



*IBM FileNet Content Manager
5.0: Java API Programming*

(Course code F143)

Student Notebook

ERC 2.0

Authorized

IBM | Training

Trademarks

IBM® and the IBM logo are registered trademarks of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DB2®

FileNet®

Tivoli®

WebSphere®

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

May 2011 edition

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by a real business enterprise is entirely coincidental.

© Copyright International Business Machines Corporation 2011.

This document may not be reproduced in whole or in part without the prior written permission of IBM.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Contents

Trademarks	xxi
Course description	xxiii
Agenda	xxv
Unit 1. Content Engine Java API Overview.....	1-1
Unit lessons	1-2
Lesson 1.1. Content Engine Java API concepts	1-3
Lesson: Content Engine Java API concepts	1-4
Activities that you need to complete	1-5
FileNet P8 architecture	1-6
Content Engine and Java EE	1-7
Content Engine-supported transport protocols	1-8
Content Engine Java APIs	1-9
Application types supported by Content Engine APIs	1-10
Required Java Archive (JAR) files	1-11
Commonly used Content Engine Java packages	1-13
Java Authentication and Authorization Service (JAAS)	1-14
JAAS configuration files	1-16
Resources	1-17
Activities	1-18
Lesson 1.2. Content Engine objects	1-19
Lesson: Content Engine objects	1-20
Activities that you need to complete	1-21
Interface hierarchy	1-22
EngineObject and RepositoryObject	1-23
IndependentObject and DependentObject	1-24
Base classes and interfaces	1-25
Core objects of the API	1-26
Instantiating objects - Factory class	1-27
Collection objects	1-28
Actions in pending state	1-29
Working with properties	1-30
Setting up a thick client development environment	1-31
Using Eclipse IDE for writing your code	1-32
Demonstrations	1-33
Activities	1-34
Unit 2. Communication with the Content Engine.....	2-1
Unit lessons	2-2
Lesson 2.1. Get a connection to a FileNet P8 domain	2-3
Lesson: Get a connection to a FileNet P8 domain	2-4
Activities that you need to complete	2-5

Connection object	2-6
Steps to get a connection to a FileNet P8 domain	2-7
Factory.Connection.getConnection(...)	2-8
UserContext.createSubject(...)	2-9
UserContext.get()	2-10
userContext.pushSubject(...)	2-11
Sample code to get a connection to a FileNet P8 domain	2-12
Demonstrations	2-13
Activities	2-14
Lesson 2.2. Retrieve the domain and the object stores	2-15
Lesson: Retrieve the domain and the object stores	2-16
Activities that you need to complete	2-17
Domain object	2-18
Global Configuration Database and Domain objects	2-19
Steps to retrieve Domain and ObjectStore objects	2-20
Factory.Domain.fetchInstance(...)	2-21
domain.get_ObjectStores()	2-22
objectStore.getAccessAllowed()	2-23
Sample code to retrieve the Domain object	2-24
Sample code to retrieve the ObjectStore objects	2-25
Demonstrations	2-26
Activities	2-27
Unit 3. Folders	3-1
Unit lessons	3-2
Lesson 3.1. Create and delete Folder objects	3-3
Lesson: Create and delete Folder objects	3-4
Activities that you need to complete	3-5
What is a Folder object?	3-6
Containment concepts	3-7
Create a Folder object	3-8
Steps to create a Folder object (1)	3-9
Steps to create a Folder object (2)	3-10
folder.createSubFolder(...)	3-11
Factory.Folder.newInstance(...)	3-12
Factory.ObjectStore.fetchInstance(...)	3-13
objectStore.get_RootFolder()	3-14
folder.set_Parent(...)	3-15
folder.set_FolderName(...)	3-16
folder.save(...)	3-17
Sample code to create a folder	3-18
Demonstrations	3-19
Deleting folders	3-20
Steps to delete a Folder object	3-21
Factory.Folder.fetchInstance(...)	3-22
folder.delete()	3-23
Sample code to delete a folder	3-24
Demonstrations	3-25

Activities	3-26
Lesson 3.2. Retrieve objects from a folder	3-27
Lesson: Retrieve objects from a folder	3-28
Activities that you need to complete	3-29
Referential containment	3-30
Steps to retrieve objects from a folder (1)	3-32
Steps to retrieve objects from a folder (2)	3-33
<code>folder.get_SubFolders()</code>	3-34
<code>folder.getProperties().getBooleanValue(...)</code>	3-35
<code>folder.get_CreatedDocuments()</code>	3-36
<code>folder.get_ContainedObjects()</code>	3-37
<code>referentialContainmentRelationship.get_Name()</code>	3-38
<code>referentialContainmentRelationship.get_Head()</code>	3-39
<code>independentObject.getClassName()</code>	3-40
Sample code to retrieve Folder objects	3-41
Sample code to retrieve Document objects	3-42
Sample code to retrieve objects from a folder	3-43
Demonstrations	3-44
Activities	3-45
Unit 4. Custom Objects.....	4-1
Unit lessons	4-2
Lesson 4.1. Create custom objects	4-3
Lesson: Create custom objects	4-4
Activities that you need to complete	4-5
What are custom objects?	4-6
Custom objects are unlike Document objects	4-7
Why use custom objects?	4-8
Creating a custom object	4-9
Steps to create a custom object (1)	4-10
Steps to create a custom object (2)	4-11
<code>Factory.CustomObject.createInstance(...)</code>	4-12
<code>customObject.getProperties ()</code>	4-13
<code>properties.putvalue(...)</code>	4-14
<code>independentlyPersistableObject.save(...)</code>	4-15
<code>folder.file(...)</code>	4-16
Sample code to create a custom object and set properties	4-17
Sample code to set a multivalued property of a custom object	4-18
Sample code to save and file a custom object to a folder	4-19
Demonstrations	4-20
Activities	4-21
Lesson 4.2. Retrieve custom object properties	4-23
Lesson: Retrieve custom object properties	4-24
Activities that you need to complete	4-25
Steps to retrieve custom object properties	4-26
<code>Factory.CustomObject.fetchInstance(...)</code>	4-27
<code>properties.getStringValue(...)</code>	4-28
<code>properties.getInteger32Value(...)</code>	4-29

properties.getDateTimeValue(...)	4-30
properties.getStringListValue(...)	4-31
Sample code to retrieve a custom object	4-32
Sample code to retrieve custom object properties	4-33
Sample code to retrieve a multivalued property	4-34
Demonstrations	4-35
Activities	4-36
Unit 5. Documents	5-1
Unit lessons	5-2
Lesson 5.1. Create Document objects	5-3
Lesson: Create Document objects	5-4
Activities that you need to complete	5-5
Document objects	5-6
Available actions on Document objects	5-7
Steps to create a document	5-8
Steps to create a Document object	5-9
Steps to save the document and file	5-10
Steps to set content to the Document object	5-11
Steps to set content reference element	5-12
Factory.Document.createInstance(...)	5-13
Factory.ContentElement.createList()	5-14
Factory.ContentTransfer.createInstance()	5-15
contentTransfer.setCaptureSource(...)	5-16
contentTransfer.set_ContentType(...)	5-17
Factory.ContentReference.createInstance()	5-18
contentReference.set_ContentLocation(...)	5-19
contentReference.set_ContentType(...)	5-20
document.set_ContentElements(...)	5-22
document.checkin(...)	5-23
document.getProperties ()	5-25
properties.putvalue(...)	5-26
document.set_MimeType(...)	5-27
folder.file(...)	5-28
independentlyPersistableObject.save(...)	5-29
Sample code to create a Document object (1)	5-30
Sample code to create a Document object (2)	5-31
Sample code to set multiple content elements	5-32
Sample code to set content reference	5-33
Demonstrations	5-34
Activities	5-35
Lesson 5.2. Retrieve a document and its content.	5-37
Lesson: Retrieve a document and its content	5-38
Activities that you need to complete	5-39
Steps to retrieve a Document object	5-40
Steps to retrieve contents of a Document object	5-41
Factory.Document.fetchInstance(...)	5-42
document.get_Name()	5-43

document.get_ContentElements()	5-44
contentElement.get_ContentType()	5-45
document.get_ContentElementsPresent()	5-46
contentTransfer.get_RetrievalName()	5-47
contentTransfer.get_ContentSize()	5-48
contentTransfer.accessContentStream()	5-49
Sample code to retrieve a document	5-50
Sample code to retrieve content elements of a document	5-51
Sample code to retrieve content stream	5-52
Demonstrations	5-53
Activities	5-54
Unit 6. Properties	6-1
Unit lessons	6-2
Lesson 6.1. Retrieve property descriptions	6-3
Lesson: Retrieve property descriptions	6-4
Activities that you need to complete	6-5
Properties defined	6-6
Setting and accessing property values	6-8
Property cache	6-9
Property Filters	6-10
Components of property filters object	6-11
propertyFilter.addIncludeProperty(...)	6-12
propertyFilter.addExcludeProperty(...)	6-13
propertyFilter.addIncludeType(...)	6-14
ClassDescription objects	6-15
PropertyDescription objects	6-16
Steps to retrieve class descriptions	6-17
Steps to retrieve property descriptions	6-18
objectStore.get_ClassDescriptions()	6-19
classDescription.get_PropertyDescriptions()	6-20
propertyDescription.get_SymbolicName()	6-21
propertyDescription.get_IsReadOnly()	6-22
propertyDescription.get_IsValueRequired()	6-23
propertyDescription.get_Setability()	6-24
propertyDescription.get_Cardinality()	6-25
propertyDescription.get_IsSystemGenerated()	6-26
Sample code to retrieve class descriptions	6-27
Sample code to retrieve a document	6-28
Sample code to retrieve property descriptions	6-29
Demonstrations	6-30
Activities	6-31
Lesson 6.2. Retrieve a choice list	6-33
Lesson: Retrieve a choice list	6-34
Activities that you need to complete	6-35
Choice list objects	6-36
Using choice lists	6-37
ChoiceList interface	6-38

Choice objects	6-39
Steps to retrieve a choice list and choices	6-40
Steps to retrieve a choice list	6-41
Steps to retrieve individual choices	6-42
propertyDescription.get_ChoiceList(...)	6-43
choiceList.get_ChoiceValues()	6-44
choiceList.get_HasHierarchy()	6-45
choice.get_ChoiceType()	6-46
choice.get_ChoiceIntegerValue()	6-47
choice.get_ChoiceStringValue()	6-48
Sample code to retrieve a choice list	6-49
Sample code to retrieve the individual choices	6-50
Demonstrations	6-51
Activities	6-52
Lesson 6.3. Retrieve object properties	6-53
Lesson: Retrieve object properties	6-54
Activities that you need to complete	6-55
Steps to retrieve a given property value	6-56
Steps to retrieve a property description for a property	6-57
Steps to retrieve a property value	6-58
propertyDescription.get_DataType()	6-59
property.getStringValue()	6-60
property.getStringListValue()	6-61
Sample code to retrieve object properties	6-62
Sample code to retrieve a property value (1)	6-63
Sample code to retrieve a property value (2)	6-64
Sample code to retrieve a property value (3)	6-65
Demonstrations	6-66
Activities	6-67
Unit 7. Searches	7-1
Unit lessons	7-2
Lesson 7.1. Search for objects	7-3
Lesson: Search for objects	7-4
Activities that you need to complete	7-5
Overview of searches	7-6
Components of a query	7-7
Searches	7-8
SearchScope class	7-9
SearchSQL	7-10
Fetching the result set - contents	7-11
Steps to search for objects based on the property	7-12
Steps to search for objects	7-13
SearchScope.fetchObjects(...)	7-14
Sample code to search for documents based on property	7-15
Additional sample SQL statements for searching objects	7-16
objectStore.set_DefaultQueryTimeLimit(...)	7-17
serverCacheConfiguration.set_NonPagedQueryMaxSize(...)	7-18

Tips for searches	7-19
Demonstrations	7-21
Activities	7-22
Lesson 7.2. Search for documents with paging.	7-23
Lesson: Search for documents with paging	7-24
Activities that you need to complete	7-25
Results sets and paging	7-26
Fetching the result set - controls	7-27
documentset.pagelimiter()	7-28
Sample code to search for documents with paging	7-29
serverCacheConfiguration.set_QueryPageMaxSize(...)	7-30
serverCacheConfiguration.set_QueryPageDefaultSize(...)	7-31
Demonstrations	7-32
Activities	7-33
Lesson 7.3. Search for objects across multiple object stores	7-35
Lesson: Search for objects across multiple object stores	7-36
Activities that you need to complete	7-37
Create alias IDs	7-38
Assign an alias ID to a class	7-40
Steps to search for objects across the object stores	7-41
Sample code to search across object stores (1)	7-42
Sample code to search across object stores (2)	7-43
Demonstrations	7-44
Activities	7-45
Lesson 7.4. Build SQL statements	7-47
Lesson: Build SQL statements	7-48
Activities that you need to complete	7-49
Building SQL statements	7-50
Steps to build an SQL statement	7-51
Elements of a basic SQL statement	7-52
Conventions for SQL query syntax statements	7-53
SELECT list	7-54
FROM clause	7-55
WHERE clause	7-56
IN operator (IBM FileNet P8 SQL extensions)	7-57
OBJECT Function Description	7-59
IsClass Function Description	7-60
Available SearchSQL helper class methods	7-61
searchSQL.setSelectList(...)	7-62
searchSQL.setFromClauseInitialValue(...)	7-63
searchSQL.setFromClauseAdditionalJoin(...)	7-64
searchSQL.setWhereClause (...)	7-66
searchSQL.setOrderByClause(...)	7-67
searchSQL.setQueryString (...)	7-68
searchSQL.setMaxRecords(...)	7-69
Sample code to construct SQL statements	7-70
Demonstrations	7-71
Activities	7-72

Lesson 7.5. Content-based retrieval	7-73
Lesson: Content based retrieval	7-74
Activities that you need to complete	7-75
Content Based Retrieval (CBR)	7-76
IBM Content Search Services	7-78
Content Engine Search Architecture	7-79
Content searches	7-81
Initiating content search	7-82
Full-Text joins	7-83
CONTAINS Function	7-85
FREETEXT function	7-87
Ranking	7-89
Common stop words	7-90
Proximity operators (Verity terminology)	7-91
Autonomy K2 query language	7-93
Query option	7-94
Steps to return object properties	7-96
Steps to retrieve properties of the objects	7-97
searchScope.fetchRows(...)	7-98
repositoryRow.getProperties()	7-99
searchSQL.setContainsRestriction(...)	7-100
searchSQL.setFreetextRestriction(...)	7-101
Sample code for content-based document searching	7-102
Sample code for retrieving the properties	7-103
Additional sample VQL statements for content-based search	7-104
IBM Content Search Services- XML Support	7-105
Demonstrations	7-106
Activities	7-107
 Unit 8. Document Versions	 8-1
Unit lessons	8-2
Lesson 8.1. Retrieve versionable objects	8-3
Lesson: Retrieve versionable objects	8-4
Activities that you need to complete	8-5
Document versions	8-6
Three levels of versioning	8-7
Two-level versioning (1)	8-9
Two-level versioning (2)	8-10
Document version states	8-11
Reservation object	8-12
Versionable object	8-13
VersionSeries object	8-14
Versionable versus VersionSeries object (1)	8-15
Versionable versus VersionSeries object (2)	8-16
Document versions example	8-17
Steps to retrieve versions of a document	8-18
versionable.get_Versions()	8-19
versionable.get_MajorVersionNumber()	8-20

versionable.get_MinorVersionNumber()	8-21
versionable.get_VersionStatus()	8-22
Sample code to retrieve versions for a document	8-23
Demonstrations	8-24
Steps to retrieve version series for a document	8-25
versionable.get_VersionSeries()	8-26
versionSeries.get_ReleasedVersion()	8-27
versionSeries.get_CurrentVersion()	8-28
Sample code to retrieve a version series (1)	8-29
Sample code to retrieve a version series (2)	8-30
Demonstrations	8-31
Activities	8-32
Lesson 8.2. Check out and check in a document	8-33
Lesson: Check out and check in a document	8-34
Activities that you need to complete	8-35
Steps to check out a document	8-36
versionable.get_IsCurrentVersion()	8-37
document.get_CurrentVersion()	8-38
versionable.get_IsReserved()	8-39
versionable.checkout(...)	8-40
Sample code to check out a document	8-42
Demonstrations	8-43
Steps to check in a document	8-44
versionable.get_Reservation()	8-45
document.checkin(...)	8-46
Sample code to check the version status	8-48
Sample code to check in a document	8-49
Demonstrations	8-50
Steps to cancel a checked-out document	8-51
versionable.cancelCheckout()	8-52
Sample code to cancel a checkout of a document	8-53
Demonstrations	8-54
Activities	8-55
Lesson 8.3. Promote and demote a document	8-57
Lesson: Promote and demote a document	8-58
Activities that you need to complete	8-59
Steps to promote a version for a document	8-60
versionable.promoteVersion()	8-61
Sample code to promote a version for a document	8-62
Demonstrations	8-63
Steps to demote a version for a document	8-64
versionable.DemoteVersion()	8-65
Sample code to demote a version for a document	8-66
Demonstrations	8-67
Activities	8-68
Unit 9. Security	9-1
Unit lessons	9-2

Lesson 9.1. Retrieve object permissions	9-3
Lesson: Retrieve object permissions	9-4
Activities that you need to complete	9-5
Object-level security	9-6
Access control objects and properties	9-7
Permission sources	9-8
Steps to get object permissions	9-9
containable.get_Permissions()	9-10
accessPermission.get_GranteeName()	9-11
accessPermission.get_GranteeType()	9-12
accessPermission.get_PermissionSource()	9-13
accessPermission.get_AccessMask()	9-14
accessPermission.get_AccessType()	9-15
Sample code to get object permissions (1)	9-16
Sample code to get object permissions (2)	9-17
Demonstrations	9-18
Activities	9-19
Lesson 9.2. Set object permissions	9-21
Lesson: Set object permissions	9-22
Activities that you need to complete	9-23
Security inheritance	9-24
Setting security with a security parent	9-25
Steps to set permissions on a folder (1)	9-26
Steps to set permissions on a folder (2)	9-27
Factory.AccessPermission.createInstance(...)	9-28
accessPermission.set_GranteeName()	9-29
accessPermission.set_AccessType(...)	9-30
accessPermission.set_InheritableDepth(...)	9-31
accessPermission.set_AccessMask()	9-32
folder.set_Permissions(...)	9-33
Sample code to set folder permissions	9-34
Steps to set the security folder to a document	9-35
document.get_Containers()	9-36
document.set_SecurityFolder(...)	9-37
Sample code to set the security folder to a document	9-38
Demonstrations	9-39
Activities	9-40
Lesson 9.3. Apply a security template	9-41
Lesson: Apply a security template	9-42
Activities that you need to complete	9-43
Security policies and templates	9-44
Types of security templates	9-45
Creating a security template	9-46
Steps to apply a security template	9-47
Steps to apply a security policy to a document (1)	9-48
Steps to apply a security policy to a document (2)	9-49
objectStore.get_SecurityPolicies()	9-50
securityPolicy.get_SecurityTemplates()	9-51

securityTemplate.get_ApplyStateID()	9-52
document.applySecurityTemplate(...)	9-53
Sample code to retrieve a security policy	9-54
Sample code to apply a security policy to a document (1)	9-55
Sample code to apply a security policy to a document (2)	9-56
Sample code to apply a security policy to a document (3)	9-57
Demonstrations	9-58
Activities	9-59
Lesson 9.4. Retrieve security users and groups	9-61
Lesson: Retrieve security users and groups	9-62
Activities that you need to complete	9-63
What is a realm?	9-64
User and Group objects	9-65
User and group names	9-66
FileNet User and Group Interfaces	9-67
Steps to get user names from the directory service	9-68
Factory.EntireNetwork.fetchInstance(...)	9-69
entireNetwork.get_MyRealm()	9-70
realm.findUsers(...)	9-71
Sample code to retrieve the realm	9-73
Sample code to retrieve user names	9-74
Getting group names from directory service	9-75
realm.findGroups(...)	9-76
Sample code to retrieve group names	9-78
Demonstrations	9-79
Activities	9-80

Unit 10. Events and Subscriptions	10-1
Unit lesson	10-2
Lesson 10.1. Implement a Java event handler	10-3
Lesson: Implement a Java event handler	10-4
Activities that you need to complete	10-5
Event-action architecture	10-6
Subscriptions	10-8
Subscriptions for classes or instances	10-9
Steps to create an event action	10-10
Applying a filter to an event subscription	10-11
Define subscription filter	10-12
Synchronous and Asynchronous mode	10-13
Event handler requirements	10-14
Event handler coding guidelines	10-15
Code modules	10-17
Update event action with a new code module version	10-18
Adding external JAR files to the Content Engine	10-19
eventActionHandler.onEvent(...)	10-20
ObjectChangeEvent	10-21
objectChangeEvent.get_SourceObject()	10-22
Sample code to implement an event action handler	10-23

EngineRuntimeException	10-24
Error logging and debugging	10-25
Sample messages in P8_server_error.log file	10-27
Demonstrations	10-28
Activities	10-29
Unit 11. Auditing	11-1
Unit lessons	11-2
Lesson 11.1. Retrieve the audit history for existing objects	11-3
Lesson: Retrieve the audit history for existing objects	11-4
Activities that you need to complete	11-5
Auditing concepts	11-6
Enable and configure auditing	11-8
Configure auditing for a specific property	11-10
Audit event logs	11-12
Event object	11-14
No title	11-15
Steps to retrieve audited events	11-17
containable.get_AuditedEvents()	11-18
event.get_InitiatingUser()	11-19
engineObject.getClassName()	11-20
event.get_DateCreated()	11-21
event.get_EventStatus()	11-22
updateEvent.get_ModifiedProperties()	11-23
updateEvent.get_OriginalObject()	11-24
updateEvent.get_SourceObject()	11-25
Sample code to retrieve audited events from an object	11-26
Sample code to retrieve the modified properties	11-27
Sample code to retrieve the each property value (1)	11-28
Sample code to retrieve each property value (2)	11-29
Sample code to retrieve audited events for document versions	11-30
Sample code to audit the changes to a specific property	11-31
Demonstrations	11-32
Activities	11-34
Lesson 11.2. Retrieve the audit history for deleted objects	11-35
Lesson: Retrieve the audit history for deleted objects	11-36
Activities that you need to complete	11-37
Using SQL queries to retrieve audit data	11-38
Steps to retrieve audit events for deleted objects	11-39
Sample code to retrieve the delete events (1)	11-40
Sample code to retrieve the delete events (2)	11-41
Demonstrations	11-42
Activities	11-43
Lesson 11.3. Work with custom events	11-45
Lesson: Work with custom events	11-46
Activities that you need to complete	11-47
Raising and auditing custom events	11-48
Steps to raise a custom event on a subscribable object	11-49

Factory.CustomEvent.CreateInstance(...)	11-50
customEvent.set_EventStatus(...)	11-51
subscribable.raiseEvent(...)	11-52
Sample code to raise custom event	11-53
Demonstrations	11-54
Activities	11-55
Unit 12. Batches	12-1
Unit lessons	12-2
Lesson 12.1. Batch update	12-3
Lesson: Batch update	12-4
Activities that you need to complete	12-5
Batch definition	12-6
Batch class and BatchItemHandle interface	12-7
Batch updates	12-8
Steps to execute a batch	12-9
objectStore.get_Domain	12-10
updatingBatch.createUpdatingBatchInstance(...)	12-11
UpdatingBatch.add(...)	12-12
updatingBatch.updateBatch()	12-13
Sample code for batch update (1)	12-14
Sample code for batch update (2)	12-15
Sample code for batch update (3)	12-16
Demonstrations	12-17
Activities	12-18
Lesson 12.2. Batch retrieval	12-19
Lesson: Batch retrieval	12-20
Activities that you need to complete	12-21
Batch retrievals	12-22
Steps to execute a batch	12-23
Steps to retrieve batch item handles and objects	12-24
retrievingBatch.createRetrievingBatchInstance(...)	12-25
retrievingBatch.add(...)	12-26
retrievingBatch.retrieveBatch()	12-27
batch.getBatchItemHandles(...)	12-28
batchItemHandle.hasException()	12-29
batchItemHandle.getException()	12-30
Sample code for batch retrieval (1)	12-31
Sample code for batch retrieval (2)	12-32
Sample code for batch retrieval (3)	12-33
Demonstrations	12-34
Activities	12-35
Unit 13. Annotations	13-1
Unit lessons	13-2
Lesson 13.1. Create Annotation objects	13-3
Lesson: Create Annotation objects	13-4
Activities that you need to complete	13-5

Image Viewer	13-6
Demonstrations	13-7
Working with annotations in Image Viewer	13-8
Image Viewer with annotations	13-9
Annotation objects	13-10
Annotated objects and annotations	13-11
Annotated objects and annotations	13-12
Steps to create an Annotation object	13-13
Factory.Annotation.createInstance(...)	13-14
annotation.set_ContentElements(...)	13-15
annotation.set_AnnotatedObject(...)	13-16
Sample code to create an Annotation object	13-17
Demonstrations	13-18
Activities	13-19
Lesson 13.2. Retrieve annotations	13-21
Lesson: Retrieve annotations	13-22
Activities that you need to complete	13-23
Steps to retrieve annotations	13-24
document.get_Annotations()	13-25
annotation.get_Name()	13-26
annotation.get_Creator()	13-27
annotation.get_DateCreated()	13-28
Sample code to retrieve annotations	13-29
Demonstrations	13-30
Activities	13-31
Lesson 13.3. Copy annotations	13-33
Lesson: Optional: Copy annotations	13-34
Activities that you need to complete	13-35
Steps to copy annotations	13-36
Sample code to copy annotations (1)	13-37
Sample code to copy annotations (2)	13-38
Sample code to copy annotations (3)	13-39
Demonstrations	13-40
Activities	13-41
Unit 14. Document Lifecycle	14-1
Unit lesson	14-2
Lesson 14.1. Change a document lifecycle state	14-3
Lesson: Change a document lifecycle state	14-4
Activities that you need to complete	14-5
Document lifecycles	14-6
Sample document lifecycle state	14-7
Document lifecycle policy and lifecycle action	14-8
Setup process for lifecycle policy and action	14-9
Steps to change a document lifecycle state	14-10
Promote or demote the document	14-11
document.get_DocumentLifecyclePolicy()	14-12
document.getProperties()	14-13

properties.getStringValue(...)	14-14
documentLifecyclePolicy.get_DocumentStates()	14-15
documentState.get_StateName()	14-16
documentState.get_CanBeDemoted()	14-17
document.changeState(...)	14-18
Sample code to get document lifecycle states	14-20
Sample code to get current lifecycle state	14-21
Sample code to promote or demote a document	14-22
Demonstrations	14-23
Activities	14-24
Lesson 14.2. Create lifecycle actions	14-25
Lesson: Create lifecycle actions	14-26
Activities that you need to complete	14-27
DocumentLifecycleActionHandler interface	14-28
Steps to create a lifecycle action	14-29
documentLifecycleActionHandler.onDocumentPromote(...)	14-30
documentLifecycleActionHandler.onDocumentDemote(...)	14-31
document.get_CurrentState()	14-32
EngineRuntimeException raised by the lifecycle handler	14-33
Error logging and debugging	14-34
Code modules	14-35
Update lifecycle action	14-36
Adding external JAR files to the Content Engine	14-37
Sample code to use lifecycle action handler (1)	14-38
Sample code to use lifecycle action handler (2)	14-39
Sample code to add custom methods	14-40
Demonstrations	14-41
Activities	14-42
Unit 15. Publishing	15-1
Unit lessons	15-2
Lesson 15.1. Publish a document	15-3
Lesson: Publish a document	15-4
Activities that you need to complete	15-5
Publishing overview	15-6
Publishing components	15-7
Publishing actions	15-9
Publish templates	15-10
MIME types and publish templates	15-12
Steps to publish a document	15-13
Publish options XML	15-14
Creating the publish options XML string	15-15
instantiatingScope.fetchObject(...)	15-16
document.publish(...)	15-17
publishRequest.get_PublishingStatus()	15-18
Sample code to get the document for publishing	15-19
Sample code to get the publish template	15-20
Sample code to publish a document	15-21

Demonstrations	15-22
Activities	15-23
Lesson 15.2. Republish a document.	15-25
Lesson: Republish a document	15-26
Activities that you need to complete	15-27
Steps to republish a document	15-28
Creating the publish options XML string	15-29
Document.get_DestinationDocuments(...)	15-30
document.republish(...)	15-31
Sample code to get the publication document	15-32
Sample code to republish a document	15-33
Sample code to get the republishing status	15-34
Sample SQL statement to retrieve publish templates	15-35
Sample code to retrieve publish templates	15-36
Demonstrations	15-37
Activities	15-38
Unit 16. Compound Documents	16-1
Unit lessons	16-2
Lesson 16.1. Create compound documents	16-3
Lesson: Create compound documents	16-4
Activities that you need to complete	16-5
Compound documents	16-6
Components of compound documents	16-7
Features of compound documents	16-9
Managing and viewing compound documents	16-11
Compound document - chapter book	16-12
Compound document - insurance policy	16-13
Building a compound document	16-14
ComponentRelationship class	16-15
Document class	16-16
Steps to create a compound document	16-17
document.set_CompoundDocumentState(...)	16-18
Factory.ComponentRelationship.createInstance(...)	16-19
componentRelationship.set_ChildComponent(...)	16-20
Setting child version relationships	16-21
componentRelationship.set_VersionBindType(...)	16-22
componentRelationship.save(...)	16-23
Sample code to set compound document state	16-24
Sample code to retrieve child documents	16-25
Sample code to set up static component relationship	16-26
Sample code to set up dynamic component relationship	16-27
Demonstrations	16-28
Activities	16-29
Lesson 16.2. Retrieve compound documents.	16-31
Lesson: Retrieve compound documents	16-32
Activities that you need to complete	16-33
Steps to retrieve component relationships	16-34

Steps to retrieve the child documents	16-35
document.get_CompoundDocumentState()	16-36
compoundDocumentState.getValue()	16-37
document.get_ChildRelationships()	16-38
componentRelationship.get_Name()	16-39
componentRelationship.get_VersionBindType()	16-40
document.get_ChildDocuments()	16-41
Sample code to retrieve the parent document	16-42
Sample code to retrieve component relationships (1)	16-43
Sample code to retrieve component relationships (2)	16-44
Sample code to retrieve child documents	16-45
Demonstrations	16-46
Activities	16-47

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® and the IBM logo are registered trademarks of International Business Machines Corporation.

