NoSQL at work with JCR and Apache Jackrabbit

Carsten Ziegeler, Adobe
cziegeler@apache.org, Nov 2011 Vancouver

Presented by

Produced by

About

- Member of the ASF
  - Sling, Felix, Portals, Incubator
  - PMC: Felix, Portals, Incubator, Sling (Chair)
- RnD Team at Adobe Research Switzerland
- Article/Book Author
- Technical Reviewer
- JSR 286 Spec Group (Portlet API 2.0)

Motivation

- Tried and trusted NoSQL solution
- Standard Java API
  - First spec released in May 2005
  - Various implementations, products, and solutions
  - Open Source implementation since 2006
- Think about your data use cases / problems
  - JCR might help!

Consider JCR

- Data structure
- Supporting the web
- ACID
- Security
- Additional features

The Structure of Data

- A data storage should be flexible and
- Allow to model app data in the “right” way
  - Optimal way of dealing with the data in the app

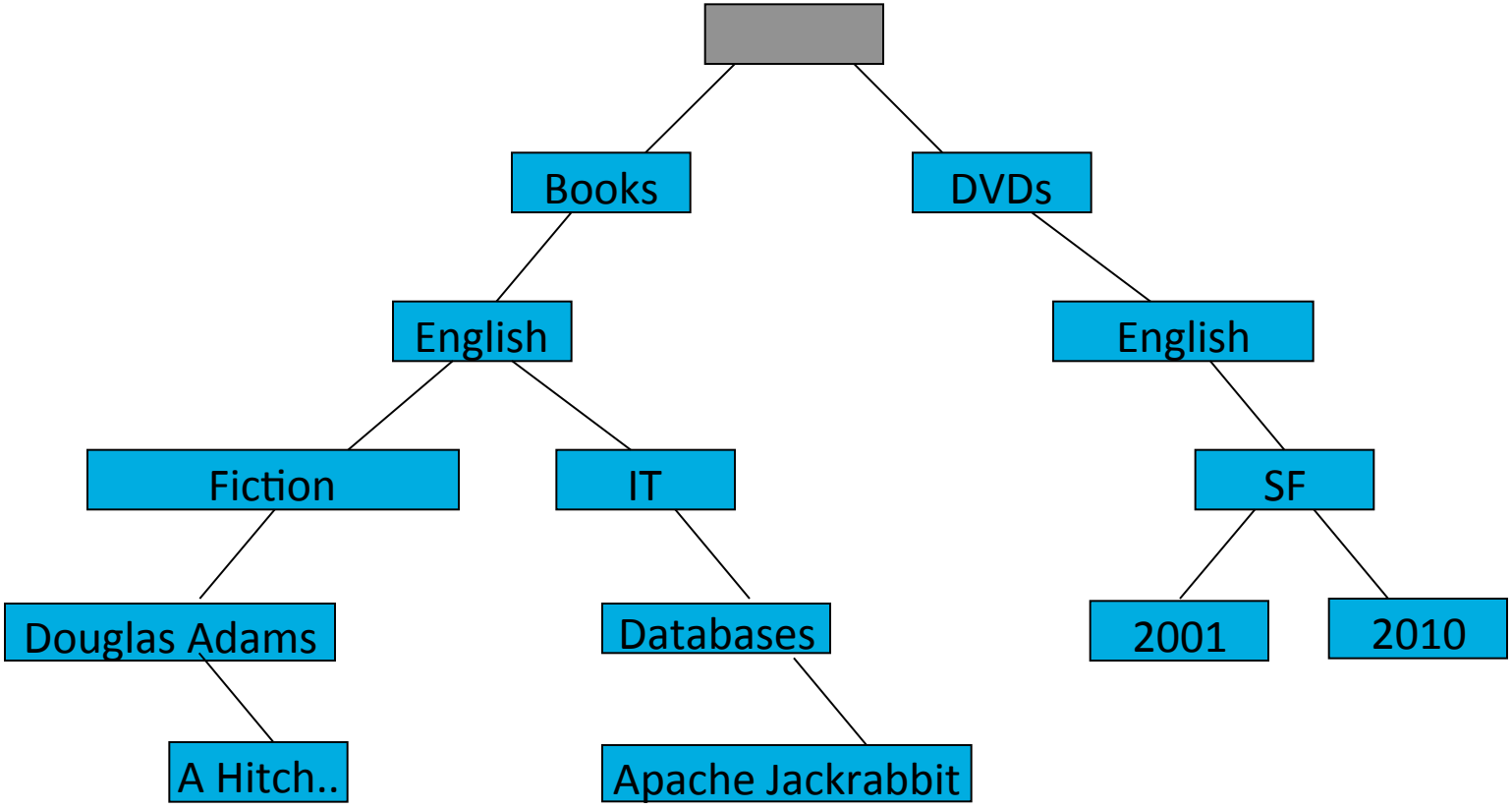
The Structure of Data

- A data storage should be flexible and
- Allow to model data in the “right” way
- What **is** the “right” way?
  - Tables?
  - Key-Value-Pairs?
  - Schema based?
  - Semi structured or even unstructured?
  - Flat, hierarchical or graph?

