An Overview of ModeShape

September 2010

Topics

- Why use JCR?
- Why use ModeShape?
- Productization

What is JCR?

JSR-170 and JSR-283

- Standard programmatic API: `javax.jcr`
- Abstracts where/how the content is stored
- Best features from databases and file systems

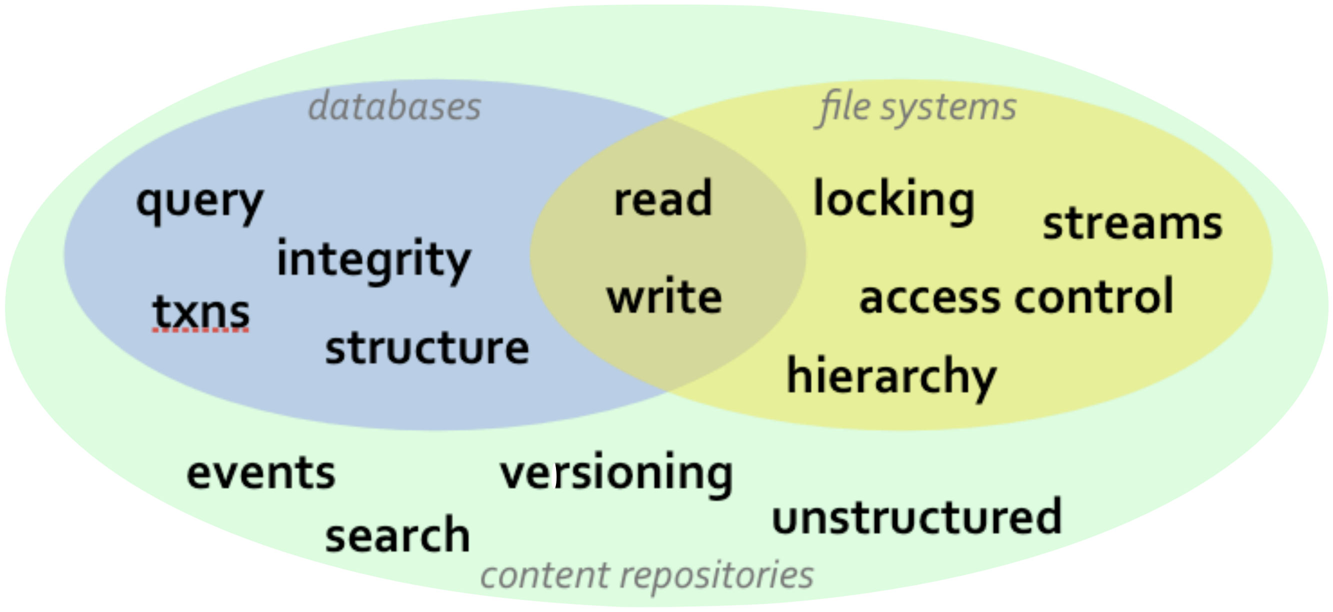


Image credit: David Nuescheler

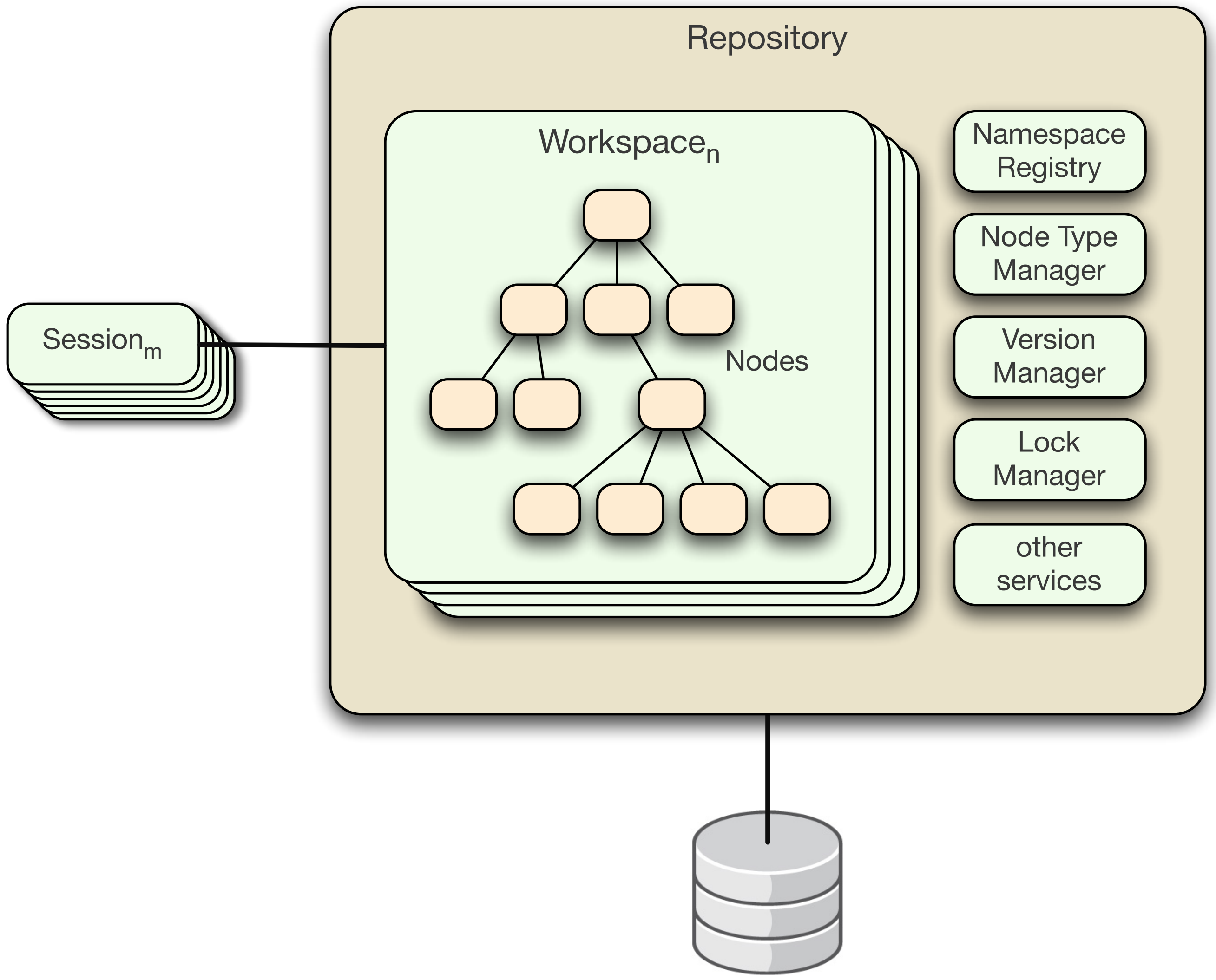
Typical uses of JCR

- Content management systems
- Portals *(e.g., GateIn)*
- Knowledge bases
- Test management systems
- Artifact repositories *(e.g., BRMS, EDS, ...)*

Why use a JCR repository?

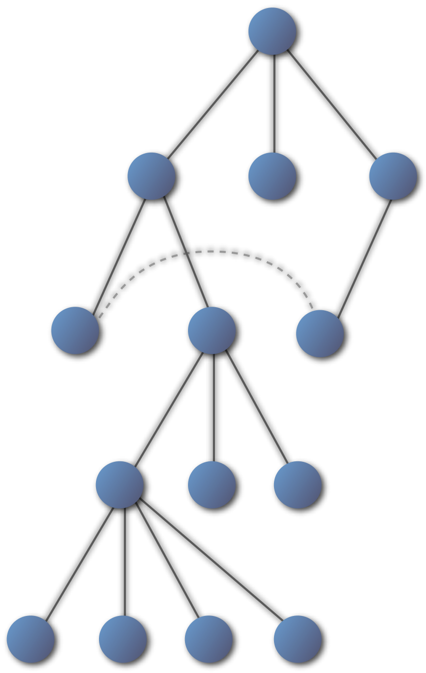