The following are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide:

DB2®

FileNet®

Tivoli®

WebSphere®

Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux® is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other product and service names might be trademarks of IBM or other companies.

Course description

IBM FileNet Content Manager 5.0: Java API Programming

Duration: 5 days

Overview

This course is designed to help you acquire the knowledge and skills necessary to create custom applications for an IBM FileNet P8 Content Manager system using the Java API.

You work with a fully functioning IBM FileNet P8 system to practice the skills required to write code and test content management functions.

Audience

This course is for programmers responsible for the design and development of custom Content Manager applications. This course is also for anyone who needs to know the capabilities of the Content Engine API for Java-based applications.

Prerequisites

- Basic programming experience in Java-based technologies
- Knowledge in content management concepts
- IBM FileNet P8 5.0 Prerequisite Skills using Workplace (F140)
- Knowledge in IBM FileNet P8 Platform Administration is recommended

Skills taught

How to develop applications using IBM FileNet P8 Content Engine Java API libraries.

Contents

- Overview of Content Engine Java API
- Communicating with Content Engine
- Content Engine Java API for working with the following:
 - Folders
 - Custom Objects

- Documents
- Properties
- Searches
- Document Versioning
- Security
- Events and Subscriptions
- Auditing
- Batches
- Annotations
- Document Lifecycle
- Publishing
- Compound Documents

Unit 1. Content Engine Java API Overview

What this unit is about

This unit introduces the Content Engine Java API and the Content Engine objects.

What you should be able to do

After completing this unit, you should be able to:

- Locate the Content Engine developer help documents.
- Locate the Java API Reference that contains Content Engine Java API classes.
- Set up Eclipse.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Content Engine Java API Overview

Unit lessons

- 
- Content Engine Java API concepts
 - Content Engine objects

© Copyright IBM Corporation 2011

Figure 1-1. Unit lessons

F1432.0

Notes:

Lesson 1.1. Content Engine Java API concepts

Lesson: Content Engine Java API concepts

- 
- Why is this lesson important to you?
 - Your company has purchased a FileNet P8 system and plans to manage their content using the FileNet P8 Content Engine. As their programmer, you want an orientation to the FileNet P8 Content Engine Java API, which you are going to use to write a custom application for the company.

© Copyright IBM Corporation 2011

Figure 1-2. Lesson: Content Engine Java API concepts

F1432.0

Notes:

Content Engine Java API concepts

Activities that you need to complete



- Locate the Content Engine developer help documents.
- Locate the Java API Reference that contains Content Engine Java API classes.

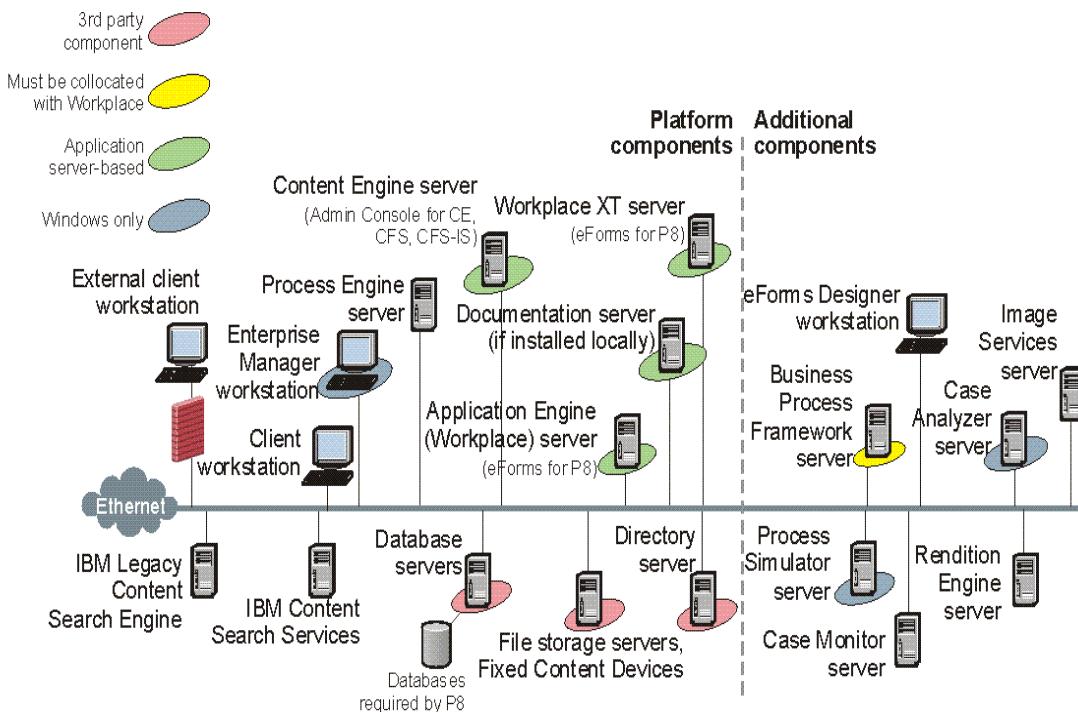
© Copyright IBM Corporation 2011

Figure 1-3. Activities that you need to complete

F1432.0

Notes:

Content Engine Java API concepts

FileNet P8 architecture

© Copyright IBM Corporation 2011

Figure 1-4. FileNet P8 architecture

F1432.0

Notes:**Help path**

- IBM FileNet P8 Version 5.0 Information Center > Planning and preparing for IBM FileNet P8 > Planning and preparing for IBM FileNet P8 installation > Planning the installation

This high-level architecture diagram shows where the various engines fit within a FileNet P8 system.

Content Engine Java API concepts

Content Engine and Java EE



- IBM FileNet P8 Content Engine (CE)
 - Built on the Java EE platform
 - Implemented within a Java EE standards-compliant application server
 - Supports a wide variety of
 - Operating systems
 - Directory service providers
 - Application servers
 - Database servers

© Copyright IBM Corporation 2011

Figure 1-5. Content Engine and Java EE

F1432.0

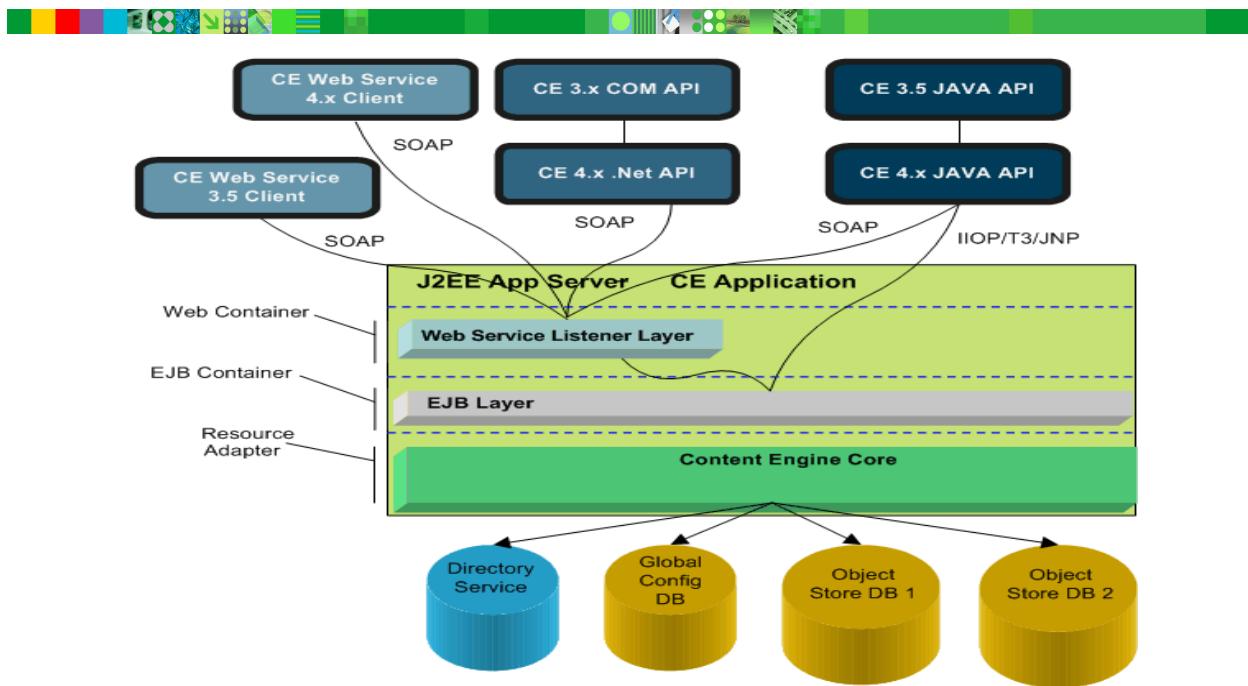
Notes:

The Content Engine can support a wide variety of environments (examples follow) because it is implemented within a Java EE standards-compliant application server environment:
Operating systems examples: Microsoft, Sun, IBM, Hewlett Packard, and Red Hat

- Directory service providers examples: Microsoft Active Directory, Novell eDirectory, IBM Tivoli Directory Server, Sun JavaSystem Directory Server
- Java EE compliant application servers examples: BEA, IBM, and JBoss
- Database servers examples: Microsoft SQL Server, Oracle Database, and IBM DB2

Content Engine Java API concepts

Content Engine-supported transport protocols



- The Content Engine supports two transport protocols:
 - Enterprise JavaBeans (EJB). This course covers APIs that work on EJB.
 - Content Engine Web Service (CEWS)

© Copyright IBM Corporation 2011

Figure 1-6. Content Engine-supported transport protocols

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Transport Protocols

The Content Engine supports two basic transport protocols:

- Enterprise JavaBeans (EJB). This course covers APIs that work on EJB.
- Content Engine Web Service (CEWS)

The diagram shows the interaction of these protocols with the Content Engine.

Content Engine Java API concepts

Content Engine Java APIs



- Content Engine Java APIs
 - Provide classes for creating, accessing, and manipulating content and objects
 - Are platform neutral
 - Are compatible with Enterprise JavaBeans (EJB) 2.0
- The Content Engine Java APIs can be installed
 - During Content Engine installation
 - Anywhere, including on a non-Content Engine system

© Copyright IBM Corporation 2011

Figure 1-7. Content Engine Java APIs

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Developer Roadmap > Development and Integration Tools > Content Engine Java API

Content Engine Java API concepts

Application types supported by Content Engine APIs



- Runtime (user) applications
 - Interact with object stores
 - Work with objects such as documents and folders
- Administration applications
 - Perform configuration tasks
 - Examples:
 - Add-ons - Product extensions to the core FileNet P8 Content Engine components
 - Defining audit-event parameters
- Metadata authoring applications
 - Create class definitions, properties, and so on

Note: This course includes mainly runtime application APIs.

© Copyright IBM Corporation 2011

Figure 1-8. Application types supported by Content Engine APIs

F1432.0

Notes:

Content Engine Java API concepts

Required Java Archive (JAR) files



For a Content Engine (CE) Java API EJB Transport Client	
JAR File Name	Description
Jace.jar	The core FileNet JAR file for the CE Java API; includes the common JAR and transport JAR
log4j.jar	The LOG4J framework for Content Engine Java API logging
An application server-specific JAR file	If already in an application server, this JAR file is not required, except in a stand-alone JVM

© Copyright IBM Corporation 2011

Figure 1-9. Required Java Archive (JAR) files

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Developer Requirements

You can get the required files for a remote system, by running Content Engine Client Installer, which is included with the Content Engine. To install the Content Engine client files that you need for application development, select the "Other Applications" option from the installer's "Select FileNet P8 Applications" panel.

By default, the JAR files are located in the lib subdirectory. JAAS configuration files are in the /config/samples subdirectory.

For the unit activities, you set up the class path for these JAR files as follows:

- In a batch file if you are using Command Prompt.
- In the console of the IDE if you are using Eclipse.

Content Engine Java API concepts

Commonly used Content Engine Java packages

Package Name	Description
com.filenet.api.core	Classes and interfaces related to the core objects of the API
com.filenet.api.collection	Type-safe interfaces for collections of objects
com.filenet.api.constants	Classes defining collections of related, type-safe constant values
com.filenet.api.query	Classes for performing CE searches
com.filenet.api.publishing	Interfaces related to publishing
com.filenet.api.security	Interfaces related to authentication, authorization, and user- and group-specific data
com.filenet.api.action	Classes for making changes to objects.
com.filenet.api.exception	Exception-reporting framework for the CE
com.filenet.api.util	Utility classes and interfaces

Note: This list is not complete.

© Copyright IBM Corporation 2011

Figure 1-10. Commonly used Content Engine Java packages

F1432.0

Notes:

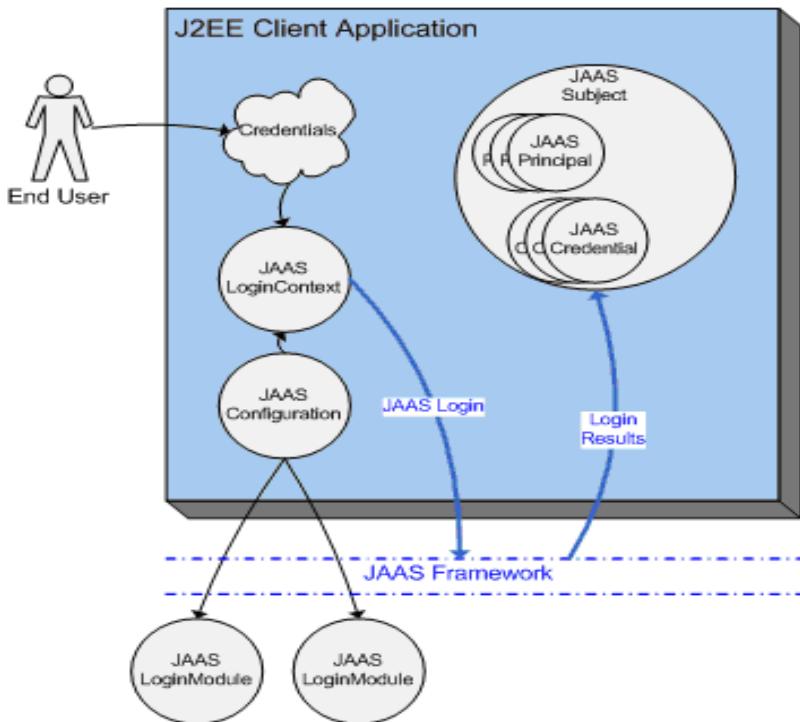
Help path

For the complete list of Java Packages, refer to the following:

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Java Packages and .NET Namespaces

Note: These Java packages are contained in the jace.jar file.

Content Engine Java API concepts

Java Authentication and Authorization Service (JAAS)

JAAS –
Provides a framework for securely determining who is invoking a Java application.

© Copyright IBM Corporation 2011

Figure 1-11. Java Authentication and Authorization Service (JAAS)

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Security > IBM FileNet P8 security > Authentication

JAAS configurations

To use a Java EE-based application, a client must first perform a JAAS login by specifying a JAAS configuration (in a file).

The JAAS configuration specifies the authentication technologies (LoginModules) that are used to verify the client credentials.

A JAAS configuration file lists one entry for each configured application. Within an application entry is a list of LoginModules for that application. Based on the contents of the configuration file, the JAAS framework dynamically determines which set of authentication technologies to invoke when a client application attempts to authenticate.

JAAS LoginContexts

A Java EE client application must specify a JAAS configuration and a mechanism through which credentials can be obtained from a user at run time. These two items constitute a JAAS LoginContext. Java EE client applications use the LoginContext to interact with JAAS and to authenticate themselves to a Java EE server.

JAAS LoginModules

JAAS-compliant LoginModules are implemented by authentication technology providers. Java EE application server vendors, such as BEA, IBM, and JBOSS, typically provide LoginModules for standard types of credentials, such as user name and password.

JAAS Subjects

When a JAAS log-in has successfully completed, a JAAS Subject is returned to the caller. The Subject is a key Java EE class, which is transparently sent between Java EE clients and Java EE servers any time an EJB is invoked.

A JAAS Subject contains the set of JAAS principals (authenticated identities) and JAAS credentials (authentication data such as cryptographic keys) that result from the log-in process. Each LoginModule that successfully authenticates the user updates the Subject with information about the user. A Java EE server application can examine the principals in the Subject to establish the caller identity.

Content Engine Java API concepts

JAAS configuration files

- Must establish a JAAS context prior to any requests to the Content Engine server
- The FileNet P8 Content Engine installation provides JAAS configuration sample files
 - Examples:
 - jaas.conf.WebSphere
 - jaas.conf.WebLogic
 - Legacy support
 - Content Engine API does an internal JAAS log-in
 - The client environment must include a proper JAAS configuration

© Copyright IBM Corporation 2011

Figure 1-12. JAAS configuration files

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Security > IBM FileNet P8 security > Authentication

The client application obtains a JAAS Subject prior to calling the Content Engine Java API.

Interactions with the Content Engine Java API cause the Content Engine EJB to be invoked.

The client JAAS subject is transparently sent to the Java EE application server with each EJB call.

The application server validates the JAAS subject and confirms the caller identity before executing any code in the Content Engine EJB.

Content Engine Java API concepts

Resources



- Developer Help for Content Engine Java API
 - IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide
- Java API reference (Javadocs)
 - IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java API Reference

© Copyright IBM Corporation 2011

Figure 1-13. Resources

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Developer Roadmap > Development and Integration Tools > Samples

Content Engine Java API concepts

Activities

Explore the Java API documentation.

- Use the help paths listed in the notes to do the following:
 - Locate the Content Engine developer help documents.
 - Locate the Java API Reference that contains Content Engine Java API classes.

© Copyright IBM Corporation 2011

Figure 1-14. Activities

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide
- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Java API Reference

Lesson 1.2. Content Engine objects

Lesson: Content Engine objects



- Why is this lesson important to you?
 - Your company has purchased a FileNet P8 system and plans to manage their content using the FileNet P8 Content Engine. As their programmer, you want an orientation to the FileNet P8 Content Engine objects, which you are going to use to write a custom application for the company.

© Copyright IBM Corporation 2011

Figure 1-15. Lesson: Content Engine objects

F1432.0

Notes:

Content Engine objects

Activities that you need to complete



- Set up the Eclipse environment.

© Copyright IBM Corporation 2011

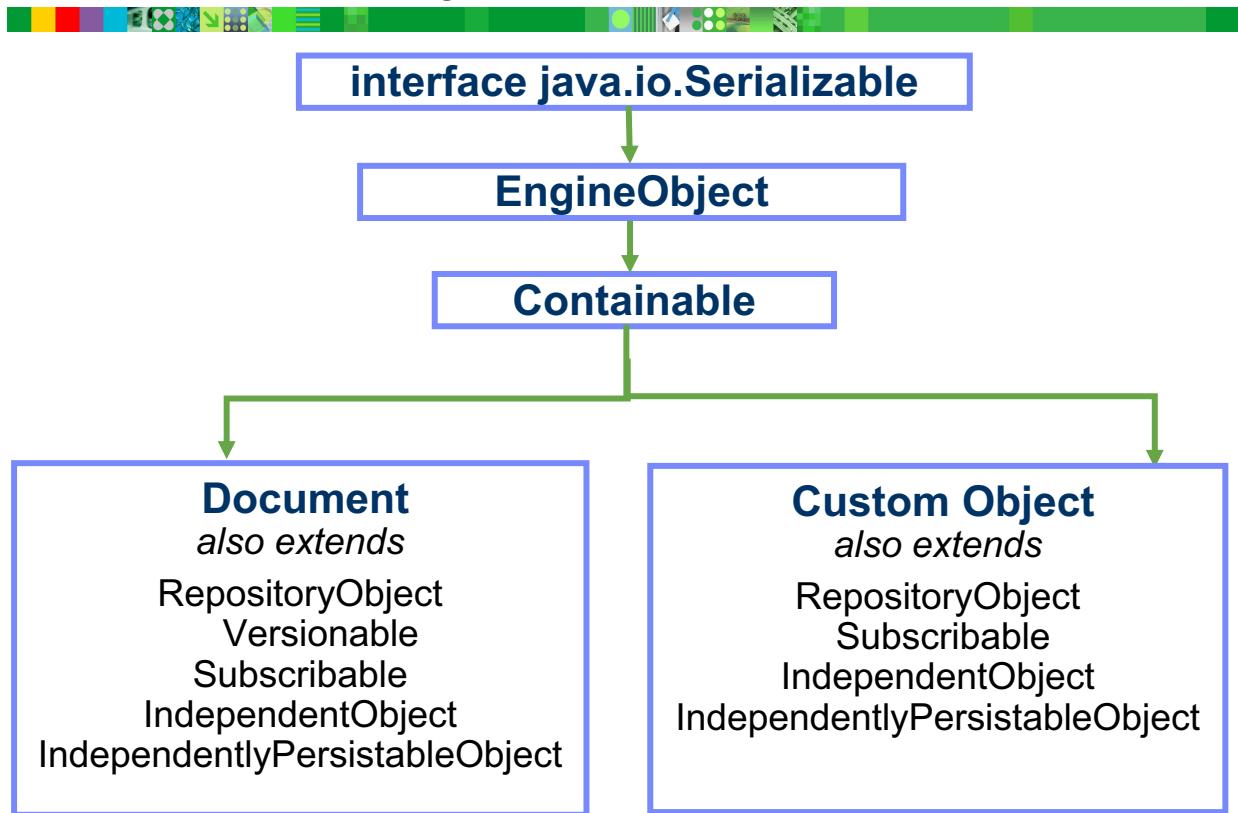
Figure 1-16. Activities that you need to complete

F1432.0

Notes:

Content Engine objects

Interface hierarchy



© Copyright IBM Corporation 2011

Figure 1-17. Interface hierarchy

F1432.0

Notes:

Help path

For more details on the interface or class hierarchy, refer to the following:

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java API Reference > Tree

All the interfaces mentioned in the diagram are part of the `com.filenet.api.*` packages.

Notice that **Document** and **CustomObject** interfaces are similar in their hierarchy, except that the **Document** interface also extends the **Versionable** interface to manage the content. **CustomObject** does not have content.

Content Engine objects

EngineObject and RepositoryObject

- **EngineObject**

- Represents the top-level interface from which most of the other Content Engine API interfaces are derived
- Is any object known to the Content Engine including objects *outside* of repositories
- Examples: Domain, MarkingSet, ObjectStore, and User objects

- **RepositoryObject**

- Is any object known to the Content Engine that resides within an object store
- Example: A IDocument object

© Copyright IBM Corporation 2011

Figure 1-18. EngineObject and RepositoryObject

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces

EngineObject

Normally, an instance of one of the EngineObject subclasses is used.

EngineObject provides base functionalities such as retrieving the following:

- The Connection object that is used for communication to the Content Engine server
- The class information and properties of an object

RepositoryObject

Normally, an instance of one of the RepositoryObject subclasses is used.

Content Engine objects

IndependentObject and DependentObject



- EngineObject includes independent objects and dependent objects
- IndependentObject
 - Has its own identity and always has an ObjectReference.
 - Examples: IDocument, IFolder, and ICustomObject.
- DependentObject
 - Exists within the scope of another object.
 - Is saved to the server when attached as a child of an IndependentObject.
 - Is fetched using a property of parent object.
 - Has a corresponding list-type collection.
 - Examples: IContentElementList, IPermissionList, and IChoiceList.
 - Has identity and security that are derived from its parent object.

© Copyright IBM Corporation 2011

Figure 1-19. IndependentObject and DependentObject

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces

Content Engine objects

Base classes and interfaces



- IndependentlyPersistableObject
 - Is a persistable IndependentObject that you can create, update, and delete directly.
- Subscribable Object
 - Serves as the target of an event subscription, represented by a subscription-based object.
- Versionable Object
 - Represents the base class for classes whose instances are versionable objects.
- Containable Object
 - Represents the base class for all objects that can be contained.

© Copyright IBM Corporation 2011

Figure 1-20. Base classes and interfaces

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces

Versionable Object

- Can have multiple associated versions in an object store.
- Can be checked out, optionally edited, and checked in.

Containable Object

- All Containable subclasses are referentially contained except the Folder object.
- Folder objects and their subclasses can be directly contained in an Folder object.

Content Engine objects

Core objects of the API



Object	Description
Connection	Holds the information required for an application to connect to the FileNet P8 domain resources
ObjectStore	An independent, domain-scoped object that provides access to resources
Folder	A container that holds other containable objects
Document	A single version of a document in a object store
CustomObject	A general object that has no content
Annotation	Information that can be attached to an object

© Copyright IBM Corporation 2011

Figure 1-21. Core objects of the API

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces

Content Engine objects

Instantiating objects – Factory class

- Content Engine Java APIs are defined as interfaces.
 - Cannot instantiate new objects using the "new" construct.
 - Must use a Factory class method to instantiate type-appropriate objects.
- Factory class has static inner classes.
 - Names parallel the standard Content Engine API types.
 - Example: Factory.Document
- Factory methods are convenient and provide type safety.
 - No need to cast the Factory method results to a desired type.
 - Identify programming errors on the returned type at compile time instead of at run time.
 - The following methods are available for instantiating objects:
 - CreateInstance
 - FetchInstance
 - GetInstance

© Copyright IBM Corporation 2011

Figure 1-22. Instantiating objects - Factory class

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Instantiating Objects

Content Engine objects

Collection objects



- Content Engine collections are
 - Groups of related elements
 - Strongly typed (contain single type objects)
 - Two types: List and Set
- List-type collection
 - Is a group of dependent objects or a list of primitive data items.
 - Has a parent object to which it is scoped.
 - Can be updated directly using type-safe methods.
 - Elements of a list are ordered and need not be unique.
 - Examples: IContentElementList, ISecurityTemplateList
- Set-type collection
 - Is a group of independent objects.
 - Cannot be updated directly.
 - Elements of a set are unordered and unique (some exceptions).
 - Examples: IObjectStoreSet, IDocumentSet, IFolderSet

© Copyright IBM Corporation 2011

Figure 1-23. Collection objects

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Collections > Concepts

Content Engine objects

Actions in pending state

- New object exists locally only.
 - Create an object instance and add properties locally
 - Check in and file into a folder locally.
 - Save object to persist object and properties.
- Save object to persist property updates.
 - Until saved, actions are pending.
- Deletion operations do not persist until saved.
- Call
`IndependentlyPersistableObject.GetPendingActions`
to review pending actions.

© Copyright IBM Corporation 2011

Figure 1-24. Actions in pending state

F1432.0

Notes:

Creation, storage, and retrieval of objects and their properties are transactional.

You create an object instance locally, add properties to it locally, check it in locally, file it into a folder locally, and then save it.

These steps do not result in update or storage at the server until the `save()` method is called.

Content Engine objects

Working with properties



- Added and updated object properties remain local until object is saved.
- When retrieved, persisted object must specify properties to return.
 - Properties are retrieved into local cache.
 - Access properties from local cache.
- Use PropertyFilter and optional FilterElement classes to specify properties for retrieval.

© Copyright IBM Corporation 2011

Figure 1-25. Working with properties

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Properties > Concepts

More information on properties is provided in the "Properties" unit.

Content Engine objects

Setting up a thick client development environment



- EJB transport protocol with WebSphere:
 - Install the IBM WebSphere Application Client for WebSphere Application Server on your development client workstation.
 - Set up the environment to use Java Runtime Environment (JRE)
 - Make sure to set the JAVA_HOME environment variable and JRE bin in the path.
 - Update the sas.client.props (Comes with IBM WebSphere).
 - Add the Java Virtual Machine (JVM) argument.
 - Add the appropriate JAR files to your classpath
 - The WebSphere library and plugins directories.
 - Pass the correct URI to the connection.
 - Example: "iiop://hostname:port/FileNet/Engine"

© Copyright IBM Corporation 2011

Figure 1-26. Setting up a thick client development environment

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 Information Center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Developer Requirements > Setting Up a Thick Client Development Environment

This page and the procedures in the Student Exercises book refer to the setup for EJB transport protocol with WebSphere.

Refer to the IBM FileNet P8 Version 5.0 Information Center for WebLogic and JBoss setup.

Your Student Exercises book contains detailed setup information to write and run the code for this course.

