Contents

➔ Introduction
➔ Features
➔ Data Model
➔ Application Development
What is Jackrabbit?

• Open Source content repository for Java.
• Hierarchical data storage
• Full implementation of JCR specification.
• JCR specified in JSR 170 and JSR 283
Jackrabbit history

2004: Apache Jackrabbit started, 8 committers

2006: Jackrabbit 1.0 released, 13 committers

2008: Jackrabbit 1.4 released, 20 committers

2010: Jackrabbit 2.0 released, 25 committers

Today: Jackrabbit 2.3 and 3.0 being developed, 29 committers, estimated 79 person-years of combined effort over time
Java Content Repository

- An API for accessing content repository (Jackrabbit)
- Content Repository
- Generic Application Data Store (binary and text data)
- How data is actually stored does not matter.
- Provides storage and retrieval of data.
JCR example application

- “How to implement a music store or library with JCR?”
- Designed to showcase JCR features and best practices
- Store and manage individual “tunes”, optionally organized in albums, etc.
- Support alternative views like predefined genres, or more ad-hoc searches
- Integrated handling of reviews, cover images, and other related content
- Staged publishing and timed releases of tunes or albums
- Personalization for things like settings, favorites, personal play-lists, etc.
- Extensibility and flexibility
Jackrabbit Architecture
How Jackrabbit works?
Contents

➔ Introduction
➔ Features
➔ Data Model
➔ Application Development
Features Of Jackrabbit

• Fine and coarse-grained content access
• Structured content/Unstructured Content
• Binary Property
• Query
• Import
• Export
Features Continued

- Referential integrity
- Transactions
- Events
- Locking
- Clustering
- Multiple persistence models
Searching

- Content is indexed
- Configurable per repository
- Support for full text search
- Also binaries indexed with automatic text extraction

Example
SELECT * FROM slingshot:photo WHERE jcr:path LIKE '/slingshot/%' AND jcr:descrNon CONTAINS 'vancouver'
Index Readers

- Jackrabbit uses Lucene as the underlying index implementation.
- The extensions also cover features that are not supported by Lucene, like hierarchical queries.
Authentication and Access Control

- Allow/Deny access on a node
- Allows structuring based on access rights
Versioning

- Applications today require versioned data:
- Given a data item, an application must be able to access the current as well as all past versions of that data item.
- Neither relational nor object databases provide standard versioning.
- Keeping track of versions, and making those versions available to applications, is an important repository feature.
Versioning

● To make a node versionable, add the `mix:versionable` mixin
● scope of “versionability” determined by node types
● A checkin freezes a piece of content and makes a copy of it in the version history
● A checkout unfreezes the content and allows it be modified
● A restore goes back in time to a previously checked in version
● A merge combines changes from another workspace to those made in this workspace
Data Model

- Repository
- Workspace
- Node
- Property
Repository

- Represents the entire content storage.
- Similar to a database server, or a filesystem.
- Consists of one or more workspaces.
Workspace

- Analogous to database, or additional file systems.
- All repositories have a “default” workspace.
Session

- All content access goes through a session
- Sessions are created with authenticated login() call
- Session based authorization of reads, writes
- Tracking of transient changes
- Use multiple sessions for concurrent operations
Node

- A node is used to build the path of the data and is similar in concepts to databases, tables and fields.
- Directories and files in a file system. A node can have one primary node type (single inheritance). A node can optionally expose additional behaviors via Mixin types (implement interfaces).
Node Types

**Primary type:**

*nt:unstructured* - Most commonly used node type.

*nt:base* - It specifies the least minimum of content that node must have.

*nt:folder* - Represents a directory.

*nt:file* - Represents the contents of a file. Stores binary content such as images, documents.

*jcr:content* - Mandatory child node of type *nt:resource* that stores the actual binary content.

*jcr:mimeType* - Property for MIME type of the binary data.

*jcr:encoding* - Property used to represent any content encoding scheme for the binary data.