- Flexible data structure
  - because your data will change
- Hierarchical graph-based storage
  - natural organization
  - flexible schema
  - handles wide variety of content
- Search & query
- Versioning, events, access control, referential integrity, locking
- Standard API

Fundamental elements of JCR



Why use JCR?

Hierarchical organization



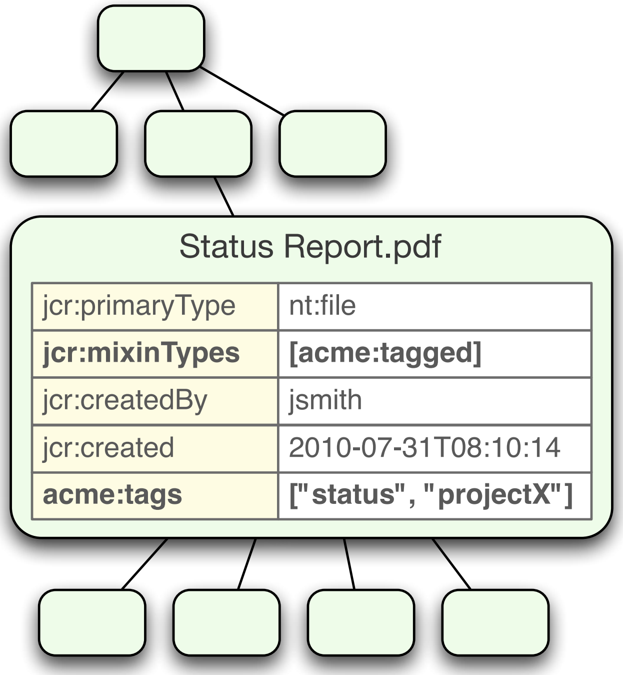
- Great for natural hierarchies
  - URLs and file systems
  - Dates
  - Composition
  - Catalogs
  - ... many others
- Examples
  - Files and folders (artifacts)
  - Database schemas
  - Web services
  - XML, YAML, JSON, etc.

Why use JCR?

