JSR 170 for Developers: An Introduction to the Content Repository for Java™ Technology API

Peeter Piegaze and Marcel Reutegger

Developers and Expert Group members
Day Software AG
www.day.com

TS-4474

Content Repository for Java™ Technology

Introduction to the API

Learn the basics of the JSR 170: Content Repository for Java Technology API

JSR 170: Content Repository for Java Technology API

What Is JSR 170?

Repository Model:
Workspaces, Nodes and Properties

Basic Programming:
Connect, Traverse, Read and Write

Advanced Topics:
Node Types

Demo:
A JSR 170 Implementation in Action

JSR 170: Content Repository for Java Technology API

What Is JSR 170?

Repository Model:
Workspaces, Nodes and Properties

Basic Programming:
Connect, Traverse, Read and Write

Advanced Topics:
Node Types

Demo:
A JSR 170 Implementation in Action

What Is JSR 170?

"The API should be a standard, implementation independent, way to access content bi-directionally on a granular level within a content repository."

Source: http://www.jcp.org/en/jsr/detail?id=170

Why Do We Need a Standard API?

- The JSR 170 Expert Group members were each asked to provide sample code from their current content access API
- A JavaServer Pages™ based snippet that outputs the “Title” of a set of “Documents” in a “Folder”
- The result…

One…

```
<%
childCount = node.getContentCount();
for(int i=0;i<childCount;i++) {
    IContent child = node.getContent(i);
    Property title = child.getPropertyByName("Title");
    %><%= title.getValue() %><br/><%
}
%>
```

Two…

```
<%
childCount = node.getContentCount();
for(int i=0;i<childCount;i++) {
```

```
<%
fndocs = new IFnObjSetDualProxy(
        fnfolder.getContents(idmFolderContent.idmFolderContentDocument));
int numDocs = fndocs.getCount();
for (int i = 1; i <= numDocs; i++) {
        IFnDocumentDual fndoc =
        new IFnDocumentDualProxy(fndocs.getItem(new Integer(i)));
        IFnPropertiesDual propset = fndoc.getProperties();
        IFnPropertyDual idmTitleProp = propset.getItem("Title");
        String title = idmTitleProp.getValue();
      %><%= title %><br/><%
      if (comCleanup) {
        cleaner.release(fndoc);
      }
}
%>
```

Three…

```
<%
childCount = node.getCon
```

```
<%
fndocs = new IFnObjSetDualProxy(
        fnfolder.getContents(idmP
int numDocs = fndocs.getCount();
for (int i = 1; i <= numDocs; i++
        IFnDocumentDual fndoc =
        new IFnDocumentDualProxy(
        IFnPropertiesDual propset
        IFnPropertyDual idmTitleP
        String title = idmTitlePr
      %><%= title %><br/><%
      if (comCleanup) {
        cleaner.release(fndoc);
      }
}
%>
```

```
<%
LAPI_DOCUMENTS documents = new LAPI_DOCUMENTS(session);
LLValue childTable = new LLValue();
documents.ListObjects(volumeID, folderID,
        null, null, LAPI_DOCUMENTS.PERM_SEE, childTable);
Enumeration children = childTable.enumerateValues();
while(children.hasMoreElements()) {
        LLValue child = (LLValue)e.nextElement();
        String title = child.toString("Name");
      %><%= title %><br/><%
}
%>
```

Four…

```
<%
childCount = node.getCon
```

```
<%
LAPI_DOCUMENTS documents = new LAPI_DOCUMENTS(session);
LLValue childTable = new LLValue();
documents.ListObjects(volumeID, folderID,
        null, null, LAPI_DOCUMENTS.PERM_SEE, childTable);
Enumeration children = childTable.enumerateValues();
whil
}
%>
```

```
<%
fndocs = new IFnObjSetDualProxy(
        fnfolder.getContents(idmF
int numDocs = fndocs.getCount();
for (int i = 1; i <= numDocs; i++
        IFnDocumentDual fndoc =
        new IFnDocumentDualProxy(
        IFnPropertiesDual propset
        IFnPropertyDual idmTitleP
        String title = idmTitlePr
      %><%= title %><br/><%
      if (comCleanup) {
        cleaner.release(fndoc);
      }
}
%>
```

```
<%
IDocuments documents = new IDocumentsProxy(binder.getDocuments());
documents.cache();
int documentCount = documents.getCount();
for (int i = 0; i<documentCount; i++) {
        document = new IDocumentProxy(documents.getItemByIndex(i));
        String title = document.getTitle()
        %><%= title %><br/><%
}
%>
```

Five…

```
<%
childCount = node.getCon
for(int i=0;i<childCount
```

```
<%
fndocs = new IFnObjSetDualProxy(
        fnfolder.getContents(idmF
int numDocs = fndocs.getCount();
for (int i = 1; i <= numDocs; i++
        IFnDocumentDual fndoc =
        new IFnDocumentDualProxy(
        IFnPropertiesDual propset
        IFnPropertyDual idmTitleP
        Str
      %><%=
      if (c
        cle
      }
}
%>
```