*jcr:data* - Contains the binary data.

*jcr:lastModified* - Represents last modification time of the content.
Example
Node Types Continued

**Mixin Type:**
- **mix:referenceable** - Required if the node is to be referenced from other properties. Adds a UUID property to the node.
- **mix:versionable** - Added for use of revision control features for a node.
- **mix:lockable** - To lock/unlock a node to prevent concurrent modifications.
Property

• Represents data associated with a node.
• Conceptually similar to attributes or simple child elements.
• Equivalent to fields, database columns.
• Can be single or multi-valued. Properties can reference other nodes/properties.
• Referential integrity checks apply.
Property Types

Constants defined in javax.jcr.PropertyType.

- LONG
- DOUBLE
- BINARY - Represents binary data.
- BOOLEAN
- DATE - Stored as a calendar object.
- STRING PATH - Stored as a string and represents the path to another node. Does not enforce referential integrity.
- NODE - Only mix:referenceable nodes may be referenced
Multi-Valued Properties

- Limit at around 10-100k values, depending on size of values
- All values must be of the same type
- Duplicates allowed
- No “null” values
- Automatically removed
- Order is preserved
Contents

➔ Introduction
➔ Features
➔ Data Model
➔ Application Development
Application Development

- Apache Sling
- Configuration
- Retrieval
WORKING WITH REPOSITORY

Login into workspace

Credentials myCredentials = new SimpleCredentials("USERID", "PASSWORD".toCharArray());

Session mySession = repository.login(myCredentials, "WORKSPACE");
Changing the Content

- Add/Remove nodes
- Add/Remove/Change properNes
- Transient space
- Then save
  // change
  albumNode.addNode("newAlbum");
  europaNode.setProperty("jcr:description", "something");
  // save
  mySession.save();
  // or revert all changes
  mySession.refresh(false);
import javax.jcr.Session;
import javax.jcr.Node;
import javax.jcr.NodeType;
import static org.apache.jackrabbit.JcrConstants.*;
javax.jcr.Session; javax.jcr.Node; javax.jcr.NodeType;
Node root = session.getRootNode();
Node node1 = root.getNode("parent");
TRAVERSING THE CONTENT

From the root or any node

Node rootNode = mySession.getRootNode();
Node albumNode = rootNode.getNode("slingshot/albums/travel");
Node europeNode = albumNode.getNode("Europe");
Retrieving the Property

```java
Property prop = albumNode.getProperty("jcr:description");
Value value = prop.getValue();
String desc = value.getString();
value.getBoolean();
value.getStream();
value.getLong();
value.getDate();
value.getDouble();
```
Retrieval

JCR version 2:

- **JCR_JQOM**: `Source` `QueryObjectModel.getSource()`
  - `Constraint` `QueryObjectModel.getConstraint()`
  - `Ordering[]` `QueryObjectModel.getOrderings()`
  - `Column[]` `QueryObjectModel.getColumns()`

- **JCR_SQL2**: `Query ::= 'SELECT' columns
  'FROM' Source
  ['WHERE' Constraint]
  ['ORDER BY' orderings]`
Issues with Jackrabbit

Ways to handle concurrent edits:
   1. Merge changes
   2. Fail conflicting changes
   3. Block concurrent changes

Jackrabbit does 1 by default, 2 when merge fails. Explicitly opt for 3 by using the JCR locking feature.

 Doesn't support very flat content hierarchies. It means problems will occur when you try to put more than 10k child nodes under a single parent node.
Improved Agility

- Elasticity: Ease of adding or removing nodes in distributed data store
- Agility
- JackRabbit offers increased agility with elasticity.
- Hadoop can easily grow its number of servers which Jackrabbit also can. But to cut down the number of servers, Hadoop faces a huge problem.
Comparison with RDBMS

- is hierarchical
- is flexible
- uses a standard Java API (e.g., javax.jcr)
- abstracts where the information is really stored
- supports queries and full-text search out of the box
THANK YOU