The Structure of Data

- The right way depends on the application:
  - Tables
  - Key-Value-Pairs
  - Schema based
  - Semi structured and unstructured
  - Flat, hierarchical, and graph
  - …
- An app might have more than one “right” way
- But: A lot of data can be modeled in a hierarchy

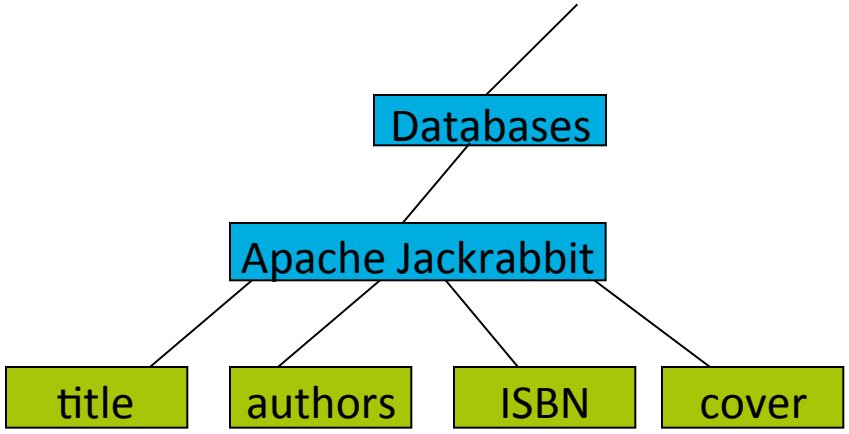
Sample: Product Catalog



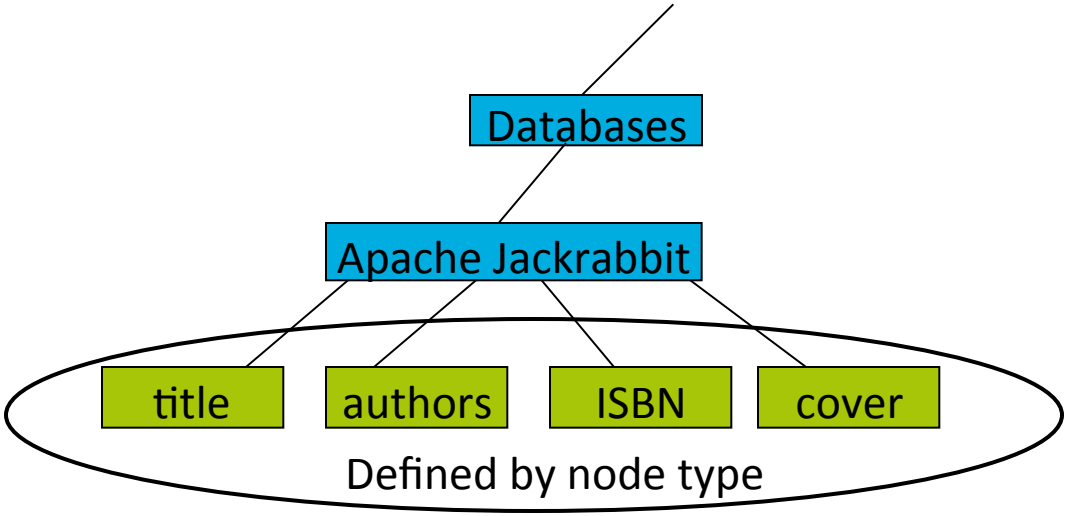
Java Content Repository

- Hierarchical content
  - Nodes with properties
  - (Table is a special tree)
- Structured
  - Nodetypes with typed properties
- And/or semi structured and unstructured
- Fine and coarse-grained
- Single repository for **all** content!

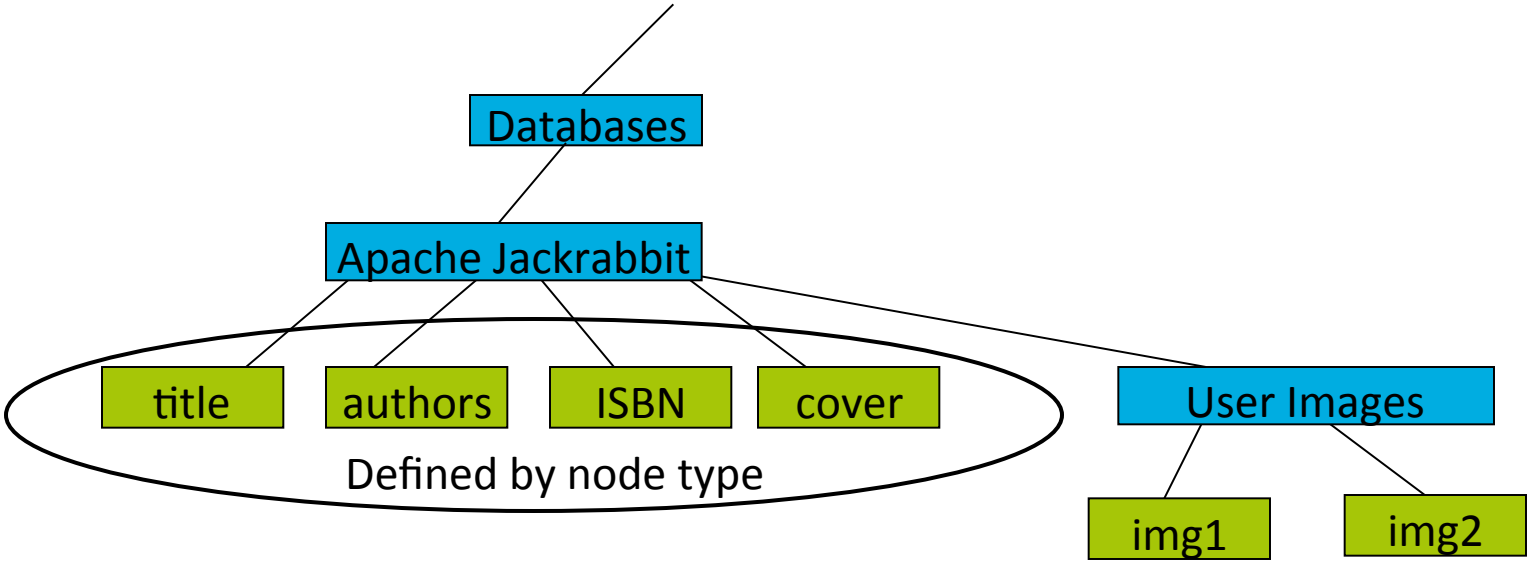
Sample: Product Catalog



Sample: Product Catalog



Sample: Product Catalog



JSR 170 / JSR 283: Content Repository for JavaTM technology API

- (Java) **Standard** – Version 1.0 and 2.0
  - Supported by many vendors
  - Used by many products and projects
  - Several open source solutions
- Data model and features
- Connecting to a CR
- Working / Interaction with a CR

