JBoss DNA

November 2008

Randall Hauch
Project Lead

Content Repositories

Content Repositories

- **Provide**
  - Hierarchical graph-based storage
  - Flexible/extensible schema (as needed)
  - Versioning, events, and access control
  - Search and query
  - Metadata
  - Multiple persistence choices

Content Repositories

- **Provide**
  - Hierarchical graph-based storage
  - Flexible/extensible schema (as needed)
  - Versioning, events, and access control
  - Search and query
  - Metadata
  - Multiple persistence choices
- **Used for**
  - websites and applications
  - content management
  - document storage
  - multi-media files

JSR-170 (and JSR-283)

JSR-170 (and JSR-283)

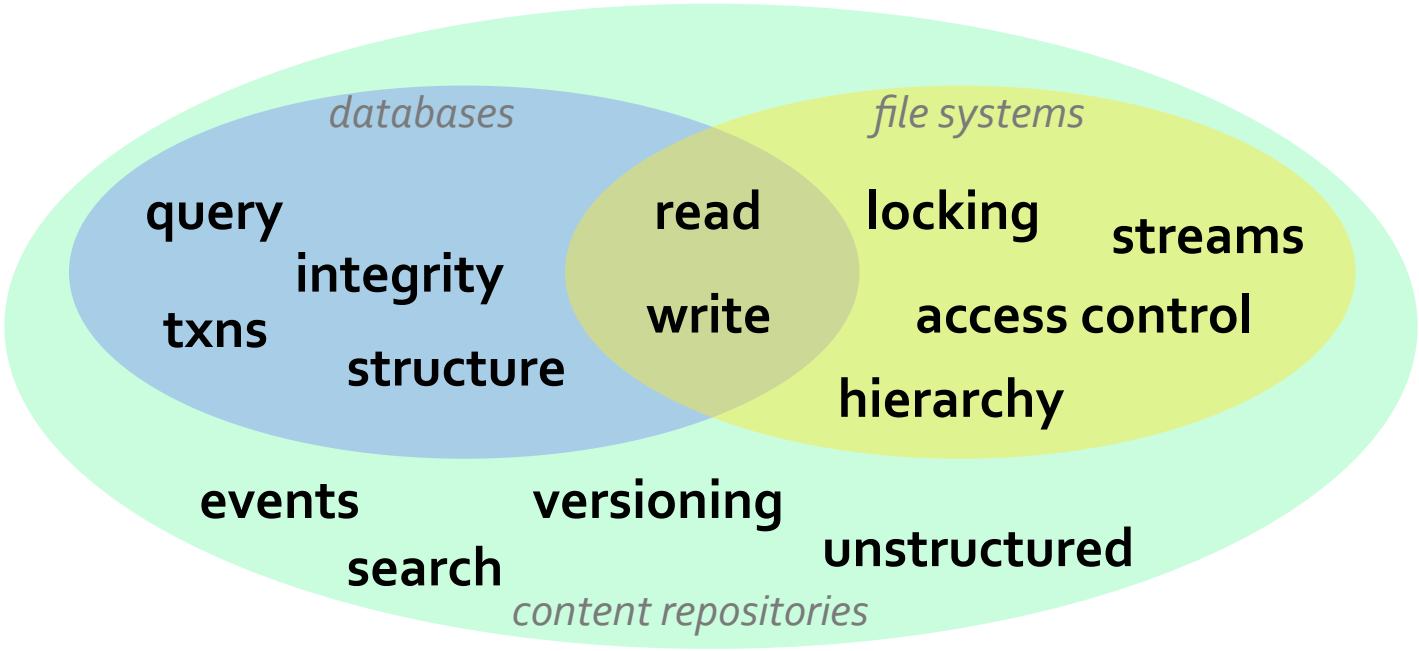
- **Standard Java API for content repositories**
  - "Content Repository API for Java" (a.k.a. "JCR")
  - `javax.jcr`

JSR-170 (and JSR-283)

- **Standard Java API for content repositories**
  - "Content Repository API for Java" (a.k.a. "JCR")
  - `javax.jcr`
- **Access content while hiding persistence layer**

JSR-170 (and JSR-283)

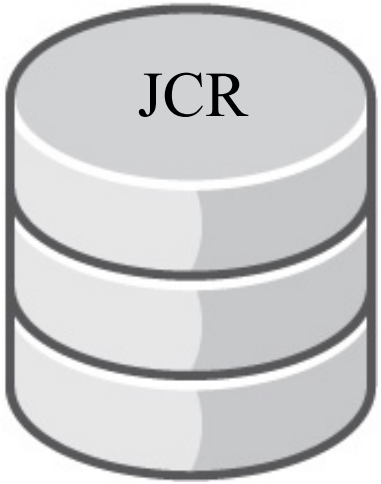
- **Standard Java API for content repositories**
  - "Content Repository API for Java" (a.k.a. "JCR")
  - `javax.jcr`
- **Access content while hiding persistence layer**
- **Offer best features of different layers**



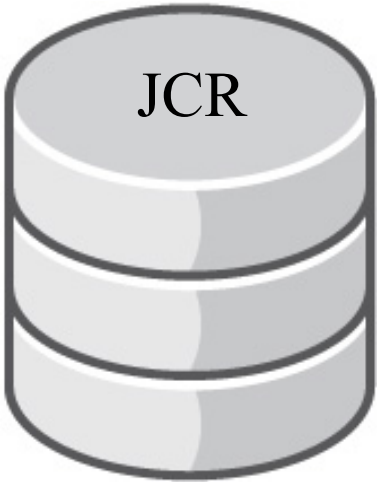
Who uses JCR?