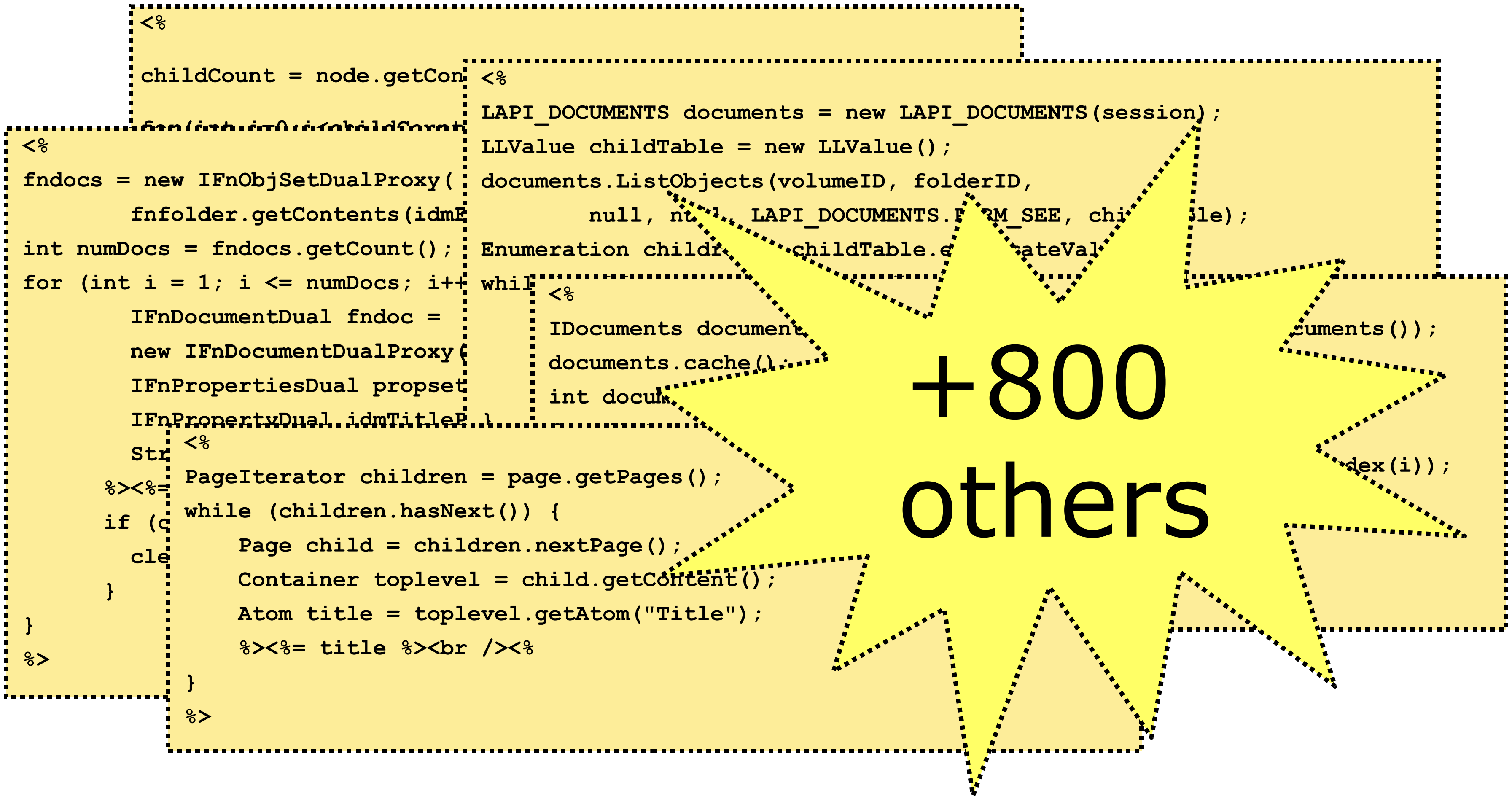
```
<%
LAPI_DOCUMENTS documents = new LAPI_DOCUMENTS(session);
LLValue childTable = new LLValue();
documents.ListObjects(volumeID, folderID,
        null, null, LAPI_DOCUMENTS.PERM_SEE, childTable);
Enumeration children = childTable.enumerateValues();
whil
```

```
<%
IDocuments documents = new IDocumentsProxy(binder.getDocuments());
documents.cache();
int documentCount = documents.getCount();

                                    nts.getItemByIndex(i));
```

```
<%
PageIterator children = page.getPages();
while (children.hasNext()) {
    Page child = children.nextPage();
    Container toplevel = child.getContent();
    Atom title = toplevel.getAtom("Title");
    %><%= title %><br /><%
}
%>
```

Eight Hundred and Five!



```
<%
childCount = node.getCon
for(int i=0;i<childCount
```

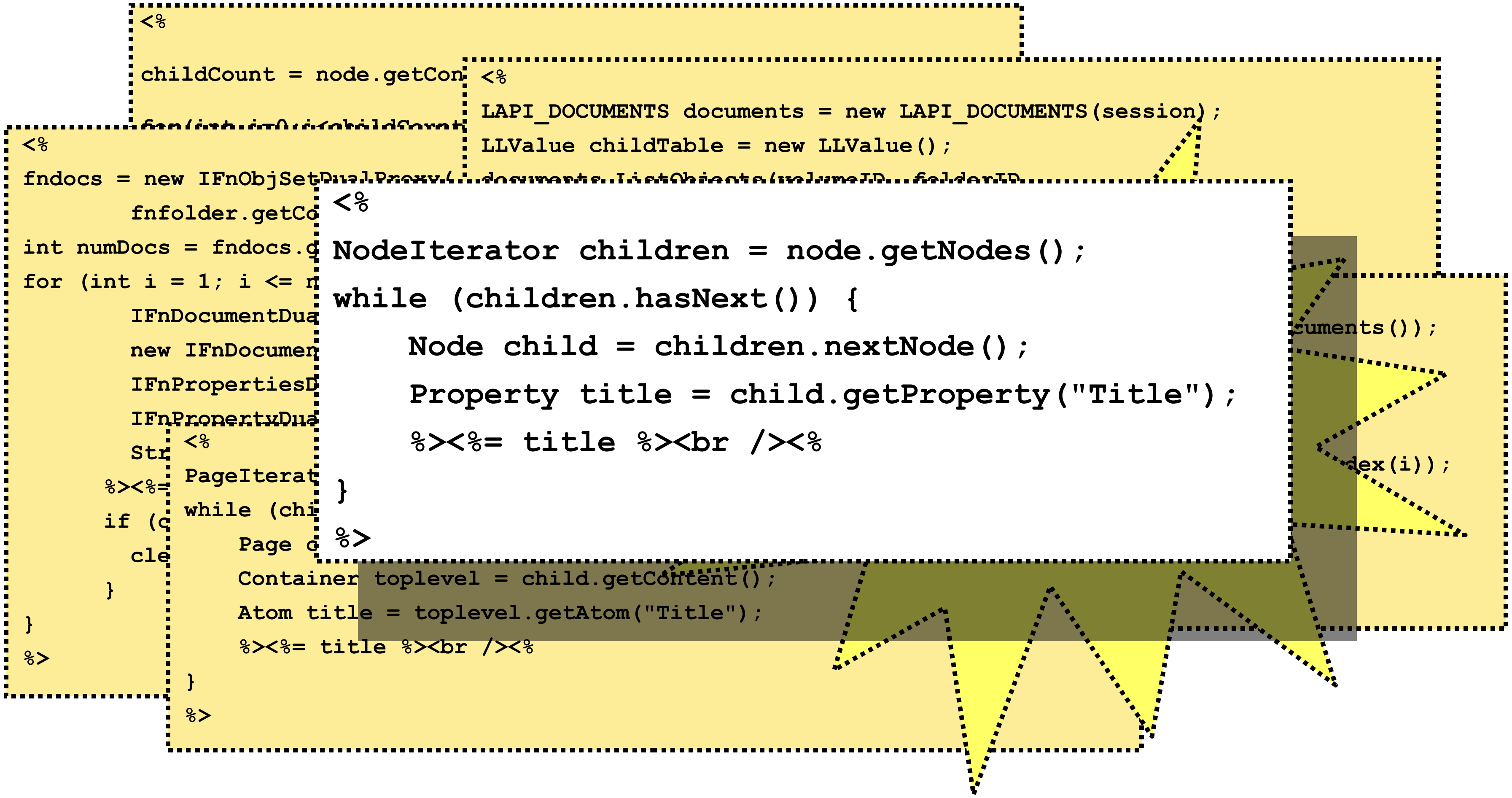
```
<%
fndocs = new IFnObjSetDualProxy(
        fnfolder.getContents(idmF
int numDocs = fndocs.getCount();
for (int i = 1; i <= numDocs; i++
        IFnDocumentDual fndoc =
        new IFnDocumentDualProxy(
        IFnPropertiesDual propset
        IFnPropertyDual idmTitleP
        Str
    %><%=
    if (c
        cle
    }
}
%>
```

```
<%
LAPI_DOCUMENTS documents = new LAPI_DOCUMENTS(session);
LLValue childTable = new LLValue();
documents.ListObjects(volumeID, folderID,
        null, nu  , LAPI_DOCUMENTS.P  RM_SEE, chi    le);
Enumeration childr    childTable.e    ateVal
whil
```

```
<%
IDocuments documen                          cuments());
documents.cache();
int docum
                                            dex(i));
```

```
<%
PageIterator children = page.getPages();
while (children.hasNext()) {
    Page child = children.nextPage();
    Container toplevel = child.getContent();
    Atom title = toplevel.getAtom("Title");
    %><%= title %><br /><%
}
%>
```