Flexible schema (part 1)

Each JCR node has:

- Name, path, & identifier
- Properties
  - name & value(s)
  - property types
- Child nodes
  - same-name-siblings
  - may be ordered
- One or more node types
  - primary type and mixins
  - dictate whether properties and children can be added/removed
  - can be changed at any time on each node



jcr:primaryType	nt:file
jcr:mixinTypes	**[acme:tagged]**
jcr:createdBy	jsmith
jcr:created	2010-07-31T08:10:14
acme:tags	**["status", "projectX"]**

Flexible schema (part 2)

JCR node types:

- define the allowed properties
  - name (or unrestricted with "*")
  - type
  - number of values
  - searchability
  - mandatory
- define the allowed child nodes
  - name (or unrestricted with "*")
  - required node types
  - same-name-siblings allowed
- inheritance and mixed together

LONG, DOUBLE, DECIMAL, DATE, BOOLEAN, NAME, PATH, URI, REFERENCE, WEAK_REFERENCE, UNDEFINED

Why use JCR?

Query languages

- XPath
- JCR-SQL

From JCR 1.0,
deprecated in JCR 2.0

- JCR-SQL2
- JCR-JQOM

New in JCR 2.0

JCR-SQL2 is based upon SQL

JCR-SQL2	SQL
node type	table
property	column
node	row

Nodes appear as rows in those tables that correspond to the node's types

- a node can appear as a row in multiple tables
- often a complete picture of a node requires joining multiple tables
- other joins are possible, such as ancestors, descendants, equivalent properties

JCR-SQL2 features

- Select
- Joins (INNER, LEFT/RIGHT OUTER)
- Property & name criteria
- Child & descendant node criteria
- Comparison operators (`=,<,<=,>,>=,<>, LIKE`)
- Logical operators (`AND, OR, NOT`)
- Full-text search criteria
- Variables

ModeShape accepts all valid JCR-SQL2 queries

Sample JCR-SQL2 queries

```
SELECT * FROM [car:Car] WHERE [car:model] LIKE ‘%Toyota%’ AND [car:year] >= 2006
```

```
SELECT [jcr:primaryType],[jcr:created],[jcr:createdBy] FROM [nt:file]
WHERE NAME() LIKE $name
```

```
SELECT file.*,content.* FROM [nt:file] AS file
JOIN [nt:resource] AS content ON ISCHILDNODE(content,file)
WHERE NAME(file) LIKE 'q*.2.vdb'
```

Why use JCR?

Other benefits

- Versioning
- Notification (events)
- Access control
- Referential integrity
- Locking
- Standard API

Using JCR 2.0 repositories

- Uses JCR 2.0 API & Java SE ServiceLoader
- No implementation dependencies

Java code:

```
Properties parameters = new Properties();
parameters.load(...);          // load from a file
Repository repository = null;
for (RepositoryFactory factory : ServiceLoader.load(RepositoryFactory.class)) {
    repository = factory.getRepository(parameters);
    if (repository != null) break;
}
```

Properties for ModeShape:

```
org.modeshape.jcr.URL = file:path/to/configRepository.xml?repositoryName=MyRepository
```

or

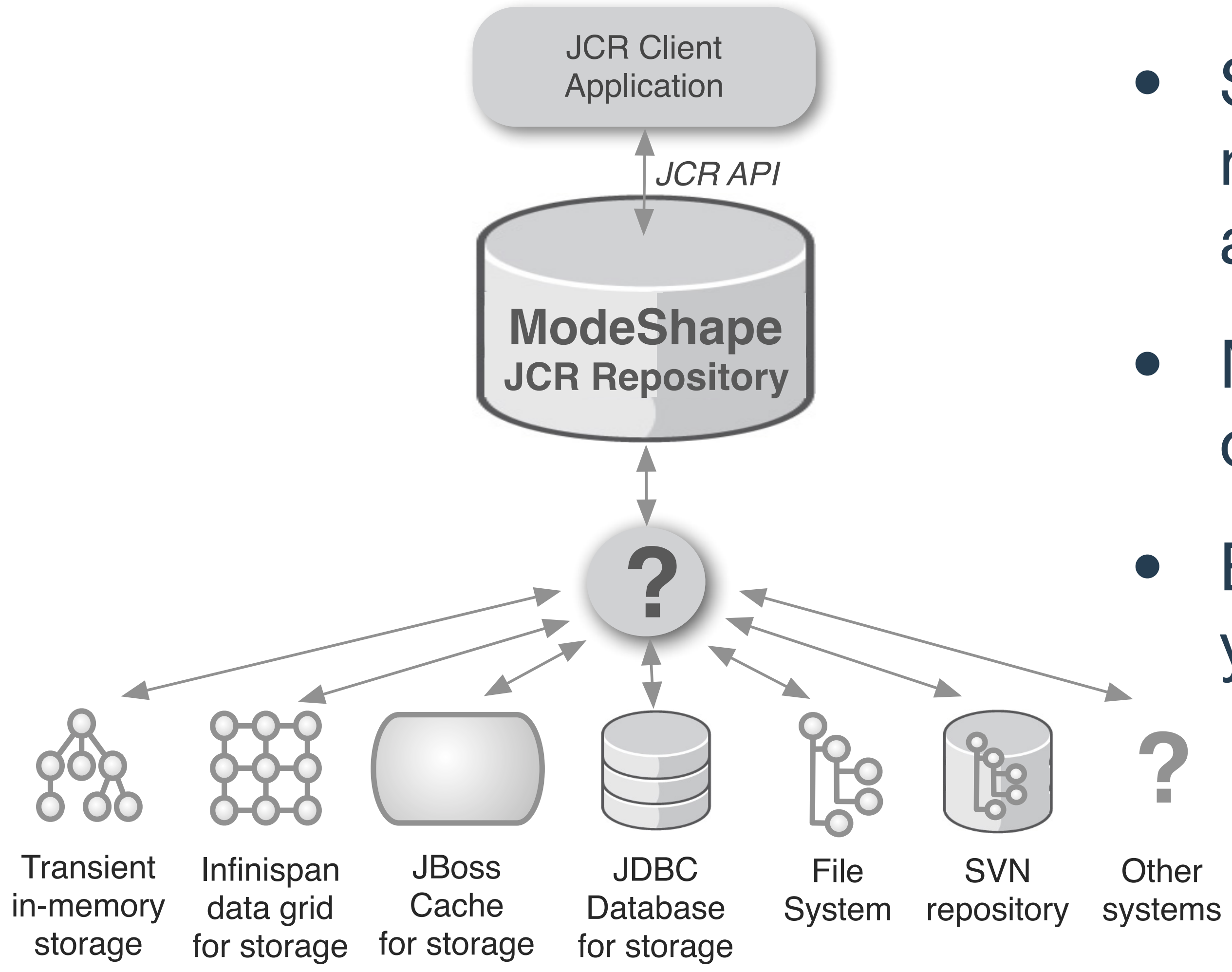
```
org.modeshape.jcr.URL = jndi:/jcr/local?repositoryName=MyRepository
```