... and many more

Primary JCR interfaces

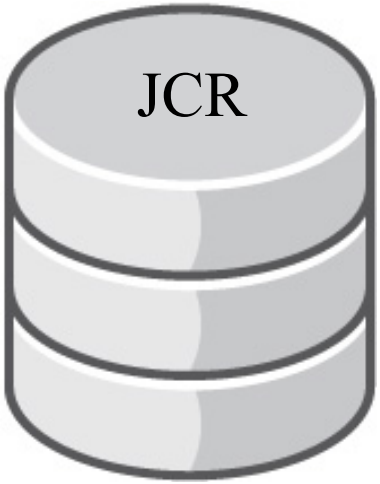


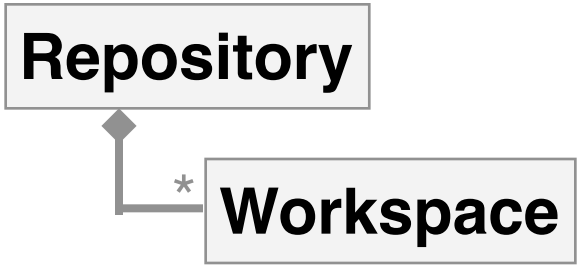
Primary JCR interfaces



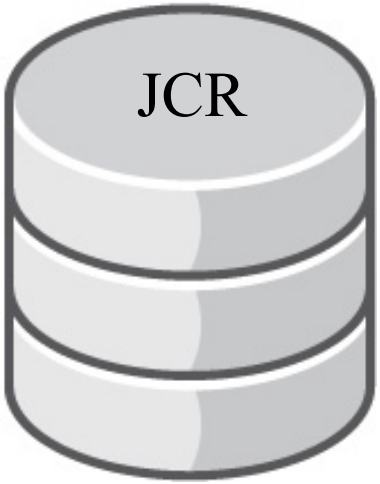
Repository

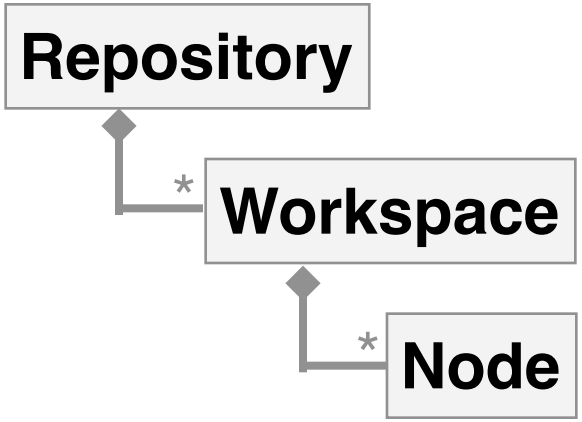
Primary JCR interfaces



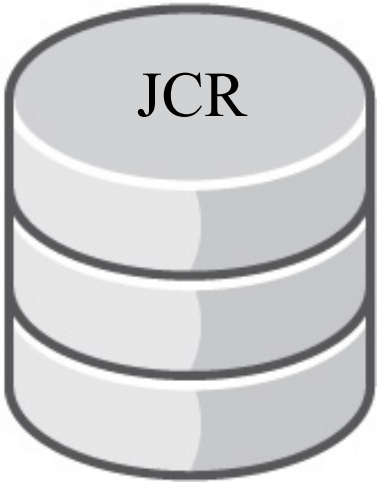


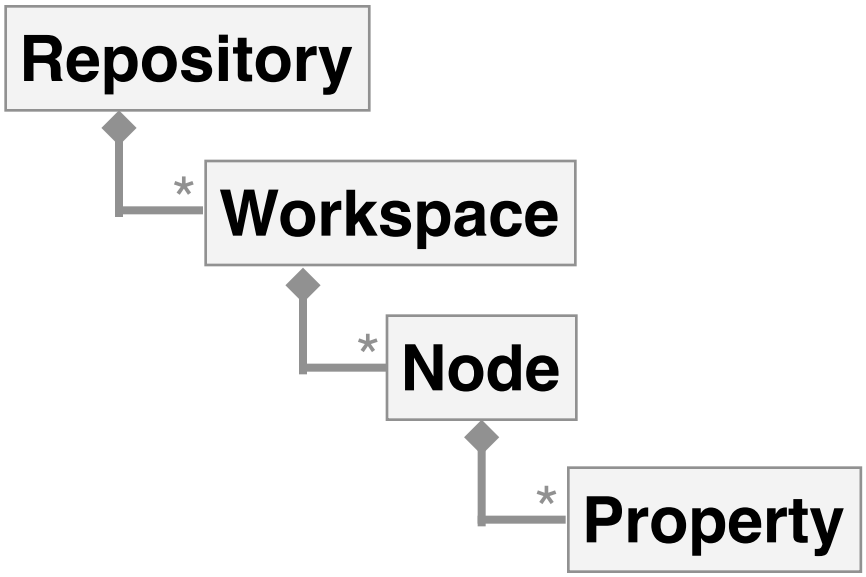
Primary JCR interfaces



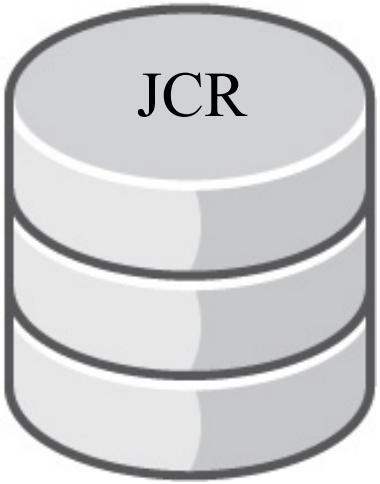


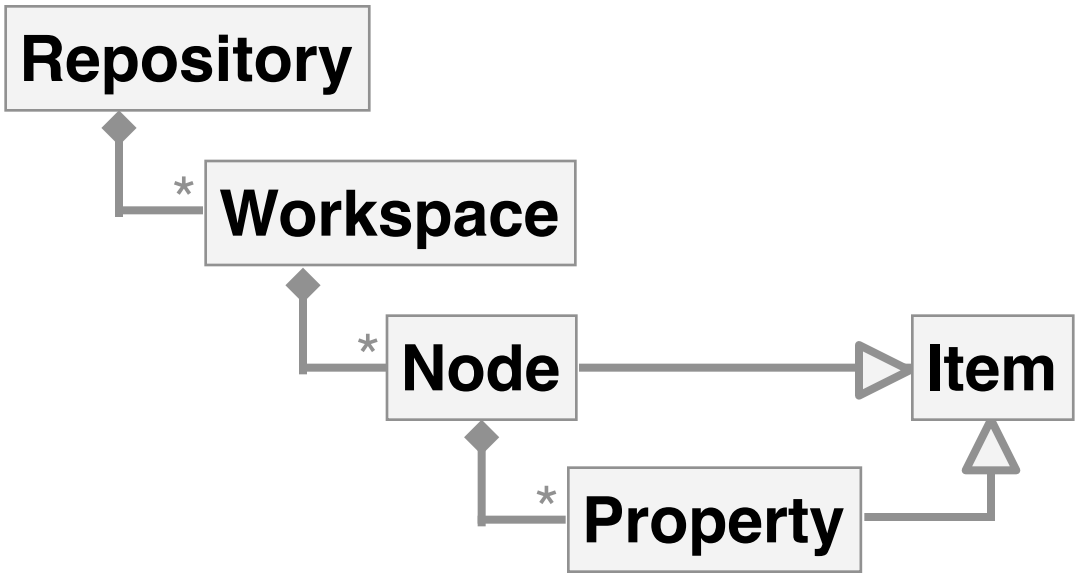
Primary JCR interfaces



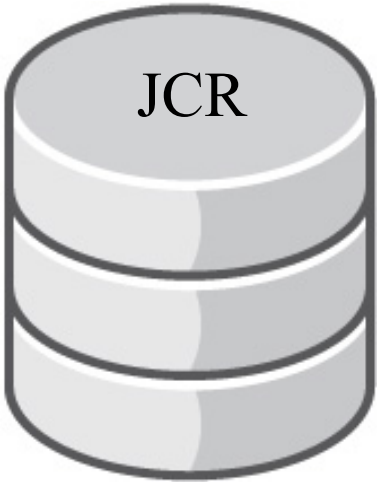


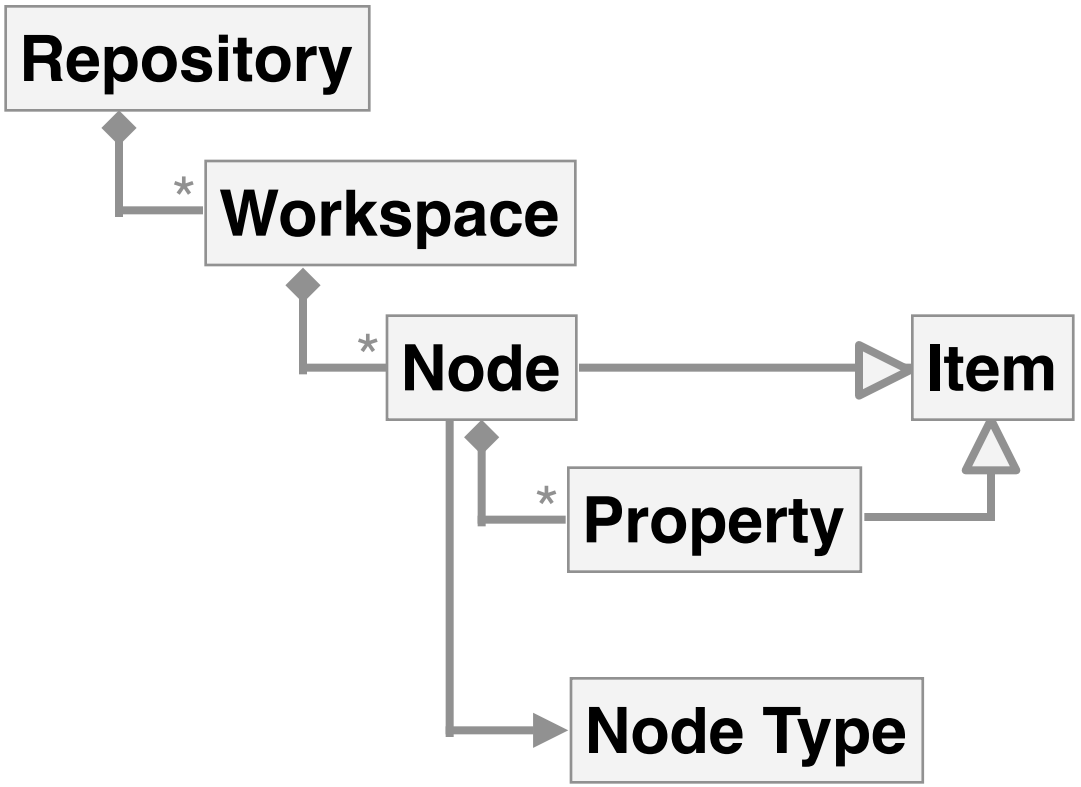
Primary JCR interfaces



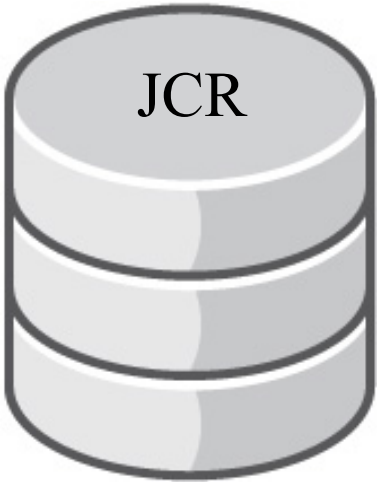


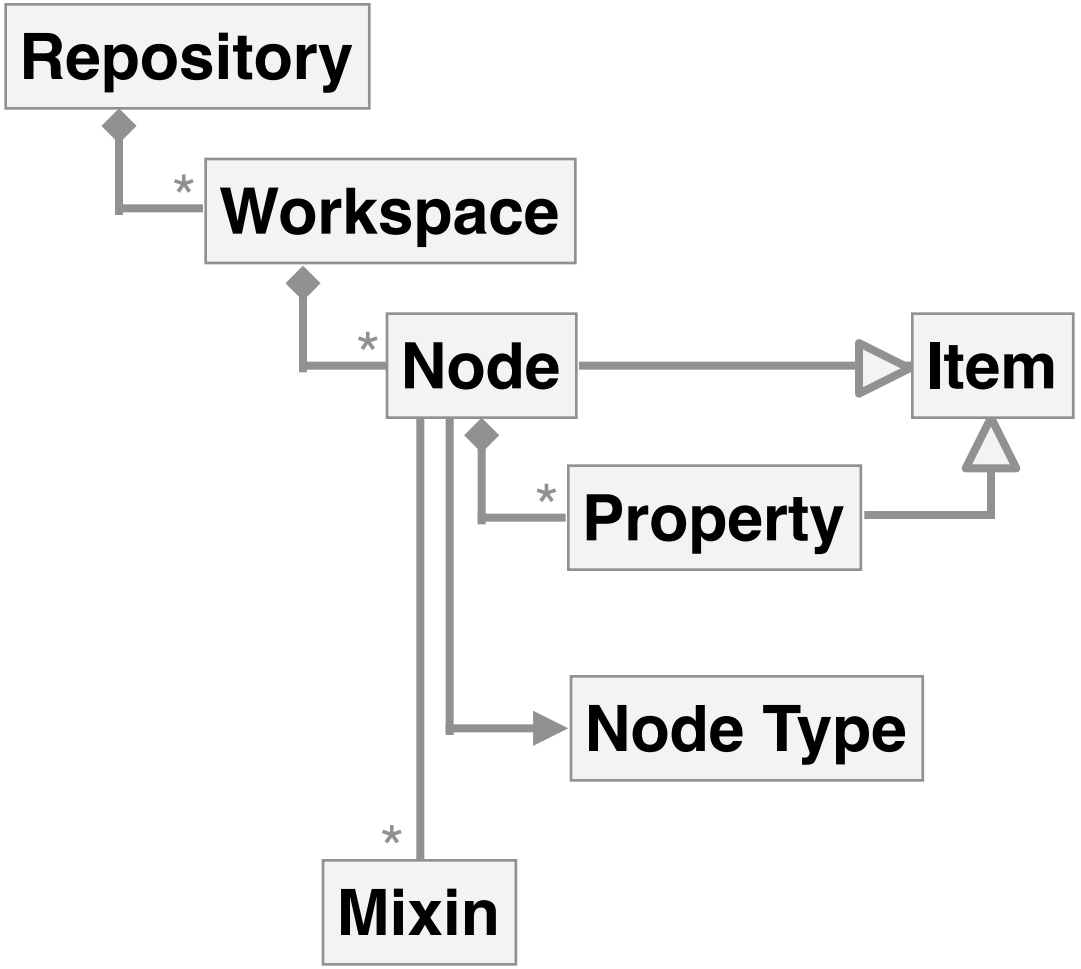
Primary JCR interfaces



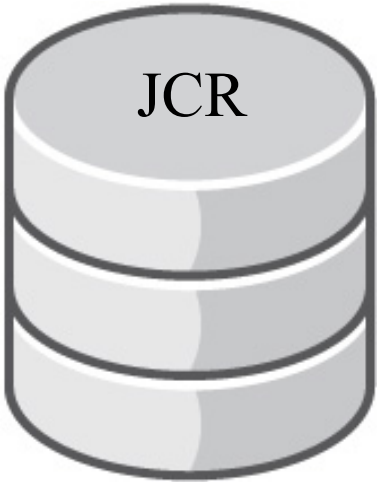


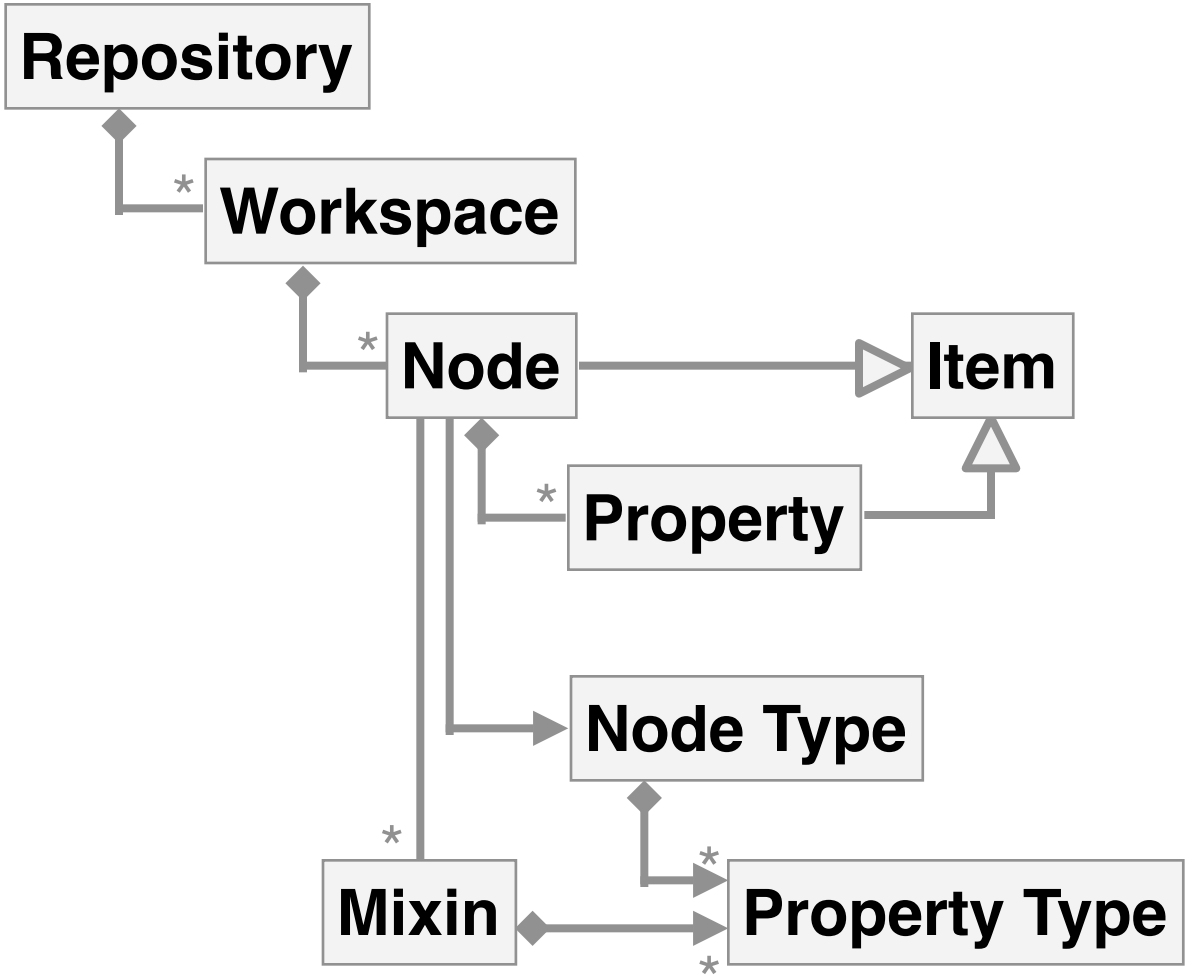
Primary JCR interfaces



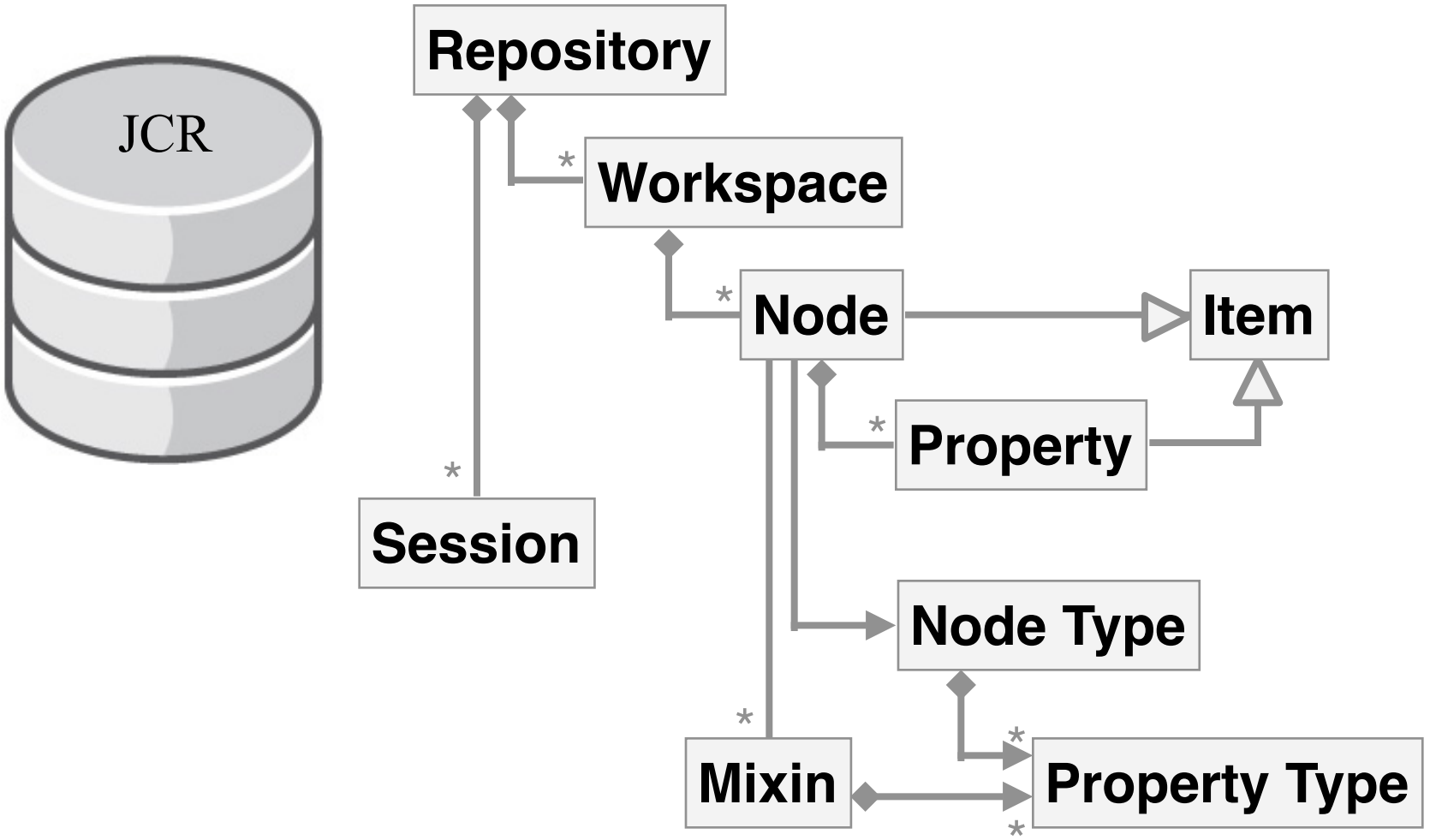


Primary JCR interfaces

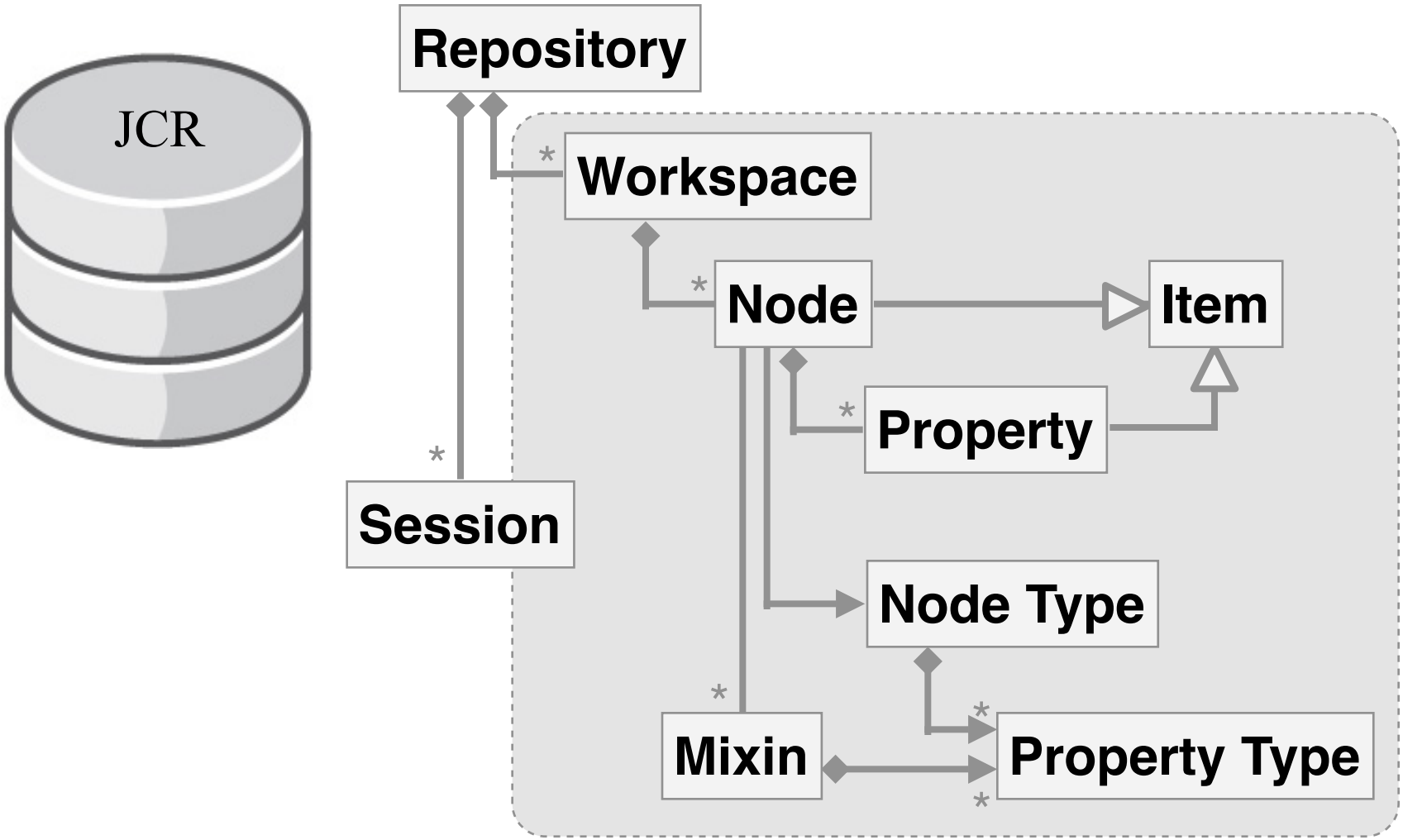




Primary JCR interfaces



Primary JCR interfaces



More JCR concepts

- **Nodes**
  - have names, including same-name siblings
  - are referenced by path of names from root
- **Namespaces**
  - isolate names of nodes, properties, node types, and mixins
- **Events**
  - allow sessions to observe changes in content
  - can be filtered by change type or location
- **Versioning**
  - of nodes and subgraphs as they change
  - policy defined on each node by adding mixin (“`mix:versionable`” or “`mix:simpleVersionable`”)

More JCR concepts

- **Searching**
  - enables full-text search
  - can use special functions and XPath-like criteria
- **Querying**
  - is via SQL-like grammar
  - will change in JCR 2.0 to allow other grammars
- **Security**
  - can leverage JAAS
  - will improve in JCR 2.0 with better access controls
- **Transactions**
  - relies upon JTA and JTS

Let’s see some code

Create or get the Repository instance

```
Repository repository = ...  // okay, it’s implementation dependent
```

Create, use and close a Session

```
Credentials credentials = new SimpleCredentials("jsmith", "secret".toCharArray());
String workspaceName = "My Repository";
Session session = repository.login(credentials, workspaceName);
try {
   // Use the session to access the repository content
} finally {
   if ( session != null ) session.logout();
}
```

Let's see some code

Create or get the Repository instance

```
Repository repository = ...  // okay, it's implementation dependent
```

Create, use and close a Session

```
Credentials credentials = new SimpleCredentials("jsmith", "secret".toCharArray());