JSR 170: a Single, Standard API



```
<%
childCount = node.getCon
for(int i=0;i<childCount
```

```
<%
LAPI_DOCUMENTS documents = new LAPI_DOCUMENTS(session);
LLValue childTable = new LLValue();
documents.ListObjects(volumeID, folderID,
```

```
<%
fndocs = new IFnObjSetDualProxy(
        fnfolder.getCo
int numDocs = fndocs.g
for (int i = 1; i <= n
        IFnDocumentDua
        new IFnDocumen
        IFnPropertiesD
        IFnPropertyDua
        Str
    %><%=
    if (c
        cle
    }
}
%>
```

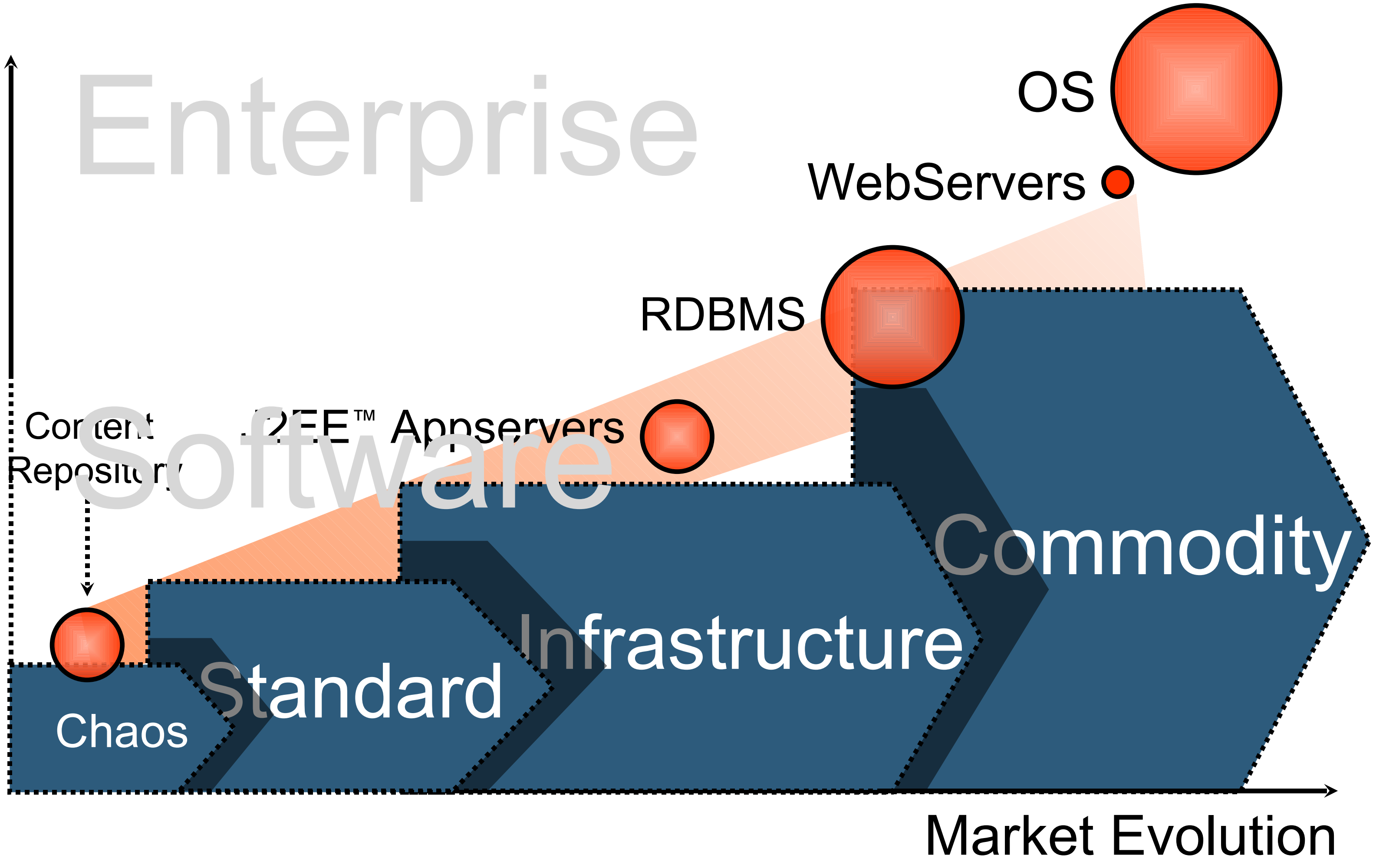
```
cuments());

dex(i));
```

```
<%
PageIterat
while (chi
    Page c
    Container toplevel = child.getContent();
    Atom title = toplevel.getAtom("Title");
    %><%= title %><br /><%
}
%>
```

```
<%
NodeIterator children = node.getNodes();
while (children.hasNext()) {
    Node child = children.nextNode();
    Property title = child.getProperty("Title");
    %><%= title %><br /><%
}
%>
```