- The Application Client version must match the WebSphere Application Server version.
- Use the Java Runtime Environment (JRE) provided with the IBM WebSphere® Application Client.

Content Engine objects

Using Eclipse IDE for writing your code



- You can use Eclipse or Notepad and the Windows Command Prompt to write and execute your code.
- Your Student Exercises book include instructions for the following:
 - Setting up a project in Eclipse
 - Writing code using a text editor
 - Compiling and executing the code in Command Prompt

© Copyright IBM Corporation 2011

Figure 1-27. Using Eclipse IDE for writing your code

F1432.0

Notes:

Content Engine objects

Demonstrations



- Set up Eclipse:
 - Create an Eclipse project
 - Add library files
 - Configure Java complier
 - Create a package
 - Add supporting files
 - Edit the Properties file

© Copyright IBM Corporation 2011

Figure 1-28. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

Set up Eclipse

1. Start Eclipse by double-clicking the icon on the desktop.
2. Follow the instructions in the "Set Up Eclipse IDE" lesson in your Student Exercises book to complete each task.

Content Engine objects

Activities



In your Student Exercises

- Unit: Set Up Eclipse IDE
- Lesson: Set Up Eclipse IDE
- Activities
 - Set up the Eclipse environment.

© Copyright IBM Corporation 2011

Figure 1-29. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 2. Communication with the Content Engine

What this unit is about

This unit describes how to get a connection to a FileNet P8 domain and how to retrieve the domain and the object stores.

What you should be able to do

After completing this unit, you should be able to:

- Get a connection to a FileNet P8 domain.
- Retrieve the domain and the object stores.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Communication with the Content Engine

Unit lessons

- 
- Get a connection to a FileNet P8 domain
 - Retrieve the domain and the object stores

© Copyright IBM Corporation 2011

Figure 2-1. Unit lessons

F1432.0

Notes:

Lesson 2.1. Get a connection to a FileNet P8 domain

Lesson:

Get a connection to a FileNet P8 domain

- 
- Why is this lesson important to you?
 - Your company manages its content using the IBM FileNet Content Manager. As their programmer, you are going to write code to communicate with the Content Engine server by getting a connection to a FileNet P8 domain.

© Copyright IBM Corporation 2011

Figure 2-2. Lesson: Get a connection to a FileNet P8 domain

F1432.0

Notes:

Get a connection to a FileNet P8 domain

Activities that you need to complete



- Edit the Property file.
- Get a connection to a FileNet P8 domain.

© Copyright IBM Corporation 2011

Figure 2-3. Activities that you need to complete

F1432.0

Notes:

Get a connection to a FileNet P8 domain

Connection object



- The Connection object
 - Represents a logical connection to a FileNet P8 domain
 - Contains information for configuration options:
 - Needed to execute an operation on the CE server
 - A URI includes the transport protocol (connection type), host name, and port number
- All engine objects maintain a reference to the connection instance.
 - You retrieve it by calling the GetConnection method of the EngineObject.

© Copyright IBM Corporation 2011

Figure 2-4. Connection object

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts>FileNet P8 Domain

An incorrect configuration is the most frequent cause of an exception related to a connection failure.

Example:

If the URI carried by the Connection object specifies the EJB transport protocol but an EJB.JAR file is not in the class path, the API throws an exception.

Get a connection to a FileNet P8 domain

Steps to get a connection to a FileNet P8 domain

1. Create a new Connection object.

```
Factory.Connection.getConnection(...)
```

2. Authenticate with the application server.

```
UserContext.createSubject(...)
```

3. Get UserContext object associated with the current thread.

```
UserContext.get()
```

4. Make the specified JAAS Subject active.

```
UserContext.pushSubject(...)
```

© Copyright IBM Corporation 2011

Figure 2-5. Steps to get a connection to a FileNet P8 domain

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Procedures > Getting a Connection

Get a connection to a FileNet P8 domain

Factory.Connection.getConnection(...)

```
public static Connection getConnection(String uri)
```

- Parameters

- uri - Specify the Universal Resource Identity (URI) for this connection.
 - WebLogic - t3://servername:port/FileNet/Engine
 - WebSphere - iiop://servername:port/FileNet/Engine

- Returns

- A Connection object

- Notes:

- A domain connection is required to obtain a reference to any object stored within that domain.
 - This method uses default configuration parameters.

© Copyright IBM Corporation 2011

Figure 2-6. Factory.Connection.getConnection(...)

F1432.0

Notes:

- A domain connection is required to obtain a reference to any object stored within that domain.
- This method uses default connection parameters.
- There is another variation of this method (takes additional parameter for configuration):

public static Connection getConnection(String uri, ConfigurationParameters parameters)

Configuration parameters represent a group of parameters that define the behavior of the client API.

The method discussed on this page is equivalent to calling the following method:

getConnection(uri, new ConfigurationParameters())

Get a connection to a FileNet P8 domain

UserContext.createSubject(...)

```
public static javax.security.auth.Subject  
    createSubject(Connection conn,  
        String user, String password,  
        String optionalJAASStanzaName)
```

- Parameters
 - conn - Specify the Connection object for this thread; cannot be null.
 - user - Specify the username to be authenticated.
 - password - Specify the user password.
 - optionalJAASStanzaName - Specify the JAAS configuration stanza name.
 - If it is null, the stanza name defaults to FileNetP8.
- Returns
 - JAAS Subject object
- Notes
 - The client must be properly configured for JAAS.

© Copyright IBM Corporation 2011

Figure 2-7. UserContext.createSubject(...)

F1432.0

Notes:

- This method uses standard JAAS calls to authenticate with the application server and return a JAAS Subject object.
- The calling code can optionally provide a JAAS configuration stanza name.
- The returned Subject has no association to the current user context. The caller must use the Subject with the UserContext.

Get a connection to a FileNet P8 domain

UserContext.get()



```
public static UserContext get()
```

- Returns
 - A UserContext object associated with the current thread
- Notes
 - When no user context has been set, a default user context exists containing the current locale default from the JVM.

© Copyright IBM Corporation 2011

Figure 2-8. UserContext.get()

F1432.0

Notes:

Get a connection to a FileNet P8 domain

userContext.pushSubject(...)



```
public void pushSubject  
    (javax.security.auth.Subject sub)
```

- Parameters

- sub - The JAAS Subject to be associated with the thread
 - Cannot be null

- Returns

- This method saves any previously active Subject and makes the specified Subject active.

© Copyright IBM Corporation 2011

Figure 2-9. userContext.pushSubject(...)

F1432.0

Notes:

To restore the previous Subject, call the *popSubject()* method.

Get a connection to a FileNet P8 domain

Sample code to get a connection to a FileNet P8 domain

```
String uri =
    "iiop://ccv01135:2809/FileNet/Engine";
String username = "p8admin";
String password = "IBMFleNetP8";
Connection conn =
    Factory.Connection.getConnection(uri);
Subject subject = UserContext.createSubject
    (conn, username, password, null);
UserContext uc = UserContext.get();
uc.pushSubject(subject);
return conn;
```

© Copyright IBM Corporation 2011

Figure 2-10. Sample code to get a connection to a FileNet P8 domain

F1432.0

Notes:

Get a connection to a FileNet P8 domain

Demonstrations



- Get a Connection to a FileNet P8 Domain.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 2-11. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getConnectionEDU()* method in the *CEConnectionEDU.java* file.
3. Run the code and verify the results as instructed in the *Communication with the Content Engine* unit in the Student Exercises book.

Get a connection to a FileNet P8 domain

Activities



In your Student Exercises book

- Unit: Communication with the Content Engine
- Lesson: Get a connection to a FileNet P8 domain
- Activities
 - Edit the Property file.
 - Get a connection to a FileNet P8 domain.

© Copyright IBM Corporation 2011

Figure 2-12. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 2.2. Retrieve the domain and the object stores

Lesson:

Retrieve the domain and the object stores



- Why is this lesson important to you?
 - You need to access the Domain object to retrieve other objects in the domain. As a programmer for the Content Manager, you are going to write code to retrieve the Domain and the ObjectStore objects.

© Copyright IBM Corporation 2011

Figure 2-13. Lesson: Retrieve the domain and the object stores

F1432.0

Notes:

Retrieve the domain and the object stores

Activities that you need to complete



- Retrieve the domain and the object stores.

© Copyright IBM Corporation 2011

Figure 2-14. Activities that you need to complete

F1432.0

Notes:

Retrieve the domain and the object stores

Domain object



- The Domain object
 - Represents a collection of resources and services sharing the same Global Configuration Database (GCD).
 - Is associated with one or more security realms for authenticating users.
- The Java API
 - Can programmatically create a new Domain object but it is not typical.
 - Can retrieve a persisted Domain object.

© Copyright IBM Corporation 2011

Figure 2-15. Domain object

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > FileNet P8 Domain

Retrieve the domain and the object stores

Global Configuration Database and Domain objects



- Global Configuration Database (GCD) defines
 - A common set of attributes that control the functional characteristics of resources and services for the domain
- Domain resources
 - Sites and their virtual servers and server instances
 - Object store databases
 - Full-text index areas
 - File storage areas
 - Content cache areas
 - Add-ons
 - Marking sets

© Copyright IBM Corporation 2011

Figure 2-16. Global Configuration Database and Domain objects

F1432.0

Notes:

Retrieve the domain and the object stores

Steps to retrieve Domain and ObjectStore objects

- 
1. Get the Domain Object.

```
Factory.Domain.fetchInstance(...)
```

2. Get the ObjectStores.

```
domain.getObjectStores()
```

© Copyright IBM Corporation 2011

Figure 2-17. Steps to retrieve Domain and ObjectStore objects

F1432.0

Notes:

An object-valued Domain property can also be retrieved by calling the *ObjectStore.getDomain()* method.

Retrieve the domain and the object stores

Factory.Domain.fetchInstance(...)

```
public static Domain fetchInstance  
(Connection conn, String name, PropertyFilter  
filter)
```

- Parameters
 - conn – Specify the Connection object.
 - name – Specify the symbolic name of the class instance to retrieve.
 - filter – Specify a PropertyFilter object.
- Returns
 - A Domain object
- Notes:
 - This method always makes a round trip to the server.

© Copyright IBM Corporation 2011

Figure 2-18. Factory.Domain.fetchInstance(...)

F1432.0

Notes:

If the PropertyFilter value is null, this method returns values for all non-object properties and returns placeholders for all object-valued properties

Any subsequent attempts to access an object-valued property will cause an automatic round-trip to the server to fetch its value.

Retrieve the domain and the object stores

domain.get_ObjectStores()



```
public ObjectStoreSet get_ObjectStores()
```

- Returns
 - The collection of ObjectStore objects associated with a given FileNet P8 domain.

© Copyright IBM Corporation 2011

Figure 2-19. domain.get_ObjectStores()

F1432.0

Notes:

Retrieve the domain and the object stores

objectStore.getAccessAllowed()



```
Integer getAccessAllowed()
```

- Returns

- Returns a value representing a bit mask of access rights granted to the user requesting this object.

- Notes

- This method is used along with `AccessLevel.USE_OBJECT_STORE.getValue()` which specifies that the user or group is granted or denied permission to use an object store.

© Copyright IBM Corporation 2011

Figure 2-20. `objectStore.getAccessAllowed()`

F1432.0

Notes:

Retrieve the domain and the object stores

Sample code to retrieve the Domain object



```
String domainName = "P8Domain";
Domain domain=Factory.Domain.fetchInstance(
    connection, domainName, null);
System.out.println(domain.get_Name());
```

© Copyright IBM Corporation 2011

Figure 2-21. Sample code to retrieve the Domain object

F1432.0

Notes:

Retrieve the domain and the object stores

Sample code to retrieve the ObjectStore objects

```
ObjectStoreSet osSet =
    domain.get_ObjectStores();
ObjectStore store;
Iterator iterator = osSet.iterator();
Iterator<ObjectStore> osIter = iterator;
while (osIter.hasNext()) {
    store = (ObjectStore)osIter.next();
    if((store.getAccessAllowed().intValue() &
AccessLevel.USE_OBJECT_STORE.getValue())>
0)
    System.out.println(store.get_Name());
}
```

© Copyright IBM Corporation 2011

Figure 2-22. Sample code to retrieve the ObjectStore objects

F1432.0

Notes:

Retrieve the domain and the object stores

Demonstrations



- Retrieve the Domain.
- Retrieve the Object Stores.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 2-23. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getDomainEDU()* method in the *CEConnectionEDU.java* file.
3. Run the code and verify the results as instructed in the *Communication with the Content Engine* unit in the Student Exercises book.
4. Repeat steps 2 and 3 for the *getObjectStoresEDU(...)* method that gets the object stores for the given domain.

Retrieve the domain and the object stores

Activities



In Student Exercises book

- Unit: Communication with the Content Engine
- Lesson: Retrieve Domain and ObjectStore objects
- Activities
 - Retrieve the domain and the object stores.

© Copyright IBM Corporation 2011

Figure 2-24. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 3. Folders

What this unit is about

This unit describes how to create and delete folders and how to retrieve the objects contained in the folders.

What you should be able to do

After completing this unit, you should be able to:

- Create and delete Folder objects.
- Retrieve objects from a folder.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Folders

Unit lessons



- Create and delete Folder objects
- Retrieve objects from a folder

© Copyright IBM Corporation 2011

Figure 3-1. Unit lessons

F1432.0

Notes:

Lesson 3.1. Create and delete Folder objects

Lesson: Create and delete Folder objects

- 
- Why is this lesson important to you?
 - The accounting department of your company wants to create a folder for each customer to store their ordering information. As their programmer, you are going to add a folder feature that allows users to create and delete folders.

© Copyright IBM Corporation 2011

Figure 3-2. Lesson: Create and delete Folder objects

F1432.0

Notes:

Create and delete Folder objects

Activities that you need to complete



- Create and delete Folder objects.

© Copyright IBM Corporation 2011

Figure 3-3. Activities that you need to complete

F1432.0

Notes:

Create and delete Folder objects

What is a Folder object?



- A Folder object
 - Is a container that holds *containable* objects:
 - Child folders or subfolders
 - Documents, Custom objects, and their subclasses
 - Workflow definitions, Publish templates, others
 - Has a parent folder
 - Has zero or more annotations associated with it
 - Is independently securable
 - Can be searched for and retrieved using a query

© Copyright IBM Corporation 2011

Figure 3-4. What is a Folder object?

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Containment > Concepts > Folders

Create and delete Folder objects

Containment concepts



- Folders are directly contained.
 - They exist inside a folder.
 - They are deleted from the object store when they are removed from a folder.
- Document, custom, and other objects are referentially contained in folders.
 - They can be filed in any number of folders.
 - Filing them in a folder does not duplicate them.
 - Unfiling them from a folder does not delete them from object store.

© Copyright IBM Corporation 2011

Figure 3-5. Containment concepts

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Containment > Concepts > Folders

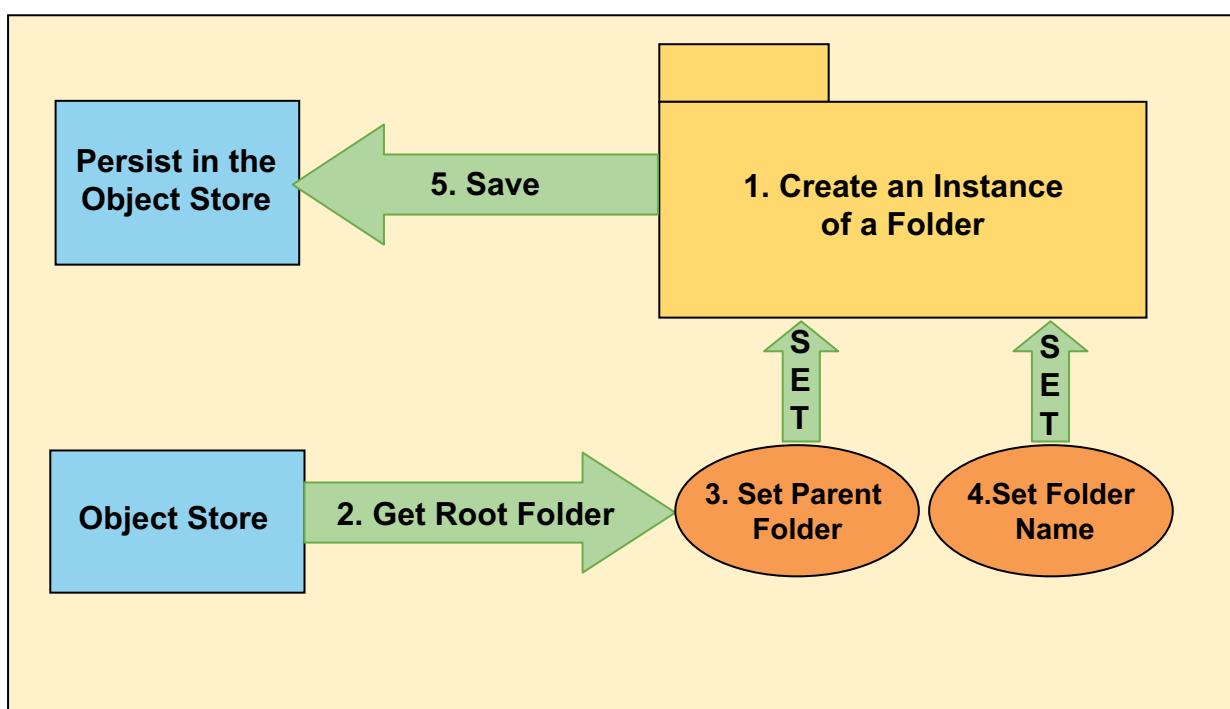
It is possible to referentially contain a folder in another folder. However, this method of containing is not recommended.

Each object store has an automatically created root folder that represents the default root container associated with the object store. You cannot create or delete a root folder, but you can access it.

Folders directly contained under the root folder are referred to as top folders. These folders typically represent the starting points for folder navigation, because, for many applications, you might not want to display or allow users to add objects to the root folder.

Create and delete Folder objects

Create a Folder object



© Copyright IBM Corporation 2011

Figure 3-6. Create a Folder object

F1432.0

Notes:

Steps to create a Folder object:

1. Create a subfolder under the root or parent folder.
2. Get the root or parent folder from the object store.
3. Set parent folder.
4. Set folder name.
5. Save the folder.

Create and delete Folder objects

Steps to create a Folder object (1)

1. Create a subfolder under the root or parent folder.

```
folder.createSubFolder(...) Or  
Factory.Folder.createInstance(...)
```

2. Get the root or parent folder from the object store.

```
Factory.ObjectStore.fetchInstance(...)  
objectStore.get_RootFolder() Or  
Factory.Folder.fetchInstance(...)
```

3. Set parent folder.

```
folder.set_Parent(...)
```

© Copyright IBM Corporation 2011

Figure 3-7. Steps to create a Folder object (1)

F1432.0

Notes:

Create and delete Folder objects

Steps to create a Folder object (2)



4. Set folder name.

```
folder.setFolderName(...)
```

5. Save the folder.

```
folder.save(...)
```

© Copyright IBM Corporation 2011

Figure 3-8. Steps to create a Folder object (2)

F1432.0

Notes:

Create and delete Folder objects

folder.createSubFolder(...)

```
public Folder createSubFolder(String name)
```

- **Parameters**

- name - Specify the name of the subfolder that you want to create
 - Cannot be an empty string
 - The following characters are not allowed: * \ / : * ? " % & > |

- **Returns**

- A Folder object that represents the new subfolder

- **Notes**

- This method creates a directly contained subfolder for this folder.

© Copyright IBM Corporation 2011

Figure 3-9. folder.createSubFolder(...)

F1432.0

Notes:

Three methods are used to create a subfolder:

- Factory.Folder.createInstance(...)
- objectStore.createObject(...)
- folder.createSubFolder(...)

There is no performance difference among these methods. However, the createSubFolder() method is preferred for type-safety reasons. It ensures that required properties are set on the newly created subfolder. Factory.Folder.createInstance() method is discussed next and is also used in the code.

Create and delete Folder objects

Factory.Folder.createInstance(...)



```
public static Folder createInstance  
    (ObjectStore os, String classId)
```

- Parameters
 - os - Specify the name of the objectstore
 - classId – Specify the identifier of the folder subclass you want to create. It can be either of the following:
 - Class name constant
 - Symbolic name of the class
- Returns
 - An object reference to a new instance of this Folder subclass
- Notes
 - This method creates a directly contained subfolder for this folder.

© Copyright IBM Corporation 2011

Figure 3-10. Factory.Folder.createInstance(...)

F1432.0

Notes:

Create and delete Folder objects

Factory.ObjectStore.fetchInstance(...)

```
public static ObjectStore  
    fetchInstance(Domain domain, String name,  
                  PropertyFilter filter)
```

- Parameters

- domain - Specify the Domain object to which this class instance is scoped
- name - Specify the symbolic name of the class instance to retrieve
- filter - Specify a PropertyFilter object that represents information for controlling which property values to return

- Returns

- The ObjectStore object requested.

© Copyright IBM Corporation 2011

Figure 3-11. Factory.ObjectStore.fetchInstance(...)

F1432.0

Notes:

If the value for the property filter is null, this method returns the following:

- Values for all non-object properties
- Placeholders for all object-valued properties

Create and delete Folder objects

objectStore.get_RootFolder()



```
public Folder get_RootFolder()
```

- Parameters
 - None
- Returns
 - The value of the RootFolder property
- Notes
 - Root folder is the automatically created Folder object representing the root of the container hierarchy associated with this Object Store.

© Copyright IBM Corporation 2011

Figure 3-12. objectStore.get_RootFolder()

F1432.0

Notes:

Create and delete Folder objects

folder.set_Parent(...)

```
public void set_Parent(Folder value)
```

- Parameters
 - value - Specify the parent folder for this folder that you created
- Notes
 - This method sets the value of the Parent property.

© Copyright IBM Corporation 2011

Figure 3-13. folder.set_Parent(...)

F1432.0

Notes:

Create and delete Folder objects

folder.set_FolderName(...)



```
public void set_FolderName(String value)
```

- Parameters

- value - Specify the name of the subfolder that you want to create
 - Cannot be an empty string
 - The following characters are not allowed: * \ / : * ? " % & > |

- Notes

- This method sets the value of the FolderName property.

© Copyright IBM Corporation 2011

Figure 3-14. folder.set_FolderName(...)

F1432.0

Notes:

Create and delete Folder objects

folder.save(...)

```
public void save(RefreshMode refreshMode)
```

- Parameters

- refreshMode - Specify whether or not to refresh all of the object properties

- Notes

- This method saves changes made to this folder object.
 - Optionally refreshes all of the object properties.

© Copyright IBM Corporation 2011

Figure 3-15. folder.save(...)

F1432.0

Notes:

Folder is derived from IndependentlyPersistableObject interface.

Create and delete Folder objects

Sample code to create a folder

Create a folder under the root folder.

```
Folder myFolder =  
    Factory.Folder.createInstance(objectStore,null);  
Folder rootFolder = objectStore.get_RootFolder();  
myFolder.set_Parent(rootFolder);  
myFolder.set_FolderName("NewFolder");  
myFolder.save(RefreshMode.REFRESH);
```

Create a subfolder to an existing folder.

```
Folder subFolder =  
    myFolder.createSubFolder("newSubFolder");  
subFolder.save(RefreshMode.REFRESH);
```

© Copyright IBM Corporation 2011

Figure 3-16. Sample code to create a folder

F1432.0

Notes:

Create and delete Folder objects

Demonstrations



- Create a folder under the root folder.
- Create a subfolder to an existing folder.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 3-17. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *createFolderEDU (...)* method in the *FoldersEDU.java* file.
3. Run the code and verify the results as instructed in the *Folders* unit in the Student Exercises book .

Create and delete Folder objects

Deleting folders



- To delete a folder, delete all its subfolders first.
- Other types of containees do not need to be deleted first.
 - Other containees are references only, and the referential relationship markers are cleaned up automatically.
- Root Folder cannot be deleted.

© Copyright IBM Corporation 2011

Figure 3-18. Deleting folders

F1432.0

Notes:

Create and delete Folder objects

Steps to delete a Folder object

- 
1. Get the folder to be deleted.

```
Factory.Folder.fetchInstance(...)
```

2. Delete the folder.

```
folder.delete()
```

3. Save the folder.

```
folder.save(...)
```

© Copyright IBM Corporation 2011

Figure 3-19. Steps to delete a Folder object

F1432.0

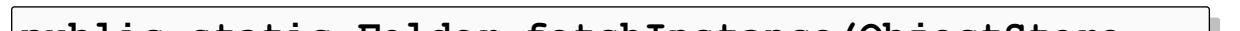
Notes:

Create and delete Folder objects

Factory.Folder.fetchInstance(...)



```
public static Folder fetchInstance(ObjectStore  
os, String path, PropertyFilter filter)
```



- **Parameters**

- os – Specify the Object Store object where this folder is located
- path - Specify the fully qualified path name for the Folder object in a String
- filter – Specify a PropertyFilter object that represents information for controlling which property values to return

- **Returns**

- The specified Folder object

- **Notes**

- If the value for the property filter is null, this method returns the following:
 - Values for all non-object properties
 - Placeholders for all object-valued properties

© Copyright IBM Corporation 2011

Figure 3-20. Factory.Folder.fetchInstance(...)

F1432.0

Notes:

Create and delete Folder objects

folder.delete()

```
public void delete()
```

- Notes

- Cannot delete the root folder.
- Must commit the change to the repository.
- In order to delete a folder with directly contained subfolders, the subfolders must first be deleted.
- Other objects are automatically unfiled from the folder.
- The user must have delete permissions on this Folder object.

© Copyright IBM Corporation 2011

Figure 3-21. folder.delete()

F1432.0

Notes:

Create and delete Folder objects

Sample code to delete a folder

```
String folderToDelete = "/DeleteFolder";
Folder delFolder= Factory.Folder.fetchInstance
    (objectStore, folderToDelete, null);
delFolder.delete();
delFolder.save(RefreshMode.REFRESH);
```

© Copyright IBM Corporation 2011

Figure 3-22. Sample code to delete a folder

F1432.0

Notes:

The *Folder.save(...)* method needs to be called to persist the changes made to this folder (deletion, in this case).

Create and delete Folder objects

Demonstrations



- Delete a folder
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 3-23. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *deleteFolderEDU (...)* method in the *FoldersEDU.java* file.
3. Run the code and verify the results as instructed in the *Folders* unit in the Student Exercises book .

Create and delete Folder objects

Activities



In your Student Exercises book

- Unit: Folders
- Lesson: Create and delete Folder objects
- Activities
 - Create and delete Folder objects.

© Copyright IBM Corporation 2011

Figure 3-24. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 3.2. Retrieve objects from a folder

Lesson: Retrieve objects from a folder

- 
- Why is this lesson important to you?
 - The billing section in your company wants to retrieve customer information from the customer folder. As their programmer, you are going to write code to retrieve the contents of a folder when an employee accesses the folder.

© Copyright IBM Corporation 2011

Figure 3-25. Lesson: Retrieve objects from a folder

F1432.0

Notes:

Retrieve objects from a folder

Activities that you need to complete



- Retrieve objects from a folder.

© Copyright IBM Corporation 2011

Figure 3-26. Activities that you need to complete

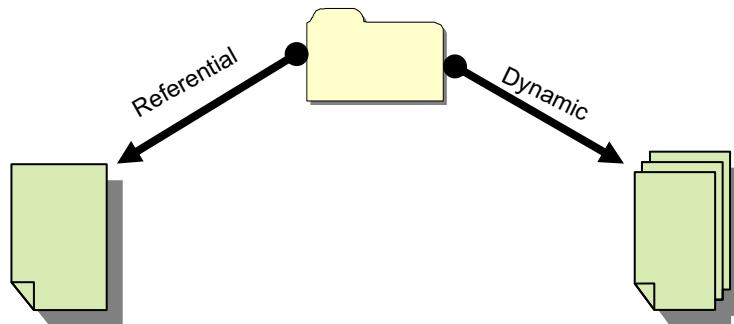
F1432.0

Notes:

Retrieve objects from a folder

Referential containment

- Most objects are contained in folders by reference.
 - Therefore, the same object can be filed in any number of folders.
- The static relationship between an object and each of its folders is represented by a *ReferentialContainmentRelationship* object.
- In a relationship, Tail points to folder and Head points to contained object.



© Copyright IBM Corporation 2011

Figure 3-27. Referential containment

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Containment > Concepts > Referential Containment Relationships

The Containable subclasses are the following:

- CodeModule
- CustomObject
- Document, Folder
- PublishStyleTemplate
- PublishTemplate
- WorkflowDefinition
- XMLPropertyMappingScript

The *ReferentialContainmentRelationship* object represents a static relationship between a folder and a custom object, a workflow definition, or a specific document version.

The *DynamicReferentialContainmentRelationship* object represents a dynamic relationship between a folder and the latest version of a document.

Retrieve objects from a folder

Steps to retrieve objects from a folder (1)



- Retrieve the child folders from a parent folder.

```
folder.get_SubFolders()
```

- Retrieve the documents from a folder.

```
folder.get_CreatedDocuments()
```

- Retrieve all the objects contained in a folder.

```
folder.get_Containedees()
```

© Copyright IBM Corporation 2011

Figure 3-28. Steps to retrieve objects from a folder (1)

F1432.0

Notes:

Retrieve objects from a folder

Steps to retrieve objects from a folder (2)



- Get information about the retrieved objects.

```
referentialContainmentRelationship.getName()  
referentialContainmentRelationship.getHead()  
independentObject.getClassName()  
independentObject.fetchProperties(...)  
independentObject.getProperties()
```

© Copyright IBM Corporation 2011

Figure 3-29. Steps to retrieve objects from a folder (2)

F1432.0

Notes:

Retrieve objects from a folder

folder.get_SubFolders()



```
public FolderSet get_SubFolders()
```

- Parameters
 - None
- Returns
 - A FolderSet collection, which has the directly contained child folders of this Folder
- Notes
 - Any referentially contained child folders are not included in the collection.