String workspaceName = "My Repository";
Session session = repository.login(credentials, workspaceName);
try {
    // Use the session to access the repository content
} finally {
    if ( session != null ) session.logout();
}
```

Commentary

- Getting a hold of Repository instances is implementation dependent
- Other credential options (e.g., using JAAS) are implementation dependent
- Creating sessions (or "connections") requires knowledge of credentials, making pooling difficult

Work with content

Getting nodes by path

```
Node root = session.getRootNode();
Node node = root.getNode("autos/sports cars/Toyota/2008/Prius");
```

Getting the children of a node

```
for (NodeIterator iter = node.getNodes(); iter.hasNext();) {
    Node child = iter.nextNode();
    // Do something fun
}
```

Creating nodes

```
Node ford = root.addNode("autos/sports cars/Ford");
Node ford08 = ford.addNode("2008");
Node volt = root.addNode("autos/sports cars/Chevy").addNode("2010").addNode("Volt","car");
```

Mixins

```
ford08.addMixin("car:year");
ford08.removeMixin("car:year");
```

Work with content

Getting nodes by path

```
Node root = session.getRootNode();
Node node = root.getNode("autos/sports cars/Toyota/2008/Prius");
```

Getting the children of a node

```
for (NodeIterator iter = node.getNodes(); iter.hasNext();) {
    Node child = iter.nextNode();
    // Do something fun
}
```

Creating nodes

```
Node ford = root.addNode("autos/sports cars/Ford");
Node ford08 = ford.addNode("2008");
Node volt = root.addNode("autos/sports cars/Chevy").addNode("2010").addNode("Volt","car");
```

Mixins

```
ford08.addMixin("car:year");
ford08.removeMixin("car:year");
```

Commentary

- Unfortunately JCR doesn't use generics
- Cannot create node if parent doesn't exist

Work with content

Reading a property

```
Property property = node.getProperty("engineSize");
String[] engineSize = null;
// Must call either 'getValue()' or 'getValues()' depending upon # of values!
if (property.getDefinition().isMultiple()) {
    Value[] jcrValues = property.getValues();
    engineSize = new String[jcrValues.length];
    for (int i = 0; i < jcrValues.length; i++) {
        engineSize[i] = jcrValues[i].getString();
    }
} else {
    engineSize = new String[] {property.getValue().getString()};
}
```

Setting a property

```
Property property = node.getProperty("engineSize");
property.setValue("V4 Hybrid");
// Or set directly via the node
node.setProperty("mpgCity",48);
```

Work with content

Reading a property

```
Property property = node.getProperty("engineSize");
String[] engineSize = null;