Apache Jackrabbit

- JSR 170 and 283 reference implementation
- Apache TLP since 2006
- Vital community
- Frequent Releases
  - 1.6.5 (JSR 170 based, EOL)
  - **2.2.9 (JSR 283)**
  - (dev branch 2.3.2)
- Components
  - Commons, API
  - RMI, WebDAV, WebApp, JCA
  - OCM
  - And more...

http://jackrabbit.apache.org/

Data and the Web?

- The web is hierarchical by nature
- Web applications provide data in different ways
  - HTML
  - JSON
- Provide your data in a RESTful way
  - http://…/products/books/english/it/databases/apachejackrabbit.(html|json)
- Avoid mapping/conversion
  - http://…/products.jsp?id=5643564

Resource Oriented Architecture I

- Every piece of information is a resource
  - News entry, book, book title, book cover image
  - Descriptive URI
- Stateless web architecture (REST)
  - Request contains all relevant information
  - Targets the resource
- Leverage HTTP
  - GET for rendering, POST for operations

Resource Oriented Architecture II

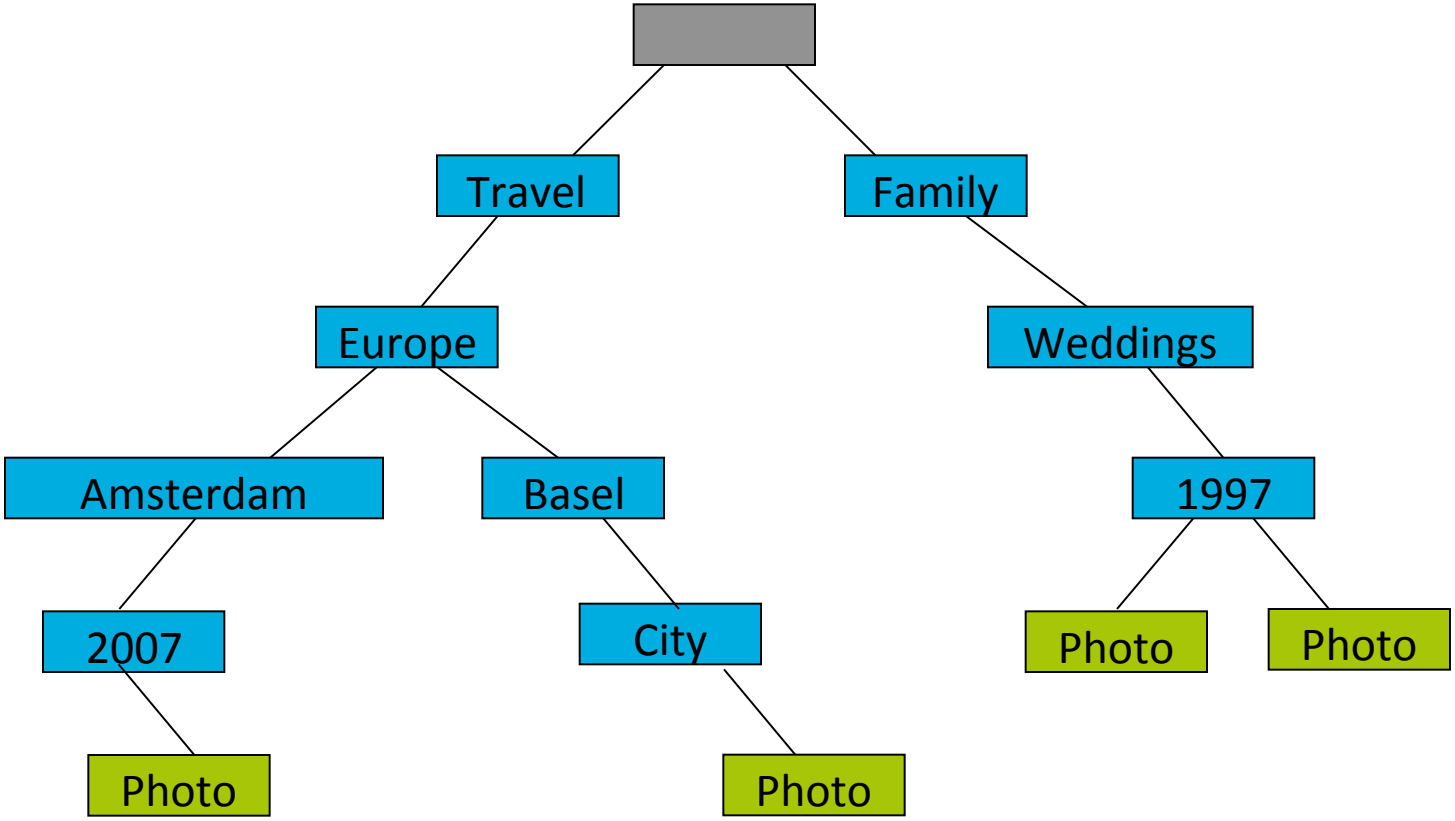
- JCR and Apache Jackrabbit are a perfect match
  - Hierarchical
  - From a single piece of information to binaries
- Elegant way to bring data to the web
- Apache Sling is (the|one) web framework

Sample Application: Slingshot

- Digital Asset Management
  - Hierarchical storage of pictures
  - Upload
  - Tagging
  - Searching
  - Automatic thumbnail generation
- Sample application from Apache Sling

Poor man's flickr...