Why use ModeShape?

- Supports JCR 2.0
- Storage options
- Access existing content thru JCR
- Unified repository via federation
- Automatically derives content
- Lightweight, embeddable & clusterable
- Plays well with JBoss AS

Why use ModeShape?

Storage options



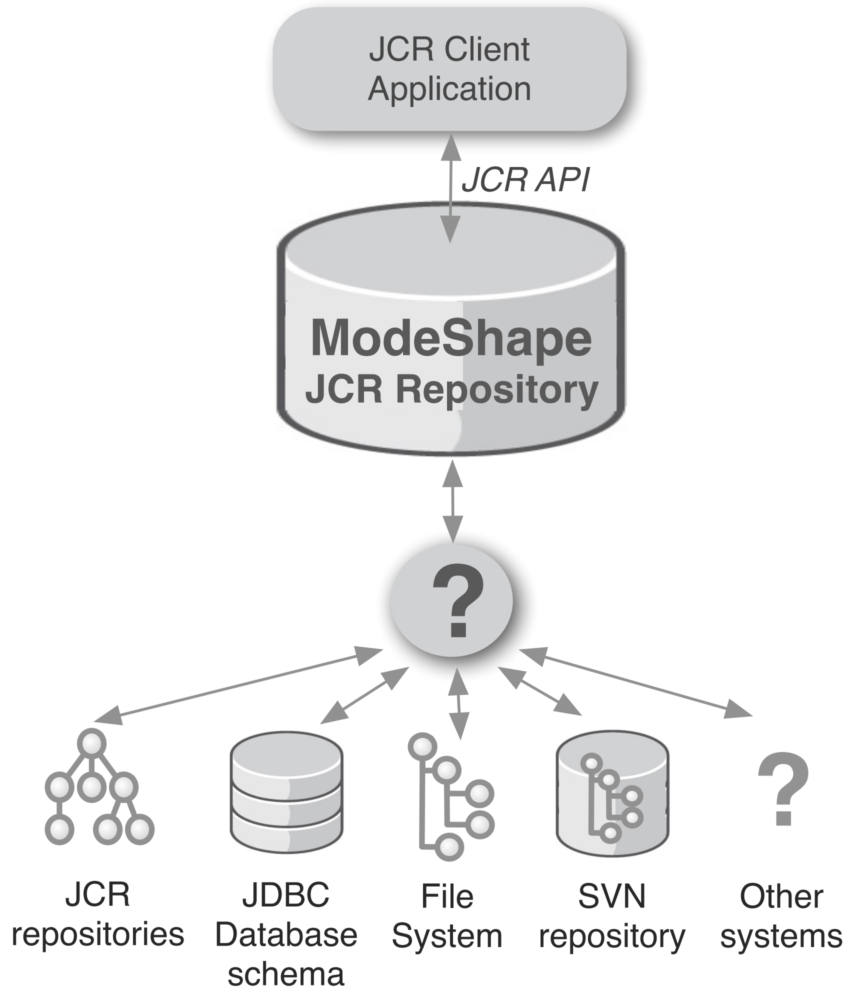
- Store content where it makes sense for your application
- Multiple connectors out-of-the-box
- Easy enough to write your own

Storage examples

- Store in RDBMS using Hibernate
  - Conventional and well-understood
- Store in Infinispan
  - Scalability, fault tolerance and speed of a data grid
- Store artifacts on file system
  - Images, documents, downloads, static pages, etc.
  - Serve files via file system, bypassing JCR altogether
  - Manipulate content through JCR (e.g., CMS)
- Transient in-memory
  - Lightweight and fast for embedded uses
  - Often used with imported data

Why use ModeShape?