© Copyright IBM Corporation 2011

Figure 3-30. folder.get_SubFolders()

F1432.0

Notes:

Retrieve objects from a folder

folder.getProperties().getBooleanValue(...)



Boolean getBooleanValue(String propertyName)

- Parameter
 - propertyName – Specify the property name for which the value is requested.
 - Example: "IsHiddenContainer"
- Returns
 - A Boolean value
- Notes
 - This method locates a PropertyBoolean property by name in this Properties collection and returns the value it holds.
 - The value of this Boolean property indicates whether this folder class should be hidden from non-administrative users (true) or not (false).

© Copyright IBM Corporation 2011

Figure 3-31. folder.getProperties().getBooleanValue(...)

F1432.0

Notes:

Get the properties collection for the folder and then get the value for the specified Boolean property.

```
Properties props = folder.getProperties();
Boolean isHidden = props.getBooleanValue("IsHiddenContainer");
```

You can hide a folder from users of applications (such as Workplace and Application Integration), or unhide a previously hidden folder, using Content Engine Enterprise Manager.

When you hide a folder using this procedure, the documents, custom objects, and folders that are contained in the folder can still be found by a search.

IsHiddenContainer indicates whether the GUI elements that represent a folder in FileNet P8 should be made visible to the user. By default, this property is set to False causing the GUI elements to be displayed. This property is not directly enforced by Content Engine, but is employed by the client application, such as Workplace.

Retrieve objects from a folder

folder.get_CreatedDocuments()



```
public DocumentSet get_CreatedDocuments ()
```

- Returns
 - A DocumentSet collection, which has the Documents contained within this folder
- Notes
 - Any referentially contained child folders are not included in the collection.

© Copyright IBM Corporation 2011

Figure 3-32. folder.get_CreatedDocuments()

F1432.0

Notes:

Retrieve objects from a folder

folder.get_Containees()

```
public ReferentialContainmentRelationshipSet  
get_Containees()
```

- Returns
 - A ReferentialContainmentRelationshipSet collection, which has the ReferentialContainmentRelationship objects contained within this folder

© Copyright IBM Corporation 2011

Figure 3-33. folder.get_Containees()

F1432.0

Notes:

Retrieve objects from a folder

referentialContainmentRelationship.get_Name()



```
public String get_Name()
```

- Returns
 - A String representing the name of this object

© Copyright IBM Corporation 2011

Figure 3-34. referentialContainmentRelationship.get_Name()

F1432.0

Notes:

Retrieve objects from a folder

referentialContainmentRelationship.get_Head()

IndependentObject get_Head()

- Returns

- The value of the Head property
 - Head is an IndependentlyPersistableObject instance that participates as the logical head (containee) in a relationship between two objects

- Notes

- For a ReferentialContainmentRelationship object, the head can be any of the Containable subclasses:
 - CustomObject, Document, or Folder object
 - If the head is a Document object, it represents a specific document version.

© Copyright IBM Corporation 2011

Figure 3-35. referentialContainmentRelationship.get_Head()

F1432.0

Notes:

For a DynamicReferentialContainmentRelationship object, the Head is always a versionable object representing the current document version.

For a Link object, this property can specify any object that is a subclass of the IndependentObject base class.

Retrieve objects from a folder

independentObject.getClassName()



```
public String getClassName()
```

- Returns
 - The name of the class from which this object is instantiated

© Copyright IBM Corporation 2011

Figure 3-36. independentObject.getClassName()

F1432.0

Notes:

Retrieve objects from a folder

Sample code to retrieve Folder objects

Get the parent folder.

```
Folder folder= Factory.Folder.fetchInstance  
          (objectStore,parentFolder, null);
```

Get the subfolders (Check to see if the folder is hidden).

```
FolderSet subFolders = folder.get_SubFolders();  
Iterator it = subFolders.iterator();  
while(it.hasNext()) {  
    Folder subFolder = (Folder)it.next();  
    String name = subFolder.get_FolderName();  
    if(retrieveFolder.getProperties().  
        getBooleanValue("IsHiddenContainer"))  
        System.out.println("Folder"+name+"is hidden");  
}
```

© Copyright IBM Corporation 2011

Figure 3-37. Sample code to retrieve Folder objects

F1432.0

Notes:

Retrieve objects from a folder

Sample code to retrieve Document objects

```
DocumentSet documents =  
    folder.get_CreatedDocuments();  
Iterator it = documents.iterator();  
while(it.hasNext()) {  
    Document retrieveDoc = (Document)it.next();  
    String name = retrieveDoc.get_Name();  
}
```

© Copyright IBM Corporation 2011

Figure 3-38. Sample code to retrieve Document objects

F1432.0

Notes:

Retrieve objects from a folder

Sample code to retrieve objects from a folder



```
ReferentialContainmentRelationshipSet  
    refConRelSet = folder.get_Containees();  
Iterator it = refConRelSet.iterator();  
while(it.hasNext()) {  
    ReferentialContainmentRelationship retrieveObj  
    =(ReferentialContainmentRelationship)it.next();  
    IndependentObject containee =  
        retrieveObj.get_Head();  
    String className = containee.getClassName();  
    String displayName = retrieveObj.get_Name();  
}
```

© Copyright IBM Corporation 2011

Figure 3-39. Sample code to retrieve objects from a folder

F1432.0

Notes:

Retrieve objects from a folder

Demonstrations



- Retrieve subfolders from a folder.
- Retrieve documents from a folder.
- Retrieve all the objects from a folder.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 3-40. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getSubFoldersEDU (...)* method in the *FoldersEDU.java* file.
3. Run the code and verify the results as instructed in the *Folders* unit in the Student Exercises book .
4. Repeat steps 2 and 3 for the *getDocsInThisFolderEDU (...)* method that gets documents from the folder.
5. Repeat steps 2 and 3 for the *getFolderContaineesEDU (...)* method that gets objects from the folder.

Retrieve objects from a folder

Activities

In your Student Exercises book

- Unit: Folders
- Lesson: Retrieve objects from a folder
- Activities
 - Retrieve objects from a folder.

© Copyright IBM Corporation 2011

Figure 3-41. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 4. Custom Objects

What this unit is about

This unit describes how to create and how to retrieve custom objects.

What you should be able to do

After completing this unit, you should be able to:

- Create custom objects.
- Retrieve custom object properties.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Custom Objects

Unit lessons

- 
- Create custom objects
 - Retrieve custom object properties

© Copyright IBM Corporation 2011

Figure 4-1. Unit lessons

F1432.0

Notes:

Lesson 4.1. Create custom objects

Lesson: Create custom objects



- Why is this lesson important to you?
 - Morgan registers online to be a customer of a company. The application uses a custom object class to store the profile of the customers. As a programmer, you are going to write code to create an instance of the class each time a customer enters this information.

© Copyright IBM Corporation 2011

Figure 4-2. Lesson: Create custom objects

F1432.0

Notes:

Create custom objects

Activities that you need to complete



- Create custom objects.

© Copyright IBM Corporation 2011

Figure 4-3. Activities that you need to complete

F1432.0

Notes:

Create custom objects

What are custom objects?



- A custom object is an object without content
 - Created to model its business process usage
 - Named for its business process usage
 - Used to store properties
- A custom object shares the following characteristics with a Document object:
 - Is independently securable
 - Can be persisted in an object store and filed into a folder
 - Can have annotations associated with it
 - Can be a target for a workflow subscription
 - Can subscribe to server-side events

© Copyright IBM Corporation 2011

Figure 4-4. What are custom objects?

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Custom Objects > Concepts

Create custom objects

Custom objects are unlike Document objects

- Unlike a Document object, a custom object does **not** have the following characteristics:
 - Carry content
 - Have a default Name property (in the CustomObject class)
 - Have multiple versions
 - Support lifecycle states
 - Have all the system properties that a document has

© Copyright IBM Corporation 2011

Figure 4-5. Custom objects are unlike Document objects

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Custom Objects > Concepts

You can configure a Name property for custom object classes. The value of this property is then used as the default Containment Name.

For example, you might define the CustomerName property as the Name property for the Customers class.

However, only one of the properties can be configured in IBM FileNet Workplace for each object store as the custom object name property.

Create custom objects

Why use custom objects?



- Use a custom object when you do the following:
 - Need a template for storing record-oriented data
 - Do not need to store file content
 - Do not need automatic versioning of the data
 - Want the data to be secured
 - Need to conserve storage space
- Applications often associate custom properties with the objects that they process
 - Create a CustomObject subclass for each set of application-specific custom properties
 - To link the subclass with another object, create a custom property that points to the associated object by using the GUID of the associated object as the value

© Copyright IBM Corporation 2011

Figure 4-6. Why use custom objects?

F1432.0

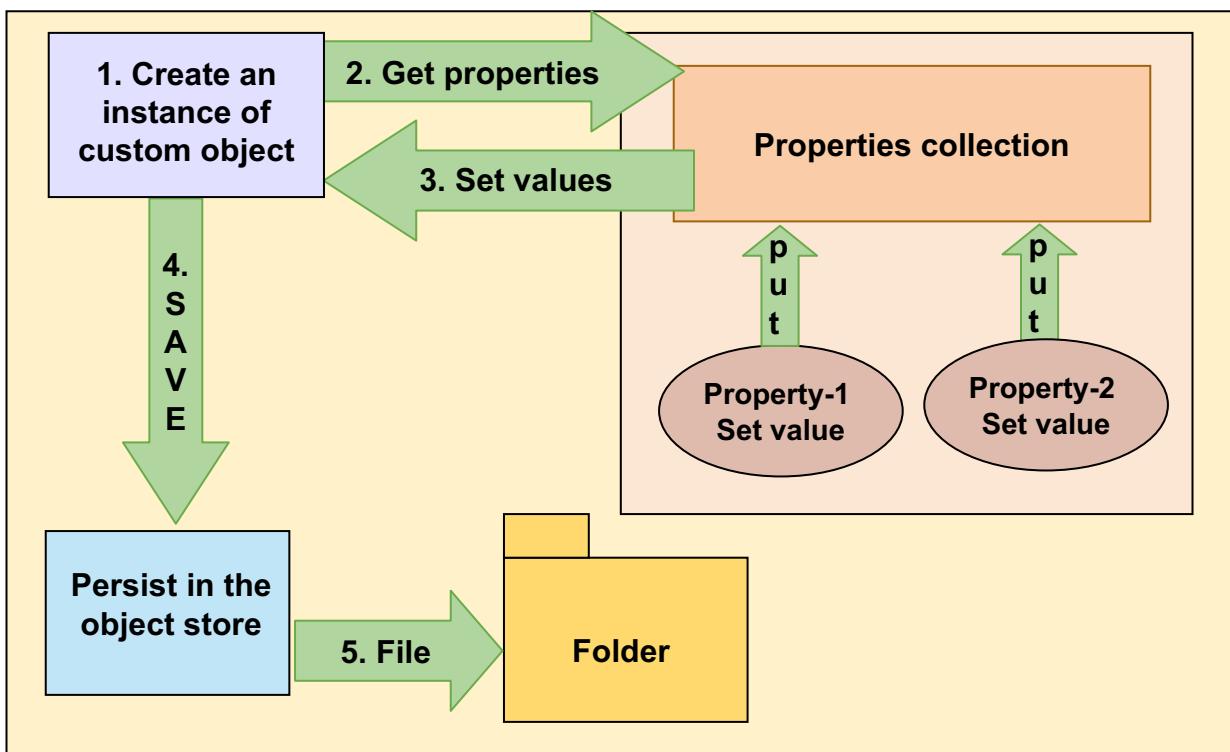
Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Custom Objects > Concepts

Create custom objects

Creating a custom object



© Copyright IBM Corporation 2011

Figure 4-7. Creating a custom object

F1432.0

Notes:

1. Create an instance of custom object.
2. Get the Properties of the custom object.
3. Set Properties to the custom object.
4. Save the custom object.
5. File the custom object in a folder.

Create custom objects

Steps to create a custom object (1)



1. Create an instance of custom object.

```
Factory.CustomObject.createInstance(...)
```

2. Get the properties for the custom object.

```
customObject.getProperties()
```

3. Set Properties to the custom object

```
properties.putValue(...)
```

© Copyright IBM Corporation 2011

Figure 4-8. Steps to create a custom object (1)

F1432.0

Notes:

Create custom objects

Steps to create a custom object (2)



4. Save the custom object.

```
customObject.save(...)
```

5. File the custom object in a folder (optional).

```
folder.file(...)
```

```
referentialContainmentRelationship.save(...)
```

© Copyright IBM Corporation 2011

Figure 4-9. Steps to create a custom object (2)

F1432.0

Notes:

Create custom objects

Factory.CustomObject.createInstance(...)

```
static CustomObject  
    createInstance(ObjectStore os, String  
    classId)
```

- Parameters
 - os – Specify the object store where the custom object is to be persisted
 - classId – Specify the class identifier of the custom object
- Returns
 - A CustomObject object reference to a new instance of this class
- Notes
 - You must call the save method to persist this object in the object store.
 - This method does not file the custom object into the folder.
 - A subclassed Custom object also inherits the properties and permissions from its superclass.

© Copyright IBM Corporation 2011

Figure 4-10. Factory.CustomObject.createInstance(...)

F1432.0

Notes:

Create custom objects

customObject.getProperties ()

Properties getProperties ()

- Returns
 - A Properties collection representing the cached properties of this object
- Notes
 - An object reference does not have values in its property collection.
 - Therefore, you must refresh the object before calling this method.
 - This method is inherited from EngineObject interface.

© Copyright IBM Corporation 2011

Figure 4-11. customObject.getProperties ()

F1432.0

Notes:

Create custom objects

properties.putvalue(...)

```
void putValue(String propertyName, String value)
```

- Parameters
 - propertyName - Specify the property name
 - value - Specify the value for the property
- Notes
 - This method creates or updates a PropertyString instance in this Properties collection.
 - If a property with the same name already exists in the collection, its value is replaced with the value provided.

© Copyright IBM Corporation 2011

Figure 4-12. properties.putvalue(...)

F1432.0

Notes:

Create custom objects

independentlyPersistableObject.save(...)



```
void save(RefreshMode refreshMode)
```

- Parameters
 - refreshMode - Specifies whether or not to refresh all of the properties of the object
- Notes
 - There is another version of this method:
 - public void save(RefreshMode refreshMode, PropertyFilter filter)
 - A PropertyFilter object that represents information for controlling which property values to refresh
 - Specifying null for property filter is the same as calling save(RefreshMode refreshMode).

© Copyright IBM Corporation 2011

Figure 4-13. independentlyPersistableObject.save(...)

F1432.0

Notes:

The CustomObject interface extends the IndependentlyPersistableObject interface.

Create custom objects

folder.file(...)



```
ReferentialContainmentRelationship
file(IndependentlyPersistableObject
    containee,AutoUniqueName autoUniqueName, String
    containmentName,DefineSecurityParentage
    defineSecurityParentage)
```

- Parameters

- containee - Specify the custom object to be filed
- autoUniqueName - Specify whether or not to resolve the containment names for uniqueness
- containmentName - Specify a name for the containment
- defineSecurityParentage – DEFINE_SECURITY_PARENTAGE

- Returns

- A ReferentialContainmentRelationship object, where this folder is the Tail and the custom object filed is the Head

© Copyright IBM Corporation 2011

Figure 4-14. folder.file(...)

F1432.0

Notes:

This method files the custom object to this folder.

Parameters

Containee: An IndependentlyPersistableObject object for the Containable subclass instance to be filed

autoUniqueName: An AutoUniqueName object indicating whether to detect and resolve naming collisions of containment names

containmentName: A String containing the containment name to assign to the new object. By default, this name is the file name of the object. The following characters are not allowed: \ / : * ? " < > |

defineSecurityParentage: A DefineSecurityParentage object indicating whether the containing folder is to be used as a security parent. If so, the SecurityFolder property of the object being added is automatically set to this folder.

Create custom objects

Sample code to create a custom object and set properties

```
CustomObject myObject = Factory.CustomObject.  
    createInstance(os, "Customer");  
com.filenet.api.property.Properties props =  
    myObject.getProperties();  
props.putValue("customer_name", "John Smith");  
props.putValue("customer_id", "CUS29");  
props.putValue("age", 29);  
DateFormat date = new  
    SimpleDateFormat("dd/MM/yyyy");  
Date memberDate = date.parse("20/12/2005");  
props.putValue("member_date", memberDate);
```

© Copyright IBM Corporation 2011

Figure 4-15. Sample code to create a custom object and set properties

F1432.0

Notes:

Create custom objects

Sample code to set a multivalued property of a custom object



```
StringList phones =  
    Factory.StringList.createList();  
phones.add("000-000-0000");  
phones.add("333-444-5555");  
props.putValue("phone_numbers", phones);
```

© Copyright IBM Corporation 2011

Figure 4-16. Sample code to set a multivalued property of a custom object

F1432.0

Notes:

Create custom objects

Sample code to save and file a custom object to a folder

```
myObject.save(RefreshMode.REFRESH);  
System.out.println("Custom object " +  
myObject.getName() + " created");  
ReferentialContainmentRelationship rel =  
folder.file(myObject,AutoUniqueName.AUTO_UN  
IQUE,customObjectName,DefineSecurityParenta  
ge.DO_NOT_DEFINE_SECURITY_PARENTAGE);  
rel.save(RefreshMode.REFRESH);
```

© Copyright IBM Corporation 2011

Figure 4-17. Sample code to save and file a custom object to a folder

F1432.0

Notes:

Create custom objects

Demonstrations



- Create custom objects.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 4-18. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

- Start Eclipse and open the *CMJavaAPISolution* project.
- Review the code for the *createCustomObjectEDU(...)* method in the *CustomObjectsEDU.java* file.
- Run the code and verify the results as instructed in the *Custom Objects* unit in the Student Exercises book.

Create Custom Objects

Activities

In your Student Exercises book

- Unit: Custom Objects
- Lesson: Create custom objects
- Activities
 - Create custom objects.

© Copyright IBM Corporation 2011

Figure 4-19. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 4.2. Retrieve custom object properties

Lesson: Retrieve custom object properties

- 
- Why is this lesson important to you?
 - The application has a custom object class that stores customer profiles. As the programmer, you are going to write code to retrieve custom object properties in order to modify the profile each time a customer updates this information.

© Copyright IBM Corporation 2011

Figure 4-20. Lesson: Retrieve custom object properties

F1432.0

Notes:

The "Retrieve object properties" topic is discussed in detail in the "Properties" unit.

Retrieve custom object properties

Activities that you need to complete



- Retrieve custom object properties.

© Copyright IBM Corporation 2011

Figure 4-21. Activities that you need to complete

F1432.0

Notes:

Retrieve custom object properties

Steps to retrieve custom object properties



1. Get the custom object.

```
Factory.CustomObject.fetchInstance(...)
```

2. Get the properties collection for the custom object.

```
customObject.getProperties(...)
```

3. Get the value for each property.

```
properties.getStringValue(...)  
properties.getInteger32Value(...)  
properties.getDateJsonValue(...)
```

© Copyright IBM Corporation 2011

Figure 4-22. Steps to retrieve custom object properties

F1432.0

Notes:

Retrieve custom object properties

Factory.CustomObject.fetchInstance(...)

```
static CustomObject fetchInstance  
    (ObjectStore os, String path,  
     PropertyFilter filter)
```

- Parameters

- os - Specify the object store in which this custom object is located
- path - Specify the full folder path for the custom object
- filter - Specify a PropertyFilter object that represents information for controlling which property values to return

- Returns

- The requested CustomObject object

© Copyright IBM Corporation 2011

Figure 4-23. Factory.CustomObject.fetchInstance(...)

F1432.0

Notes:

The folder path must contain the folder name where the custom object is filed and the containment used to file this custom object.

If the PropertyFilter value is null, this method returns values for all non-object properties and returns placeholders for all object-valued properties.

Any subsequent attempts to access an object-valued property causes an automatic round-trip to the server to fetch its value.

Retrieve custom object properties

properties.getStringValue(...)



```
String getStringValue(String propertyName)
```

- Parameters
 - propertyName – Specify the property name for which you want to retrieve the value
- Returns
 - A String specifying the value of this property
- Notes
 - This method locates a PropertyString property by name in this Properties collection and returns the value that it holds.

© Copyright IBM Corporation 2011

Figure 4-24. properties.getStringValue(...)

F1432.0

Notes:

Retrieve custom object properties

properties.getInteger32Value(...)



```
Integer getInteger32Value(String propertyName)
```

- Parameters

- propertyName – Specify the property name for which you want to retrieve the value

- Returns

- An Integer specifying the value of this property

- Notes

- This method locates a PropertyInteger32 property by name in this Properties collection and returns the value that it holds.

© Copyright IBM Corporation 2011

Figure 4-25. properties.getInteger32Value(...)

F1432.0

Notes:

Retrieve custom object properties

properties.getDateTimeValue(...)



```
Date getDateTimeValue(String propertyName)
```

- Parameters
 - propertyName – Specify the property name for which you want to retrieve the value
- Returns
 - A Date specifying the value of this property
- Notes
 - This method locates a PropertyDateTime property by name in this Properties collection and returns the value it holds.

© Copyright IBM Corporation 2011

Figure 4-26. properties.getDateTimeValue(...)

F1432.0

Notes:

Retrieve custom object properties

properties.getStringListValue(...)



```
StringList getStringListValue(String  
    propertyName)
```

- Parameters
 - propertyName – Specify the property name for which you want to retrieve the value
- Returns
 - A StringList collection specifying the value of this property
- Notes
 - This method locates a PropertyStringList property by name in this Properties collection and returns the value that it holds.

© Copyright IBM Corporation 2011

Figure 4-27. properties.getStringListValue(...)

F1432.0

Notes:

Retrieve custom object properties

Sample code to retrieve a custom object



```
CustomObject myObject =  
    Factory.CustomObject.fetchInstance(store,  
                                      folderPath, null);  
  
String CustomObjName =  myObject.get_Name();  
System.out.println(CustomObjName + "  
                    CustomObject is retrieved");
```

© Copyright IBM Corporation 2011

Figure 4-28. Sample code to retrieve a custom object

F1432.0

Notes:

Retrieve custom object properties

Sample code to retrieve custom object properties

```
com.filenet.api.property.Properties props =  
    myObject.getProperties();  
  
String name = props.getStringValue("customer_name");  
String custID = props.getStringValue("customer_id");  
Integer age = props.getInteger32Value("age");  
Date dateCreated = props.getDate TValue  
    (PropertyNames.DATE_CREATED);  
Date memberDate = props.getDate TValue  
    ("member_date");  
  
System.out.println("Customer Name: " + name);  
System.out.println("Customer age: " + age);  
System.out.println("Customer ID: " + custID);  
System.out.println("Date created: " + dateCreated);  
System.out.println("Member since : " +  
    memberDate);
```

© Copyright IBM Corporation 2011

Figure 4-29. Sample code to retrieve custom object properties

F1432.0

Notes:

Retrieve custom object properties

Sample code to retrieve a multivalued property

```
...
StringList phoneNumbers =
    props.getStringListValue("phone_numbers");
Iterator it = phoneNumbers.iterator();
System.out.println("Customer phone numbers: ");
while (it.hasNext())
{
    String phoneNumber = (String)it.next();
    System.out.println(phoneNumber);
}
```

© Copyright IBM Corporation 2011

Figure 4-30. Sample code to retrieve a multivalued property

F1432.0

Notes:

Retrieve custom object properties

Demonstrations



- Retrieve custom object properties.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 4-31. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

- Start Eclipse and open the *CMJavaAPISolution* project.
- Review the code for the *getCustomObjectEDU(...)* method in the *CustomObjectsEDU.java* file.
- Run the code and verify the results as instructed in the *Custom Objects* unit in the Student Exercises book.

Retrieve custom object properties

Activities



In your Student Exercises book

- Unit: Custom Objects
- Lesson: Retrieve custom object properties
- Activities
 - Retrieve custom object properties.

© Copyright IBM Corporation 2011

Figure 4-32. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 5. Documents

What this unit is about

This unit describes how to create documents and how to retrieve them.

What you should be able to do

After completing this unit, you should be able to:

- Create Document objects
- Retrieve a document and its content

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Documents

Unit lessons



- Create Document objects
- Retrieve a document and its content

© Copyright IBM Corporation 2011

Figure 5-1. Unit lessons

F1432.0

Notes:

Lesson 5.1. Create Document objects

Lesson: Create Document objects

- 
- Why is this lesson important to you?
 - Your application requires a feature that creates a Document object on the Content Engine whenever someone adds content. You, as the programmer, are going to write code to accomplish this.

© Copyright IBM Corporation 2011

Figure 5-2. Lesson: Create Document objects

F1432.0

Notes:

Create Document objects

Activities that you need to complete



- Create document objects.

© Copyright IBM Corporation 2011

Figure 5-3. Activities that you need to complete

F1432.0

Notes:

Create Document objects

Document objects



- A Document object
 - Is an instance of the Document class or a subclass
- A Document object can have the following:
 - Zero or more content elements
 - Associated annotations
 - Custom metadata or properties that are used for identification
- Document content elements are stored
 - Locally, inside an object store
 - In an external repository and referenced from the object store
- A Document object can be versioned
 - Each version of Document
 - Is assigned a version number
 - Is saved as a separate, accessible, unique entity

© Copyright IBM Corporation 2011

Figure 5-4. Document objects

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content Engine overview > Features > Documents
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Documents > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces

The main difference between the Document objects and CustomObject objects:

CustomObject objects do not have content or versions but Document objects do.

Create Document objects

Available actions on Document objects



- A Document object can be
 - Filed in a folder
 - Secured
 - Repurposed for publishing
 - Searched for and retrieved
 - Associated with other documents to form compound documents
- In addition, a Document object can be
 - Subclassed to add additional custom properties
 - Associated with lifecycle policies
 - Subscribed to a server-side event
 - Used as an attachment in a workflow step
 - Designated as the target for a workflow subscription

© Copyright IBM Corporation 2011

Figure 5-5. Available actions on Document objects

F1432.0

Notes:

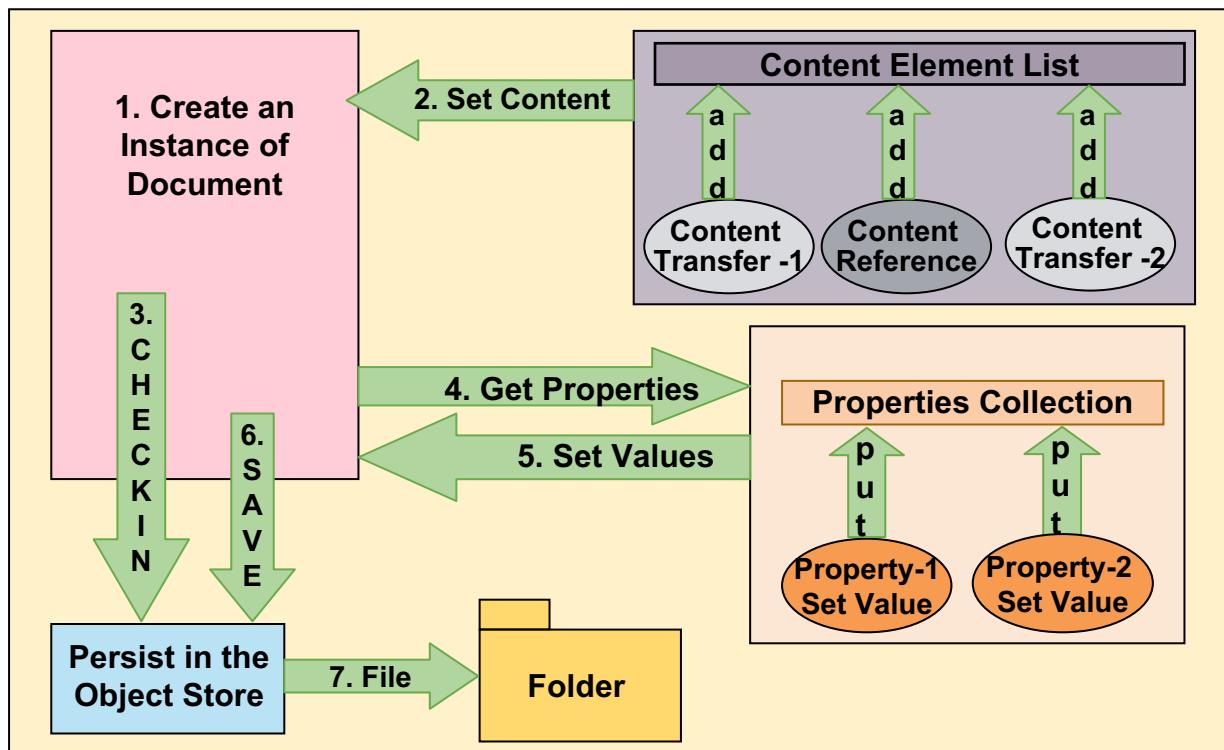
Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content Engine overview > Features > Documents
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Documents > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces

Security, search, and publish concepts are presented in separate units in this course.