Slingshot Content Structure



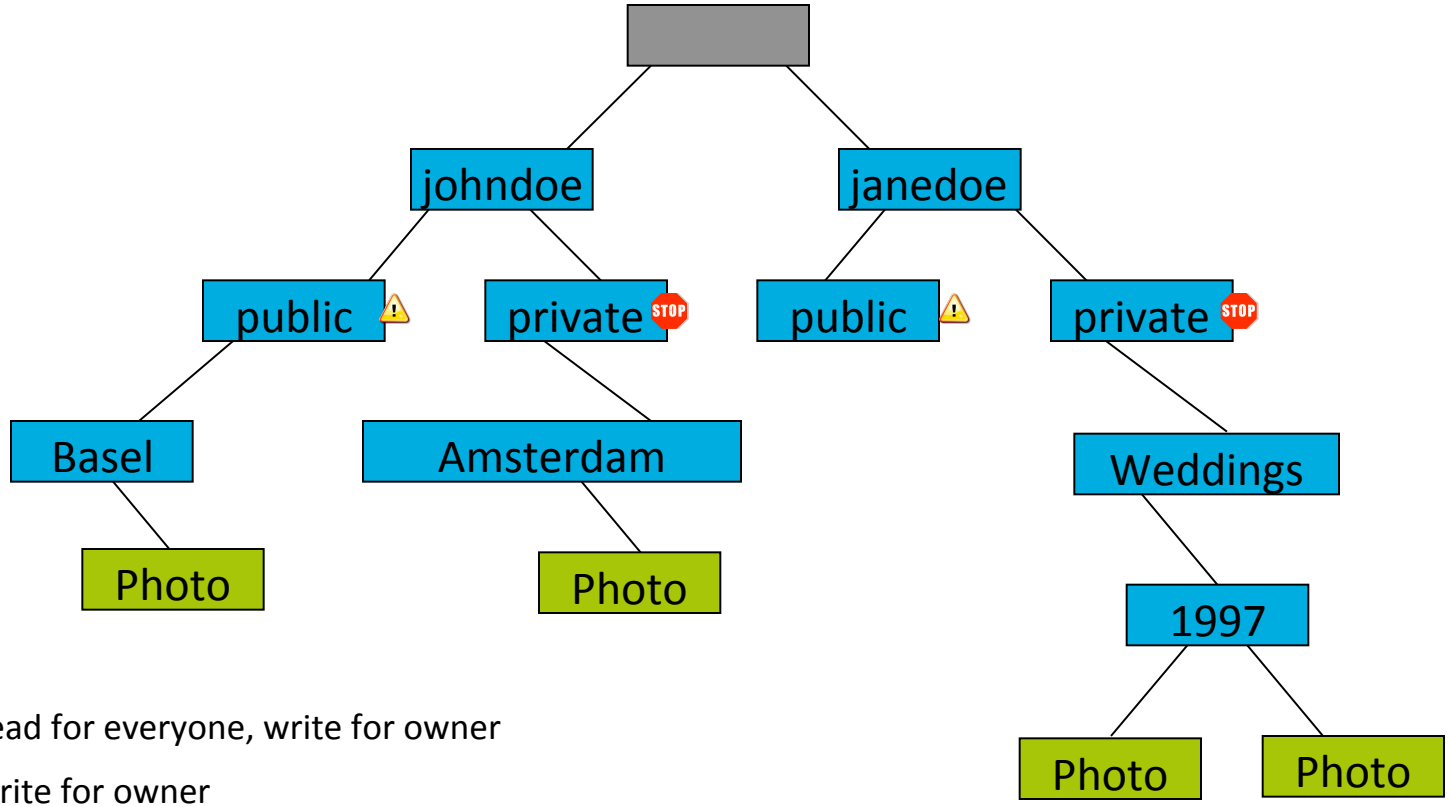
Facts About Slingshot

- Java web application
- Uses Apache Sling as web framework
- Content repository managed by Apache Jackrabbit
- Interaction through the JCR API

Authentication and Access Control

- Apache Jackrabbit supports JAAS
  - Custom login modules possible
- Deny / Allow of privileges on a node
  - Like read, write, add, delete
  - Inheritance from parent
- Tree allows structuring based on access rights
- Access control is done in the data tier!