Accessing other data



- Use JCR API to access data from other systems
- Leave the data where it is
- Just another connector

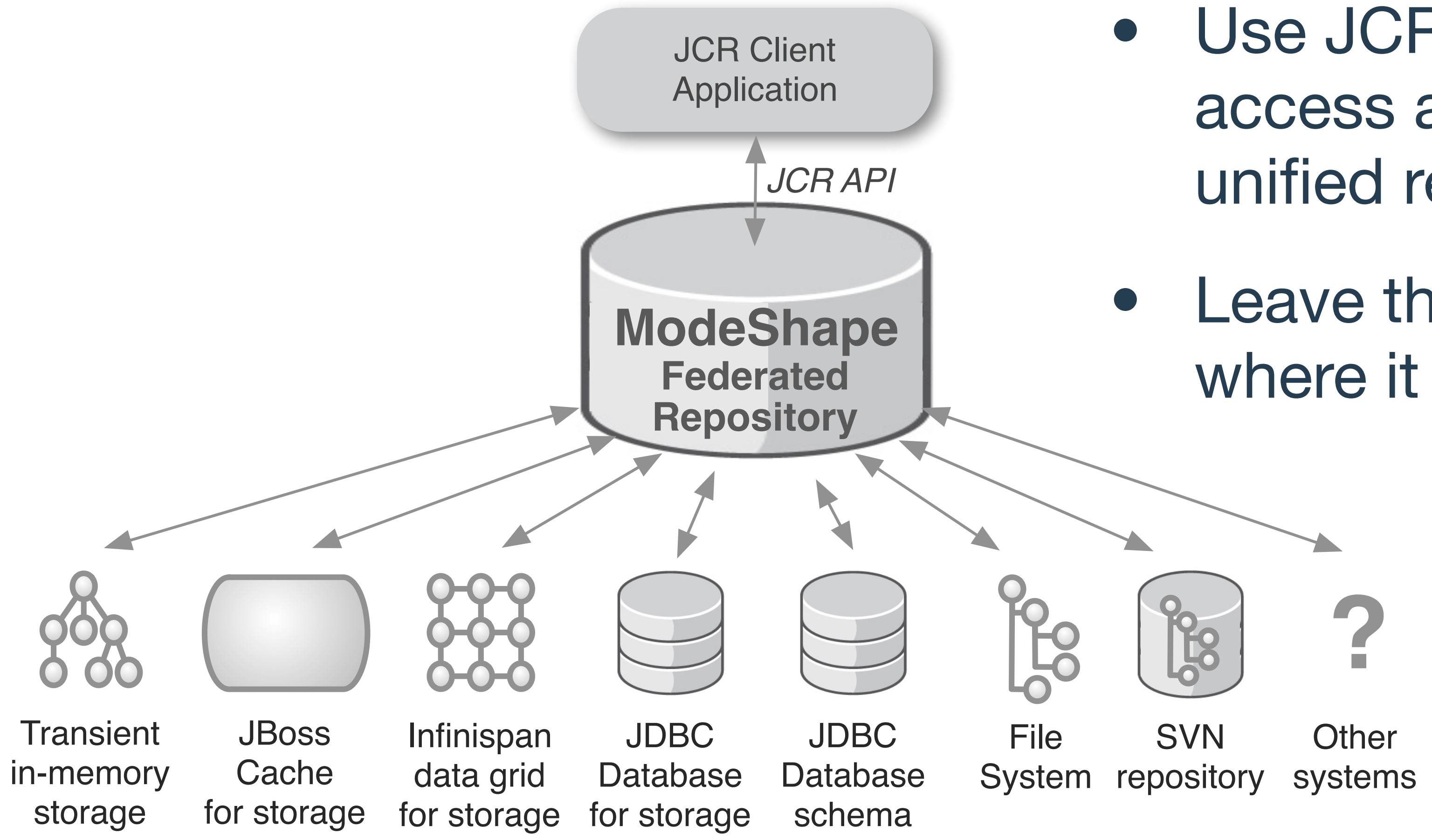
Why use ModeShape?

Access examples

- Access files already stored on file system
  - Use the JCR API within your application
  - Use search and query
- Access database metadata (live)
  - Leverage session-based caching
  - Consistent model
- Access (& manipulate) content in other JCRs
  - Got data already in another JCR system?
  - Makes most sense with federation ...

Why use ModeShape?

Federation


JCR Client Application
JCR API
ModeShape Federated Repository
Transient in-memory storage
JBoss Cache for storage
Infinispan data grid for storage
JDBC Database for storage
JDBC Database schema
File System
SVN repository
Other systems

- Use JCR API to access a virtual, unified repository
- Leave the data where it is

Why use ModeShape?