Goal: The Content Repository Evolves Standard → Infrastructure → Commodity



Level 1 Features

Read only
Simple and Covers a Large Number of Usecases

Fine and Coarsegrained
Content Items Small to Large

Hierarchical
Parent Child Relationships, Sort Order

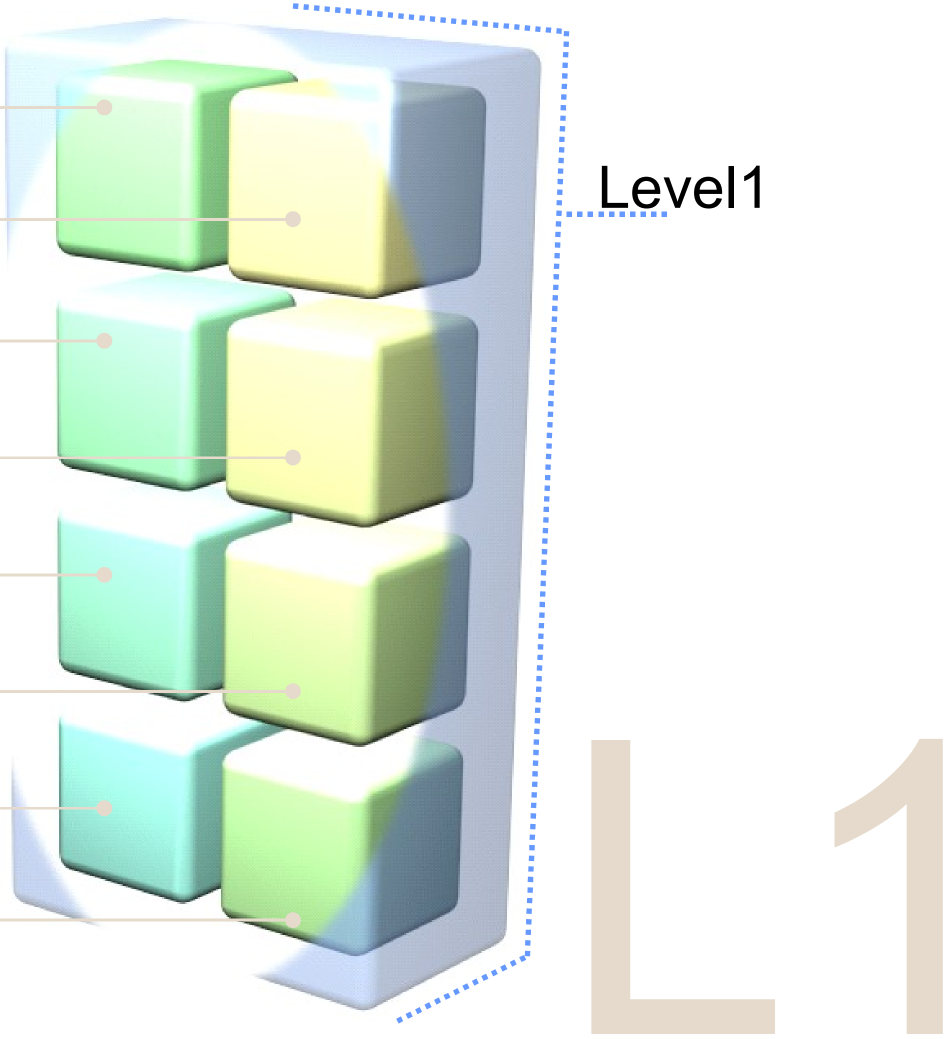
Structured
Strong Typed Information

PropertyTypes
String, Binary, Numbers, Calendar, ...

NodeTypes
Introspect Complex Content Structures

Query (XPath)
Search and Query the Repository

Export
Standardized XML Content Export



Level 1 Applications

File Edit View Favorites Tools Help
Back Search Favorites Media
Google Search Web PageRank
Address Go Links
Solutions | Product | Services | Customers | Partners | About us | C

Day : Global Content Management : JSR 170 Specification Lead - Microsoft Internet Explorer

http://www.day.com/en.html

Day

Global Content Management

Day provides Content Management, Portal Management and Digital Asset Management solutions to the world's largest companies - unifying content from any source, regardless of location, language or platform.

LATEST NEWS

- 2005-02-07 : Day Global Partner Summit 2005
- 2005-01-13 : Day and Unisys (Switzerland) AG Partner
- 2004-12-07 : WestLB Banks on Day
- 2004-11-30 : CME Content Management Europe 2004
- 2004-11-18 : Day, Pictet and Linkvest Content Management Seminar in Geneva
- 2004-11-05 : St.Gallen Website Is Best Website of All Swiss Cantons

CUSTOMERS AND PARTNERS

Deloitte Touche Tohmatsu | Partner | Top five global consulting group. Day customer and partner

InterContinental Hotels Group | Customer | Leading global hospitality group, with over 3,300 hotels across nearly 100 countries and territories ...

Consileon | Partner | Provides services and knowledge to master challenges in Software Development, Software Integration and Management Consulting...

FUTURETECH

CRX Content Repository Extreme
Day's CRX is the first industry stren
implementation of the groundbreal
170 standard for content repositori
makes content accessible for any a

Exclusive Interview: Gartner on JS
Gartner analyst Ray Valdes on the
standard, JSR 170, including indust
the standard will address and integ
customers' existing infrastructure.