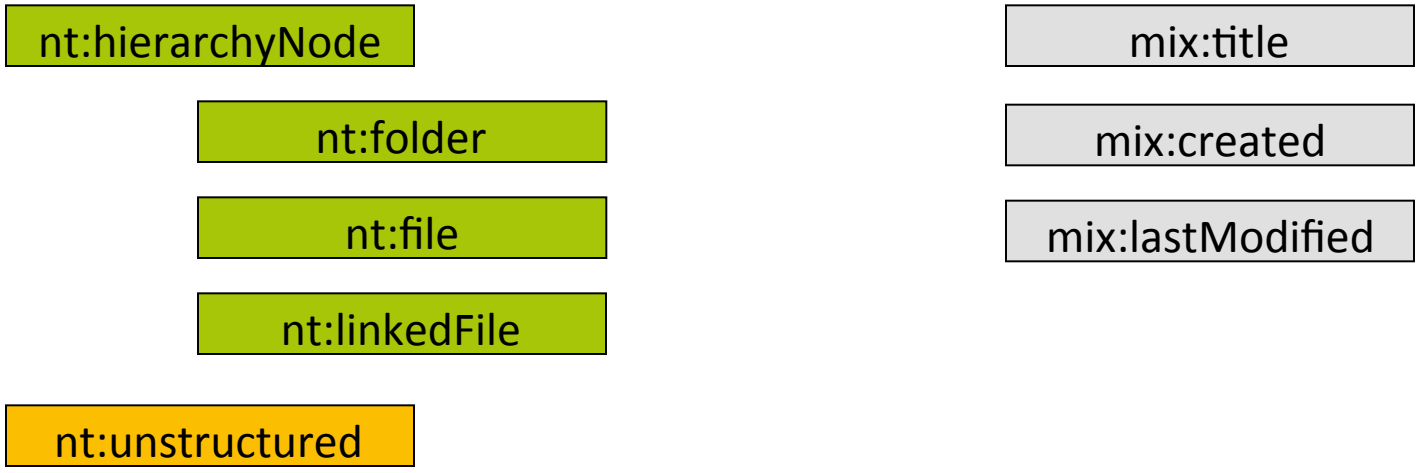
Slingshot Content Structure



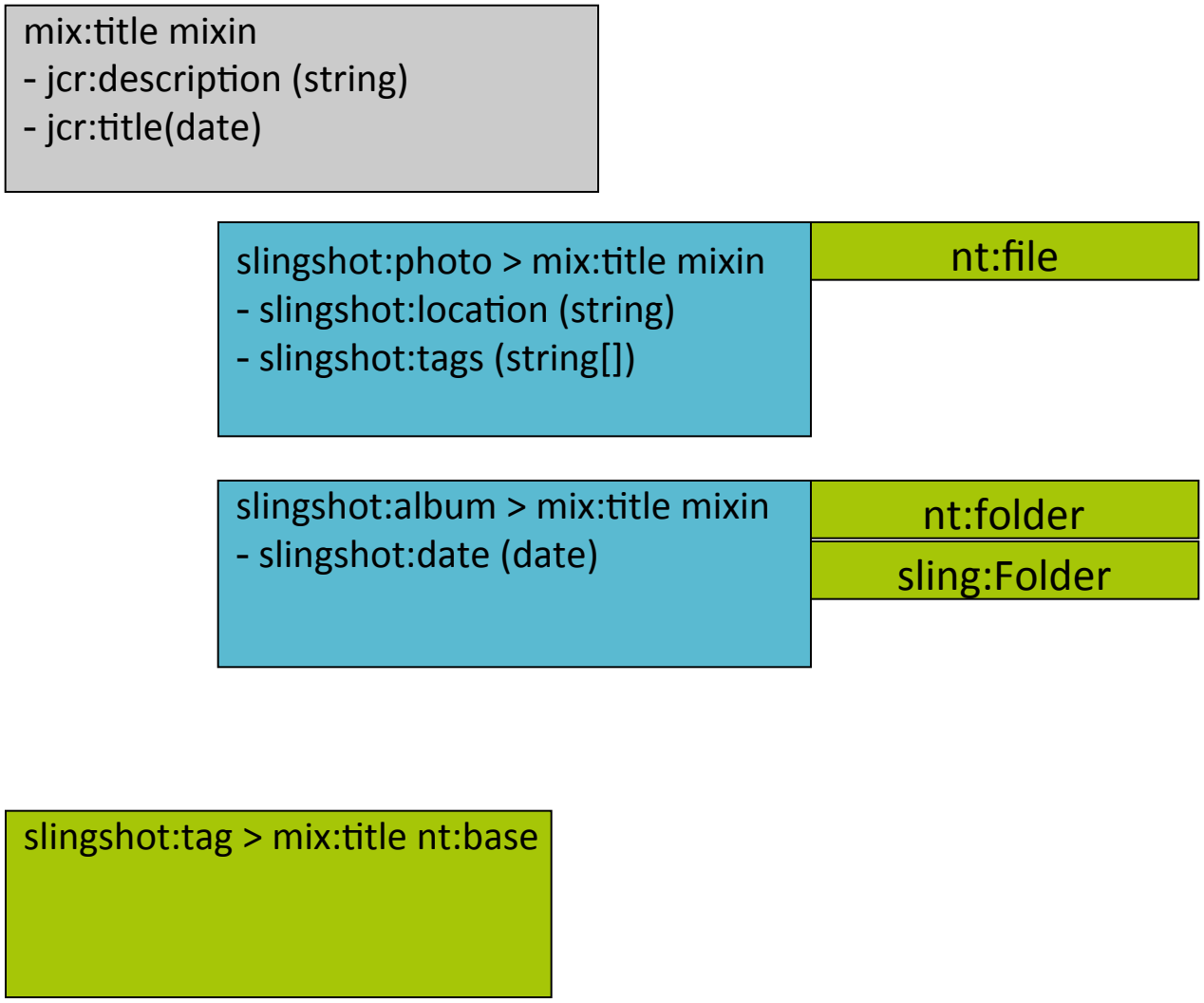
Content Modeling

- Create a content hierarchy
  - Based on how the data is used
  - Access rights
- Node type
  - Defines the structure of a node (schema)
  - Leverage existing types
  - Start unstructured
  - Use mixin node types where appropriate (semi structure)

Leverage Existing Node Types



Modeling: Node Types and Mixins



Namespaces

- Namespaces can be used for
  - Node types
  - Node and property names
- Single namespace per company or app
- Reasonably unique namespace prefix
- Prefixed names for
  - Node types (always!)
  - Structured content
- No namespace for unstructured content

Content Modeling: Words of Advice

- Use an application root node
  - /my:content or /slingshot
  - Good for searching, backup, and migration
- Avoid flat hierarchies
  - User interface complexity
- Content-driven design
  - Design your content before your application

Content Modeling: Words of Advice

- Embrace change
- Look at existing node types (JSR 2.0)
- Start unstructured
- Semi structure through mixin node types
- Checkout Apache Jackrabbit wiki and mailing lists
  - "Davids Model"

(Nearly) Everthing is Content

- Application domain specific content
- Presentation support (HTML, CSS, JavaScript, Images)
- Documentation, translations
- Server side scripts
- Application code
- …