Federation examples

- Own some content, access other data
  - Use the JCR API consistently throughout your application
  - Single virtual and unified repository
  - Leverage JCR's flexible schema
  - Use search and query across all data
  - Leave existing data where it lives
- Use the best data store for each kind of data
  - Use the JCR API consistently throughout your application
  - Single virtual and unified repository
  - Match access and capabilities with requirements

Query languages

- XPath
- JCR-SQL

From JCR 1.0, deprecated in JCR 2.0

- JCR-SQL2
- JCR-JQOM

New in JCR 2.0

- Full-text search

Like web search engines, based on JCR-SQL2's full-text search expression grammar

Separate parsers, but all use the same canonical AST, planner, optimizer, and processor

Why use ModeShape?

Enhanced JCR-SQL2

ModeShape accepts all valid JCR-SQL2 queries, plus:

- Additional joins (`FULL OUTER, CROSS`)
- Subqueries in criteria
- `SELECT [DISTINCT]`
- `UNION/INTERSECT/EXCEPT [ALL]`
- `LIMIT` and `OFFSET`
- `DEPTH` and `PATH` criteria
- `REFERENCE` criteria
- `IN` and `NOT IN`
- `BETWEEN val1 [EXCLUSIVE] AND val2 [EXCLUSIVE]`
- Arithmetic expressions in criteria

Why use ModeShape?

Sample enhanced JCR-SQL2 queries

```
SELECT * FROM [car:Car] WHERE [car:model] LIKE ‘%Toyota%’ AND [car:year] >= 2006
```

```
SELECT [jcr:primaryType],[jcr:created],[jcr:createdBy] FROM [nt:file]
WHERE PATH() LIKE $path
```

```
SELECT file.*,content.* FROM [nt:file] AS file
JOIN [nt:resource] AS content ON ISCHILDNODE(content,file)
WHERE PATH(file) LIKE '/files/q*.2.vdb'
```

```
SELECT [jcr:primaryType],[jcr:created],[jcr:createdBy] FROM [nt:file]
WHERE PATH() IN (
    SELECT [vdb:originalFile] FROM [vdb:virtualDatabase]
    WHERE [vdb:version] <= $maxVersion
    AND CONTAINS([vdb:description],'xml OR xml maybe')
)
```

Why use ModeShape?

Use the data you already have

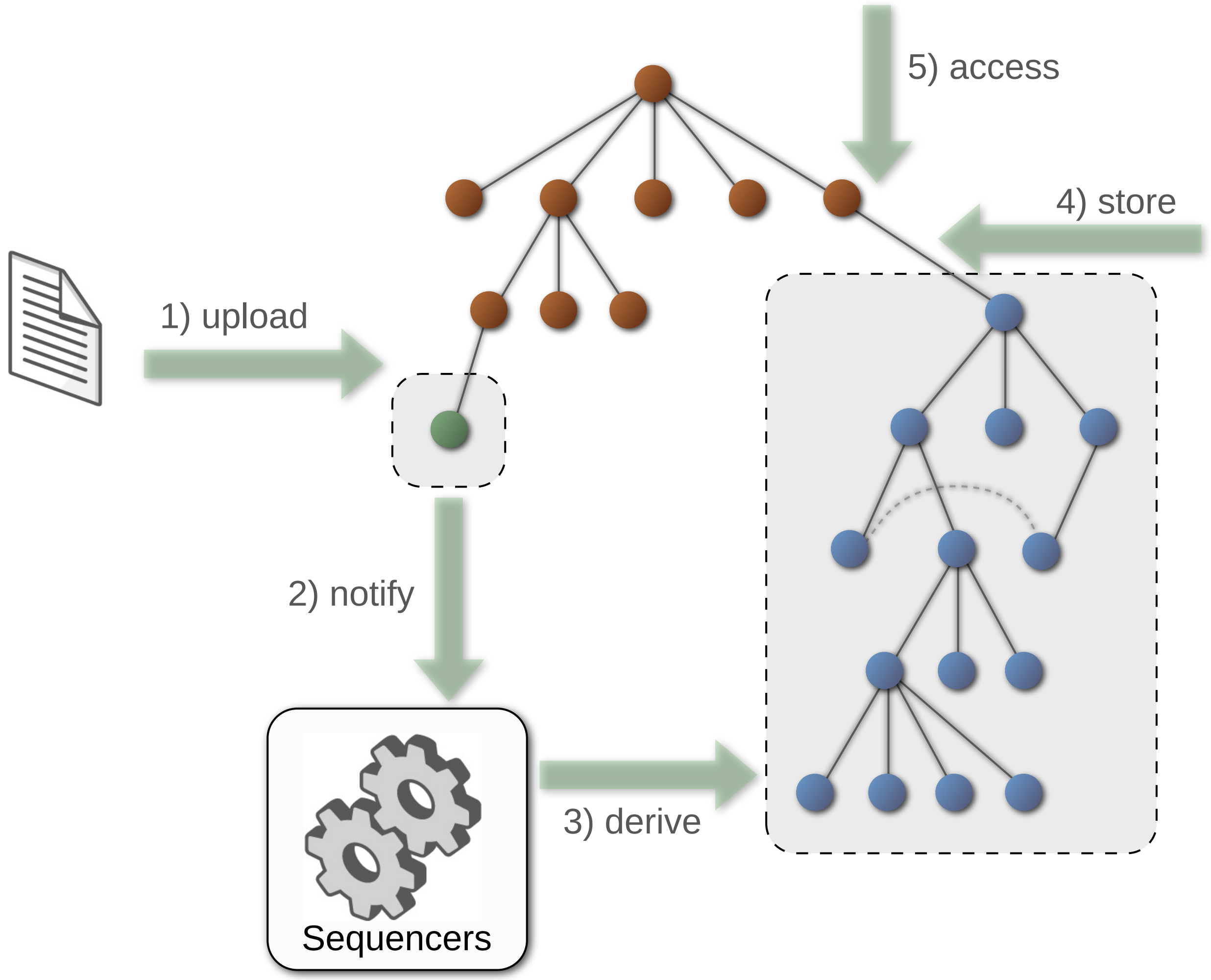
- Repositories frequently store files
  - Many of those files contain structured information
- How do you use what’s inside those files?
  - Download and parse in your application?
  - Rely upon full-text search?
  - Do it whenever the files change?
- ModeShape sequencers get at that info
  - Parsers for various file formats
  - Extract the information and place into graph form using node types
  - Store derived information inside the repository
  - Access, navigate,and query