// Must call either 'getValue()' or 'getValues()' depending upon # of values!
if (property.getDefinition().isMultiple()) {
    Value[] jcrValues = property.getValues();
    engineSize = new String[jcrValues.length];
    for (int i = 0; i < jcrValues.length; i++) {
        engineSize[i] = jcrValues[i].getString();
    }
} else {
    engineSize = new String[] {property.getValue().getString()};
}
```

Setting a property

```
Property property = node.getProperty("engineSize");
property.setValue("V4 Hybrid");
// Or set directly via the node
node.setProperty("mpgCity",48);
```

Commentary

- Getting the property values could be ***way*** less verbose
- But `Value` does have methods to get the values in the Java types I want

Work with content (continued)

Reading all properties

```
for (PropertyIterator iter = node.getProperties(); iter.hasNext();) {
    Property property = iter.nextProperty();
    // Same as before
}
```

Reading some properties

```
for (PropertyIterator iter = node.getProperties("jcr:*|*Mpg"); iter.hasNext();) {
    Property property = iter.nextProperty();
    // Same as before
}
```

Visiting nodes and properties

```
node.accept( new ItemVisitor() {
    public void visit(Property property) throws RepositoryException {
        // Do something with the property
    }
    public void visit(Node node) throws RepositoryException {
        // Do something with the node
    }
});
```

Work with content (continued)

Reading all properties

```
for (PropertyIterator iter = node.getProperties(); iter.hasNext();) {
    Property property = iter.nextProperty();
    // Same as before
}
```

Reading some properties

```
for (PropertyIterator iter = node.getProperties("jcr:*|*Mpg"); iter.hasNext();) {
    Property property = iter.nextProperty();
    // Same as before
}
```

Visiting nodes and properties

```
node.accept( new ItemVisitor() {
    public void visit(Property property) throws RepositoryException {
        // Do something with the property
    }
    public void visit(Node node) throws RepositoryException {
        // Do something with the node
    }
});
```

Commentary

- Again, generic iterators would simplify things

David's rules for content modeling

http://wiki.apache.org/jackrabbit/DavidsModel

Rule #1: Data First, Structure Later. Maybe.

Rule #2: Drive the content hierarchy, don't let it happen.

Rule #3: Workspaces are for clone(), merge() and update().

Rule #4: Beware of Same Name Siblings.

Rule #5: References considered harmful.

Rule #6: Files are Files are Files.

Rule #7: ID's are evil.