Create Document objects

Steps to create a document



© Copyright IBM Corporation 2011

Figure 5-6. Steps to create a document

F1432.0

Notes:

1. Create a Document object.
2. Set Contents to the Document object.
 - a. Create an empty ContentElementList collection.
 - b. Create a ContentTransfer object.
 - c. Specify the source of the content.
 - d. Add contentTransfer element to the contentElementList collection.
 - e. Set content to the Document object.
3. Check in the document.
4. Get Properties from the document.
5. Set Properties to the document.
6. Save the document.
7. File the document in a folder (optional).

Create Document objects

Steps to create a Document object

1. Create a Document object.

```
Factory.Document.createInstance(...)
```

2. Set content to Document object. (See the following pages.)
3. Check in the document.

```
document.checkin(...)
```

4. Get properties from the document.

```
document.getProperties()
```

5. Set properties to the document.

```
properties.putValue()
```

```
document.set_MimeType(...)
```

© Copyright IBM Corporation 2011

Figure 5-7. Steps to create a Document object

F1432.0

Notes:

Create Document objects

Steps to save the document and file



6. Save the document.

```
document.save(...)
```

7. File the document in a folder (optional).

```
folder.file(...)
```

```
referentialContainmentRelationship.save(...)
```

© Copyright IBM Corporation 2011

Figure 5-8. Steps to save the document and file

F1432.0

Notes:

Create Document objects

Steps to set content to the Document object

1. Create a ContentElementList and ContentTransfer objects.

```
Factory.ContentElement.createList()
```

```
Factory.ContentTransfer.createInstance()
```

2. Specify the source of the content.

```
contentTransfer.setCaptureSource(...)
```

3. Add a contentTransfer element to the contentElementList.

```
contentElementList.add(...)
```

4. Set content to the Document object.

```
document.set_ContentElements(...)
```

© Copyright IBM Corporation 2011

Figure 5-9. Steps to set content to the Document object

F1432.0

Notes:

Create Document objects

Steps to set content reference element

- 
1. Use the existing ContentElementList collection.
 2. Create a ContentReference object.

```
Factory.ContentReference.createInstance()
```

3. Specify the source of the content.

```
contentReference.set_ContentLocation(...)
```

4. Add a ContentReference element to the ContentElementList.

```
contentElementList.add(...)
```

5. Set content to the Document object.

```
document.set_ContentElements(...)
```

© Copyright IBM Corporation 2011

Figure 5-10. Steps to set content reference element

F1432.0

Notes:

Create Document objects

Factory.Document.createInstance(...)

```
public static Document  
createInstance(ObjectStore os, String classId)
```

- Parameters

- os - Specify the object store where this Document is to be persisted
- classId - Specify the symbolic name of the class instance to create

- Returns

- An object reference to a new instance of this Document class

- Notes

- This method creates a new instance of the specified Document class.
- You must call the save method to persist this object in the object store.

© Copyright IBM Corporation 2011

Figure 5-11. Factory.Document.createInstance(...)

F1432.0

Notes:

Create Document objects

Factory.ContentElement.createList()



```
public static ContentElementList createList()
```



- Returns
 - ContentElementList object
- Notes
 - This method creates a new instance of this collection class.
 - This list is a dependent object.
 - It is persisted on the server when its parent object has been saved with this new list.

© Copyright IBM Corporation 2011

Figure 5-12. Factory.ContentElement.createList()

F1432.0

Notes:

Create Document objects

Factory.ContentTransfer.createInstance()

```
public static ContentTransfer createInstance()
```

- Returns
 - A ContentTransfer object
- Notes
 - This method creates a new instance of ContentTransfer class.
 - This object is a dependent object.
 - It is persisted on the server when its parent object has been saved.

© Copyright IBM Corporation 2011

Figure 5-13. Factory.ContentTransfer.createInstance()

F1432.0

Notes:

Create Document objects

contentTransfer.setCaptureSource(...)



```
void setCaptureSource(java.io.InputStream  
                      source)
```

- **Parameters**

- **source** - Specify the source of the content data in an **InputStream** object.

© Copyright IBM Corporation 2011

Figure 5-14. contentTransfer.setCaptureSource(...)

F1432.0

Notes:

Create Document objects

contentTransfer.set_ContentType(...)

```
void set_ContentType(String type)
```

- Parameters

- type - Specify the Multipurpose Internet Mail Extensions (MIME) format string of the content data indicates to the server what kind of data this content element represents.

- Notes

- The specified Multipurpose Internet Mail Extensions (MIME) format string of the content data indicates to the server what kind of data this content element represents.
 - If you do not set the ContentType property, it is automatically set by the server.

© Copyright IBM Corporation 2011

Figure 5-15. contentTransfer.set_ContentType(...)

F1432.0

Notes:

Create Document objects

Factory.ContentReference.createInstance()

```
public static ContentReference  
    createInstance()
```

- Returns
 - A ContentReference object
- Notes
 - This method creates a new instance of the ContentReference class.
 - This object is a dependent object.
 - It is persisted on the server when its parent object has been saved.

© Copyright IBM Corporation 2011

Figure 5-16. Factory.ContentReference.createInstance()

F1432.0

Notes:

The ContentReference object represents external content that exists outside of an object store (and therefore outside the control of the Content Engine server), but to which an object store maintains a reference.

The URL of the resource that contains the content data is stored in the ContentLocation property.

The object is persisted to an object store when its parent object (the independently persistable object that references it) is persisted.

The most frequently used form of this method takes no arguments. Because a dependent object is persisted to the object store of its parent, there is no need to specify an object store.

A second form of this method, provided for backward compatibility with the IBM FileNet P8 Content Engine COM API, takes an object store argument. If the specified object store is not that of the parent object, an exception is thrown when the parent is persisted.

Create Document objects

contentReference.set_ContentLocation(...)



```
void set_ContentLocation(String location)
```

- Parameter

- location - Specify, in URL format, the name of the resource that contains the content data represented by this ContentReference object.

© Copyright IBM Corporation 2011

Figure 5-17. contentReference.set_ContentLocation(...)

F1432.0

Notes:

Create Document objects

contentReference.set_ContentType(...)

```
void set_ContentType(String type)
```

- Parameters

- type - Specify the MIME format of the content data carried by this content element

- Notes

- The specified MIME format string of the content data indicates to the server what kind of data this content element represents.
 - If you do not set the ContentType property, it is automatically set by the server.

© Copyright IBM Corporation 2011

Figure 5-18. contentReference.set_ContentType(...)

F1432.0

Notes:

The ContentType property specifies the MIME format string of the content data carried by this content element, which indicates to the server what kind of data this content element represents. If you do not set the ContentType property, it is automatically set by the server.

The following MIME types are specific to IBM FileNet:

- application/x-filenet-declarerecordtemplate: Record template
- application/x-filenet-documentassembly: Document assembly
- application/x-filenet-external: An object that contains a single ContentReference content element
- application/x-filenet-external-is: External Image Services document
- application/x-filenet-publishtemplate: Publish template
- application/x-filenet-scenariodefinition: Scenario definition document
- application/x-filenet-search: Stored search

- application/x-filenet-searchtemplate: Search template
- application/x-filenet-workflowdefinition: Workflow definition document
- multipart/x-filenet-external: An object that contains multiple ContentReference content elements only

Create Document objects

document.set_ContentElements(...)



```
void set_ContentElements(ContentElementList  
    value)
```

- Parameters

- value - Specify the ContentElementList that has the content elements

- Notes

- This method sets the content to the Document object.
 - Each content element in the list represents content data.
 - Data can be either of the following:
 - Local to an object store (represented by IContentTransfer object)
 - External (represented by an IContentReference object)

© Copyright IBM Corporation 2011

Figure 5-19. document.set_ContentElements(...)

F1432.0

Notes:

Create Document objects

document.checkin(...)



```
void checkin(AutoClassify autoClassify,
CheckinType checkinType)
```

- Parameters

- autoClassify - Specify an AutoClassify constant, which denotes whether or not to enable autoclassification of a document during check-in
- checkinType - Specify a CheckinType constant, which denotes whether to check in a document as a new minor version or as a new major version

© Copyright IBM Corporation 2011

Figure 5-20. document.checkin(...)

F1432.0

Notes:

Parameters:

AutoClassify constants

- AutoClassify.DO_NOT_AUTO_CLASSIFY
- AutoClassify.AUTO_CLASSIFY

CheckinType constants

- CheckinType.MINOR_VERSION
- CheckinType.MAJOR_VERSION

The autoClassify parameter specifies whether the document must have automatic document classification enabled.

Do the following before you check in a document:

- You must set the content for the document if a document has content.

- You must explicitly set the value if you do not want the Content Engine to assign a MIME type to the checked-in document.
- You must have the appropriate access rights to check in a document.
- This method checks in this document reservation object by saving it as a new document version.

Create Document objects

document.getProperties ()

Properties getProperties ()

- Returns
 - A Properties collection representing the cached properties of this object
- Notes
 - An object reference does not have values in its property collection.
 - Therefore, you must refresh the object before calling this property.

© Copyright IBM Corporation 2011

Figure 5-21. document.getProperties ()

F1432.0

Notes:

Create Document objects

properties.putvalue(...)



```
void putValue(String propertyName, String value)
```

- Parameters

- propertyName - Specify the property name
 - value - Specify the value for the property

- Notes

- This method creates or updates a PropertyString instance in this Properties collection.
 - If a property with the same name already exists in the collection, its value is replaced with the value provided.

© Copyright IBM Corporation 2011

Figure 5-22. properties.putvalue(...)

F1432.0

Notes:

Create Document objects

document.set_MimeType(...)

```
void set_MimeType(String value)
```

- Parameters

- value - Specify the content type

- Notes

- This method sets the content type for the document content element.

© Copyright IBM Corporation 2011

Figure 5-23. document.set_MimeType(...)

F1432.0

Notes:

The MimeType property specifies the MIME format string of the content data carried by this document.

You can set the MimeType property for a specific document version at the time of creation and on subsequent check-outs (when the document is in reservation).

However, every time you check in a document, its MimeType property value reverts to the system-assigned value unless you have explicitly set it again.

Each content element that is attached to a document has its own MIME type, which is specified by its ContentType property.

Create Document objects

folder.file(...)

```
ReferentialContainmentRelationship file  
    (IndependentlyPersistableObject  
     containee, AutoUniqueName autoUniqueName, String  
      containmentName, DefineSecurityParentage  
      defineSecurityParentage)
```

- Parameters
 - containee - Specify the Document object to be filed
 - autoUniqueName - Specify whether or not to resolve the containment names for uniqueness
 - containmentName - Specify a name for the containment
- Returns
 - A ReferentialContainmentRelationship object where this folder is the Tail and the document filed is the Head.

© Copyright IBM Corporation 2011

Figure 5-24. folder.file(...)

F1432.0

Notes:

This method files the containee in the given folder.

Parameters

containmentName - By default, the name is the file name of the document.

Create Document objects

independentlyPersistableObject.save(...)



```
void save(RefreshMode refreshMode)
```



- Parameters

- refreshMode - Specify whether or not to refresh all of the object properties
 - RefreshMode.NO_REFRESH
 - RefreshMode.REFRESH

- Notes

- This method saves changes made to this object.
- Optionally, it refreshes all of the object properties.

© Copyright IBM Corporation 2011

Figure 5-25. independentlyPersistableObject.save(...)

F1432.0

Notes:

ReferentialContainmentRelationship is derived from IndependentlyPersistableObject interface.

RefreshMode.NO_REFRESH

This parameter specifies that, upon saving or committing the object, its property cache is not refreshed with updated data from the Content Engine server.

RefreshMode REFRESH

This parameter specifies that, upon saving or committing the object, its property cache is refreshed with updated data from the Content Engine server.

Create Document objects

Sample code to create a Document object (1)

- Create a Document object.

```
Document myDoc =  
    Factory.Document.createInstance  
        (objectStore, "Product");
```

- Set content to the Document object.

```
ContentElementList contentList =  
    Factory.ContentElement.createList();  
ContentTransfer content =  
    Factory.ContentTransfer.createInstance();  
content.setCaptureSource(new  
    FileInputStream("Model 200.GIF"));  
contentList.add(content);  
myDoc.set_ContentElements(contentList);
```

© Copyright IBM Corporation 2011

Figure 5-26. Sample code to create a Document object (1)

F1432.0

Notes:

Create Document objects

Sample code to create a Document object (2)

- Check in the document.

```
myDoc.checkin(AutoClassify.AUTO_CLASSIFY,
               CheckinType.MAJOR_VERSION);
```

- Set Properties to the document and Save the document.

```
myDoc.getProperties().putValue("DocumentTitle",
                                "New Model");
myDoc.setMimeType("image/gif");
myDoc.save(RefreshMode.REFRESH);
```

- File the document to a folder.

```
ReferentialContainmentRelationship rel =
    folder.file(myDoc, AutoUniqueName.AUTO_UNIQUE,
                "NewModel", DefineSecurityParentage.DO_NOT_
                    DEFINE_SECURITY_PARENTAGE);
rel.save(RefreshMode.NO_REFRESH);
```

© Copyright IBM Corporation 2011

Figure 5-27. Sample code to create a Document object (2)

F1432.0

Notes:

Create Document objects

Sample code to set multiple content elements

- Set Content data that is local to an object store.

```
... ContentTransfer content1 =
    Factory.ContentTransfer.createInstance();
content1.setCaptureSource(new
    FileInputStream("Winter.jpg"));
content1.set_ContentType("image/jpeg");
contentList.add(content1);
ContentTransfer content2 =
    Factory.ContentTransfer.createInstance();
content2.setCaptureSource(new
    FileInputStream("Spring.xml"));
content2.set_ContentType("text/xml");
contentList.add(content2);
```

- Set content elements to the document.

```
myDoc.set_ContentElements(contentList);
```

© Copyright IBM Corporation 2011

Figure 5-28. Sample code to set multiple content elements

F1432.0

Notes:

The steps to create a Document object with multiple content elements are similar to creating a Document object with a single content element.

The difference is that you are adding several content elements to the Content Element List.

Refer to the sample code for creating a document on the previous slides.

Create Document objects

Sample code to set content reference

- Create a reference for the content that exists outside of an object store.

```
ContentReference contentRef =  
    Factory.ContentReference.createInstance();  
contentRef.set_ContentLocation  
    ("http://www.ibm.com");  
contentRef.set_ContentType("text/html");  
contentList.add(contentRef);
```

- Set content to the document.

```
myDoc.set_ContentElements(contentList);
```

© Copyright IBM Corporation 2011

Figure 5-29. Sample code to set content reference

F1432.0

Notes:

Create Document objects

Demonstrations



- Create a Document object.
- Set content to the document and check in.
- Set properties to the document and save.
- File the document to a folder.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 5-30. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *createDocumentEDU(...)* method in the *DocumentsEDU.java* file.
3. Run the code and verify the results as instructed in the *Documents* unit in the Student Exercises book.

Create Document objects

Activities

In your Student Exercises book

- Unit: Documents
- Lesson: Create Document objects
- Activities
 - Create document objects.

© Copyright IBM Corporation 2011

Figure 5-31. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 5.2. Retrieve a document and its content

Lesson: Retrieve a document and its content

- 
- Why is this lesson important to you?
 - Your company employees need to access document objects to review their content. As their programmer, you are going to write code to retrieve a document object and its content when an employee accesses the document.

© Copyright IBM Corporation 2011

Figure 5-32. Lesson: Retrieve a document and its content

F1432.0

Notes:

Retrieve a document and its content

Activities that you need to complete



- Retrieve a document and its content.

© Copyright IBM Corporation 2011

Figure 5-33. Activities that you need to complete

F1432.0

Notes:

Retrieve a document and its content

Steps to retrieve a Document object



1. Get the Document object.

```
Factory.Document.fetchInstance(...)
```

2. Get the document name (optional).

```
document.getName()
```

© Copyright IBM Corporation 2011

Figure 5-34. Steps to retrieve a Document object

F1432.0

Notes:

Retrieve a document and its content

Steps to retrieve contents of a Document object

1. Get the document content elements.

```
document.get_ContentElements()
```

2. Get the Content type for the content element.

```
contentElement.get_ContentType()
```

3. Get the file name where the content data is stored.

```
contentTransfer.get_RetrievalName()
```

4. Get the content size for the document.

```
contentTransfer.get_ContentSize()
```

5. Get an input stream for reading the content data.

```
contentTransfer.accessContentStream()
```

© Copyright IBM Corporation 2011

Figure 5-35. Steps to retrieve contents of a Document object

F1432.0

Notes:

Retrieve a document and its content

Factory.Document.fetchInstance(...)

```
public static Document fetchInstance  
    (ObjectStore os, Id objectId,  
     PropertyFilter filter)
```

- Parameters

- os - Specify the ObjectStore object where this document is located
- Id - Specify GUID or folder path for the document
- filter - Specify a PropertyFilter object for which property values to return

- Returns

- The specified Document object

© Copyright IBM Corporation 2011

Figure 5-36. Factory.Document.fetchInstance(...)

F1432.0

Notes:

The folder path of the document consists of the folder name and the containment name of the document (not the document title).

If the value for the property filter is null, this method returns

- Values for all nonobject properties
- Placeholders for all object-valued properties

When the “add document” code is run more than once with the same document name, two documents are added to the same folder. The program does not display an error. However, an error is thrown when the document is retrieved using the path.

These documents with the same name can be retrieved using the GUID.

Retrieve a document and its content

document.get_Name()



```
String get_Name()
```

- Returns
 - A String representing the name of this object
- Notes
 - Represents the name for this object.
 - For most classes, this property is read-only.

© Copyright IBM Corporation 2011

Figure 5-37. document.get_Name()

F1432.0

Notes:

Retrieve a document and its content

document.get_ContentElements()



```
ContentElementList get_ContentElements()
```

- Returns
 - A ContentElementList object containing the list of content elements associated with this document.
- Notes
 - Each content element represents content data, which can be either of the following:
 - Local to an object store
 - External

© Copyright IBM Corporation 2011

Figure 5-38. document.get_ContentElements()

F1432.0

Notes:

Retrieve a document and its content

contentElement.get_ContentType()

```
String get_ContentType()
```

- Returns

- The value of the ContentType property as a MIME format string

© Copyright IBM Corporation 2011

Figure 5-39. contentElement.get_ContentType()

F1432.0

Notes:

MIME is a communications protocol that allows for the transmission of data in many forms.

Examples: audio, binary, or video

A MIME format string consists of a content type, a content subtype, and an optional parameter in the following format:

"MIME::content type/subtype[;parameter]"

Example: "MIME::text/html"

For more information on the supported MIME types, see "ContentType Property" in Content Engine .NET API help.

Retrieve a document and its content

document.get_ContentElementsPresent()



```
StringList get_ContentElementsPresent()
```

- Returns

- A StringList object containing the MIME type of each content element associated with this document at the time that it was last saved.

- Notes

- This method is similar to the one on the previous slide:

`contentElement.get_ContentType()`

- The previous method retrieves the MIME type for a given content element
 - This method retrieves the MIME type for all the content elements of the given document.

© Copyright IBM Corporation 2011

Figure 5-40. document.get_ContentElementsPresent()

F1432.0

Notes:

This property also returns the MIME type of the content elements (as does the method on the previous slide):

IContentElement.ContentType

The difference is that the previous property returns the MIME type for a given content element in a String.

This property retrieves the MIME type for all the content elements of a given document in a StringList collection.

Retrieve a document and its content

contentTransfer.get_RetrievalName()

```
String get_RetrievalName()
```

- Returns
 - The file path and file name from which the content data for this content element can be retrieved
- Notes
 - The path including the file name can contain up to 2083 characters.
 - The file name can be up to 255 characters long.

© Copyright IBM Corporation 2011

Figure 5-41. contentTransfer.get_RetrievalName()

F1432.0

Notes:

Retrieve a document and its content

contentTransfer.get_ContentSize()



```
Double get_ContentSize()
```

- Returns
 - The size (in bytes) of the content data associated with this ContentTransfer object

© Copyright IBM Corporation 2011

Figure 5-42. contentTransfer.get_ContentSize()

F1432.0

Notes:

Retrieve a document and its content

contentTransfer.accessContentStream()



```
java.io.InputStream accessContentStream()
```

- Returns
 - An inputStream object for reading content data
- Notes
 - This method uses an input stream to obtain read access to the content data.
 - If content data is not present in the object property cache, it is fetched from the server.

© Copyright IBM Corporation 2011

Figure 5-43. contentTransfer.accessContentStream()

F1432.0

Notes:

Retrieve a document and its content

Sample code to retrieve a document



- Retrieve a document.

```
Document document =  
    Factory.Document.fetchInstance  
(objectstore, "/Research/Model.doc", null);
```

- Retrieve a document name.

```
String documentName = doc.get_Name();
```

© Copyright IBM Corporation 2011

Figure 5-44. Sample code to retrieve a document

F1432.0

Notes:

Retrieve a document and its content

Sample code to retrieve content elements of a document

- Retrieve individual content elements.

```
ContentElementList contentElements =
    document.get_ContentElements();
ContentElement contentElement;
Iterator itContent =
    contentElements.iterator();
while (itContent.hasNext()){
    contentElement =
        (ContentElement) itContent.next();
```

- Retrieve the content type for the content element.

```
System.out.println("content type = " +
    contentElement.get_ContentType());
```

© Copyright IBM Corporation 2011

Figure 5-45. Sample code to retrieve content elements of a document

F1432.0

Notes:

Retrieve a document and its content

Sample code to retrieve content stream

- Get the file name where the content data is stored.

```
String fileName =  
    contentTransfer.get_RetrievalName();
```

- Get the content size for the document (optional).

```
Double contentSize =  
    contentTransfer.get_ContentSize();
```

- Get an input stream for reading the content data and flush out.

```
contentStream =  
    content.AccessContentStream()
```

```
...
```

© Copyright IBM Corporation 2011

Figure 5-46. Sample code to retrieve content stream

F1432.0

Notes:

Retrieve a document and its content

Demonstrations



- Retrieve a document and its content
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 5-47. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getDocumentEDU(...)* method in the *DocumentsEDU.java* file.
3. Run the code and verify the results as instructed in the *Documents* unit in the Student Exercises book.
4. Repeat steps 2 and 3 for the *getDocumentContentEDU (...)* method that gets objects from the folder.

Retrieve a document and its content

Activities



In your Student Exercises book

- Unit: Documents
- Lesson: Retrieve a document and its content
- Activities
 - Retrieve a document and its content.

© Copyright IBM Corporation 2011

Figure 5-48. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 6. Properties

What this unit is about

This unit describes how to retrieve property descriptions, choice list and property values.

What you should be able to do

After completing this unit, you should be able to:

- Retrieve property descriptions.
- Retrieve a choice list.
- Retrieve object properties.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Properties

Unit lessons

- 
- Retrieve property descriptions
 - Retrieve a choice list
 - Retrieve object properties

© Copyright IBM Corporation 2011

Figure 6-1. Unit lessons

F1432.0

Notes:

Some of the properties concepts, such as getting and setting values to multivalued properties and getting and setting DateTime data type, are discussed in the "Custom Objects" unit.

Lesson 6.1. Retrieve property descriptions

Lesson: Retrieve property descriptions

- 
- Why is this lesson important to you?
 - To manage the properties of the Content Engine objects, you need to retrieve the property descriptions. As a programmer, you are going to write code to retrieve and display the property descriptions of a given Document class.

© Copyright IBM Corporation 2011

Figure 6-2. Lesson: Retrieve property descriptions

F1432.0

Notes:

Retrieve property descriptions

Activities that you need to complete



- Retrieve property descriptions.

© Copyright IBM Corporation 2011

Figure 6-3. Activities that you need to complete

F1432.0

Notes:

Retrieve property descriptions

Properties defined



- Properties in an object store are
 - Individual values used to describe an object
 - Defined on a class and are inheritable by subclasses
- Definition of a property includes the following:
 - Data type: Scalar or object-valued
 - Cardinality: Single or multivalue
 - Settability: Read-only, read-write
 - Whether it is the “Name” property
 - If it has a ChoiceList
- Definition of a property is accessed using the PropertyDescription object of the API.

© Copyright IBM Corporation 2011

Figure 6-4. Properties defined

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Properties > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Properties > Concepts

Scalar data types:

- Binary
- Boolean
- DateTime
- Float
- ID

- Integer
- String

A single object-valued type is used to represent an object. For example, the annotations associated with a document can be represented as object type properties for the document.

Retrieve property descriptions

Setting and accessing property values



- Property values are saved on the server when the associated object is saved.
 - When first set, the properties are in memory
 - The object has to be saved for the properties to persist
- Properties can be retrieved for display and update in the following ways:
 - In a group, along with an object
 - By name, individually or collectively
 - In filtered sets
- Local property cache
 - Fetched from server into local cache
 - Accessed from there by your program

© Copyright IBM Corporation 2011

Figure 6-5. Setting and accessing property values

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Properties > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Properties > Concepts

Retrieve property descriptions

Property cache



- Temporary local store for object and property states
- Exists to reduce or eliminate round trips to the server
- Ways to add object properties to the property cache:
 - Call object refresh method for an existing object
 - Call fetch property or fetch properties method
 - Call object setter methods
- Retrieve from cache:
 - Individually or in collections

© Copyright IBM Corporation 2011

Figure 6-6. Property cache

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Properties > Concepts > Property Cache Concepts

To save system resources, each Content Engine object maintains a local (client-side) property cache from which one or more properties of an object, if available, can be retrieved.

The object refresh method pulls properties from the server. It replaces the object properties currently in cache and adds any new properties not in cache.

Fetch property or fetch properties methods merge with existing properties and add any new properties that are not in cache.

The system has an internal property called the update sequence number that is used to coordinate concurrent updates among different applications.

Retrieve property descriptions

Property Filters



- Purpose
 - Limit properties and property detail retrieved from server
 - Improve retrieval performance
- A property filter consists of
 - PropertyFilter object
 - Filter element represented by a FilterElement object
 - Each filter element represents a specification for adding properties
- Attributes of a PropertyFilter object
 - Act as global defaults for the PropertyFilter object
 - Defaults values can be overridden

© Copyright IBM Corporation 2011

Figure 6-7. Property Filters

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Properties > Concepts > Property Filter Concepts

A property filter allows you to control which properties of a Content Engine object (and with what level of detail) to return from the server during an object retrieval or an object refresh.

Because the number and size of properties can be large for certain objects, using a property filter to retrieve a subset of the available properties can result in better performance by reducing the amount of data that is retrieved from the server.

Retrieve property descriptions

Components of property filters object



- **IncludeProperty** specifications (zero or more)
 - Specifies, by identifier or **FilterElement**, properties to retrieve from the server
AddIncludeProperty(...)
- **IncludeType** specifications (zero or more)
 - Specifies, by property type or **FilterElement**, a class of properties to retrieve from the server
AddIncludeType(...)
- **ExcludeProperty** specifications (zero or more)
 - Specifies, by identifier, properties to be excluded from server retrieval
AddExcludeProperty(...)

© Copyright IBM Corporation 2011

Figure 6-8. Components of property filters object

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Properties > Concepts > Property Filter Concepts

Retrieve property descriptions

propertyFilter.addIncludeProperty(...)

```
public void addIncludeProperty(int  
    maxRecursion, maxSize, Boolean  
    levelDependents, String value, Integer  
    pageSize)
```

- Parameters

- maxRecursion - Specify the maximum allowable recursion depth for property retrieval
- maxSize - Specify the maximum size of content data to be retrieved
- levelDependents – Specify the recursion level
- value – Specify a space-separated list of the symbolic names or GUID-string identifiers of the properties to include
- pageSize - Specify the iterator page size for independent object sets returned

© Copyright IBM Corporation 2011

Figure 6-9. propertyFilter.addIncludeProperty(...)

F1432.0

Notes:

Parameters

levelDependents:

True if the recursion level is the same as that of the independent object to which it belongs

False if the recursion level is one level deeper

Retrieve property descriptions

propertyFilter.addExcludeProperty(...)

```
public void addExcludeProperty(String value)
```

- Parameters

- Value - Specify a space-separated list of the symbolic names in a String

- Notes

- Any properties specified for exclusion override those properties that are specified in one of the following:
 - An IncludeProperty
 - An IncludeType specification

© Copyright IBM Corporation 2011

Figure 6-10. propertyFilter.addExcludeProperty(...)

F1432.0

Notes:

Retrieve property descriptions

propertyFilter. addIncludeType(...)



```
public void addIncludeType(int maxRecursion,  
                           Long maxSize, Boolean levelDependents,  
                           FilteredPropertyType value,  
                           Integer pageSize)
```

- Parameters

- maxRecursion - Specify the maximum allowable recursion depth for property retrieval
- maxSize - Specify the maximum size of content data to be retrieved
- levelDependents – Specify the recursion level
- value – Specify a FilteredPropertyType constant specifying the type of the properties to include
- pageSize - Specify the iterator page size for independent object sets returned

© Copyright IBM Corporation 2011

Figure 6-11. propertyFilter. addIncludeType(...)