Content Silos **without** JCR

- Structured content
  - Usually schema based
- Unstructured content
- Large data (images, movies etc.)
- Different storage for each use case

Content Repository: Combined Features

- Features of a RDBMS
  - Structure, integrity, transactions
  - Queries
- Features of a file system
  - Hierarchy, binaries
  - Access control, locking
- And more good stuff
  - Observation, versioning
  - Unstructured, multi values, sort order

Content Repository: Combined Features

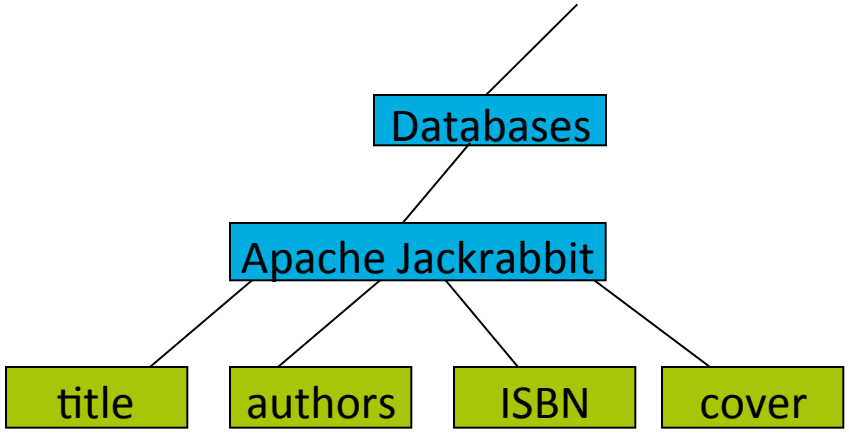
- Features of a RDBMS
  - Structure, integrity, transactions
  - Queries
- Features of a file system
  - Hierarchy, binaries
  - Access control, locking
- And more good stuff
  - Observation, versioning
  - Unstructured, multi values, sort order

No content silos

The Repository Model

- Repository: one (or more) workspaces
- Workspace contains a tree of nodes
- Node have properties
- Nodes provide the content structure
  - May have children
- Actual data is stored as values of properties
- Types and namespaces!

Sample: Product Catalog



Connecting to the Repository

- JCR 2.0 provides RepositoryFactory
- Uses Service Provider Mechanism
  - META-INF/services/javax.jcr.RepositoryFactory
- Just use
  - RepositoryFactory.getRepository(null)
- Or specify connection parameters
  - RepositoryFactory.getRepository(Map)
- With Apache Sling: OSGi service!

Working with the Repository

- Interaction is session based
  - Assemble credentials
  - Login into workspace

```
Credentials myCredentials =
        new SimpleCredentials("USERID",
                              "PASSWORD".toCharArray());
Session mySession =
        repository.login(myCredentials, "WORKSPACE");
```

Traversing the Content

- Traverse the repository
  - From the root or any node

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

Retrieve a Property

- Various ways to get a property
  - Different methods for each type
  - Automatic type conversion if possible

```
Property prop =
        albumNode.getProperty(jcr:description");
Value value = prop.getValue();
String desc = value.getString();

value.getBoolean();
value.getStream();
value.getLong();
value.getDate();
value.getDouble();
```

Changing Content

- Change everything you want
  - Add/Remove nodes
  - Add/Remove/Change properties
  - Transient space
  - Then save

```
// change
albumNode.addNode("newAlbum");
europeNode.setProperty(“jcr:description",
                "something");
// save...
mySession.save();
// ... or revert all changes
mySession.refresh(false);
```

Interaction Summary

- Get the repository
- Login to a workspace
  - Provides a session
- Use the session to
  - Access nodes and their properties
  - Navigation/traversal or query
  - CRUD and save changes
- Close the session

Advanced Development

- Apache Sling
  - REST based web framework
  - Powered by OSGi
  - Scripting Inside
- Apache Jackrabbit OCM
  - Map content to Java objects and vice versa
  - Similar to database ORMs

Observation

- Enables applications to register interest in changes
- Monitor changes
- Act on changes

API:

```
ObservationManager:
addEventListener(EventListener listener,
    int eventTypes,
    java.lang.String absPath, boolean isDeep,
    java.lang.String[] uuid,
    java.lang.String[] nodeTypeName, boolean noLocal)
```

Event Types

- Six different events
  - Node added
  - Node removed
  - Node moved
  - Property added
  - Property changed
  - Property removed

Event Listeners

- Registered with a session
- Registration with optional filters
  - Like node types, paths, event types
- Receive events for every change
  - Set of changes

API:
public void onEvent(EventIterator events);

Observation Events

- Describe changes to a workspace
  - Dispatched **after** changes are persisted
- Provide the path of the item
- Provide the user ID
- Only provided to sessions with sufficient access privileges
- Events may not be complete!
  - Example: removal of a tree of nodes

Events Advice

- Events occur after a save
  - Modification based on events is a new operation
  - Such events can be filtered
- Events may not tell the complete story
  - Tree removal
- Most common pattern
  - Node added, removed, changed
  - (Sling: additionally sends OSGi events)

