Filesystem Federation

- Overview
- Limits of Federation
- Filesystem Federation
    - Configuration
        - Repository Configuration
        - Multiple Directories
    - External Datastreams
    - Separate Properties Store
    - How-Tos
    - Other Connectors
        - Custom Connector References

Overview

Fedora 4 has the ability to expose external content, as if it were a part of the repository. Federation may be useful for migrating content into Fedora 4 or serving large files already on disk.

Modeshape's federation overview provides more background on how federation works and the underlying concepts.

Note: The term "projection" is sometimes used as a synonym for the "federation" feature.

Limits of Federation

Federated content is accessible through the REST API – however, some features will not work:

- Performance testing on federations has shown a performance impact when more than ~3000 files are found in any given filesystem directory. It is recommended that the files on the filesystem be structured in a balanced hierarchy of directories.
- Filesystem federation is read-only, currently.
- Access control can currently only be applied to an entire federated filesystem, not sub-directories/files within the resource.
- Transactions are not available.
- Versioning is not supported.

Filesystem Federation

Filesystem federation maps a node in the repository to a directory on disk. This allows files on disk to be served and updated by Fedora 4 as though they were in the repository. Filesystem federation circumvents the need to transfer files using HTTP – and with larger file sizes (or a large number of files), this feature can improve performance significantly. If you are ingesting a large number of multi-gigabyte files, we recommend you consider filesystem federation.

Another use for filesystem federation is interoperability with another system. If you have files on disk managed by another application or workflow, you can use filesystem federation to serve them with Fedora 4 without having to ingest them using the REST API or create another copy of the files.

Configuration

Once you have run Fedora with a federated filesystem configuration, do not change the configuration name ("`federated-directory`" in the example below), or federation will not be loaded correctly.

An example filesystem federation configuration to include in your Modeshape repository.json :

```
"externalSources" : {
    "federated-directory" : {
        "classname" : "org.fcrepo.connector.file.FedoraFileSystemConnector",
        "directoryPath" : "/path/to/files",
        "propertiesDirectoryPath" : "/path/to/external/properties",
        "projections" : [ "default:/federated => /" ],
        "contentBasedSha1" : "false",
        "readonly" : true,
        "extraPropertiesStorage" : "none",
        "cacheTtlSeconds" : 5
    }
},
```

- `directoryPath` - base directory for all files shared with the repository

- propertiesDirectoryPath - (optional) a path, that causes computed properties to be stored in an external file structure
- `projections` - lists one or more mappings from the repository to the filesystem. The format is "{workspace}:{repository path} => {path relative to `directoryPath`}". See the section "Multiple Directories" below for how to handle multiple mappings.
- `contentBasedSha1` - controls how internal identifiers are computed for files. By default (`contentBaseSha1` = true), Modeshape computes the SHA-1 checksum of a file's content every time the file is accessed. For small files this creates a modest overhead. For large files, however, this dramatically reduces performance, since generating the checksum can take several seconds per gigabyte of data. For this reason, we recommend setting `contentBasedSha1` to false when serving files larger than 100MB
- `readonly` - controls whether the contents of the filesbase directory for all files shared with the repository are read-only (⚠️ currently read-only is the only supported mode)
- `extraPropertiesStorage` - sets the format for storing "extra" properties (properties that can't be set using filesystem attributes). Recommended values are "json" for the current JSON properties format, or "none" for disabling extra properties. This property is ignored if propertiesDirectoryPath is set, since an external properties store will be used. (A warning or notice will appear in the logs indicating that the asserted preference here is overridden.)
- `cacheTtlSeconds` - the maximum time that cached entr ies are held before being refreshed. Setting to a low value will make changes to the filesystem (like adding new files) show up more quickly in the REST API. Setting to a higher value will improve performance for files that don't change often.

Modeshape's FileSystemConnector documentation and configuration provide additional information about configuring the filesystem connector.

Repository Configuration

The most direct way of customizing the repository configuration ('repository.json') file is to specify the location of an external configuration to the servlet container at startup. See the configuration documentation ("Configuration Elements -> fcrepo.modeshape.configuration") for details.

Multiple Directories

If you want to map multiple directories, the first entry in the projections array should map the parent directory (i.e, the directory in `directoryPath`). Subse quent entries can map subdirectories to other repository paths. For example, if you have a directory `/pub/` that contains two directories (`/pub /project1/` and `/pub/project2/`) which you want to map the `project1` and `project2` directories to the top level of the repository:

```
"externalSources" : {
    "federated-1" : {
        "classname" : "org.fcrepo.connector.file.FedoraFileSystemConnector",
        "directoryPath" : "/pub",
        "projections" : [ "default:/pub => /", "default:/project1 => /project1", "default:/project2 =>
/project2" ],
        "contentBasedSha1" : "false",
        "readonly" : true,
        "extraPropertiesStorage" : "none",
        "cacheTtlSeconds" : 5
    }
},
```

This configuration would provide the following mappings:

Repository URL	Filesystem Path
http://localhost:8080/rest/pub/	/pub/
http://localhost:8080/rest/project1/	/pub/project1/
http://localhost:8080/rest/project2/	/pub/project2/

External Datastreams

One use case for filesystem federation is to store objects and metadata in the repository, but link to large files on disk instead of ingesting them as datastream content. To do this, create an external content binary, with a a URL that resolves to the federated filesystem:

```
Content-Type: message/external-body; access-type=URL; URL="http://localhost:8080/rest/federated/file"
```

For more information, see External Content and Example 4 of the RESTful HTTP API - Containers PUT method documentation.

Separate Properties Store

For a connector, even a read-only connector, some properties are computed and stored. These include a modification date and a checksum (if enabled) which is only recomputed when the file is altered. If it is not desirable or possible to store these properties on the same file system path that is being projected over, you may specify an alternate location (propertiesDirectoryPath) in which those properties will be cached. This is especially important when using checksums in directories of large files.

How-Tos

- External Content
- How to federate over a filesystem that is updated externally
- How to audit fixity in a filesystem federation
- How to incorporate static filesystem resources into your repository

Other Connectors

In addition to the filesystem connector, Modeshape includes several other connectors, for federating content from Git, CMIS repositories, and relational databases.

Custom connectors can also be developed to support other system. The filesystem connector is a good reference implementation, particularly for file-based resources.

Custom Connector References

- Modeshape Connector reference
- FileSystemConnector JavaDoc reference
- Sample Connector Project