Sequencer examples

- Web service artifacts
  - WSDL, XSD, WS Policy, ...
- Data management artifacts
  - DDL, Teiid models & VDBs, services, data sources, ...
- Software artifacts
  - JAR, WAR, EAR, POM, manifests, Java source/classfiles, ...
- Rule and process artifacts
  - Business/technical rules, functions, flows, DSLs, ...

Why use ModeShape?

Automatically extract content

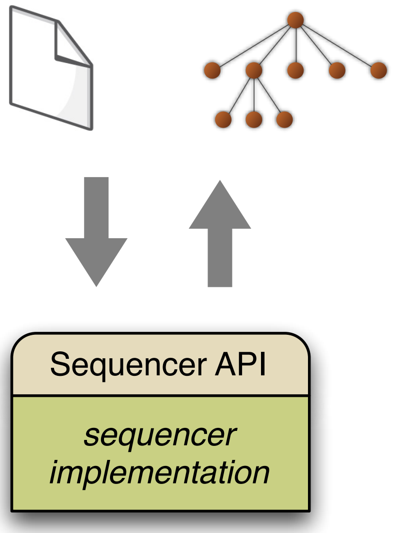


Why use ModeShape?

ModeShape sequencers

(out-of-the-box)

- ZIP, JAR, EAR, and WAR files
- Java class and source files
- DDL files
- Teiid Relational models and VDBs
- Text files (CSV and delimited)
- JCR Compact Node Definition files
- XML documents
- Microsoft Office® documents
- Image metadata



ModeShape components

- JCR engine
- RESTful service
- WebDAV service
- JDBC driver
- JBoss AS 5.1 kit
- Eclipse plugin

Why use ModeShape?

RESTful service

- Remote access to repository
- Standard HTTP methods
  - GET repositories
  - GET workspaces
  - GET, PUT, POST, DELETE nodes
  - POST queries
- JSON responses (HTML, XML planned)
- Uses RESTEasy
- Deployed as a WAR

WebDAV service

- Remote access to repository
- Can mount as network share
- Download, upload, create or delete files
- Navigate, create and delete folders
- Deployed as a WAR

Why use ModeShape?

JDBC driver

- Query repository via JDBC using JCR-SQL2
- Connects to one workspace
- Can be used in same VM as ModeShape

```
jdbc:jcr:jndi:{path/in/jndi}?repositoryName={name}&user=...
```

- Or in remote VM

```
jdbc:jcr:http://{hostname}:{port}/modeshape-rest/?repositoryName={name}&user=...
```