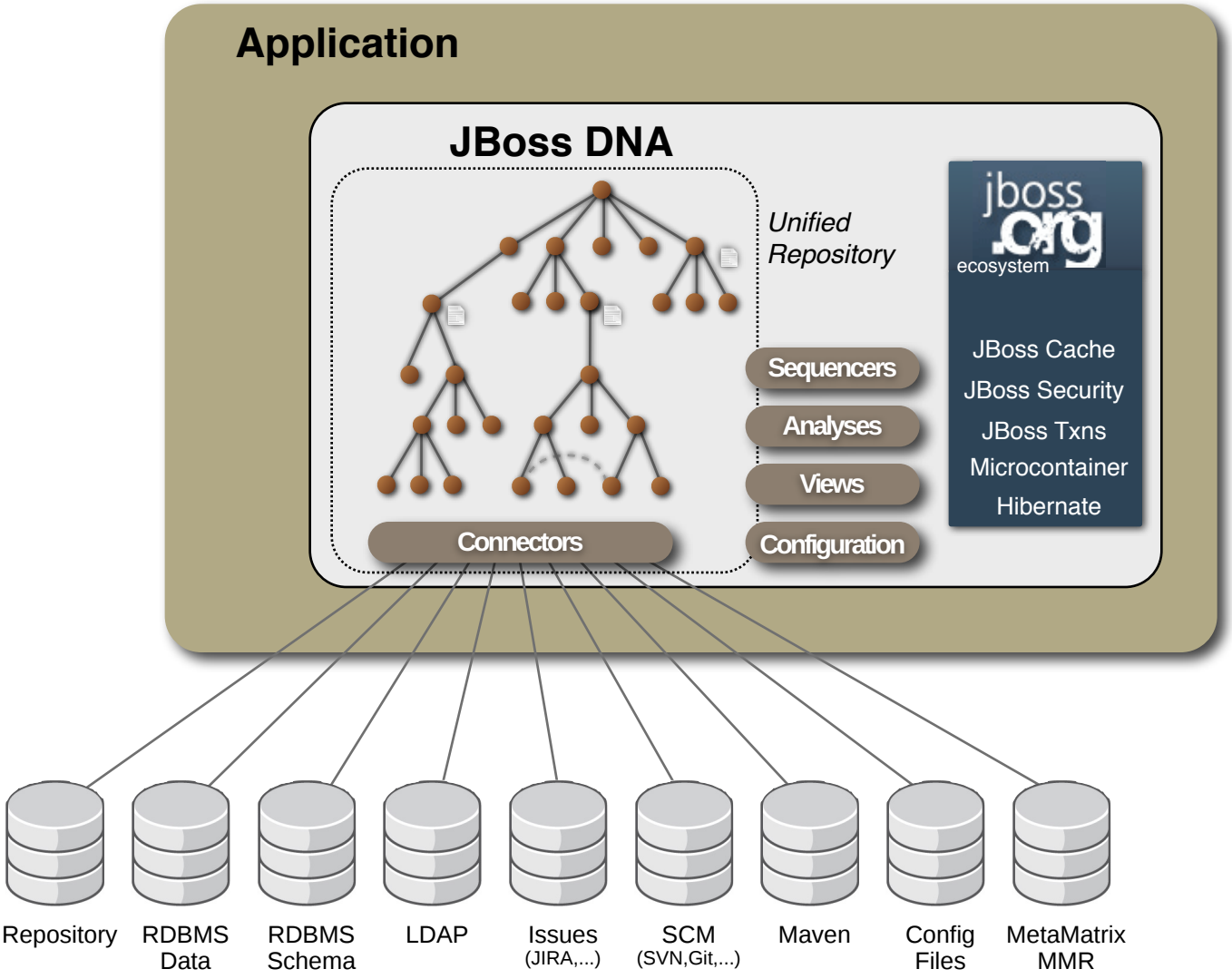
JBoss
a division of Red Hat

JBoss DNA

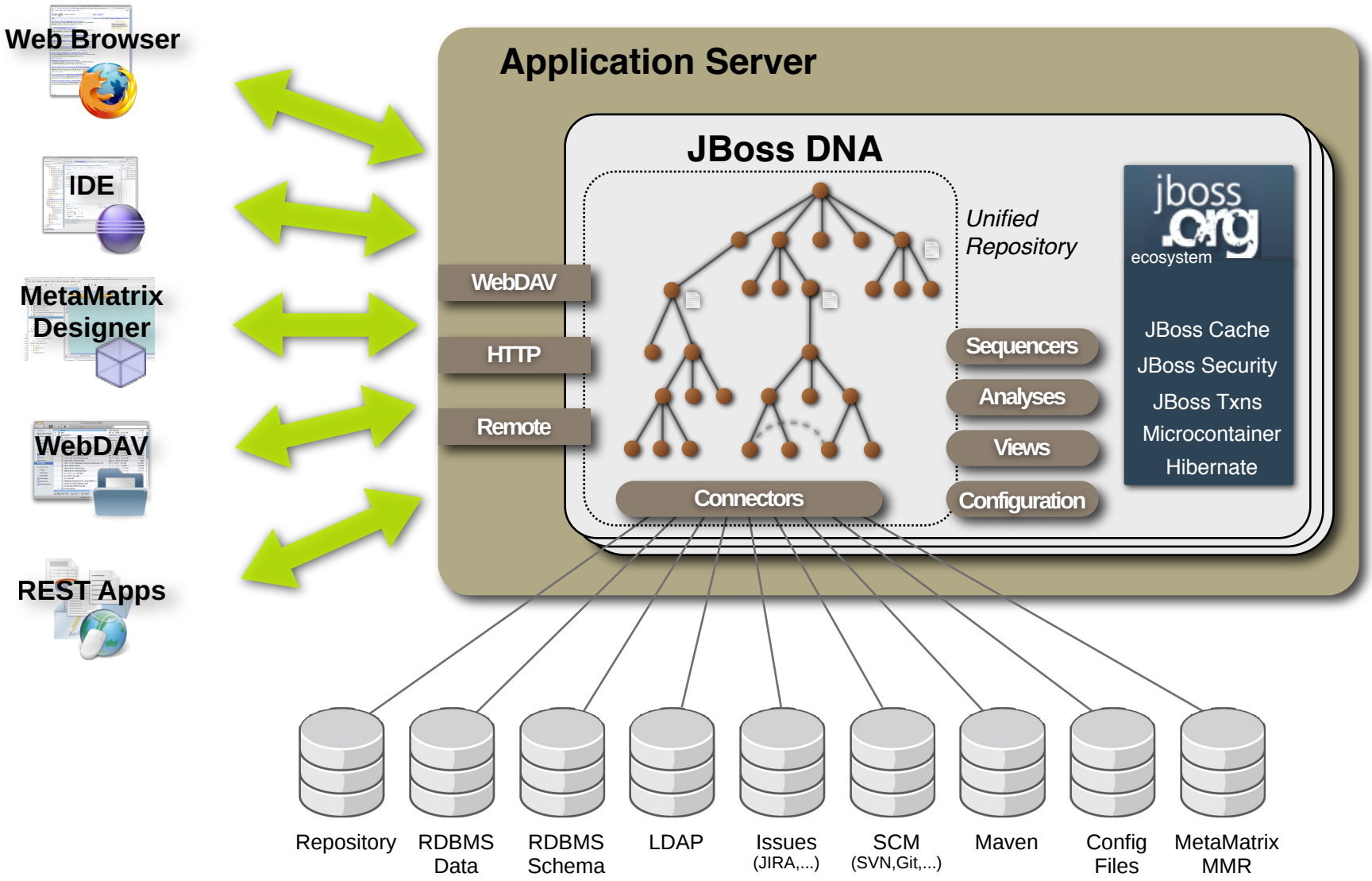
JBoss DNA

- **New JCR implementation that**
  - looks and behaves like a regular JCR repository
  - unifies content from a variety of systems
  - extracts the most benefit from the content
- **So what's different?**
  - where the content is stored (lots of places)
  - federation!
  - use of existing best-of-breed technology
    - cache, clustering, persistence, deployment
  - enterprise-class repositories
  - micro-repository for embedded use

JBoss DNA architecture



JBoss DNA architecture



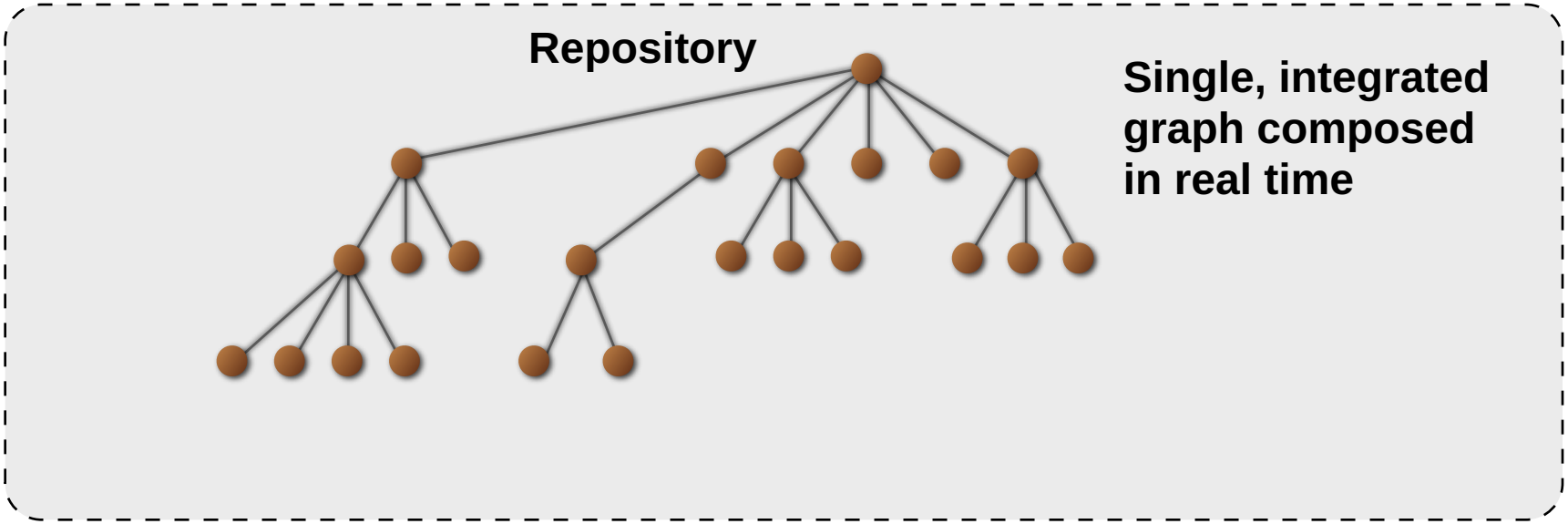
Configuring JBoss DNA

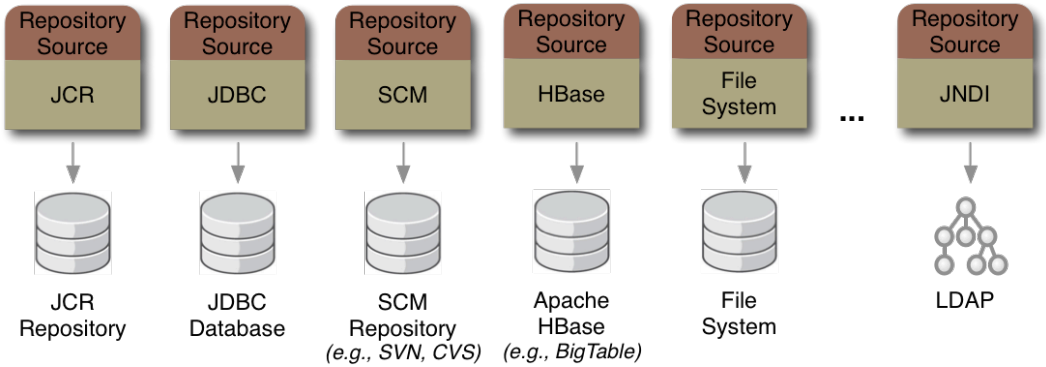
- **Configuration repository**
  - contains content describing the components
  - observed so updates are reflected in components
  - just a regular repository
- **Enables clustering**
  - Processes use the same configuration repository
  - Add and remove processes as required
- **Repository management**
  - One repository containing configurations for multiple repositories
  - Manage configuration simply as content (edit, copy, etc.)
  - Versioning supports rolling back to previous configuration