Download CRX, a native JSR 170 R
JSR 170 is the coming standard for
industry. CRX is brought to you fro
of the expert group – from the com
"invented" the standard - from the

DAY HOSTS W3C: ARCHITECTURE OF THE WORLD WIDE WEB, VOLUME ONE

The World Wide Web Consortium announces the publication of "Architecture of the World

JSR 170 WEBINARS WITH DAY'S CTO DAVID NÜSCHELER

Today, (web) applications have to adapt to vendors' proprietary APIs to interact with

RESOURCES (*PASSWORD F

JSR 170 Web

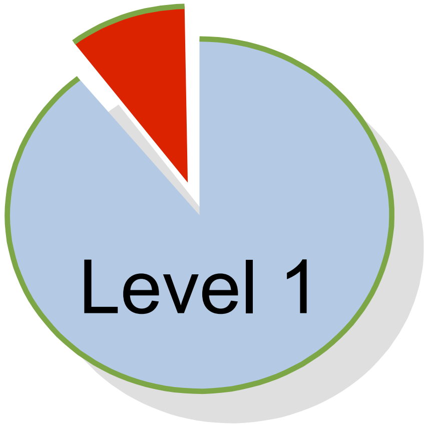
PartnerNet*

Sitemap | Webmaster | Disclaimer | Home | Deutsch | Search Day go

powered by | communiqué

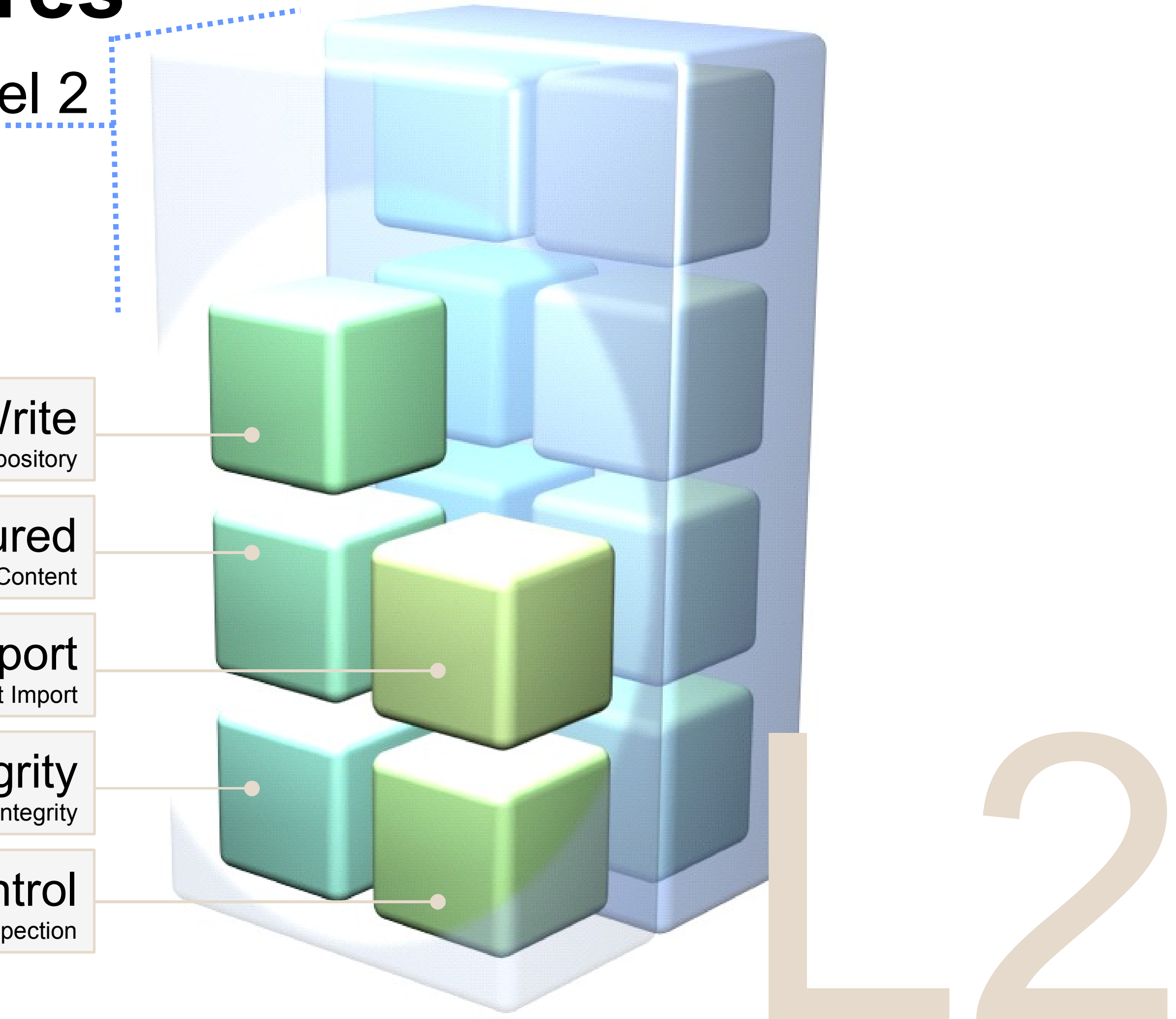
http://mediaproducts.gartner.com/gc/webletter/daysoftware/issue2/gartner1.html

Internet



- Typical level 1 applications:
  - CMS-templates, content delivery
  - Display portlets
  - Repository export
  - Reporting applications
  - Federated repositories
- Overwhelming majority of application source code is written using Level 1 calls

