary content in a JCRWiki entry.
*__nt:folder__ node types can represent folders, like those in a conventional filesystem.
*__nt:resource__ commonly represents the actual content of a file.
*__nt:version__ is a required node type for repositories that support versioning.

The entire node type hierarchy can be found in section 6.7.22.1 of the JSR-170 specification.



An older api overview http://www.theserverside.com/tt/articles/article.tss?l=JCRPract

- *I have concerns that JCR's use of mixin properties is a bit arduous and obtuse, I have yet to find adequate description of the mixin properties*--Mark Diggory 15:36, 26 September 2008 (EDT)

Relationship Models

JCR Relationships

JCR Relation objects define a directional relation between two Nodes. This allows the creation of non-hierarchical structures (graphs) out of what is basically a hierarchical object store. JCR relation object are seperate fromt he object they relate and do not alter modification timstamps of those objects went added/removed. This allows relations to be used to express things like version histories, symlinks, bassically "non-containment" like associations.

- _I think JCR relation objects should be evaluated in comparison to the REL-EXT section of a Fedora Object. Though, I'm unsure of the update policy on object with RELS_EXT, specifically given that the relationship is stored within the Fedora Object itself. _ --Mark Diggory 17:06, 26 September 2008 (EDT)

Sakai

Sakai has migrated their Content Host Service to JCR, so they support relations in the same way as JCR states above.

Fedora Relationships (RELS-EXT)

http://www.fedora-commons.org/confluence/display/FCR30/Digital+Object+Relationships

RELS-EXT gives an opportunity to express relationships between fedora objects in RDF/XML, it looks something like:

```
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:fedora="info:fedora/fedora-system:def/relations-external#"
xmlns:myns="http://www.nsdl.org/ontologies/relationships#">
<rdf:Description rdf:about="info:fedora/demo:99">
<fedora:isMemberOfCollection rdf:resource="info:fedora/demo:c1"/>
<myns:isPartOf rdf:resource="info:fedora/mystuff:100"/>
<myns:owner>Jane Doe<myns:owner/>
</rdf:Description>
</rdf:RDF>
```

This shares a conceptual similarity to the JCR Relationship Objects. Which can be used to assert relationships between JCR nodes independently of the nodes themselves. This suggests again that Relationships between DSpaceObjects/FedoraObjects/JCR Nodes is best kept functionally separate from the descriptive metadata properties attached to the object itself.

- *Per my statement above, I think that JCR is onto something here, specifically that relationships may be expressed by some "authority" other than the Object owner, as such such relationships need to be modifiable and maintained independent of the objects themselves, allowing for various authorities o assert relations independently.* --Mark Diggory 17:11, 26 September 2008 (EDT)

Versioning Content

DSpace

A Google+Summer+of+Code project in 2007 has implemented a versioning prototype, for DSpace Items, DSpace Items have two identifiers, on permanent, the other is a version lineage id. The Lineage is comprised of Items, each with unique metadata and bundles, Bitstreams within the Items will be either linked from the previous version or added anew.

http://wiki.dspace.org/index.php/Google+Summer+of+Code+2007_Versioning

JCR

JCR Appears to Version Nodes, its unclear ATM if this results in new node ids I assume it does not, There is an explicit VersionHistory API available that gives one thability to navigate the Versions (the VersionHistory of the Node). Versions are Nodes as well.

http://day.com/maven/jsr170/javadocs/jcr-1.0/javax/jcr/version/package-summary.html

Fedora

Fedora Versions Datastreams and not whole Digital Objects, this eliminates the need to generate/maintain new Object identifiers...

See for more detail http://www.fedora-commons.org/confluence/display/FCR30/Versioning

Sakai CHS

Now based on JCR so is similarly versioned

Change Notification

In 1.5 we implemented an event notification API in DSpace that allows one to listen for changes in the model upon serialization of that model. One thought on this event mechanism is that it should actually exist int he Service Layer rather than directly int he model itself (beyond the model being able to flag if an item changed or not internally. Here is a survey of Change notification in the following compared technologies:

Fedora

The Fedora API support change notification as a JMS message provider. This works via Topics or Queues.

http://www.fedora-commons.org/documentation/3.0/userdocs/server/messaging/index.html

JCR

There is an explicit Change Notification API for JCR. I did uncover a recent thread exploring the topic here:

http://www.theserverside.com/news/thread.tss?thread_id=46303

http://day.com/maven/jsr170/javadocs/jcr-1.0/javax/jcr/observation/Event.html http://day.com/maven/jsr170/javadocs/jcr-1.0/javax/jcr/observation/EventListener.html

DSpace

In 1.5 the Event Mechanism was provided as a subsystem capable of issuing events to consumers. Originally, there was a JMS consumer solution, but it was not brought into 1.5 implementation because of tractability and complexity issues with the API.

Ideally, DSpace needs to be a change listening "client" on an event notification service that may be providing events from the DSpace Content Management Service API. In this case, pluggin into either the Fedora JMS service or proposed JCR JMS Service would provide this level of functionality, allowing DSpace to receive and act on messaging from its storage layer. This is different than the previous JMS Messaging support service for DSpace in that DSpace is acting as a consumer of repository specific messages rather than a provider of those messages. Perhaps, in its default implementation, a native DSpace storage layer would emit JMS messages in a similar role to that of Fedora/JCR.

</html>