Federation Use Cases

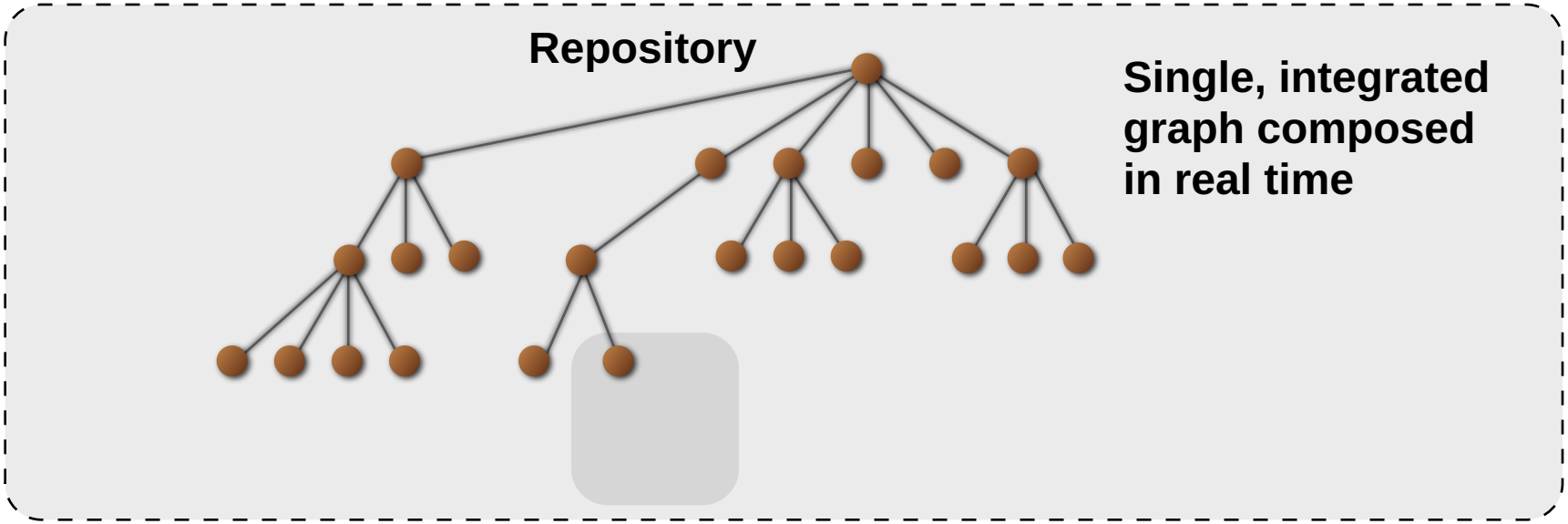
- **Unify content in multiple external systems**
  - Content still managed in current *system of record*
  - Benefits of a single repository
- **Local caching repository**
  - Remote repository is the master
  - Application wants a local copy/cache of data it uses
- **Images, large files for web content**
  - Store in JCR (versioning, events, auditing, access control)
  - Copy latest to file system (direct access by web server)
- **Segregating data by type**
  - Images in one repository, user info in another, etc.
  - Application still uses one repository
- **Segregating data by region/owner**
  - Multiple repositories structured similarly
  - Each region owns its data, but reads other regions' data