Searching

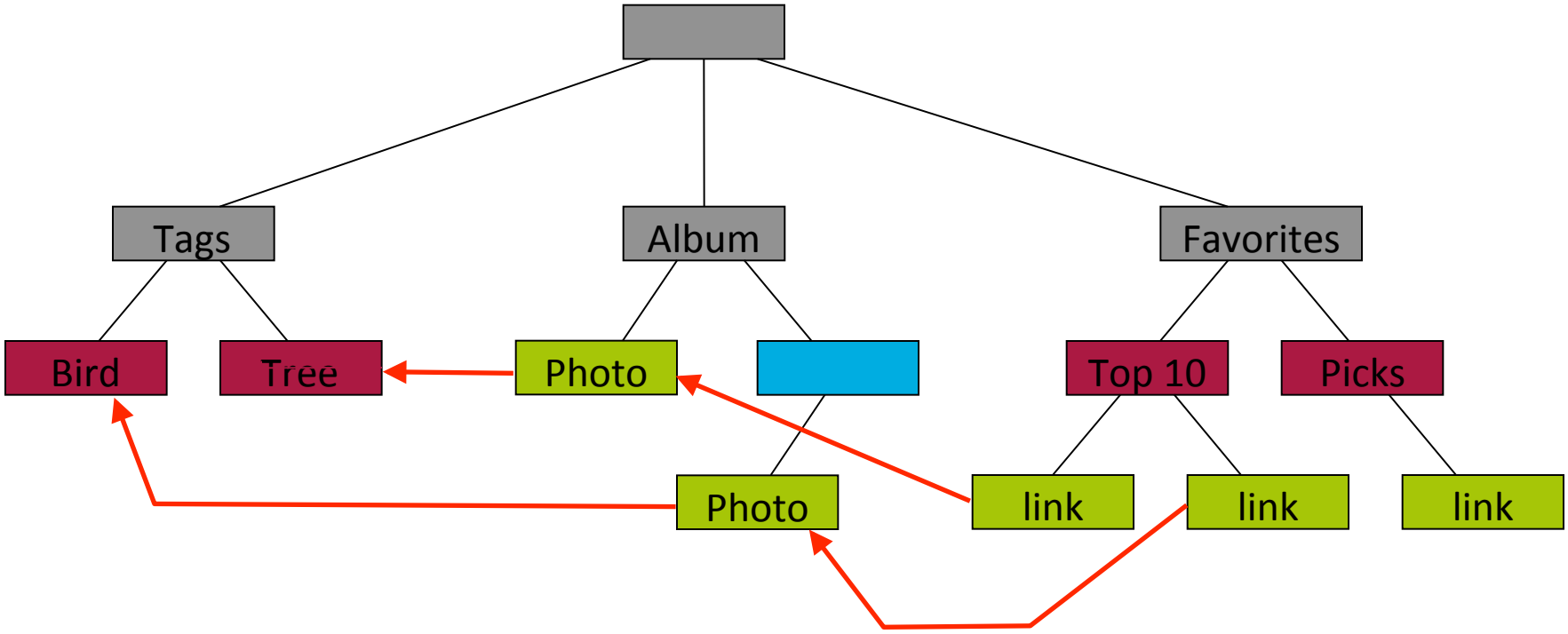
- Query API
  - Java Query Object Model
  - XPath or SQL queries
- Examples
  - SELECT * FROM slingshot:photo
    WHERE jcr:path LIKE '/slingshot/%'
  - ..AND jcr:description CONTAINS 'vancouver'

API:
Session.getWorkspace().getQueryManager():QueryManager
QueryManager.createQuery(String stmt, String language):Query;
Query.execute():QueryResult

Versioning

- Nodes can be versioned
  - Together with their child nodes
- Version history
- Getting a specific version
- Restoring a version