- Supports result set metadata
- Supports database metadata

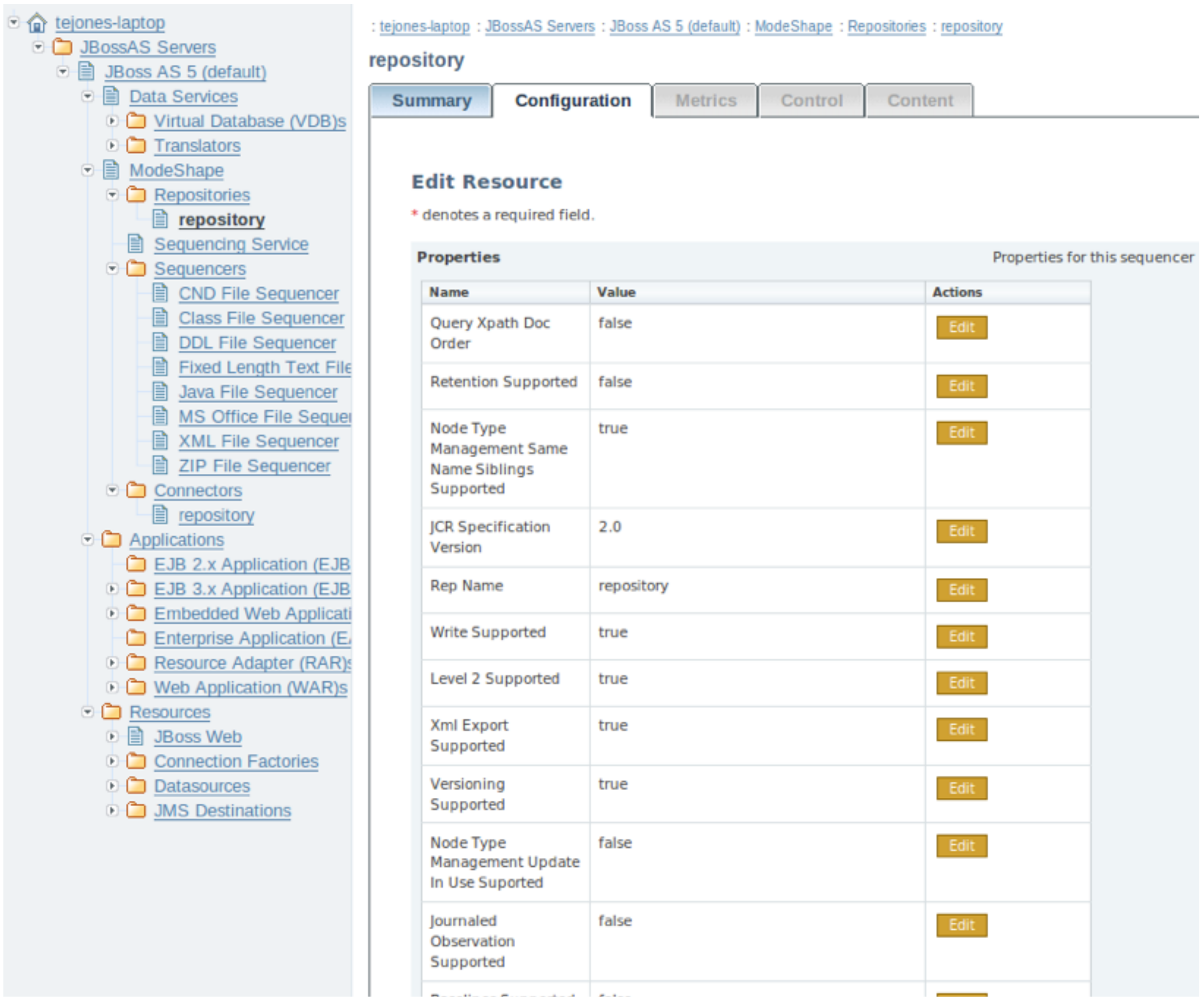
Why use ModeShape?

JBoss AS kit

- JCR service (the engine)
- JOPR plugin for management & monitoring
- REST service (WAR)
- WebDAV service (WAR)
- JAAS authentication & authorization
- JDBC data source (1 per workspace)
- JCR API on classpath
- Sequencers
- Disk-based storage by default (HSQLDB & Lucene)
- Packaged as ZIP
- Script to install into a profile

Why use ModeShape?

JOPR plugin



- tejones-laptop
  - JBossAS Servers
    - JBoss AS 5 (default)
      - Data Services
        - Virtual Database (VDB)s
        - Translators
      - ModeShape
        - Repositories
          - **repository**
        - Sequencing Service
        - Sequencers
          - CND File Sequencer
          - Class File Sequencer
          - DDL File Sequencer
          - Fixed Length Text File
          - Java File Sequencer
          - MS Office File Seque
          - XML File Sequencer
          - ZIP File Sequencer
        - Connectors
          - repository
      - Applications
        - EJB 2.x Application (EJB
        - EJB 3.x Application (EJB
        - Embedded Web Applicati
        - Enterprise Application (E
        - Resource Adapter (RAR)s
        - Web Application (WAR)s
      - Resources
        - JBoss Web
        - Connection Factories
        - Datasources
        - JMS Destinations

: tejones-laptop : JBossAS Servers : JBoss AS 5 (default) : ModeShape : Repositories : repository

repository

Summary | **Configuration** | Metrics | Control | Content

Edit Resource

* denotes a required field.

Properties — Properties for this sequencer

Name	Value	Actions
Query Xpath Doc Order	false	Edit
Retention Supported	false	Edit
Node Type Management Same Name Siblings Supported	true	Edit
JCR Specification Version	2.0	Edit
Rep Name	repository	Edit
Write Supported	true	Edit
Level 2 Supported	true	Edit
Xml Export Supported	true	Edit
Versioning Supported	true	Edit
Node Type Management Update In Use Suported	false	Edit
Journaled Observation Supported	false	Edit

Productization

SOA-P 5.1 EDS

- Includes Teiid & ModeShape
- Slightly customized ModeShape JBoss AS kit
  - modified configuration and select connectors, sequencers
- Repository for data service artifacts
  - relational model files
  - virtual database (VDB) files
- Other files can be managed (but not sequenced)
  - XSD, WSDL, CSV, docs, etc.
- Publish and unpublish from JBoss Dev Studio
- Can access repository content from within VDBs

BRMS 5.1

- Offer ModeShape as an option (in place of Jackrabbit)
- Same binaries as in SOA-P 5.1
- Enable future enhancements (if desired):
    - sequencers for rules, flows, DSLs, etc.
    - query via JDBC
    - connectors & federation
    - RESTful access

Questions?