Level 2 Features



Write
Save changes to a repository

Unstructured
Support for weakly or unstructured Content

Import
Standardized Finegrained Content Import

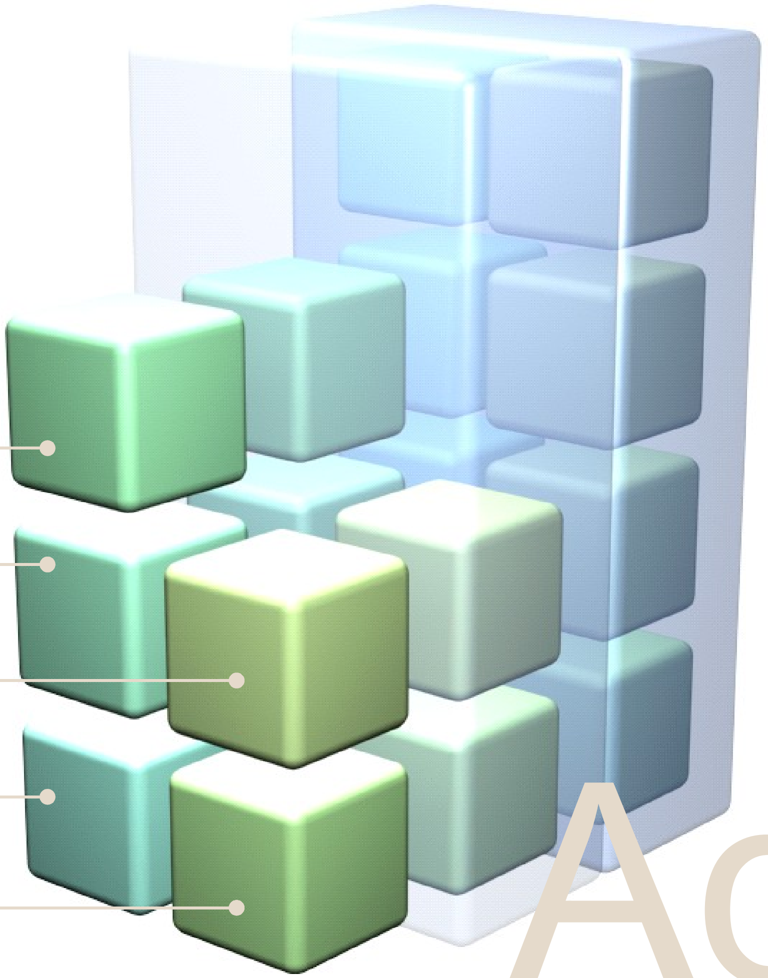
Referencial Integrity
Reference Tracking and Integrity

Access Control
Introspection

Level 2 Applications

- Typical level-2 applications:
  - Entry level content management
  - Entry level document management
  - Workflow
  - Collaboration
  - Content aggregation (content warehouse)
  - …

Advanced Features



Versioning
Workspaces, Merge, Update, Label, ...

JTA Support
(XA) Transactions

Observation
Monitor Changes in the Repository

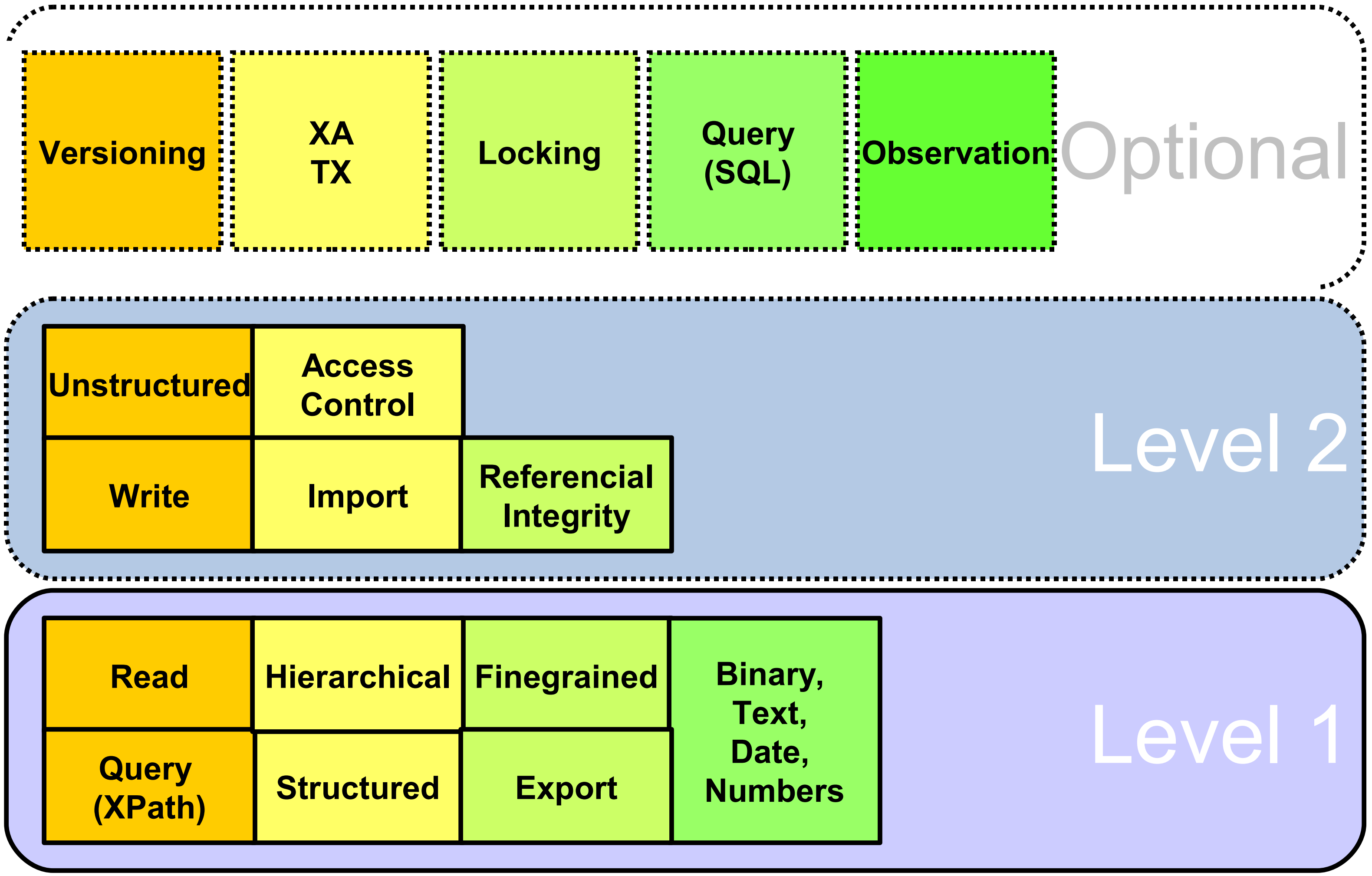
Query (SQL)
Search the Content Repeository Using SQL

Locking
Session Based and Persistent

Full JSR 170: Applications

- Typical applications that require full JSR 170 compliant repositories:
  - Complete ECM suites
  - Transactional applications
  - Source control management systems

Functional Overview



What JSR 170 Is Not:

- JSR 170 is a Content Repository API, not a CMS API
- JSR 170 does not deal with repository administration
- JSR 170 does not prescribe a semantic information model

JSR 170: the Content Repository API

What Is JSR 170?