F1432.0

Notes:

Parameters

levelDependents:

True if the recursion level is the same as that of the independent object to which it belongs

False if the recursion level is one level deeper

Retrieve property descriptions

ClassDescription objects



- Are collections of metadata that describe individual classes
- Provide access to PropertyDescription objects associated with class
- Return default permission for an object of a particular class
- Determine if versioning is enabled for a particular Document class
- Assign a document lifecycle policy to a document class or subclass

© Copyright IBM Corporation 2011

Figure 6-12. ClassDescription objects

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Classes > Concepts > Class Descriptions

Retrieve property descriptions

PropertyDescription objects



- Are collections of metadata that describe individual properties
- Are contained in the ClassDescription PropertyDescriptions property
- Provide information on how the property needs to be represented to the user
- Contain information such as choice lists
- Each instance of a PropertyDescription object describes a specific property for a single class

© Copyright IBM Corporation 2011

Figure 6-13. PropertyDescription objects

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Properties > Concepts > Property Descriptions

When you create a new class (for example, a document class), the PropertyDescription that you choose is copied into the class.

After the class is created, if you modify that PropertyDescription, it is not automatically updated in the Document class. You need to delete the old PropertyDescription and add the new or modified one to the class.

After updating the class, only newly created objects (for example, documents) have the updated property.

Retrieve property descriptions

Steps to retrieve class descriptions



1. Get the object store.

```
Factory.ObjectStore.fetchInstance(...)
```

2. Get all the class descriptions for this object store.

```
objectStore.get_ClassDescriptions(...)
```

3. Iterate through the collection.

```
classDescriptions.iterator()  
iterator.next()
```

4. Get the name of each class description.

```
classDescription.get_DisplayName()
```

© Copyright IBM Corporation 2011

Figure 6-14. Steps to retrieve class descriptions

F1432.0

Notes:

Retrieve property descriptions

Steps to retrieve property descriptions

1. Retrieve property descriptions collection from the class.

```
classDescription.get_PropertyDescriptions()
```

2. Iterate through the collection.

```
propertyDescriptions.iterator()  
iterator.next()
```

3. Display individual property descriptions.

```
propertyDescription.get_SymbolicName()  
propertyDescription.get_IsReadOnly()  
propertyDescription.get_IsValueRequired()  
propertyDescription.get_Setability()  
propertyDescription.get_Cardinality()  
propertyDescription.get_IsSystemGenerated()
```

© Copyright IBM Corporation 2011

Figure 6-15. Steps to retrieve property descriptions

F1432.0

Notes:

Retrieve property descriptions

objectStore.get_ClassDescriptions()



```
ClassDescriptionSet get_ClassDescriptions()
```



- Returns
 - A ClassDescription collection for the Content Engine classes and subclasses
- Notes
 - The ClassDescriptions property returns a snapshot of the latest class metadata for a given object store.

© Copyright IBM Corporation 2011

Figure 6-16. objectStore.get_ClassDescriptions()

F1432.0

Notes:

Retrieve property descriptions

classDescription.get_PropertyDescriptions()



```
PropertyDescriptionList  
get_PropertyDescriptions()
```

- Returns

- An ordered collection containing all of the property descriptions for this ClassDescription object

© Copyright IBM Corporation 2011

Figure 6-17. classDescription.get_PropertyDescriptions()

F1432.0

Notes:

Retrieve property descriptions

propertyDescription.get_SymbolicName()



```
String get_SymbolicName ()
```

- Returns
 - A String that represents the programmatic identifier for this PropertyDescription

© Copyright IBM Corporation 2011

Figure 6-18. propertyDescription.get_SymbolicName()

F1432.0

Notes:

Retrieve property descriptions

propertyDescription.get_IsReadOnly()



```
Boolean get_IsReadOnly()
```

- Returns
 - A Boolean value that indicates whether or not you can modify the value of the property
 - If true, the property value can be changed only by the server.

© Copyright IBM Corporation 2011

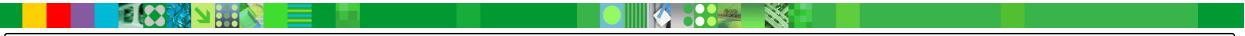
Figure 6-19. propertyDescription.get_IsReadOnly()

F1432.0

Notes:

Retrieve property descriptions

propertyDescription.get_IsValueRequired()



```
Boolean get_IsValueRequired()
```



- Returns

- A Boolean value that indicates whether the property is required to have a value (true) or not (false)

- Notes

- When you attempt to save an object containing a property that requires a value and if that property does not have a value assigned to it, an exception occurs.

© Copyright IBM Corporation 2011

Figure 6-20. propertyDescription.get_IsValueRequired()

F1432.0

Notes:

Retrieve property descriptions

propertyDescription.get_Setability()



PropertySetability get_Setability()

- Returns
 - A constant that indicates when the value of a property can be set

Name	Value
READ_ONLY	3
READ_WRITE	0
SETTABLE_ONLY_BEFORE_CHECKIN	1
SETTABLE_ONLY_ON_CREATE	2

© Copyright IBM Corporation 2011

Figure 6-21. propertyDescription.get_Setability()

F1432.0

Notes:

READ_ONLY indicates that a property is read-only. Only the server can set its value.

READ_WRITE indicates that a property is read/write. You can set its value at any time.

SETTABLE_ONLY_BEFORE_CHECKIN indicates that you can set the value of a property only before you check in the object to which it belongs.

SETTABLE_ONLY_ON_CREATE indicates that you can set the value of a property only when you create the object to which it belongs. After you save the object for the first time, the property value cannot be changed.

Retrieve property descriptions

propertyDescription.get_Cardinality()

Cardinality get_Cardinality()

- Returns
 - A constant that indicates the cardinality of the PropertyDescription

Cardinality name	Value
ENUM	1
LIST	2
SINGLE	0

© Copyright IBM Corporation 2011

Figure 6-22. propertyDescription.get_Cardinality()

F1432.0

Notes:

Retrieve property descriptions

propertyDescription.get_IsSystemGenerated()



```
Boolean get_IsSystemGenerated()
```

- Returns
 - A Boolean value that indicates whether the property has its value set automatically by the Content Engine server (true) or not (false)
 - If the value is true, the described property does one of the following:
 - Provides system information
 - Has its value determined by the server

© Copyright IBM Corporation 2011

Figure 6-23. propertyDescription.get_IsSystemGenerated()

F1432.0

Notes:

Retrieve property descriptions

Sample code to retrieve class descriptions

```
ClassDescriptionSet classDescriptions =  
    objectStore.get_ClassDescriptions();  
    ClassDescription classDesc;  
    Iterator it = classDescriptions.iterator();  
    while (it.hasNext()) {  
        classDesc = (ClassDescription)it.next();  
        System.out.println("Class name = " +  
                           classDesc.get_DisplayName());  
    }
```

© Copyright IBM Corporation 2011

Figure 6-24. Sample code to retrieve class descriptions

F1432.0

Notes:

Retrieve property descriptions

Sample code to retrieve a document

```
PropertyFilter pf = new PropertyFilter();
pf.addIncludeType(0, null, null,
                  Filtered.PropertyType.ANY,1);
Document document =Factory.Document.fetchInstance
    (store,"/APIFolder/APIOrderDoc", pf);
String documentName = document.get_Name();
System.out.println(documentName + " Document has
                           been retrieved");
return document;
```

© Copyright IBM Corporation 2011

Figure 6-25. Sample code to retrieve a document

F1432.0

Notes:

Retrieve property descriptions

Sample code to retrieve property descriptions

```
...
ClassDescription classDescription =
    document.get_ClassDescription();
PropertyDescriptionList propDescs =
    classDescription.get_PropertyDescriptions();
PropertyDescription propDesc;
Iterator propIt = propDescs.iterator();
while (propIt.hasNext()) {
    propDesc = PropertyDescription)propIt.next();
    System.out.println("Property name = " +
        propDesc.get_SymbolicName());
    System.out.println("Read only? " +
        propDesc.get_IsReadOnly().toString());
...
}
```

© Copyright IBM Corporation 2011

Figure 6-26. Sample code to retrieve property descriptions

F1432.0

Notes:

Retrieve property descriptions

Demonstrations



- Retrieve class descriptions.
- Retrieve a document using property filter.
- Retrieve property descriptions.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 6-27. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getClassesEDU(...)* method that retrieves class descriptions in the *PropertiesEDU.java* file.
3. Run the code and verify the results as instructed in the *Properties* unit in the Student Exercises.
4. Repeat steps 2 and 3 for the *getDocumentEDU (...)* method that gets the document you requested.
5. Repeat steps 2 and 3 for the *getPropertyDescriptionsEDU(...)* method that retrieves property descriptions.

Retrieve property descriptions

Activities



In your Student Exercises book

- Unit: Properties
- Lesson: Retrieve property descriptions
- Activities
 - Retrieve property descriptions.

© Copyright IBM Corporation 2011

Figure 6-28. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 6.2. Retrieve a choice list

Lesson: Retrieve a choice list

- 
- Why is this lesson important to you?
 - The custom application provides its customers with payment options when ordering a cruise. A choice list for the payment options has been created on the Content Engine using IBM FileNet Enterprise Manager. As the programmer, you are going to write code to retrieve and display the choice list of payment options.

© Copyright IBM Corporation 2011

Figure 6-29. Lesson: Retrieve a choice list

F1432.0

Notes:

Retrieve a choice list

Activities that you need to complete



- Retrieve a choice list.

© Copyright IBM Corporation 2011

Figure 6-30. Activities that you need to complete

F1432.0

Notes:

Retrieve a choice list

Choice list objects

- Each choice list object
 - Is typically created using IBM FileNet Enterprise Manager
 - Contains a list of choices from which users select a value
- Elements in the collection are Choice objects that
 - Represent a set of choices for a settable property
 - Can be hierarchical: A Choice element can itself contain a collection of choices



© Copyright IBM Corporation 2011

Figure 6-31. Choice list objects

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Choice lists > Concepts

It is also possible to create a choice list with a custom program written using the API. This lesson describes how to retrieve the choice list.

Retrieve a choice list

Using choice lists



- Usage requirements
 - A choice list must be assigned to a property template.
 - A choice list must use either a string or an integer data type.
 - A choice list data type must match the data type for its associated property template.
- Usage options
 - One choice list can be assigned to multiple property templates.

© Copyright IBM Corporation 2011

Figure 6-32. Using choice lists

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Choice lists > Concepts

Retrieve a choice list

ChoiceList interface



- The API has two versions of the ChoiceList interface:
 - `com.filenet.api.admin.ChoiceList`
 - Has the methods needed to access choice lists
 - `com.filenet.api.collection.ChoiceList`
 - Defines the ChoiceList collection returned by `com.filenet.api.admin.get_ChoiceValues()`
 - Provides access to individual choices

© Copyright IBM Corporation 2011

Figure 6-33. ChoiceList interface

F1432.0

Notes:

Retrieve a choice list

Choice objects

- Choice objects (choices)
 - Represent possible values of a property
 - Can be String or Integer data type
 - Can provide access to a nested choice list
- The Content Engine API
 - Includes methods to
 - Retrieve a ChoiceList from an IPropertyDescription
 - Check for a ChoiceList collection contained in a Choice object
 - Return the contained ChoiceList collection if found

© Copyright IBM Corporation 2011

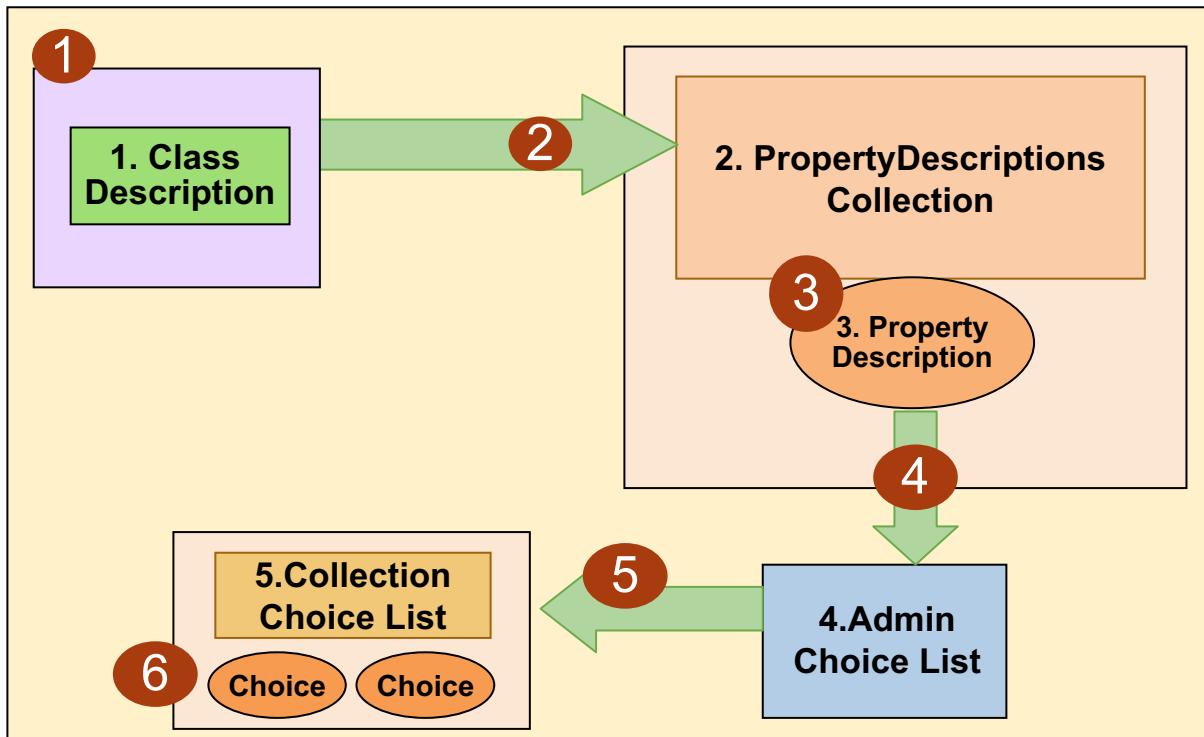
Figure 6-34. Choice objects

F1432.0

Notes:

Retrieve a choice list

Steps to retrieve a choice list and choices

© Copyright IBM Corporation 2011

Figure 6-35. Steps to retrieve a choice list and choices

F1432.0

Notes:

1. Retrieve the class description for a given Document class from the Document object.
2. Retrieve the property description collection from the class description.
3. Retrieve the individual property description from the collection.
4. Retrieve the choice list from the property description.
5. Retrieve the choice list collection from the admin choice list.
6. Retrieve individual choices from the choice list collection.

Retrieve a choice list

Steps to retrieve a choice list

1. Get the class description for the specified class.

```
document.get_ClassDescription()
```

2. Get the property descriptions collection for that class.

```
classDescription.get_PropertyDescriptions()
```

3. Get an individual property description.

```
propertyDescriptionList.iterator()  
iterator.next()
```

4. Get the choice list from the property description.

```
propDescription.get.ChoiceList()
```

© Copyright IBM Corporation 2011

Figure 6-36. Steps to retrieve a choice list

F1432.0

Notes:

Retrieve a choice list

Steps to retrieve individual choices

- 
5. Get the Choice collection.

```
choiceList.get.ChoiceValues()
```

6. Get the individual choice and its value.

```
choice.get.ChoiceStringValue()  
choice.get.ChoiceIntegerValue().toString()
```

7. Check to see whether the choice list has hierarchy.

Repeat step 6 in a recursive method to get the nested choice list.

```
choiceList.get.HasHierarchy()
```

© Copyright IBM Corporation 2011

Figure 6-37. Steps to retrieve individual choices

F1432.0

Notes:

Retrieve choice list

propertyDescription.get.ChoiceList(...)



```
ChoiceList get.ChoiceList()
```



- Returns
 - A ChoiceList collection that contains the Choice objects for this PropertyDescription object
- Notes
 - A ChoiceList object can have nested ChoiceList objects.

© Copyright IBM Corporation 2011

Figure 6-38. propertyDescription.get.ChoiceList(...)

F1432.0

Notes:

Retrieve choice list

choiceList.get_ChoiceValues()



```
ChoiceList get_ChoiceValues()
```



- Returns
 - A ChoiceList collection object representing a set of allowable values
- Notes
 - The ChoiceValues define the set of possible values in a choice list group.
 - This ChoiceList class belongs to the com.filenet.api.collection package.

© Copyright IBM Corporation 2011

Figure 6-39. choiceList.get_ChoiceValues()

F1432.0

Notes:

Retrieve choice list

choiceList.get_HasHierarchy()



```
Boolean get_HasHierarchy()
```

- Returns

- A Boolean value that specifies whether a given choice list has a hierarchical structure (true) or not (false).

© Copyright IBM Corporation 2011

Figure 6-40. choiceList.get_HasHierarchy()

F1432.0

Notes:

Retrieve choice list

choice.get.ChoiceType()



```
ChoiceType get.ChoiceType()
```

- Returns
 - A ChoiceType constant that indicates the type of data that a Choice object represents
- Notes
 - Examples of ChoiceType constants:
 - **ChoiceType.INTEGER**
 - **ChoiceType.STRING**

© Copyright IBM Corporation 2011

Figure 6-41. choice.get.ChoiceType()

F1432.0

Notes:

INTEGER

Indicates a Choice object that holds an Integer data type.

STRING

Indicates a Choice object that holds a String data type.

MIDNODE_STRING

Indicates a Choice object that acts as a group node for Choice objects that hold String data types.

MIDNODE_INTEGER

Indicates a Choice object that acts as a group node for Choice objects that hold Integer data types.

Retrieve choice list

choice.get_ChoiceIntegerValue()

```
Integer get_ChoiceIntegerValue()
```

- Returns

- The value of an Choice object that holds an Integer data type in a choice list

© Copyright IBM Corporation 2011

Figure 6-42. choice.get_ChoiceIntegerValue()

F1432.0

Notes:

Retrieve choice list

choice.get.ChoiceStringValue()



```
String get.ChoiceStringValue()
```

- The value of a Choice object that holds a String data type in a choice list

© Copyright IBM Corporation 2011

Figure 6-43. choice.get.ChoiceStringValue()

F1432.0

Notes:

Retrieve a choice list

Sample code to retrieve a choice list

```
...
com.filenet.api.admin.ChoiceList choiceList
    = propDescription.get.ChoiceList();
System.out.println("ChoiceList name = "+
    propDescription.get.SymbolicName());
com.filenet.api.collection.ChoiceList choices =
    choiceList.get.ChoiceValues();
Choice choice;
Iterator choiceIt = choices.iterator();
```

© Copyright IBM Corporation 2011

Figure 6-44. Sample code to retrieve a choice list

F1432.0

Notes:

The code for the `displayChoiceList(choices)` method is given on the next page.

Retrieve a choice list

Sample code to retrieve the individual choices

```
while (choiceIt.hasNext()) {  
    choice = (Choice)choiceIt.next();  
    if (choice.get.ChoiceType().equals  
        (ChoiceType.INTEGER)) {  
        Integer choiceInt =  
            choice.get.ChoiceIntegerValue();  
        System.out.println("choice = " +  
                           choiceInt.toString());  
    }  
    else{  
        String choiceString =  
            choice.get.ChoiceStringValue();  
        System.out.println("choice = " +  
                           choiceString);  
    }  
}
```

© Copyright IBM Corporation 2011

Figure 6-45. Sample code to retrieve the individual choices

F1432.0

Notes:

The code represents the `displayChoiceList(choices)` method. This method retrieves individual choice values. Also, this method is used for recursive to retrieve any nested choice list associated with the choice object.

Retrieve a choice list

Demonstrations



- Retrieve a choice list.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 6-46. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

- Start Eclipse and open the *CMJavaAPISolution* project.
- Review the code for the *getChoiceListEDU(...)* method in the *PropertiesEDU.java* file.
- Run the code and verify the results as instructed in the *Properties* unit in the Student Exercises book.

Retrieve a choice list

Activities



In your Student Exercises book

- Unit: Properties
- Lesson: Retrieve a choice list
- Activities
 - Retrieve a choice list.

© Copyright IBM Corporation 2011

Figure 6-47. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 6.3. Retrieve object properties

Lesson: Retrieve object properties



- Why is this lesson important to you?
 - To manage the properties of the Content Engine objects, you need to retrieve the properties for a given object. As a programmer, you are going to write code to retrieve and display the values of properties of a given Document class.

© Copyright IBM Corporation 2011

Figure 6-48. Lesson: Retrieve object properties

F1432.0

Notes:

Retrieve object properties

Activities that you need to complete



- Retrieve object properties.

© Copyright IBM Corporation 2011

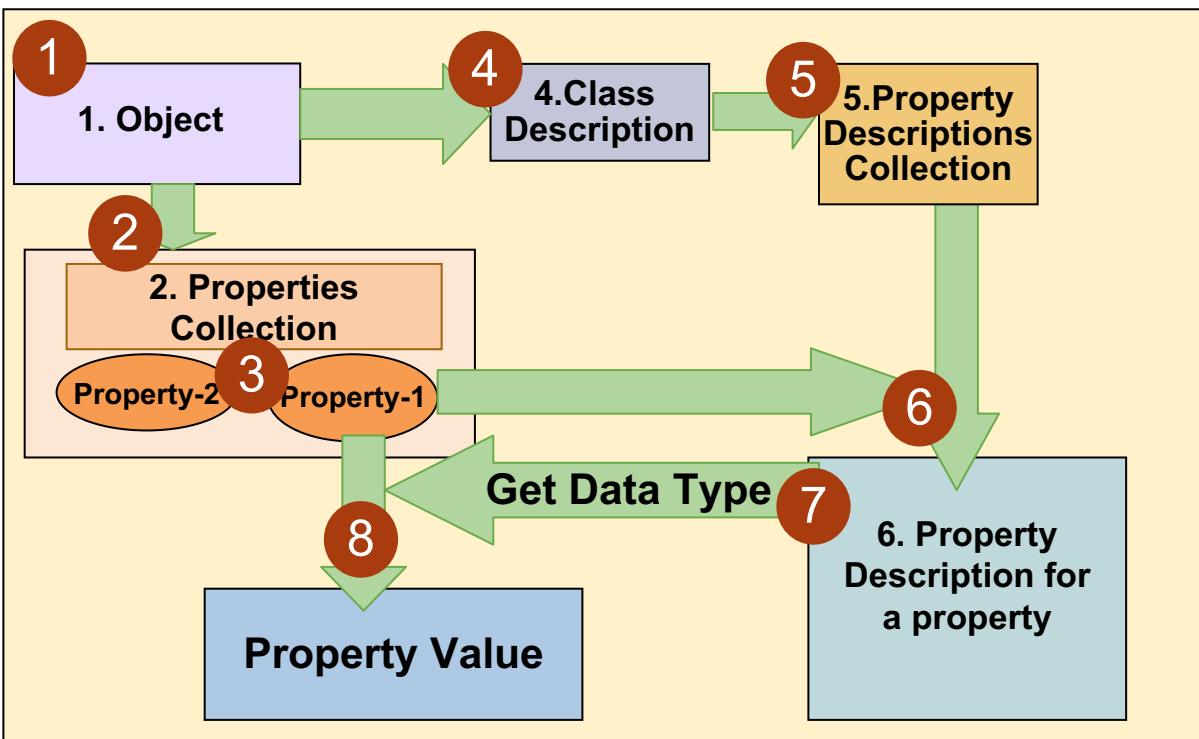
Figure 6-49. Activities that you need to complete

F1432.0

Notes:

Retrieve object properties

Steps to retrieve a given property value



© Copyright IBM Corporation 2011

Figure 6-50. Steps to retrieve a given property value

F1432.0

Notes:

1. Retrieve the object.
2. Retrieve the property collection from the object.
3. Retrieve individual properties from the collection
4. Retrieve the class description from the target object.
5. Retrieve the property descriptions from the class description.
6. Retrieve the property description from the collection for the given property, using the property name.
7. Retrieve the data type for the property from the property description.
8. Retrieve the value for the property based on the data type.

Note: Steps 4, 5, 6, and 7 are optional if you know the data type for the property.

Retrieve object properties

Steps to retrieve a property description for a property

1. Get the Document.

```
Factory.Document.fetchInstance(...)
```

2. Get the properties collection for this Document.

```
document.getProperties()
```

3. Get the individual property from collection.

```
properties.iterator()  
iterator.next()  
property.getPropertyName()
```

4. Get the document class description.

```
document.get_ClassDescription();
```

5. Get the property descriptions.

```
classDescription.getPropertyDescriptions();
```

© Copyright IBM Corporation 2011

Figure 6-51. Steps to retrieve a property description for a property

F1432.0

Notes:

Retrieve object properties

Steps to retrieve a property value

6. Get the property description for the given property.

```
propertyDescription.get_SymbolicName()
```

7. Get the data type for the property.

```
propertyDescription.get_Cardinality()
```

8. Get the value for the property after checking each data type.

```
if(propertyDesc.get_Cardinality().getValue()  
== Cardinality.SINGLE_AS_INT)  
    property.getDate TValue()  
...  
...
```

© Copyright IBM Corporation 2011

Figure 6-52. Steps to retrieve a property value

F1432.0

Notes:

Retrieve object properties

propertyDescription.get_DataType()

TypeID get_DataType()

- Returns
 - The data type of the value of a property or choice list this PropertyDescription represents.
- Possible values for the TypeID constants:
 - TypeID.BINARY
 - TypeID.BOOLEAN
 - TypeID.DATE
 - TypeID.DOUBLE
 - TypeID.GUID
 - TypeID.LONG
 - TypeID.OBJECT
 - TypeID.STRING

© Copyright IBM Corporation 2011

Figure 6-53. propertyDescription.get_DataType()

F1432.0

Notes:

Retrieve object properties

property.getStringValue()



```
String getStringValue()
```

- Returns
 - A String specifying the property value

© Copyright IBM Corporation 2011

Figure 6-54. property.getStringValue()

F1432.0

Notes:

Similar methods for the other data types:

DATE

GetDateTimeValue()

BOOLEAN

GetBooleanValue()

DOUBLE

GetFloat64Value()

LONG

GetInteger32Value()

GUID

GetIdValue()

Retrieve object properties

property.getStringListValue()



```
StringList getStringListValue()
```

- Returns
 - A collection containing all the values for this property in a StringList object

© Copyright IBM Corporation 2011

Figure 6-55. property.getStringListValue()

F1432.0

Notes:

Similar methods for the other data types:

DATE

GetDateTimeListValue()

BOOLEAN

GetBooleanListValue()

DOUBLE

GetFloat64ListValue()

LONG

GetInteger32ListValue()

GUID

GetIdListValue()

Retrieve object properties

Sample code to retrieve object properties

```
com.filenet.api.property.Properties  
properties = document.getProperties();  
Property property;  
Iterator it = properties.iterator();  
while (it.hasNext())  
{  
    property = (Property) it.next();  
    displayPropertyEDU(document, property);  
}
```

Note: The displayPropertyEDU(...) method refers to the code that is shown on next two pages.

© Copyright IBM Corporation 2011

Figure 6-56. Sample code to retrieve object properties

F1432.0

Notes:

Retrieve object properties

Sample code to retrieve a property value (1)

```
public void displayPropertyEDU(Document document,
    Property property) {
    String propertyName = property.getPropertyName();
    ClassDescription classDescription =
        document.get_ClassDescription();
    PropertyDescriptionList propDescs =
        classDescription.get_PropertyDescriptions();
    Iterator propIt = propDescs.iterator();
    PropertyDescription propertyDescription = null;
    while (propIt.hasNext()) {
        propertyDescription = (PropertyDescription)
            propIt.next();
        String symbolicName =
            propertyDescription.get_SymbolicName();
        if (symbolicName.equalsIgnoreCase(propertyName)) {
...    Continued on the next page
```

© Copyright IBM Corporation 2011

Figure 6-57. Sample code to retrieve a property value (1)

F1432.0

Notes:

Retrieve object properties

Sample code to retrieve a property value (2)

```
...
System.out.println("Property name = " +
                     symbolicName);
switch(propertyDescription.get_DataType()
                     .getValue())
{
    case TypeID.STRING_AS_INT:
        if (propertyDescription.get_Cardinality()
            .getValue() == Cardinality.SINGLE_AS_INT)
            System.out.println(property.getStringValue
                               ());
    else
        displayListValuesEDU
            (property.getStringListValue());
    break;
...
}
```

© Copyright IBM Corporation 2011

Figure 6-58. Sample code to retrieve a property value (2)

F1432.0

Notes:

This sample code represents only the String data type. Refer to the solution code for the complete `displayPropertyEDU(...)` method containing all the data types.

The `displayListValuesEDU(...)` method refers to the code that is shown in the next slide.

Retrieve object properties

Sample code to retrieve a property value (3)

```
public void displayListValuesEDU(List list) {  
    Iterator listIt = list.iterator();  
    Object value;  
    while (listIt.hasNext())  
    {  
        value = (Object)listIt.next();  
        System.out.println(value.toString());  
  
    }  
}
```

© Copyright IBM Corporation 2011

Figure 6-59. Sample code to retrieve a property value (3)

F1432.0

Notes:

Retrieve object properties

Demonstrations



- Retrieve object properties.
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 6-60. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *displayPropertiesEDU(...)*, *displayPropertyEDU(...)* and *displayListValuesEDU(...)* methods in the *PropertiesEDU.java* file.
3. Run the code and verify the results as instructed in the *Properties* unit in the Student Exercises.

Retrieve object properties

Activities

In your Student Exercises book

- Unit: Properties
- Lesson: Retrieve object properties
- Activities
 - Retrieve object properties.