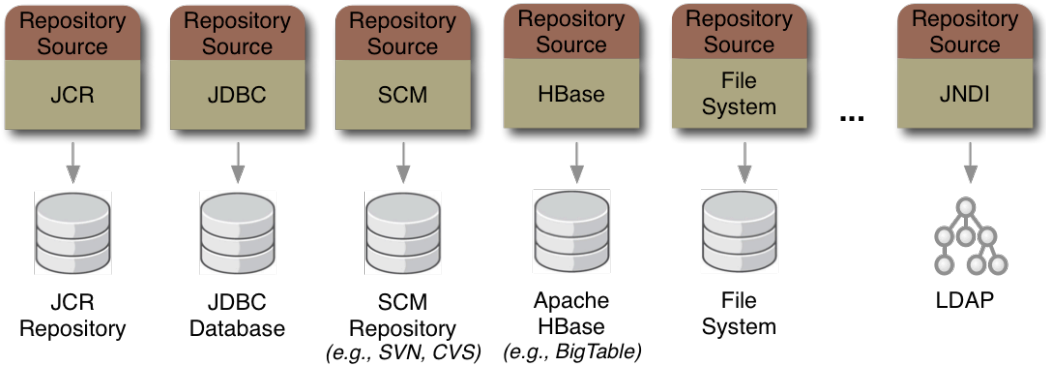
Federation and integration



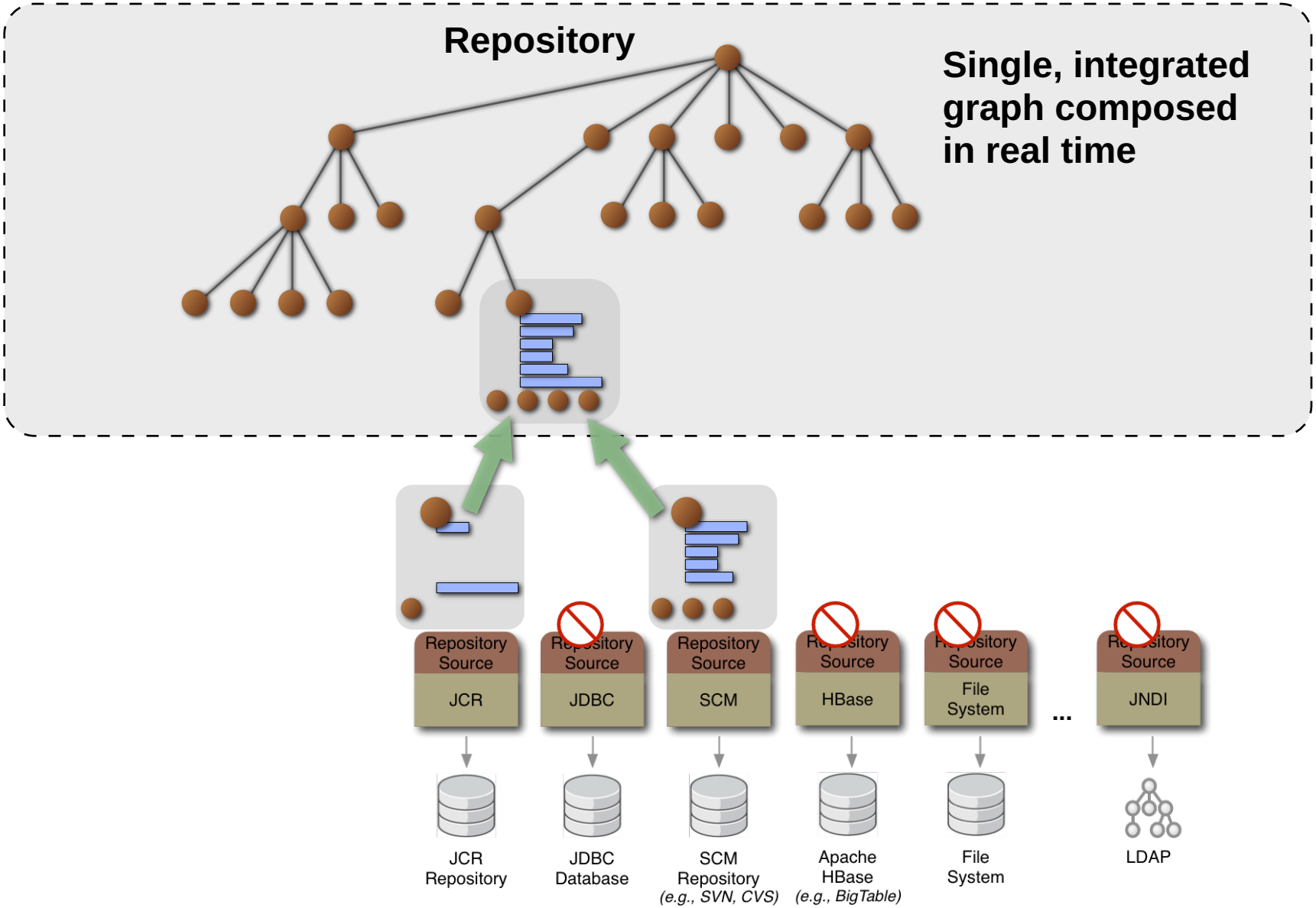


Federation and integration

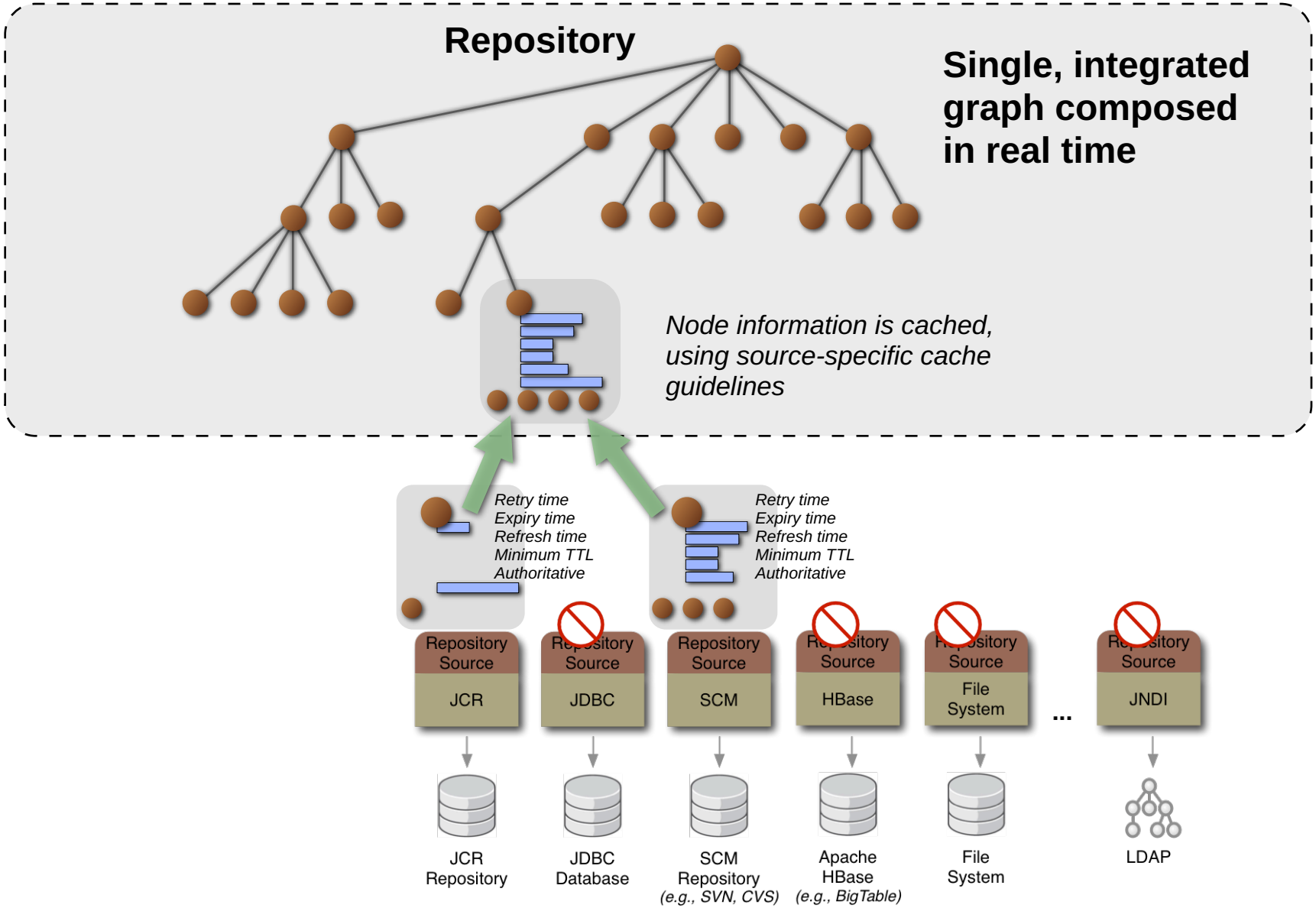




Federation and integration

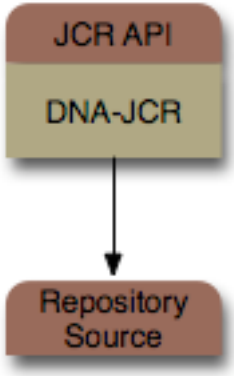


Federation and integration



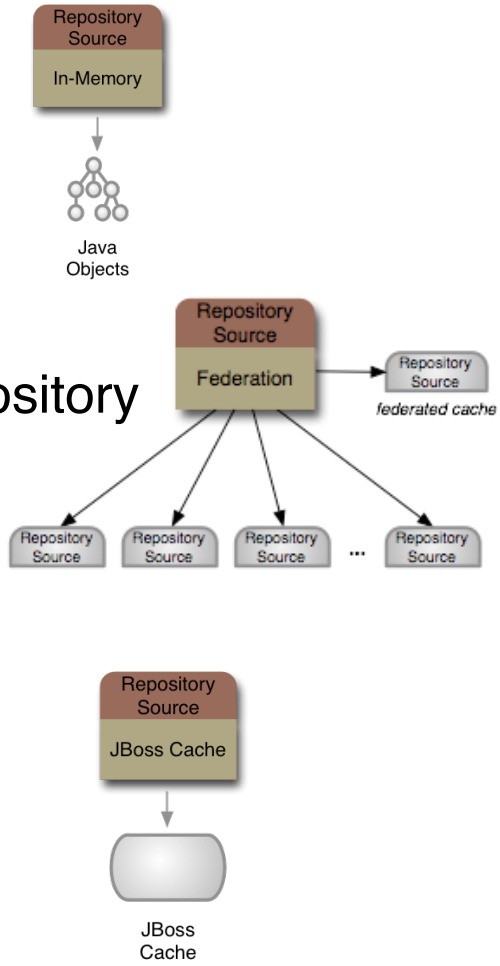
Connector framework

- **RepositorySource**
  - represents a connectable external system
  - creates connections
  - a JavaBean that's analogous to JDBC DataSource
- **RepositoryConnection**
  - represents a connection to a source
  - process requests by translating to source language
  - adapts content changes into events
- **RepositoryService**
  - manages RepositorySource instances
  - maintains pools of connections for each source
  - can reflect what's defined in a configuration repository



JBoss DNA connectors (as of 0.3)

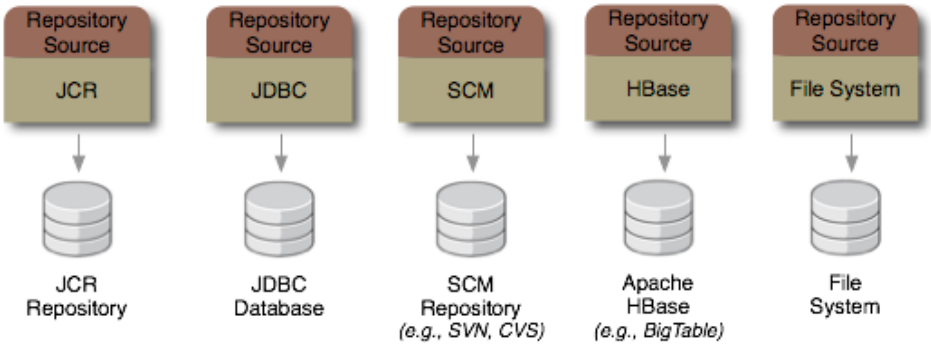
- **In-memory**
  - simple transient repository
- **Federated connector**
  - Merges content from multiple other sources
  - Each projects its content into “federated” repository
  - Strategies for merging nodes
  - Uses another source as the cache
- **JBoss Cache**
  - support for distribution, clustering, replication
  - ability to persistent information in databases



Future connectors

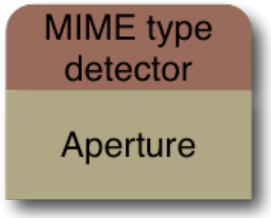
(not in a particular order)

- **Relational databases**
  - schema information (metadata)
  - data
- **File system connector**
  - expose files and directories
  - store content on file system
- **JCR repositories**
- **SCM systems**
  - SVN, CVS, Git
  - maps directory structure into `nt:folder` and `nt:file` nodes
  - includes version history
- **Maven repositories**
- **JNDI/LDAP**
- **Apache HBase**

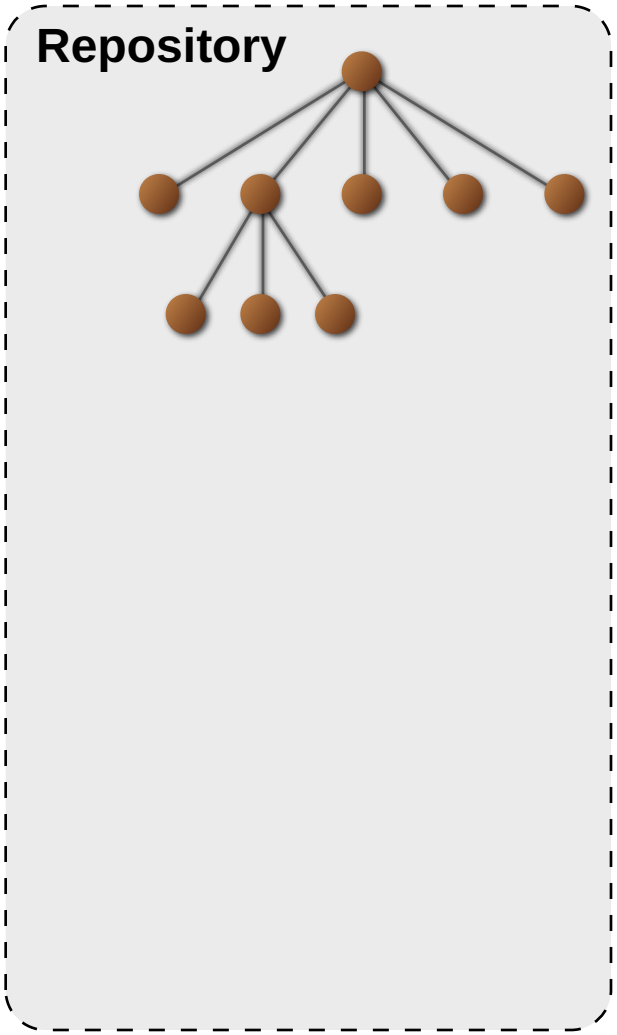


Detecting media types

- **Content often includes files**
- **Often want correct MIME type as metadata**
- **Typical approaches**
  - map extensions
  - interpret content
- **JBoss DNA uses MIME type detectors**
  - extensions that determine MIME type given filename and/or content
  - default implementation uses the Aperture open-source library