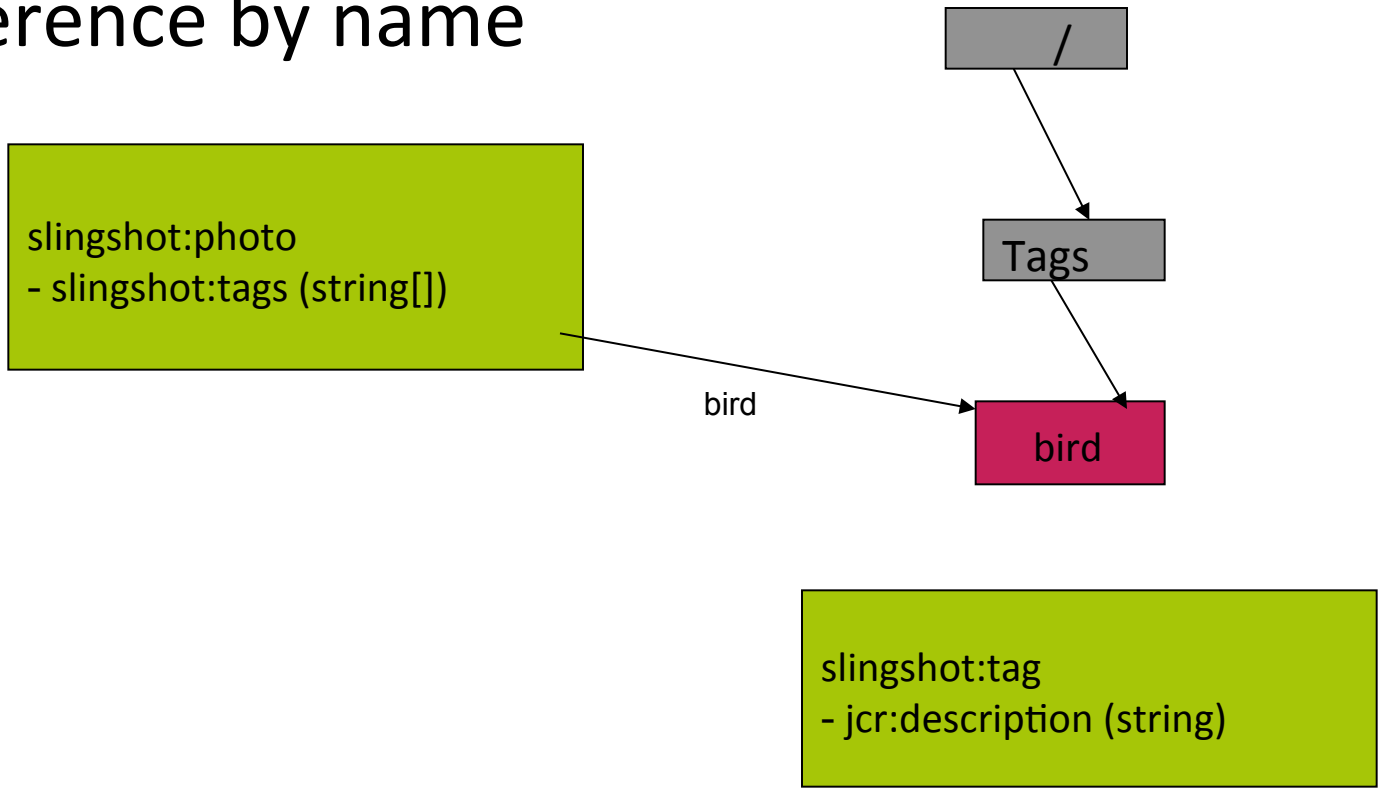
References



API:
Node.getReferences():PropertyIterator
Property.getNode():Node
Node.setProperty(String name, Node)

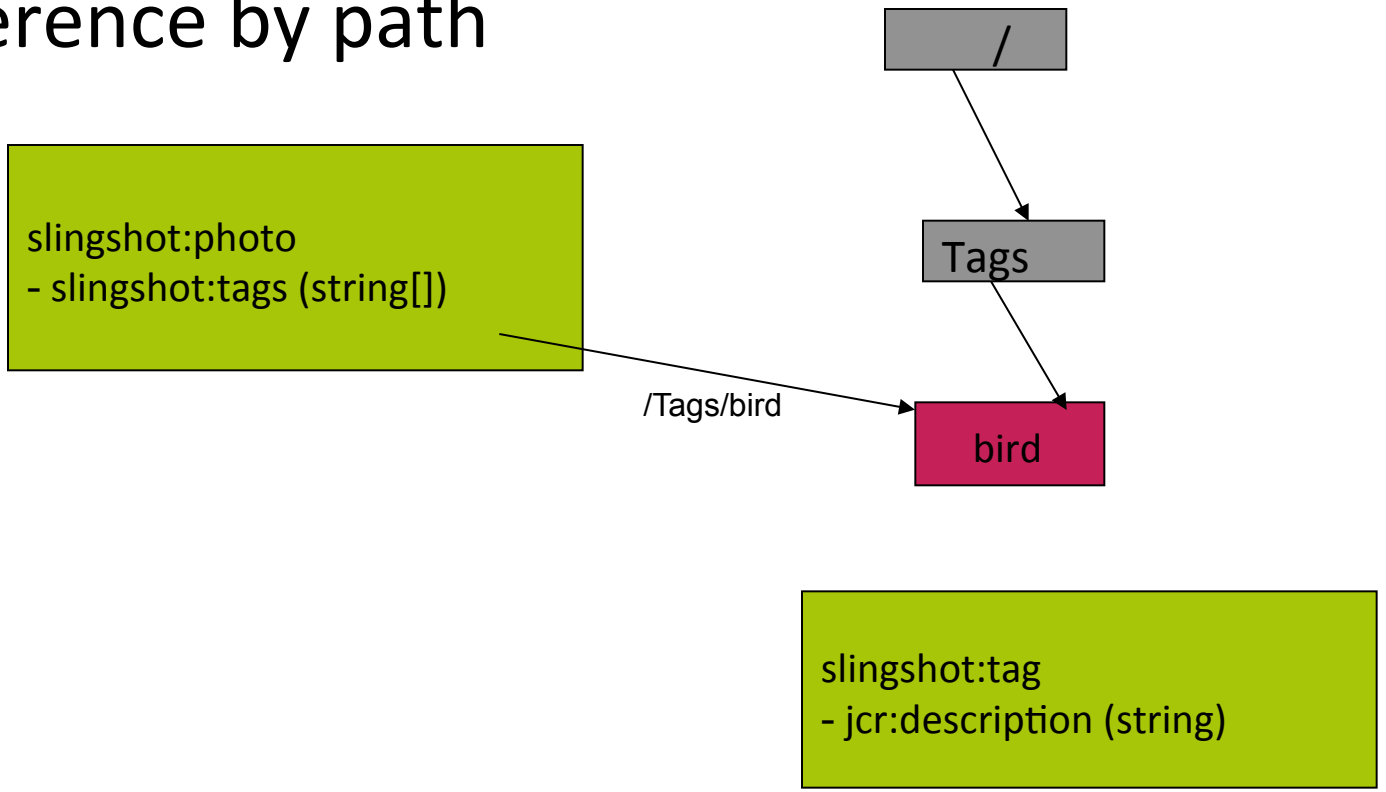
Alternative References I

- Reference by name



Alternative References II

- Reference by path



Additional Features

- Retention Policies and Hold
- API for the Nodetype Registry
- Shareable Nodes (Node with several parents)
- Journalling Observation

And remember...

- (Nearly) everything is content
  - Application Domain Specific Content
  - Presentation Support (HTML, CSS, JavaScript, Images)
  - Documentation, Translations
  - …

Consider JCR

- Data structure
- Supporting the web
- ACID
- Security
- Additional features

Famous Last Words

- Content Repositories
  - Combine advantages from FS and DB
  - Add important features
  - Structure/Access your data the way your domain requires it
  - Single repository for all your content!
- JCR – The Java API
- Apache Jackrabbit – The implementation
- Apache Sling – The web framework
- Download and join the Apache communities

Sources

- Links
  - http://jackrabbit.apache.org
  - http://sling.apache.org
- Specs
  - http://jcp.org/en/jsr/detail?id=170
  - http://jcp.org/en/jsr/detail?id=283

Contact

- Carsten Ziegeler
  - cziegeler@apache.org