© Copyright IBM Corporation 2011

Figure 6-61. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 7. Searches

What this unit is about

This unit describes how to search for the Content Engine objects based on a property and based on document content.

What you should be able to do

After completing this unit, you should be able to:

- Search for objects.
- Search for documents with paging
- Search for objects across multiple object stores.
- Build SQL statements.
- Content-based retrieval

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Searches

Unit lessons



- Search for objects
- Search for documents with paging
- Search for objects across multiple object stores
- Build SQL statements
- Content-based retrieval

© Copyright IBM Corporation 2011

Figure 7-1. Unit lessons

F1432.0

Notes:

Lesson 7.1. Search for objects

Lesson: Search for objects

- 
- Why is this lesson important to you?
 - Your application requires a feature that searches for Document objects on the given Content Engine. You are going to write code to accomplish this.

© Copyright IBM Corporation 2011

Figure 7-2. Lesson: Search for objects

F1432.0

Notes:

Search for objects

Activities that you need to complete



- Search for objects.

© Copyright IBM Corporation 2011

Figure 7-3. Activities that you need to complete

F1432.0

Notes:

Search for objects

Overview of searches



- Search for documents, custom objects, folders, and other types based on properties.
- Search for documents based on content.
- Search one or more object stores at a time.
- Search using SQL statements (with content extensions for documents).
- Search results can be collections of objects or properties.

© Copyright IBM Corporation 2011

Figure 7-4. Overview of searches

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries >Concepts

Search for objects

Components of a query



- Query criteria
 - What are you searching for?
 - Objects, properties, class definitions (SELECT)
 - Where do you want to look for it?
 - Which object store or stores
 - Which table (FROM)
 - How do you want to find it?
 - Properties or content
- Query results
 - How do you want results returned?
 - Independent objects
 - Property rows

© Copyright IBM Corporation 2011

Figure 7-5. Components of a query

F1432.0

Notes:

Search for objects

Searches



- The essential parts of a Content Engine search:
 - An SQL statement contained in a SearchSQL instance
 - A SearchScope object that contains which object store or object stores are searched

© Copyright IBM Corporation 2011

Figure 7-6. Searches

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries > Concepts

Search for objects

SearchScope class



- SearchScope Class
 - Used to launch the search
 - Determines which object store or stores to be searched
 - Can search one or more object stores using a single query
 - Returns three kinds of result sets:
 - Subclasses of EngineObjects (for example, Documents)
 - Properties objects
 - ClassDescription objects (class metadata)
- SearchScope Methods
 - Execute the SQL statement on one or more object stores
 - SearchScope(ObjectStore objectStore)
 - SearchScope(ObjectStore[] objectStores, MergeMode mergeMode)
 - MergeMode to specify union or intersection of object stores

© Copyright IBM Corporation 2011

Figure 7-7. SearchScope class

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries >Concepts >The Search Scope

The SearchScope methods execute the SQL statement on one or more object stores to find objects (IndependentObject instances), database rows (RepositoryRow instances), or metadata (ClassDescription instances).

Search for objects

SearchSQL



- SearchSQL is a helper class
 - Used to assist in building valid SQL statements
 - Used to pass an instance of this class into the SearchScope.fetchObjects method to perform a query

© Copyright IBM Corporation 2011

Figure 7-8. SearchSQL

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries >Concepts

There are helper methods on the SearchSQL class to assist you in constructing an SQL statement. The next lesson discusses these methods.

Alternatively, you can construct an SQL statement independently and pass it to a SearchSQL instance as a string.

SQL query syntax needs to follow IBM FileNet standard, which generally conforms SQL-92 with extensions for IBM FileNet-specific constructs.

Search for objects

Fetching the result set – contents

- Contents of result set can be the following:
 - [IndependentObjectSet](#)
 - Collection of EngineObject objects – object references
 - [RepositoryRowSet](#)
 - Collection of Properties objects – property rows

© Copyright IBM Corporation 2011

Figure 7-9. Fetching the result set - contents

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries >Concepts

Search for objects

Steps to search for objects based on the property



1. Instantiate a SearchScope object with the Objectstore to search from.

```
new SearchScope(...)
```

2. Instantiate a SearchSQL object with the SQL statement.

```
new SearchSQL(...)
```

3. Retrieve the objects.

```
SearchScope.fetchObjects(...)
```

© Copyright IBM Corporation 2011

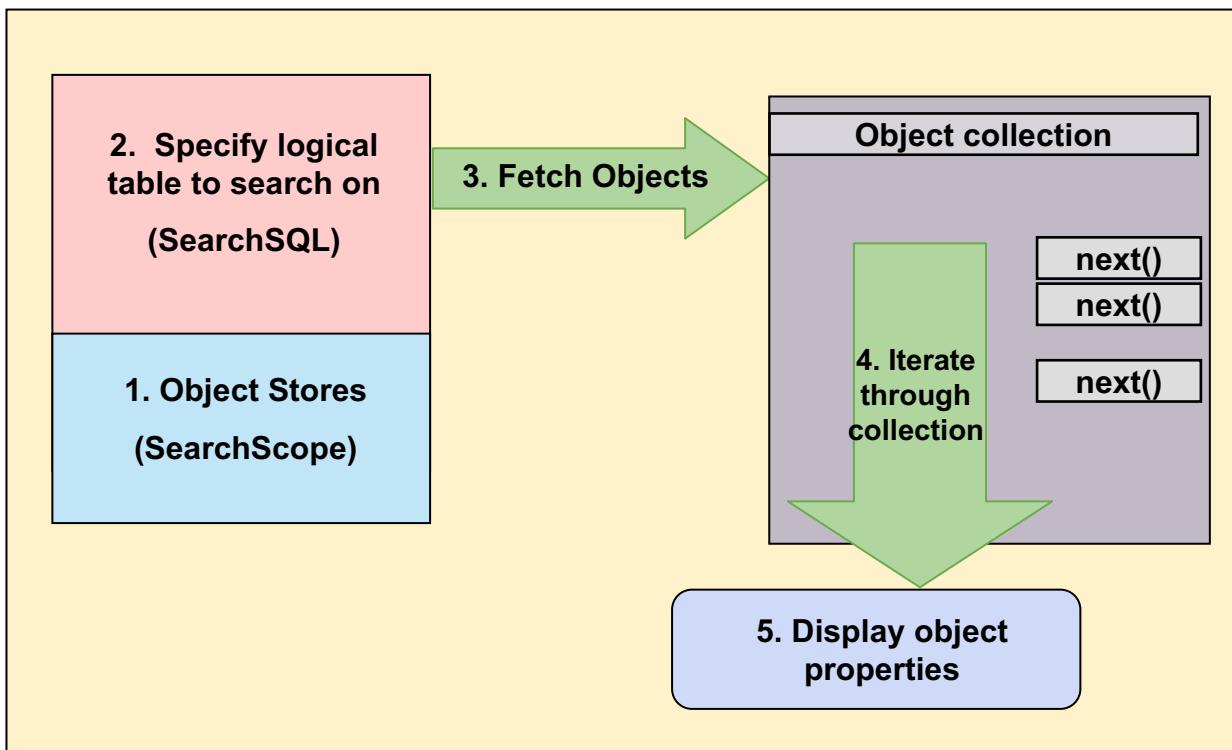
Figure 7-10. Steps to search for objects based on the property

F1432.0

Notes:

Search for objects

Steps to search for objects



© Copyright IBM Corporation 2011

Figure 7-11. Steps to search for objects

F1432.0

Notes:

Steps to search for objects:

1. SearchScope object defines the object store or object stores to search from.
2. SQL statement that is passed in as a parameter defines which table and criteria to retrieve.
3. SearchScope.fetchObjects() method retrieves the collection of objects meeting search criteria.
4. Iterate through object collection to retrieve each object.
5. Retrieve and display the object name and its properties.

Search for objects

SearchScope.fetchObjects(...)

```
public IndependentObjectSet fetchObjects(SearchSQL  
    searchSQL, Integer pageSize, PropertyFilter  
    filter, Boolean continuable)
```

- Parameters

- searchSQL – Specify a SearchSQL instance containing the SQL statement to use for the search.
- pageSize - An integer indicating the maximum number of objects per page to retrieve. This integer can be null.
- filter - A PropertyFilter object that represents information for controlling which property values to return and with what level of detail and recursion.
- continuable - A Boolean value. If false or null, the query is not paged.

- Returns

- IndependentObject objects collection from the object store or stores

© Copyright IBM Corporation 2011

Figure 7-12. SearchScope.fetchObjects(...)

F1432.0

Notes:

Search for objects

Sample code to search for documents based on property

```
SearchScope search = newSearchScope(objectStore);
String sql1 = "Select * from Product where
              Creator='P8Admin'";
SearchSQL searchSQL = new SearchSQL(sql1);
DocumentSet documents = (DocumentSet)
    search.fetchObjects
    (searchSQL, Integer.valueOf("50"),
     null, Boolean.valueOf(true));
Document doc;
Iterator DocIt = documents.iterator();
while (DocIt.hasNext()){
    doc = (Document)DocIt.next();
    System.out.println("document="+
    doc.getName());
}
```

© Copyright IBM Corporation 2011

Figure 7-13. Sample code to search for documents based on property

F1432.0

Notes:



Search for objects

Additional sample SQL statements for searching objects

Searching for subfolders for a given parent folder

```
"SELECT * from Folder WHERE
    Parent =OBJECT('/Products')"
```

Searching for documents for a given parent folder

```
"SELECT D.DocumentTitle from Product D INNER JOIN
    ReferentialContainmentRelationship r ON D.This =
    r.Head WHERE r.Tail = OBJECT('/Manuals')"
```

© Copyright IBM Corporation 2011

Figure 7-14. Additional sample SQL statements for searching objects

F1432.0

Notes:

Optional search for folders

...

```
String sqlFolder = "Select * from Folder WHERE Parent=OBJECT('/Products')";
SearchSQL searchSQL = new SearchSQL(sqlFolder); FolderSet folders;
folders = (FolderSet)search.fetchObjects(searchSQL, Integer.valueOf("50"), null,
Boolean.valueOf(true));

Folder folder;
Iterator folderIt = folders.iterator();
while (folderIt.hasNext()){
    folder = (Folder)folderIt.next();
    System.out.println("Folder= "+ folder.get_FolderName());
}
```

Search for objects

objectStore.set_DefaultQueryTimeLimit(...)

```
void set_DefaultQueryTimeLimit(  
    Integer timeLimit)
```

- Parameters

- timeLimit - An integer specifying the maximum duration of the query

- Notes

- This method sets the default value (in seconds) for the maximum amount of time that the server allows a query to run before canceling the query.
 - This default value cannot be overridden by a query.

© Copyright IBM Corporation 2011

Figure 7-15. objectStore.set_DefaultQueryTimeLimit(...)

F1432.0

Notes:

This default value cannot be overridden by a query.

Search for objects

serverCacheConfiguration.set_NonPagedQueryMaxSize(...)



```
void set_NonPagedQueryMaxSize( Integer maxSize )
```

- This method sets the maximum size of a non-paged query result set.

© Copyright IBM Corporation 2011

Figure 7-16. serverCacheConfiguration.set_NonPagedQueryMaxSize(...)

F1432.0

Notes:

Search for objects

Tips for searches



- Use the Enterprise Manager QueryBuilder tool to construct the query or validate the query.
 - Alternatively, use SearchSQL helper methods.
- Limit rows returned.
- Avoid searching or ordering on non-indexed columns.
- Avoid unnecessary object-type searches.
- Avoid unnecessary column returns (in SELECT clause).
- Avoid unnecessary row ordering (especially nested ordering).
- Avoid Complex Table Linkages.
- Avoid Subqueries (Oracle).

© Copyright IBM Corporation 2011

Figure 7-17. Tips for searches

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries > Working with Queries> Best Practices for Searches

Using the Enterprise Manager's Query Builder tool:

Include the object reference "this" in the SELECT clause when you execute queries. in Query Builder > View > SQL View.

For example, this query:

`SELECT d.Id FROM Document d WHERE d.DocumentTitle = 'MyDoc'` must be entered into Query Builder in this form:

`SELECT d.this, d.Id FROM Document d WHERE d.DocumentTitle = 'MyDoc'`

Limit Rows Returned

To restrict the result set to a specific class type (excluding subclasses), use the EXCLUDESUBCLASSES operator.

The default is INCLUDESUBCLASSES.

Avoid Subqueries (Oracle)

Avoid using subqueries and operators that indirectly generate subqueries whenever you are querying an object store that uses Oracle as the database engine. Specifically, use an INNER JOIN instead of a subquery.

Example:

Rewrite the following query, SELECT d.id FROM Document d WHERE d.This INFOLDER '/sub1/sub1a'

in this more efficient form: SELECT d.Id FROM Document d INNER JOIN ReferentialContainmentRelationship r ON d.This = r.Head WHERE r.Tail = OBJECT('/sub1/sub1a')

Search for objects

Demonstrations



- Searching for documents based on property
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 7-18. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *searchDocumentsEDU(...)* method in the *SearchEDU.java* file.
3. Run the code and verify the results as instructed in the *Searches* unit in the Student Exercises book.

Search for objects

Activities



In your Student Exercises book

- Unit: Searches
- Lesson: Search for objects
- Activities
 - Search for objects.

© Copyright IBM Corporation 2011

Figure 7-19. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 7.2. Search for documents with paging

Lesson: Search for documents with paging

- 
- Why is this lesson important to you?
 - Your application requires a feature that searches for Document objects on the given Content Engine and returns documents as sets per page. You are going to write code to accomplish this.

© Copyright IBM Corporation 2011

Figure 7-20. Lesson: Search for documents with paging

F1432.0

Notes:

Search for documents with paging

Activities that you need to complete



- Search for documents with paging.

© Copyright IBM Corporation 2011

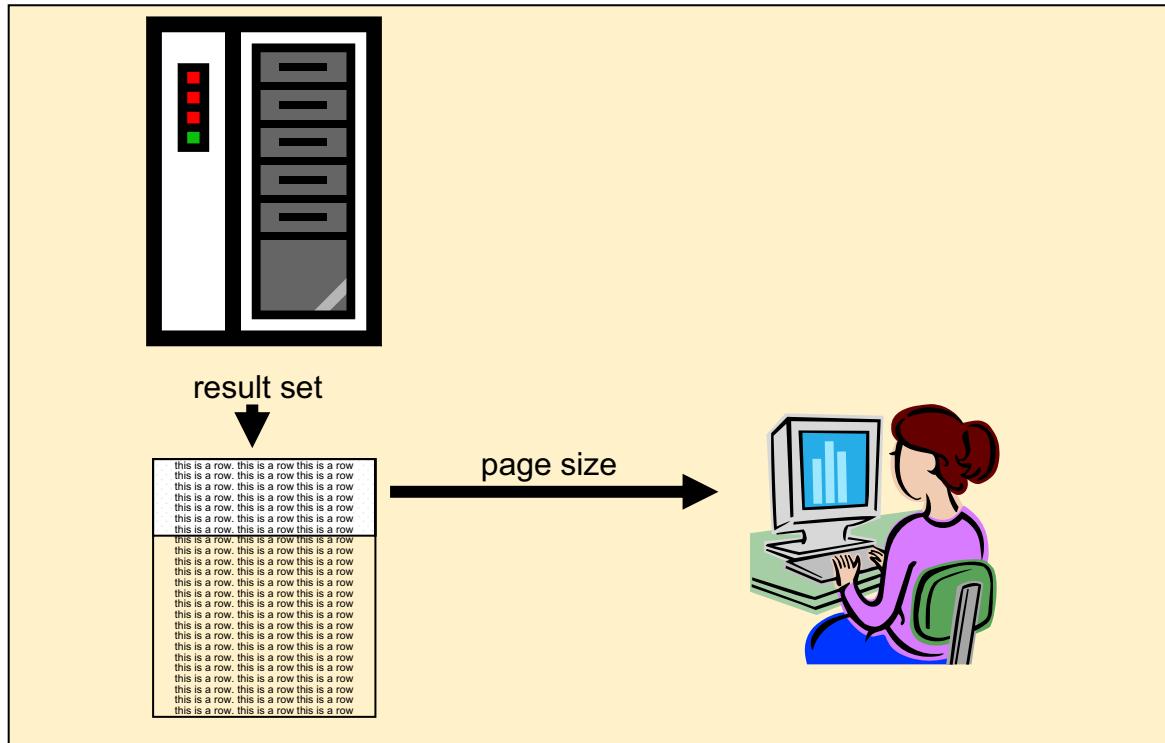
Figure 7-21. Activities that you need to complete

F1432.0

Notes:

Search for documents with paging

Results sets and paging



© Copyright IBM Corporation 2011

Figure 7-22. Results sets and paging

F1432.0

Notes:

Search for documents with paging

Fetching the result set – controls

- For performance reasons, limits can be set in code or at the server.
- Set paging to control row size of page and whether results are paged.
 - SearchScope has pageSize parameter
 - ServerCacheConfiguration.NonPagedQueryMaxSize Property
 - ServerCacheConfiguration.QueryPageSize Property
 - ServerCacheConfiguration.QueryPageMaxSize Property
- Set limits inside the SQL statement.
 - SearchSQL.SetMaxRecords – maximum return rows
 - SearchSQL.SetTimeLimit – maximum seconds query can run

© Copyright IBM Corporation 2011

Figure 7-23. Fetching the result set - controls

F1432.0

Notes:

Search for documents with paging

documentset.pageliterator()



PageIterator pageIterator()

- Returns
 - A Pageliterator object for iterating through a set-type collection.
- Notes
 - A set is a collection of independent objects, the elements of which are unordered and unique.
 - You can not directly update a set.
 - You can iterate a set one page at a time (instead of one object or one row at a time).

© Copyright IBM Corporation 2011

Figure 7-24. documentset.pageliterator()

F1432.0

Notes:

Search for documents with paging

Sample code to search for documents with paging

```
DocumentSet documents= (DocumentSet) search.fetchObjects  
    (sql, Integer.valueOf("5"), null,  
     Boolean.valueOf(true));  
PageIterator pageIter = documents.pageIterator();  
pageIter.getPageSize();  
pageIter.setPageSize(5);  
int pageCount = 0;  
while (pageIter.nextPage() == true) {  
    pageCount++;  
    int elementCount = pageIter.getElementCount();  
    Object[] pageObjects =  
        pageIter.getCurrentPage();  
    for (int index=0; index<pageObjects.length;  
         index++) {  
        Document document = (Document)pageObjects[index];  
        System.out.println("document = "+  
                           document.get_Name());
```

© Copyright IBM Corporation 2011

Figure 7-25. Sample code to search for documents with paging

F1432.0

Notes:

Search for documents with paging

serverCacheConfiguration.set_QueryPageMaxSize(...)



```
void set_QueryPageMaxSize(Integer maxSize)
```

- Parameters

- maxSize – Specifies the maximum size for a query page.
 - Default value is 1000.

- Notes

- This method sets the maximum size for a query page in a result set.

© Copyright IBM Corporation 2011

Figure 7-26. serverCacheConfiguration.set_QueryPageMaxSize(...)

F1432.0

Notes:

Search for documents with paging

serverCacheConfiguration.set_QueryPageDefaultSize(...)

```
void set_QueryPageDefaultSize(  
    Integer defaultSize)
```

- This method set the default size of a query page in a result set.

© Copyright IBM Corporation 2011

Figure 7-27. serverCacheConfiguration.set_QueryPageDefaultSize(...)

F1432.0

Notes:

Search for documents with paging

Demonstrations



- Search for documents with paging
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 7-28. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *searchPagedDocumentsEDU(...)* method in the *SearchEDU.java* file.
3. Run the code and verify the results as instructed in the *Searches* unit in the Student Exercises book.

Search for documents with paging

Activities

In your Student Exercises book

- Unit: Searches
- Lesson: Search for documents with paging
- Activities
 - Search for documents with paging.

© Copyright IBM Corporation 2011

Figure 7-29. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 7.3. Search for objects across multiple object stores

Lesson:

Search for objects across multiple object stores

- 
- Why is this lesson important to you?
 - Your application requires a feature that searches for Document objects across multiple object stores on the given Content Engine. You are going to write code to accomplish this.

© Copyright IBM Corporation 2011

Figure 7-30. Lesson: Search for objects across multiple object stores

F1432.0

Notes:

Search for objects across multiple object stores

Activities that you need to complete



- Search for objects across multiple object stores.

© Copyright IBM Corporation 2011

Figure 7-31. Activities that you need to complete

F1432.0

Notes:

Search for objects across multiple object stores

Create alias IDs



- Alias IDs enable the following:
 - Associate a property or class in one object store with an equivalent property or class in another object store.
 - Support for multi-object store searches.
- Only user-defined and built-in properties and classes can be aliased.
 - System classes cannot be aliased.
 - An object in one object store can only be aliased to one object in another object store.
 - The mapping must be one-to-one between object stores.

© Copyright IBM Corporation 2011

Figure 7-32. Create alias IDs

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Properties > How to > Create alias IDs

Create alias IDs

TIP When entering an alias ID, include the curly braces ({ }). These braces are part of the Primary ID.

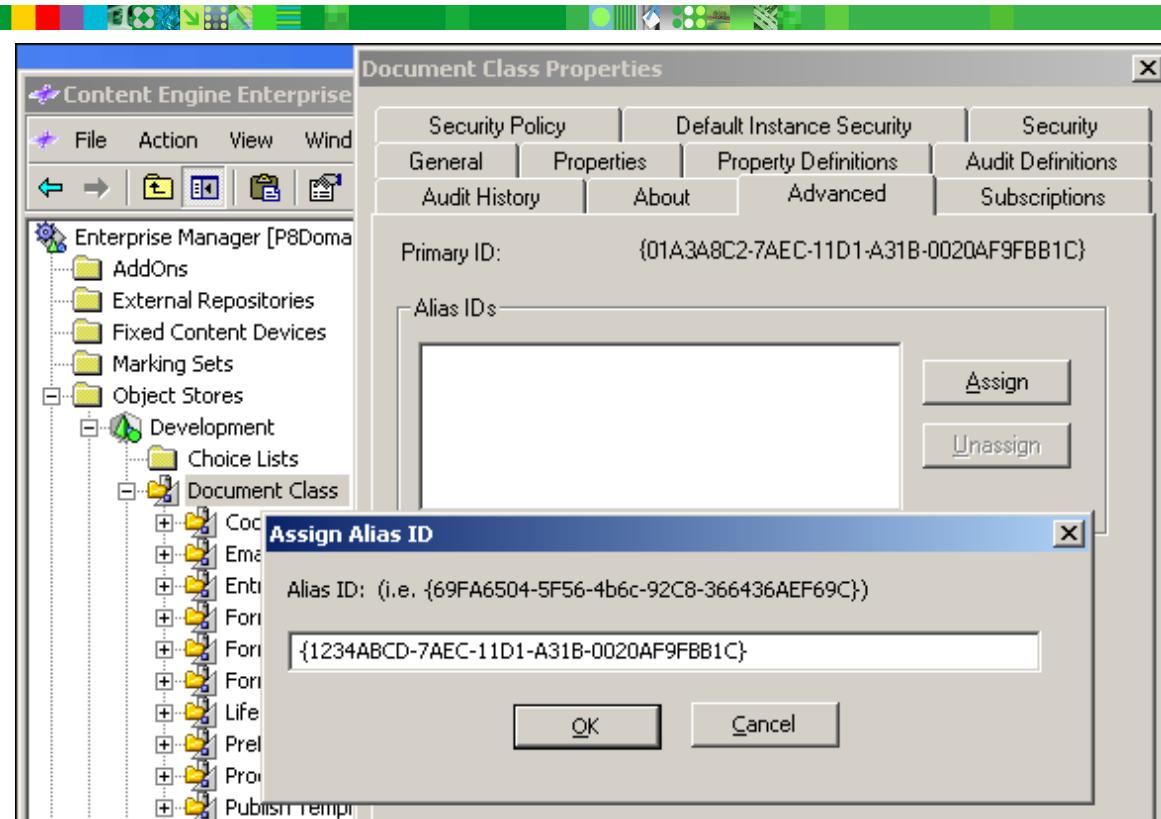
Aliasing properties

To perform a cross object store search and retrieve the same value for equivalent properties in each of the object stores, create an alias between the properties.

If you add an alias ID to a property template already assigned to a class, the alias ID is not automatically applied to that class property. You must manually assign the alias ID to the class property.

Search for objects across multiple object stores

Assign an alias ID to a class



© Copyright IBM Corporation 2011

Figure 7-33. Assign an alias ID to a class

F1432.0

Notes:

The diagram shows how to assign an alias ID to a class.

Steps to assign an alias ID to a class

1. Expand the Enterprise Manager tree to display the class to assign an alias ID to.
2. Right-click the class and select Properties from the content menu.
3. Select the Advanced tab.
4. Click Assign.
5. Enter the Primary ID (GUID) of the class you are aliasing in the Assign Alias ID dialog and click OK.
6. Click OK to apply the new alias ID and close the property sheet.

Note: For the lab exercise searching across the multiple object stores, alias ID has been assigned for the Product document class.

Search for objects across multiple object stores

Steps to search for objects across the object stores

1. Get the object stores and assign it to an array.

```
ObjectStore objectStores[] = new  
ObjectStore[2];
```

2. Instantiate a SearchScope object with the Objectstores to search from.

```
new SearchScope(...)
```

3. Instantiate a SearchSQL object with the SQL statement.

```
new SearchSQL(...)
```

4. Retrieve the objects.

```
searchScope.fetchObjects(...)
```

© Copyright IBM Corporation 2011

Figure 7-34. Steps to search for objects across the object stores

F1432.0

Notes:

Search for objects across multiple object stores

Sample code to search across object stores (1)

```
...ObjectStore objectStores[] = new
                                ObjectStore[2];
objectStores[0] = "Development";
objectStores[1] = "Sales";
PropertyFilter pf = new PropertyFilter();
pf.addIncludeProperty(new
    FilterElement(Integer.getInteger("2"),
null, Boolean.valueOf(true), "Name"));
SearchScope search = new SearchScope
    (objectStores, MergeMode.UNION);
SearchSQL sql = new SearchSQL("select * from
Document d where Creator='P8Admin' AND NOT
(IsClass(d, CodeModule))");
```

Continued on the next page.

© Copyright IBM Corporation 2011

Figure 7-35. Sample code to search across object stores (1)

F1432.0

Notes:

Search for objects across multiple object stores

Sample code to search across object stores (2)

```
DocumentSet documents = (DocumentSet)
    search.fetchObjects(sql, Integer.getInteger
        ("50"), pf, Boolean.valueOf(true));

Document doc;

Iterator it = documents.iterator();

while (it.hasNext()){

    doc = (Document)it.next();

    System.out.println("document=" +
        doc.get_Name());

}
```

© Copyright IBM Corporation 2011

Figure 7-36. Sample code to search across object stores (2)

F1432.0

Notes:

Search for objects across multiple object stores

Demonstrations



- Search for objects across multiple object stores
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 7-37. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start the Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *multipleObjectStoresSearchEDU(...)* method in the *SearchEDU.java* file.
3. Run the code and verify the results as instructed in the *Searches* unit in the Student Exercises book.

Search for objects across multiple object stores

Activities



In your Student Exercises book

- Unit: Searches
- Lesson: Search for objects across multiple object stores
- Activities
 - Search for objects across multiple object stores.

© Copyright IBM Corporation 2011

Figure 7-38. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 7.4. Build SQL statements

Lesson: Build SQL statements

- 
- Why is this lesson important to you?
 - Your application requires a feature that searches for Document objects on the given Content Engine. You want to use the IBM FileNet P8 capabilities for constructing SQL statements to run the search. You, as the programmer, are going to write code to accomplish this.

© Copyright IBM Corporation 2011

Figure 7-39. Lesson: Build SQL statements

F1432.0

Notes:

Build SQL statements

Activities that you need to complete



- Build SQL statements.

© Copyright IBM Corporation 2011

Figure 7-40. Activities that you need to complete

F1432.0

Notes:

Build SQL statements

Building SQL statements



- Use SearchSQL helper methods to develop statement:
 - setSelectList
 - setFromClauseInitialValue
 - setWhereClause
 - setMaxRows
- Use SearchSQL.toString() to see resulting SQL.
- View SQL using Enterprise Manager QueryBuilder > SQL View
- Manually refine the SQL statement and pass into SearchSQL.setQueryString.
 - Supersedes helper method settings, including MaxRows

© Copyright IBM Corporation 2011

Figure 7-41. Building SQL statements

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference

SQL statements need to follow the IBM FileNet standard, which generally conforms to SQL-92, with extensions for FileNet-specific constructs.

The SearchSQL helper methods are supplied for assistance in building SQL statements and cannot provide the level of specification that you can achieve with an independently constructed statement.

However, in a development environment, you can use the helper methods for initial construction of the SQL statement, and then use the SearchSQL.toString() method to get the SQL statement string and manually refine the SQL statement.