Repository Model:
Workspaces, Nodes and Properties

Basic Programming:
Connect, Traverse, Read and Write

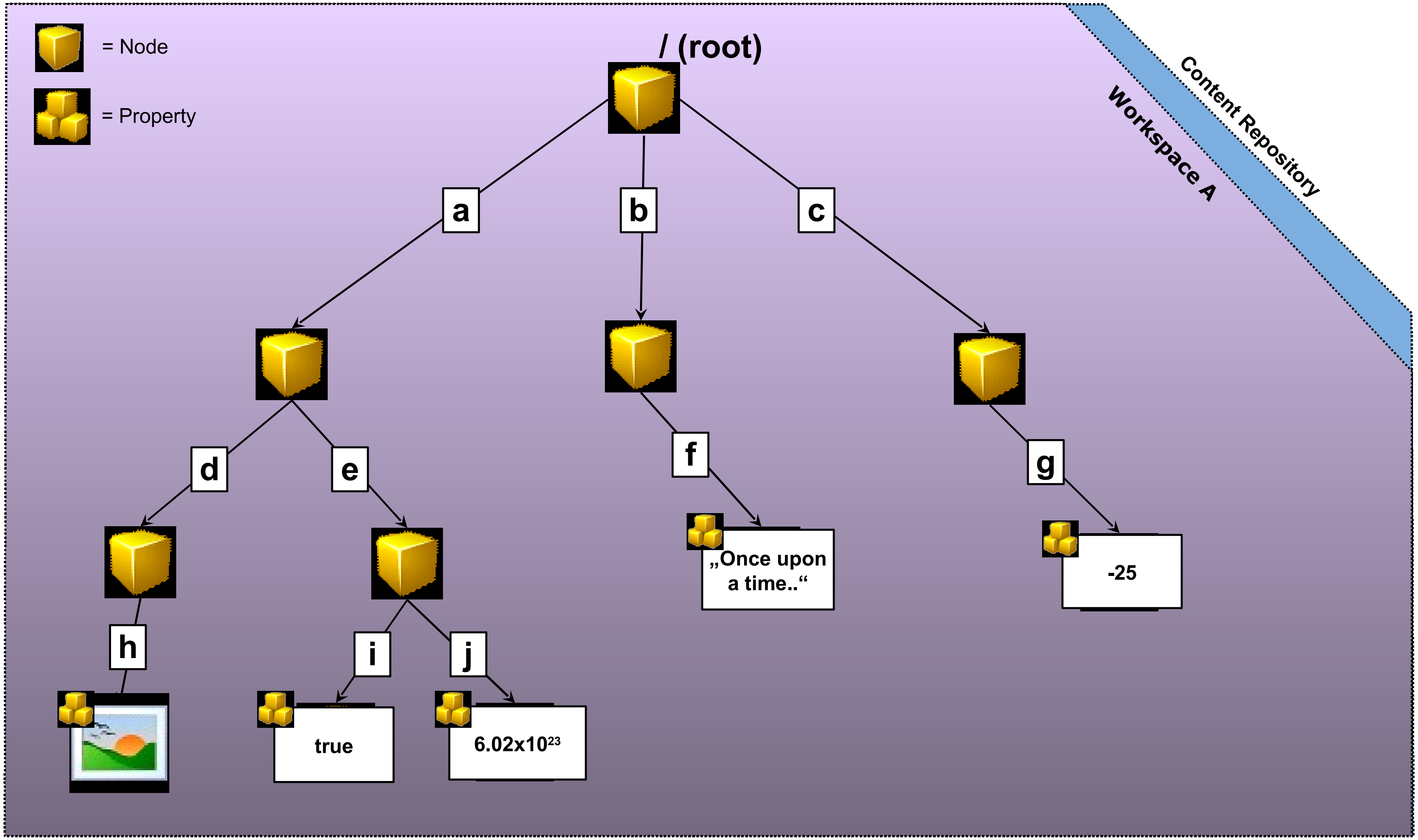
Advanced Topics:
Node Types

Demo:
A JSR 170 Implementation in Action

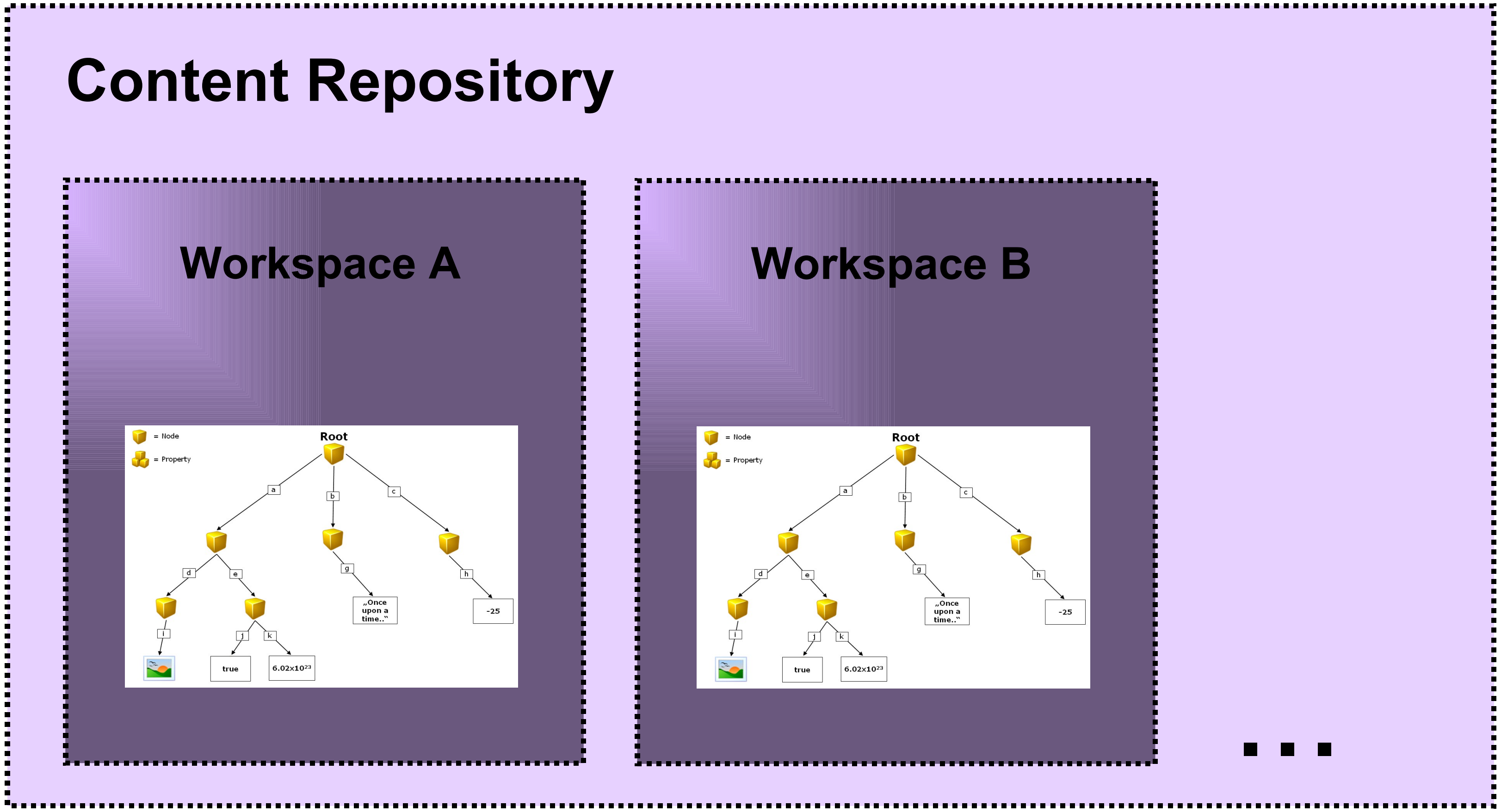
The Basic Repository Model

- A repository consists of one or more workspaces
- A workspace consists of a tree of items
- An item is either a node or a property
- A node can have child nodes and child properties
- Properties cannot have children; they are the leaves of the tree
- Nodes provide the content structure
- The “actual data” is stored as the values of properties

The Repository Model: Nodes and Properties



The Repository Model: Workspaces



JSR 170: The Content Repository API

What Is JSR 170?

Repository Model:
Workspaces, Nodes and Properties

Basic Programming:
Connect, Traverse, Read and Write

Advanced Topics:
Node Types

Demo:
A JSR 170 Implementation in Action

Connect to the Repository

- Acquire the Repository object: Exact mechanism is outside spec, but one possible way: Java Naming and Directory Interface™ API

```
// Get the Repository object
InitialContext ctx = ...

Repository repository =
       (Repository)ctx.lookup("MyRepository");
```

Get a Session

- First get your credentials
- Credential is a marker interface—we use a simple example
- Provide credentials and workspace name to get session

```
// Get a Credentials object
Credentials credentials =
  new SimpleCredentials("MyName",

"MyPassword".toCharArray());
// Get a Session
Session mySession =
  repository.login(credentials, "Workspace A");
```

Traverse the Hierarchy

- We have a Session object mySession, bound to the workspace called “Workspace A”
- We begin by getting the root node of the workspace:

```
// Get root node
Node root = mySession.getRootNode()
```

- And continue down the hierarchy:

```
// Go to the node you want
Node myNode = root.getNode("a/e");
```

Retrieve a Property

- We have Node object myNode, the node located at **`/a/e`**
- We get one of its properties, **`j :`**

```
// Retrieve a property of myNode
Property myProperty = myNode.getProperty("j");

// Get the value of the property
Value myValue = myProperty.getValue();

// Convert the value to the desired type
double myDouble = myValue.getDouble();
```

- myDouble will contain the value 6.022 x 10^23

Property Types

- STRING
- BINARY
- DATE
- LONG
- DOUBLE
- REFERENCE
- NAME
- PATH

Direct Access by Absolute Path

- In addition to incremental tree traversal, the API also supports direct access
- By absolute path:

```
// Retrieve a property by absolute path
Property myProperty =
         (Property)mySession.getItem("/a/e/j");
