Backup and Restore

NOTE: The endpoints discussed below - /fcr:backup and /fcr:restore - will be moving to an extension module in a future release of Fedora. The Backup Format (also discussed below) will change at that time.

Overview

The Fedora Backup capability allows a user, such as the repository manager, to make a REST call to have the repository binaries and metadata exported to the local file system. The Restore capability allows a user to make a REST call to have the repository restored from the contents of a previous Backup operation. In addition, with the default configuration, files are stored on disk named according to their SHA1 digest, so a filesystem backup approach can also be used.

Design Considerations

Historically, Fedora married the concepts of persistence and durability by choosing transparent forms of persistence. Fedora 3 wrote human-readable XML to disk, which could be indexed and manipulated by the Fedora repository software but which systems beyond the repository could also readily penetrate if needed. Transparency in support of durability is as valid a principle as ever, but there is a weakness to this strategy: transparent forms of persistence are not performant. Additionally, many users didn't particularly care for that principle, but they still incurred the performance costs associated with it. So in the move from Fedora 3 we divorced the two concepts, moving responsibility for transparent persistence away from the core repository software.

Today, the operational form of persistence used by the core Fedora repository component - essentially a specially designed table and index within a relational database system for metadata and a curated filesystem structure for binaries - is not meant to be manipulated directly by a human except in the most unusual situations. It's designed, instead, for use by the software to provide as performant a response as possible for the repository's defined API. Comparison to a traditional RDBMS is appropriate: if you are concerned for the durability of your data in a relational database, you export/dump backups in a transparent format to disk and use *those* to ensure durability. You would never attempt to run your system based exclusively on a disk backup/dump sitting on disk. The Backup procedure discussed on this page serializes the metadata and binaries from persistence layer to disk principally to support the Restore procedure into a repository-specific persistence layer.

If you like to maintain transparent persistence - an export that is a simple and human-readable form of your repository - this is supported with an integration around the core.

Usage

Backup

If a POST body specifying a writeable directory (local to Fedora server) is not included in the request, the backup will be written to the system temp directory.

Perform a backup of a running Fedora repository

Request

```
POST /rest/fcr:backup

> optional POST body pointing to the local filesystem backup directory to which the export will be written.
```

Response

On success

- HTTP/1.1 200 OK
- Path where the backup was written

Response body

- Absolute path of local backup directory

Restore

Note: Restoring a backup replaces the repository content with the contents of the backup, so any data in the repository will be lost.

Perform a restore of a running Fedora repository

Request

```
POST /rest/fcr:restore

> optional POST body pointing to the local filesystem backup directory from which the export will be read.
```

Response

On success

- HTTP/1.1 204 No Content

Backup Format

Regardless of the repository configuration, the output of the backup process creates resources of the same format. Further details on backup contents and the underlying implementation can be found in ModeShape's documentation.

The backup directory will contain

- 'binaries' directory that contains the repository "content" binaries stored in a pair-tree like structure. The filename of the binary is the SHA-1 of the content with the extension '.bin'. The directory structure in which each binary is found is three levels deep based on the SHA-1.
    - For example, binary content in the repository with a SHA-1 of "5613537644c4d081c1dc3530fdb1a2fe843e570170d3d054", would look like

      ```
      binaries
          44
              c4
                  d0
                      44c4d081c1dc3530fdb1a2fe843e570170d3d054.bin
      ```

- One or more "documents_00000n.bin.gz" files which contains a concatenated listing of the metadata for each of the repository objects in JSON format
    - For example

```
{ "metadata" :
  { "id" : "87a0a8c317f1e7/jcr:system/jcr:nodeTypes/nt:unstructured//undefined/1" ,
    "contentType" : "application/json" } ,
  "content" :
  { "key" : "87a0a8c317f1e7/jcr:system/jcr:nodeTypes/nt:unstructured//undefined/1" ,
    "parent" : "87a0a8c317f1e7/jcr:system/jcr:nodeTypes/nt:unstructured" ,
    "properties" :
    { "http://www.jcp.org/jcr/1.0" :
      { "primaryType" :
        { "$name" : "nt:propertyDefinition" } ,
        "onParentVersion" : "COPY" ,
        "multiple" : false ,
        "protected" : false ,
        "availableQueryOperators" :
          [ "jcr.operator.equal.to" ,
            "jcr.operator.greater.than" ,
            "jcr.operator.greater.than.or.equal.to" ,
            "jcr.operator.less.than" ,
            "jcr.operator.less.than.or.equal.to" ,
            "jcr.operator.like" ,
            "jcr.operator.not.equal.to" ] ,
        "requiredType" : "UNDEFINED" ,
        "mandatory" : false ,
        "autoCreated" : false }
    }
  }
}
```

Filesystem Backup

By default, files larger than 4KB are stored on disk named after their SHA1 digest, in the directory `fcrepo.binary.directory`. (4KB is the default, but can be changed by updating the `minimumBinarySizeInBytes` property in repository.json). That is, a file with the SHA1 "a1b2c369563c0465ab82cdb2789d45ce1c3585b1" would be stored on disk in `/path/to/fcrepo4-data/fcrepo.binary.directory/a1/b2/c3/a1b2c369563c0465ab82cdb2789d45ce1c3585b1`. So files in the repository can be backed up backing up the directory `fcrepo.binary.directory`.