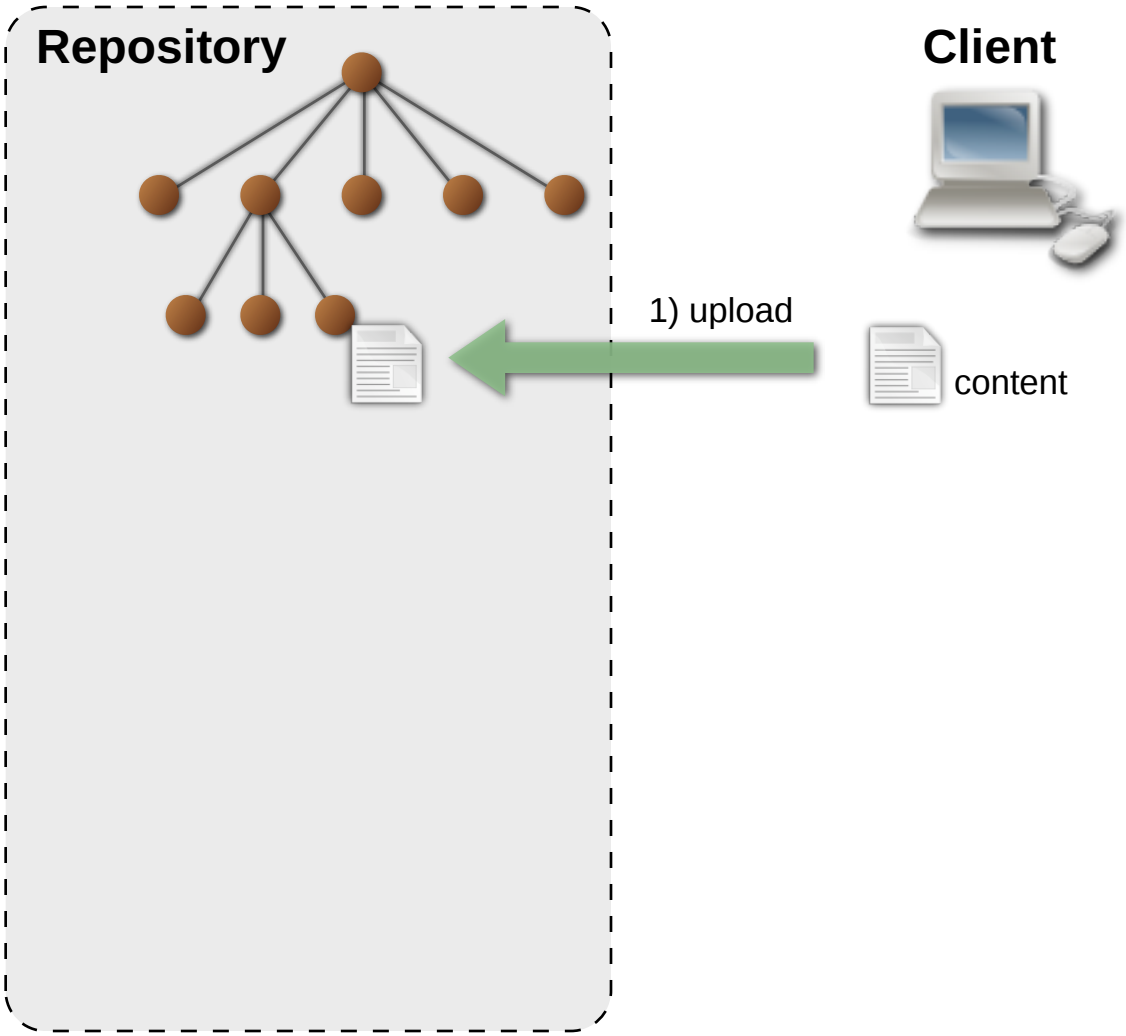
Sequencing content



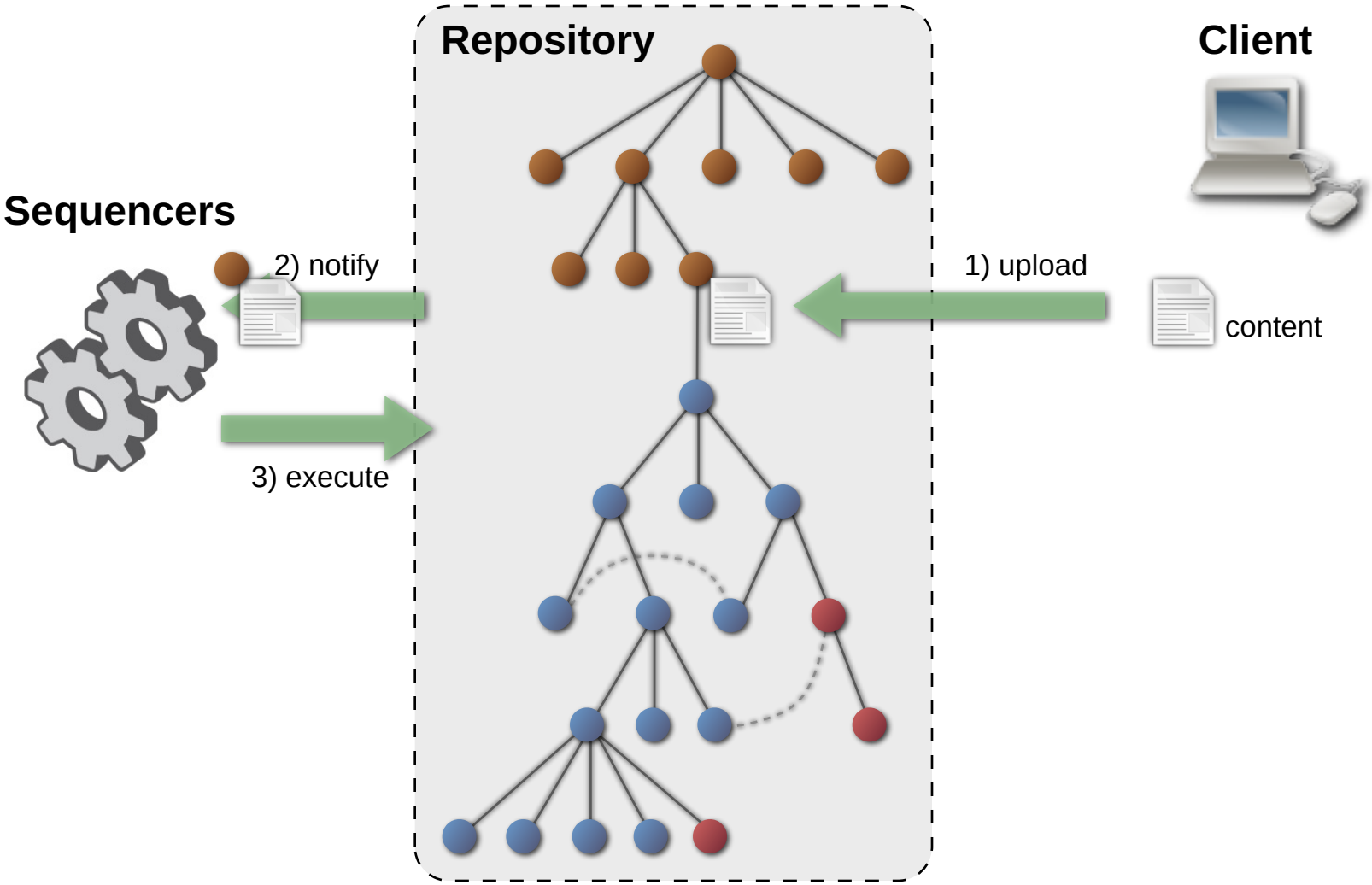
Client

content

Sequencing content



Sequencing content



Configure sequencers

- **Path expressions describe**
  - paths of content to be sequenced
  - path where to put generated output

```
inputRule => outputRule
```

  - examples:

```
//(*.(jpg|jpeg|gif|bmp|pcx|png))[*]/jcr:content[@jcr:data] => /images/$1
//(*.mp3)[*]/jcr:content[@jcr:data] => /mp3s/$1
//(*.(doc|ppt|xls))[*]/jcr:content[@jcr:data] => ./
```

- **Register a sequencer configuration**
  - Name, description, classname, and path expressions
- **Make available at runtime**
  - put sequencer implementation on the classpath
  - or use a `ClassLoaderFactory`

Writing a sequencer

Implement interface

```
public interface StreamSequencer {
    /**
     * Sequence the data found in the supplied stream,
     * placing the output information into the supplied output map.
     */
    void sequence( InputStream stream, SequencerOutput output,
                   ProgressMonitor progressMonitor );
}
```

- Read the stream
- Create output structure using SequencerOutput parameter:

```
output.setProperty(path, propertyName, propertyValue);
```

JBoss DNA sequencers (as of 0.3)

ZIP archives

Java source

Microsoft Office documents

MP3 audio files

JCR Compact Node Definition

jBPM Process Definition Language

Images (JPEG, GIF, BMP, PCX, PNG, IFF, RAS, PBM, PGM, PPM & PSD)

JBoss DNA timeline

Feb 2008 - Project Announced

May 2008 - Release 0.1

Sequencers

Sept 2008 - Release 0.2

Federation

TODAY

Nov 2008 - Release 0.3

Graph API

Dec 2008

Jan 2008

More frequent releases with smaller but incremental features, including: configuration, persistence, clustering, search, more sequencers, more connectors

For more information

- **Project: www.jboss.org/dna**
- **Downloads (0.2)**
  - Binary, source, documentation, examples
- **JBoss Maven2 Repository**
  - "org.jboss.dna" group ID (several artifacts)
- **Documentation**
  - Getting Started describes the design, the different components, and how to use them with a trivial example application
  - Reference Guide describes how JBoss DNA works internally from a developer perspective
- **Blogs:** jbossdna.blogspot.com
- **IRC:** irc.freenode.net#jbossdna

Questions?