```

Direct Access by UUID

- Assuming that the node at `/a/e` is referenced and we know its UUID, we get it:

```
Node myNode =
  mySession.getNodeByUUID(
    "0e877cc0-b055-11da-a746-0800200c9a669999");
```

- and then get the its property j:

```
Property myProperty = myNode.getProperty("j");
```

Direct Access by Reference

- Properties of type reference store UUIDs that can be used to point to nodes
- Assuming that property **`/c/r`** is a reference property pointing to **`/a/e`**:

```
Property refProp =
   (Property)mySession.getItem("c/r/");
```

- and then jump to /a/e:

```
Node myNode = refProp.getNode();
```

Writing (In a level 2 Repository)

- If the repository is level 2 -compliant we can write to it:

```
// Retrieve a node
Node myNode = (Node) mySession.getItem("/a/e");

// Add a child node
Node newNode = myNode.addNode("n");

// Add a property
newNode.setProperty("x", "Hello");

// Persist the changes
mySession.save();
```

Transient Session Space

- Changes are held in a transient storage layer associated with the Session, until save is called:
  - On save, the changes are pushed to the persistent workspace associated with that session
  - Allows complex changes to be made that may not be valid until they are completely finished
  - Needed to enforce structural restrictions on the repository. Which leads us to…

JSR 170: the Content Repository API

What Is JSR 170?

Repository Model:
Workspaces, Nodes and Properties

Basic Programming:
Connect, Traverse, Read and Write

Advanced Topics:
Node Types

Demo:
A JSR 170 Implementation in Action

Node Types

Enforcing repository structure…or not

- A node type defines the allowed substructure of the node:
  - What properties (of what property types) the node may or must have
  - What child nodes (of what node types) the nod may or must have

An Example Node Type: Enforcing Structure

```
[Document] > Authored
 + content (Resource) mandatory
 - department (String)
 - dateCreated (Date) mandatory autocreated
 - language (String) = "en"
```

Another Node Type: Supporting Unstructured Content

```
[nt:unstructured]
  + * (nt:base) =nt:unstructured multiple
  - * (undefined)
  - * (undefined) multiple
```

JSR 170: the Content Repository API

What Is JSR 170?

Repository Model:
Workspaces, Nodes and Properties

Basic Programming:
Connect, Traverse, Read and Write

Advanced Topics:
Node Types

Demo:
A JSR 170 Implementation in Action

DEMO

Q&A

Peeter Piegaze
Marcel Reutegger

JSR 170 for Developers: An Introduction to the Content Repository for Java™ Technology API

Peeter Piegaze and Marcel Reutegger

Developers and Expert Group members
Day Software AG
www.day.com

TS-4474