Training - Introduction and Feature Tour

These training archives may be out of date, but have been retained and kept available for the community's benefit in reviewing previous sessions.

Current training documentation can be found here: Training

Learning Outcomes

- Understand the purpose of a repository
- Learn what Fedora can do for you
- Understand the key capabilities of the software

Course Outline

Introduction to Fedora 4

What is a Repository?

- Secure software that stores, preserves, and provides access to digital materials
- Supports complex semantic relationships between objects both within and outside the repository
- Supports millions of objects, both large and small
- Capable of interoperating with other applications and services

Fedora 4 Guiding Principles

- Improved performance, enhanced vertical and horizontal scalability
- More flexible storage options
- Features to accommodate research data management
- Better capabilities for participating in the world of linked open data
- An improved platform for developers—one that is easier to work with and which will attract a larger core of developers.

Exposing and Connecting Content with Fedora 4

- Flexible, extensible object modelling
- Atomic objects with semantic connections using standard ontologies
- RDF-based metadata using Linked Data
- RESTful API with native RDF response format

Core Components

Durable Storage

One of the core components of Fedora 4 is its long-term storage and preservation capability. A number of features support this capability; they have been grouped here under the notion of Durable Storage.

Fixity

- Over time, digital objects can become corrupt and unusable by suffering from bit rot and other digital preservation dangers
- Fixity checks help preserve digital objects by verifying their integrity using techniques such as checksumming
- On content ingest, Fedora can verify a user-provided checksum against the calculated value
- A checksum can be recalculated and compared at any time via a REST-API request

Backup and Restore

- A full backup, including all Datastreams as well as a compact serialization of all objects, can be performed at any time
- A full restore from a repository backup can be performed at any time

Export and Import

- A specific Fedora object, its children objects, and associated Datastreams can be exported
    - The serialization of the Fedora object is more portable than the compact form found in the backup/restore feature
    - Exported objects are serialized in a standard JCR/XML format
- An exported object or hierarchy of objects can be imported at any time

Versioning

- Versions can be created across the entire repository or on particular API calls.
- A previous version can be restored via the REST-API.

Policy-Driven Storage

- Different types of content can be routed to different back-end stores on ingest
- Policies can be written to route content based on properties (e.g. filetype)

Data Modelling

Nodes

- Both objects and datastreams are represented as nodes.
- Object nodes can have both Objects and Datastreams as children.
- The tree structure allows for inheritance of things like security policies.

Properties

- Nodes have a number of properties, which are expressed as RDF triples.
    - The node itself is the implicit subject of each triple.
- Properties can be RDF literals (e.g. dc:title) or they can express relationships both internal and external to the repository.
- Any number of RDF namespaces can be defined and used.

Content Models

- Content can be modelled using Compact Node Definitions (CNDs).
- Mixins can be used to define any number of properties. A mixin can be added to a CND to be applied to objects.
- An object can inherit properties from any number of mixins; their effects are cumulative.

Linked Data

- Fedora 4.0 is compliant with the LDP 1.0 spec.
- Metadata can be represented as RDF triples that point to objects outside the repository.
- Many possibilities for exposing, importing, sharing resources with other web applications.

User Interface

Administrative Console

Tour of the HTML administrative interface.

Internal Search

- Internal search can search across all node properties.
- It also functions as a limited SPARQL endpoint.

External Components

Indexing

- Indexing repository content for external applications can be accomplished by using the JMS Message Consumer web application.
    - This is just one possible implementation - different message consumer implementations could be written.
- The JMS Message Consumer receives JMS messages on repository updates and relays these messages to one or more external applications.
- Repository content needs to be assigned the rdf:type property "indexible" in order to be indexed.

Triplestore

- An external triplestore can be used to index the RDF triples of content managed by Fedora.
- Any triplestore that supports SPARQL-update can be used; Fuseki and Sesame have been tested.

External Search

- An external search application can be used to perform more complex search queries on repository content.
- Any search application that supports SPARQL-update can be used; Solr has been tested.

Authorization

- Authentication (not to be confused with authorization) is assumed to take place in a layer above the application.
- The authorization framework provides a plug-in point within the repository that calls out to an optional authorization enforcement module.
- Currently, two authorization implementations exist.

Basic Authorization

- Basic authorization compares the user's role(s) with an Access Control List (ACL) defined on a Fedora resource.
- ACLs can be inherited; if a given node does not have an associated ACL, Fedora will examine parent nodes until it finds one.

XACML Authorization

- XACML policies can provide much more complex and granular authorization.
- A default policy must be defined for the repository, and each node can override the default with another policy.
- A XACML policy referenced by a node will also apply to all the node's children, unless they define their own XACML policies that override the parent policy.

Performance

Transactions

- Multiple actions can be bundled together into a single repository event (transaction).
- Transactions offer performance benefits by cutting down on the number of times data is written to the repository filesystem (which tends to be the slowest action).

Clustering

- Two or more Fedora instances can be configured to work together in a cluster.
- Fedora 4 currently supports clustering for high-availability use cases.
    - A load balancer can be setup in front of two or more Fedora instances to evenly distribute read requests across each instance.
    - If one Fedora instance in the cluster goes down, read requests can be directed to the other instance.
    - Ingests are replicated across all instances in the cluster.