Build SQL statements

Steps to build an SQL statement

1. Instantiate a SearchSQL object.

```
new SearchSQL(...)
```

2. Specify the SELECT list.

```
searchSQL.setSelectList(...)
```

3. Specify the FROM clause.

```
searchSQL.setFromClauseInitialValue(...)
```

4. Specify the WHERE clause.

```
searchSQL.setWhereClause(...)
```

5. Specify the ORDER BY clause and the maximum number of records to be returned (optional).

```
searchSQL.setOrderByClause(...)
```

```
searchSQL.setMaxRecords(100)
```

© Copyright IBM Corporation 2011

Figure 7-42. Steps to build an SQL statement

F1432.0

Notes:

Build SQL statements

Elements of a basic SQL statement



- SQL query syntax

```
SELECT [DISTINCT | ALL] [TOP<integer>]  
<select_list> FROM <class_reference>  
[WHERE <search_condition>]  
[ORDER BY <orderby> { ',' <orderby> } ]
```

- SELECT <select_list>
- FROM <class reference>
- WHERE <search conditions>
- ORDER BY

© Copyright IBM Corporation 2011

Figure 7-43. Elements of a basic SQL statement

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries > SQL Statement Grammar

Build SQL statements

Conventions for SQL query syntax statements

Convention	Symbol	Description
UPPER CASE		Keyword tokens. Example: SELECT
Single quotation marks	',	Surround symbols forming part of the grammar. Example: '*'
Square brackets	[]	Surrounding a token or series of tokens indicates an optional clause. Example: [ORDER BY <orderby>]
Curly Braces	{ }	Surrounding a token or series of tokens indicates an optional clause that can be repeated more than once.
Single vertical bars		Separate alternatives. For example, AND OR
Parentheses without quotation marks	()	Defines precedence. Example: (AND OR)
Italic type	< <i>italic</i> >	Indicates production rules whose definitions are well known. Example: < <i>integer</i> >

© Copyright IBM Corporation 2011

Figure 7-44. Conventions for SQL query syntax statements

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries

The FileNet P8 platform supports a query syntax that conforms generally to SQL-92 and specifically is aligned with SQL Server query syntax wherever there is equivalence of function, with classes and properties playing the role of tables and columns.

Note: If a class has a property of the same name as a reserved SQL word and you want to search for that property, you must surround the property name in your SELECT statement with square brackets ([]) to prevent errors. (For a list of reserved words, refer to your SQL documentation.) For example, if your Document class has a property named "From", which is a reserved word in SQL, structure your SELECT statement in the following manner:

```
SELECT d.[DocumentTitle], d.[From], d.[Id] FROM Document d WHERE d.[Creator] = "jsmith"
```

Build SQL statements

SELECT list



- **SELECT [DISTINCT | ALL] [TOP<integer>] <select_list>**
<select_list> ::= <select_prop> { ','
<select_prop> }

- Examples

```
SELECT [This], [capacity], [Creator],  
[credit_card_num]
```

© Copyright IBM Corporation 2011

Figure 7-45. SELECT list

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries > SQL Statement Grammar

Build SQL statements

FROM clause



- FROM <class_reference>

```
<class_reference> ::= <from_class> |  
<qualified_join>
```

- Examples

```
FROM CustomObject WITH INCLUDESUBCLASSES  
FROM {D32E4F58-AFB2-11D2-8BD6-00E0290F729A}
```

© Copyright IBM Corporation 2011

Figure 7-46. FROM clause

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries > Class and Property Identification in Query Text

Build SQL statements

WHERE clause

- 
- WHERE <search_condition>
<search_condition> ::=
<basic_search_condition> [(AND | OR)
<search_condition>]
 - Example

```
WHERE ([Creator] = 'administrator' OR  
[Creator] = 'clara' AND [DateCreated] <  
20070719T032844Z)
```

© Copyright IBM Corporation 2011

Figure 7-47. WHERE clause

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries > SQL Statement Grammar

Build SQL statements

IN operator (IBM FileNet P8 SQL extensions)

- To check for existence of a value in a list
- value IN listproperty
 - Required to search multivalued properties
 - Example: `WHERE "red" IN [ColorList]`
- value IN (constant1, constant2, ...)
- value IN (SELECT someproperty FROM ...)

© Copyright IBM Corporation 2011

Figure 7-48. IN operator (IBM FileNet P8 SQL extensions)

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries > In Operator

The following formats are used for the IN operator:

- value IN listproperty

Where "value" is either a property name or a constant (<property_spec> | <literal>), and listproperty is a property name (<property_spec>) of a property that has cardinality of LIST. This format is used to search a property that is of cardinality LIST. If the list has a value equal to "value", the expression is true. Note that you cannot use "value = listproperty" to compare a property of type LIST to a value. The IN operator must be used.

- value IN (constant1, constant2, constant3,...)

Where constant1, constant2, constant3 are constants (<literal>s). This format is used to determine if "value" is equal to one of the constants in the list. Note that when multiple items are between the parentheses, the items must be constants.

- value IN (SELECT someproperty FROM ...)

This format is used to determine if "value" is one of the results returned by the subquery. Note that the subquery must select only one property.

IBM FileNet P8 SQL extensions - Folder operators

The INFOLDER operator matches an object contained within a specified folder. The left-hand operand specifies an object-valued property or an expression that results in a single object value. The right-hand operand identifies the folder in one of the following forms:

- Folder ID
- Full path name of the folder

INSUBFOLDER can be used only on subfolders that have the Folder.Parent property set to the parent of the folder (direct containment). The INSUBFOLDER operator does not recognize a subfolder linked to a parent by referential containment (a ReferentialContainmentRelationship object having the ReferentialContainmentRelationship.Tail property equal to the subfolder and the ReferentialContainmentRelationship.Head property equal to the parent folder).

Note: Use of the INSUBFOLDER operator results in a complex query that can take a long time to execute. Before deploying a solution that uses an INSUBFOLDER search across a large number of folders and containees, ensure that this search works well on a large database and is properly optimized. The database is not capable of properly optimizing all queries.

Build SQL statements

OBJECT Function Description

- Create object constant to compare with an object-value property
- `SELECT ... FROM Folder WHERE Parent = OBJECT('/root/sub1/sub1a')`

© Copyright IBM Corporation 2011

Figure 7-49. OBJECT Function Description

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries > Object Function Description

The Object function provides a means to create the effect of an object constant, which can then be compared to an object-valued property or expression. There are two logical forms of the Object function. One takes an argument of type GUID (Id), which can be either a constant or a GUID-valued property. The other takes a string argument, which can be one of the following:

- GUID constant (Id) of the object
- Full path name of the folder for Folder objects
- A <property_spec> for a property of data type Id. An Object function can be used to compare objects to Ids, which can be especially useful in joins.

Build SQL statements

IsClass Function Description



- IsClass
- INCLUDESUBCLASSES
- SELECT ... FROM Document D WHERE
IsClass(D,DocSubclass1) OR IsClass(D,DocSubclass2)
- SELECT ... FROM Document WITH INCLUDESUBCLASSES
WHERE DocumentTitle = 'MyDoc'

© Copyright IBM Corporation 2011

Figure 7-50. IsClass Function Description

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > Relational Queries > IsClass Function Description

The INCLUDESUBCLASSES and EXCLUDESUBCLASSES operators, along with ISCLASS, are used to limit results to specific classes or class hierarchies. INCLUDESUBCLASSES is the default operator.

The IsClass function acts as a filter for finding specific classes. The IsClass function is useful when you are querying base classes with the operator INCLUDESUBCLASSES and the results must include objects that belong to one or more particular subclasses of the base class.

Build SQL statements

Available SearchSQL helper class methods

- Methods to assist in building valid SQL statements:
 - `searchSQL.SetSelectList(...)`
 - `searchSQL.SetFromClauseInitialValue(...)`
 - `searchSQL.setFromClauseAdditionalJoin (...)`
 - `searchSQL.setWhereClause (...)`
 - `searchSQL.setOrderByClause (...)`
 - `searchSQL.setQueryString (...)`
 - `searchSQL.setMaxRecords (...)`
 - `searchSQL.setContainsRestriction(...)`
 - `searchSQL.setFreeTextRestriction(...)`

© Copyright IBM Corporation 2011

Figure 7-51. Available SearchSQL helper class methods

F1432.0

Notes:

The SearchSQL helper methods are supplied for assistance in building SQL statements and cannot provide the level of specification that you can achieve with an independently constructed statement.

However, in a development environment, you can use the helper methods for initial construction of the SQL statement, and then use the `SearchSQL.toString()` method to get the SQL statement string and manually refine the SQL statement.

Build SQL statements

searchSQL.setSelectList(...)

```
public void setSelectList(String selectList)
```

- Parameters

- selectList – Specify a String containing the query SELECT list

- Notes

- This method sets the SELECT list for the SQL statement to the string specified.
 - This method must be called prior to calling SetFromClauseAdditionalJoin.

© Copyright IBM Corporation 2011

Figure 7-52. searchSQL.setSelectList(...)

F1432.0

Notes:

Build SQL statements

searchSQL.setFromClauseInitialValue(...)

```
public void setFromClauseInitialValue( String symbolicClassName, String aliasName boolean includeSubclasses null)
```

- Parameters

- symbolicClassName - A String containing the symbolic name of the class
- aliasName - A String containing the alias name of the class. This can be null.
- includeSubclasses - true if the FROM clause needs to include any subclasses of the class specified in symbolicClassName

- Notes

- Sets the first class to be used in the FROM clause for the statement.

© Copyright IBM Corporation 2011

Figure 7-53. searchSQL.setFromClauseInitialValue(...)

F1432.0

Notes:

Alias names cannot be used when EngineObject objects are to be returned by the query operation.

Build SQL statements

searchSQL.setFromClauseAdditionalJoin(...)

```
public void setFromClauseAdditionalJoin(
    JoinOperator joinOperator,
    String symbolicClassName, String aliasName,
    String joinVar1, JoinComparison joinComparison,
    String joinVar2, boolean includeSubclasses)
```

- Parameters

- joinOperator - Specifies the type of join to use for the additional class.
- joinComparison - Specifies the comparison to use for the constituents of the ON clause (joinVar1 and joinVar2).

- Notes

- Adds another class to the FROM clause used in the SQL statement Parameters.

© Copyright IBM Corporation 2011

Figure 7-54. searchSQL.setFromClauseAdditionalJoin(...)

F1432.0

Notes:

JoinComparison - Specifies the operation to use when comparing the constituents of an SQL join operation.

EQUAL, GREATER_THAN, GREATER_THAN_EQUAL_TO, LESS_THAN, LESS_THAN_EQUAL_TO, NOT_EQUAL

Parameters

JoinOperator - INNER, LEFT OUTER, RIGHT OUTER, FULL OUTER

symbolicClassName - A String containing the symbolic name of the class

aliasName

- A String containing the alias name of the class. This string can be Null.
- Alias names cannot be used when EngineObject objects are to be returned by the query operation.

joinVar1 - A String containing the name of a property on the initial class specified in SetFromClauseInitialValue. In tandem with the property specified in joinVar2, this property specifies the ON clause constituents of the join.

joinVar2 - A String containing the name of a property on the class specified in symbolicClassName. In tandem with the property specified in joinVar1, this property specifies the ON clause constituents of the join.

includeSubclasses - A Boolean value of True if the FROM clause must include any subclasses of the class specified in symbolicClassName; False otherwise.

Build SQL statements

searchSQL.setWhereClause (...)



```
public void setWhereClause( String whereClause)
```

- Parameters
 - whereClause - A String containing the WHERE clause to use
- Notes
 - This method sets the WHERE clause to be used for the SQL statement to the specified string.

© Copyright IBM Corporation 2011

Figure 7-55. searchSQL.setWhereClause (...)

F1432.0

Notes:

Build SQL statements

searchSQL.setOrderByClause(...)



```
public void setOrderByClause(String orderBy)
```

- Parameters
 - orderBy - A String containing the property or properties to use for the ORDER BY clause
- Notes
 - This method sets the ORDER BY clause to be used for the SQL statement to the specified string.
 - You can specify multiple property names separated by commas for the parameter.

© Copyright IBM Corporation 2011

Figure 7-56. searchSQL.setOrderByClause(...)

F1432.0

Notes:

Build SQL statements

searchSQL.setQueryString (...)

```
public void setQueryString(String queryString)
```

- Parameters
 - queryString - A String containing the SQL statement to use
- Notes
 - Specifies the complete SQL statement. This method cannot be used in conjunction with any other set method on this class.

© Copyright IBM Corporation 2011

Figure 7-57. searchSQL.setQueryString (...)

F1432.0

Notes:

Any values that are set by calling another SearchSQL set method (such as, setMaxRecords) and then calling this method overwrite or nullify the value that was initially set.

Calling this method and then calling any other SearchSQL set method overwrites or nullifies the SQL statement specified here.

No SQL validation is performed on the specified string.

Build SQL statements

searchSQL.setMaxRecords(...)

```
public void setMaxRecords(int maxRecords)
```

- Parameters

- maxRecords - An integer specifying the maximum number of rows to be returned

- Notes

- This method sets the maximum number of rows that can be returned in the result set.

© Copyright IBM Corporation 2011

Figure 7-58. searchSQL.setMaxRecords(...)

F1432.0

Notes:

When unspecified, all records that satisfy the query are returned, subject to the limit of ServerCacheConfiguration property NonPagedQueryMaxSize.

Build SQL statements

Sample code to construct SQL statements

```
...
SearchSQL sql = new SearchSQL();
sql.setSelectList("Name, DocumentTitle");
sql.setFromClauseInitialValue("Product,
                               "d",True);
sql.setWhereClause("Creator='P8Admin'
                   AND NOT (IsClass(d, CodeModule))");
```

- Resulting SQL statement from the sample code

```
"SELECT Name, DocumentTitle FROM Product d
 WHERE Creator='P8Admin' AND
 NOT(IsClass(d, CodeModule))"
```

© Copyright IBM Corporation 2011

Figure 7-59. Sample code to construct SQL statements

F1432.0

Notes:

Refer to the previous lesson for the code that searches the documents using the SQL statements.

Build SQL statements

Demonstrations



- Build SQL statements
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 7-60. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *searchSQLLEDU(...)* method in the *SearchEDU.java* file.
3. Run the code and verify the results as instructed in the *Searches* unit in the Student Exercises book.

Build SQL statements

Activities



In your Student Exercises book

- Unit: Searches
- Lesson: Build SQL statements
- Activities
 - Build SQL statements.

© Copyright IBM Corporation 2011

Figure 7-61. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 7.5. Content-based retrieval

Lesson: Content based retrieval

- 
- Why is this lesson important to you?
 - Your application requires a feature that searches for Document objects on the given Content Engine based on the content. You, as the programmer are going to write code to accomplish this.

© Copyright IBM Corporation 2011

Figure 7-62. Lesson: Content based retrieval

F1432.0

Notes:

Content-based retrieval

Activities that you need to complete



- Search for documents based on content.

© Copyright IBM Corporation 2011

Figure 7-63. Activities that you need to complete

F1432.0

Notes:

Content-based retrieval

Content Based Retrieval (CBR)



- Content Engine supports CBR for the following:
 - Documents, annotations, folders, and properties of custom objects
- Each object store that is configured to support CBR requires the creation of full-text indexes.
- Content Engine creates and maintains these full-text indexes using one of the following search engines:
 - IBM Content Search Services (CSS)
 - Lucene based
 - IBM Legacy Content Search Engine (CSE)
 - Autonomy -Verity K2 Enterprise

© Copyright IBM Corporation 2011

Figure 7-64. Content Based Retrieval (CBR)

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content-based retrieval
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries > Concepts > Content Searches

The FileNet Content Manager V5.0 provides two content search solutions

- the same one available in prior releases, based on third-party technologies,
- a new IBM-developed Search engine.

The search capability that has been in FileNet CM for the past several releases is called “IBM Legacy Content Search Engine (CSE)”

The new search solution in version 5.0 is called “IBM Content Search Services (CSS)”

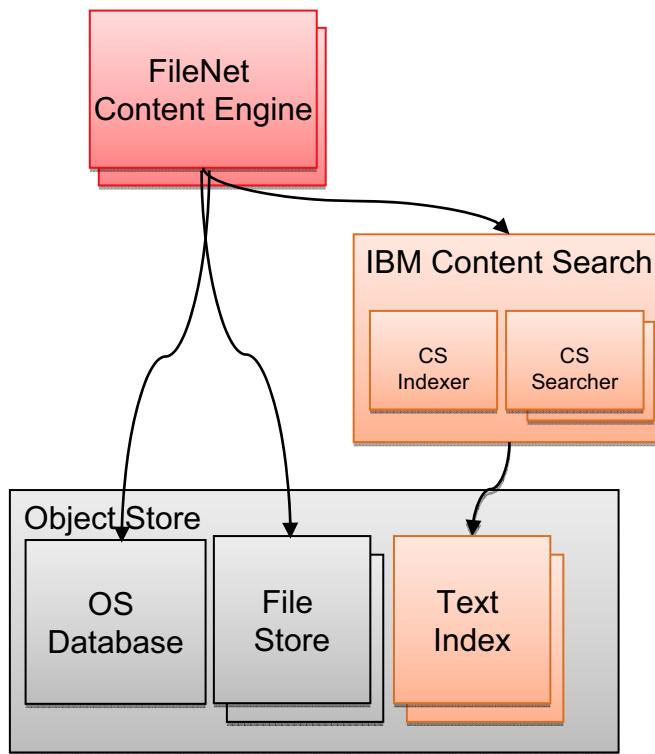
CSS is based on Lucene as part of a broader IBM open source-based search.

Reference: On your IBM FileNet P8 Linux server system, the documentation is located at /opt/IBM/FileNet/ContentEngine/verity/data/docs/pdf/VerityQueryLanguage.pdf

With content-based retrieval, you can search an object store for objects that contain specific words or phrases embedded in document or annotation content; or embedded in string properties of objects that have been configured for full-text indexing.

Content-based retrieval

IBM Content Search Services



© Copyright IBM Corporation 2011

Figure 7-65. IBM Content Search Services

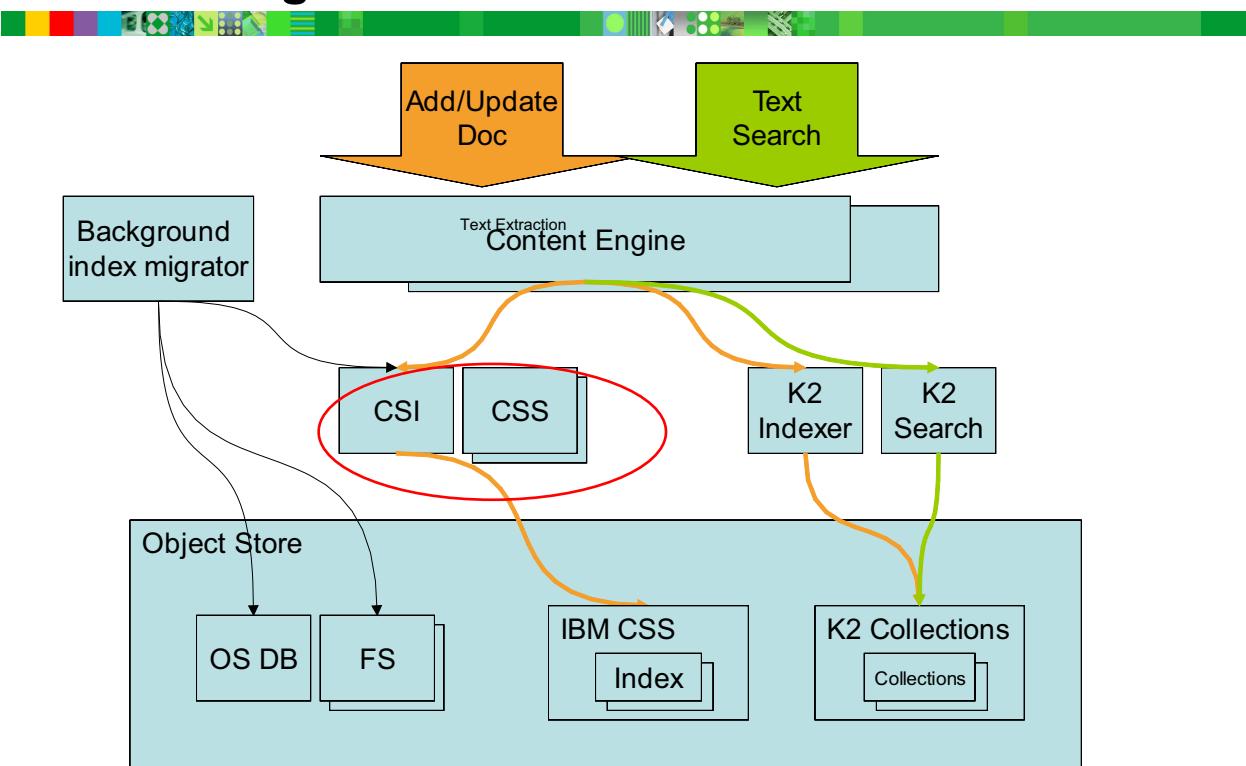
F1432.0

Notes:

In this release of IBM Content Search Services, the new search capability is only exposed at the API level. In subsequent releases of IBM FileNet client applications such as Workplace XT) will expose the new CSS capability.

Content-based retrieval

Content Engine Search Architecture



CSI- Content Search indexer CSS – Content Search Searcher

© Copyright IBM Corporation 2011

Figure 7-66. Content Engine Search Architecture

F1432.0

Notes:

Indexing Basics

1. Text extraction from the original document is performed by the Content Engine.
2. Extracted text is delivered to the assigned index writer (a Content Search Services indexing process).
3. Written to the current open index.
4. Content Engine clustering (farming) delivers scale out of text extraction.
5. Text Extraction requires INSO technology to be installed in the EAR using the CSE installer.

Search Basics

1. Search requests are presented to the Content Engine as part of a Content Engine SQL search.
2. The Content Search Services portion is delivered to one of the Content Search Services server processes.

3. The result is returned to the Content Engine for storage in a temp table.
4. The temp table is joined with the relational portion of the query .
5. Security authorization is applied and anything the user doesn't have access to is dropped.
6. Matching results are returned to the client application.

Content-based retrieval

Content searches



- A content (full-text) search
 - Returns documents that contain the text that you specify
 - Includes in the query the words or phrases that might be stored in objects or in their string properties
 - Is initiated by either a CONTAINS or FREETEXT operator in the SQL statement contained in SearchSQL
- To do a content search, you must do the following:
 - Enable content-based retrieval (CBR) for the object and its string properties.
 - Index the documents.

© Copyright IBM Corporation 2011

Figure 7-67. Content searches

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content-based retrieval
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries > Concepts > Content Searches

The CONTAINS operator can search content in all properties or in a single property, but the FREETEXT operator can search content only in all properties. Note that attempting to specify a property name for FREETEXT generates an exception.

Content-based retrieval

Initiating content search



- Initiated by including either a CONTAINS or FREETEXT operator (but not both) in the WHERE clause
- CONTAINS operator searches for exact or "fuzzy" matches of words or phrases
- FREETEXT operator searches for words or phrases with the same meaning

© Copyright IBM Corporation 2011

Figure 7-68. Initiating content search

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Queries > Concepts > Content Searches

Content-based retrieval

Full-Text joins

- `SELECT ... FROM Document D INNER JOIN ContentSearch CS ON D.This=CS.QueriedObject WHERE CONTAINS(*,'text') AND D.Creator = 'Frank'`

- `SELECT ... FROM Document D LEFT OUTER JOIN ContentSearch CS ON D.This=CS.QueriedObject WHERE CONTAINS(*,'text') OR D.Creator = 'Frank'`

© Copyright IBM Corporation 2011

Figure 7-69. Full-Text joins

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > CBR Queries > CBR Queries using IBM Legacy Content Search Engine (Autonomy) > Full-Text Joins

Whenever a CONTAINS or FREETEXT clause is used, a join is performed on the (internal) ContentSearch or VerityContentSearch classes.

Instances of the ContentSearch or VerityContentSearch classes cannot be retrieved. However, the properties of these classes are defined in the metadata for a RepositoryRow object.

The join is necessary because, when a query with a full-text search is executed, the full-text search is done first, and then the data from the full-text search is copied to a temporary database table that is referenced by either the ContentSearch or VerityContentSearch class name (both behave identically). The remainder of the search

statement is then executed against the repository, joining to this temporary table to access the full-text search data.

Content-based retrieval

CONTAINS Function

- Search for precise or fuzzy matches
- Word or phrase
- Prefix of word or phrase
- Word near another word
- Inflected word
- ```
SELECT ... FROM Document D INNER JOIN
ContentSearch CS ON D.This=CS.QueriedObject
WHERE CONTAINS(*,'text')
```

© Copyright IBM Corporation 2011

Figure 7-70. CONTAINS Function

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > CBR Queries > CBR Queries using IBM Legacy Content Search Engine (Autonomy) > CONTAINS Function

Use the CONTAINS operator to search for precise or fuzzy (less precise) matches for the following types of text:

- A word or phrase, or a prefix of a word or phrase.
- A word near another word.
- A word that is the inflected form of another. (For example, "drive" is the root for the inflected forms "drives", "drove", "driving", and "driven".)
- A word that has a higher designated weighting than another word.

The first parameter in the CONTAINS operator identifies the property or pseudo-property to which the content filter is to be applied, and the second parameter specifies the filter expression.

Content-based retrieval

## FREETEXT function

- Search for document content or string property.
- Matches the meaning, not the exact wording.
- Content Engine supports CBR for the following objects and their properties:
  - Documents
  - Annotations
  - Folders
  - Custom objects

© Copyright IBM Corporation 2011

Figure 7-71. FREETEXT function

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > CBR Queries > CBR Queries using IBM Legacy Content Search Engine (Autonomy) > FREETEXT Function

With content-based retrieval (CBR), you can search an object store for objects that contain specific words or phrases embedded in document or annotation content. Or you can search for specific words or phrases embedded in string properties of objects that have been configured for full-text indexing.

Content Engine creates and maintains the full-text indexes using the FileNet P8 Content Search Engine.

Use the FREETEXT operator to search the text of the string property or document content for values that match the meaning, but not the exact wording in the filter expression. When

using FREETEXT, the full-text query engine internally divides the filter expression into a number of search terms, assigns each term a weight, and then finds the matches.

Content-based retrieval

## Ranking



- The results of a CBR query are returned in an ordered set or rows.
- The results are ranked according to the closeness of fit with the search.
- The method by which the ranking is determined is dependent upon the query and the operators used in the full-text clause (the WHERE clause) of the query.
- A "Rank" property is returned for each match:
  - It has a value between 0.0 and 1.0 for the Legacy Search.
  - The numbers closer to 1.0 are the better matches.
  - Workplace presents the ranking as a percentage.

© Copyright IBM Corporation 2011

Figure 7-72. Ranking

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content-based retrieval > Finding objects: CBR with IBM Legacy Content Search Engine > Concepts
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content-based retrieval > Finding objects: CBR with IBM Content Search Services > Querying for object text

The FileNet P8 Content Search Engine provides search and indexing capability for Content Engine.

All FileNet P8 clients, such as Workplace, communicate with Content Search Engine through Content Engine, which manages the search operations and the creation and deletion of indexes.

Content-based retrieval

## Common stop words



- Search engine does not look for common words.
- Stop words are not added to a collection when the document is indexed.
  - Examples: a, about, above, be, because, few, for, from
- One excluded words file per language
  - English: style.stp

---

© Copyright IBM Corporation 2011

Figure 7-73. Common stop words

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > CBR Queries > CBR Queries using IBM Content Search Services > Search Results
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content-based retrieval > Finding objects: CBR with IBM Content Search Services > Querying for object text

By default, the search engine does not look for certain common words, called stop words. When the index is built, the stop words are not added to the collection, so they can never be found.

If one or more words in your search requires a stop word, such as when you use the All or Near modifiers or the AND operator between lines, the search fails because these words never appear in the index.

Content-based retrieval

## Proximity operators (Verity terminology)



| Modifier         | Desired Search Condition                                              |
|------------------|-----------------------------------------------------------------------|
| <b>None</b>      | <b>Search for phrase with the exact spelling and case-sensitivity</b> |
| <b>Near</b>      | <b>Search for words located near each other</b>                       |
| <b>Sentence</b>  | <b>Search for words located in the same sentence</b>                  |
| <b>Paragraph</b> | <b>Search for words located in the same paragraph</b>                 |
| <b>Any</b>       | <b>Search for any of the specified words or phrases</b>               |
| <b>All</b>       | <b>Search for all of the specified words or phrases</b>               |
| <b>In Zone</b>   | <b>Search for one or more words in an HTML or XML tag</b>             |

© Copyright IBM Corporation 2011

Figure 7-24. Proximity operators (Verity terminology)

F1432.0

### Notes:

**Reference:** On your IBM FileNet P8 Linux server system, the documentation is located at /opt/IBM/FileNet/ContentEngine/verity/data/docs/pdf/VerityQueryLanguage.pdf

If you enter one word as your search criteria, the search engine finds various forms of that word (stemmed variations). You can turn off stemming and require matching capitalization using double quotation marks, or look for patterns using the asterisk (\*) as a wildcard character, or exclude a word using the exclamation point (!) at the beginning.

When you enter two or more words, you can specify a relationship between those words.

Your object stores can be configured to use either Near or Sentence and Paragraph (Near is the default). If your site uses Near, Sentence and Paragraph act like Near. If the specified object stores use Sentence and Paragraph, then Near operates like Sentence and Paragraph. The words qualify if they do not cross a sentence or paragraph boundary.

By default, Near is defined as words within 1000 words of each other.

When you choose In Zone, you are prompted for the name of the XML/HTML Zone name (often called a tag). You need to know what tags are used in your documents to take advantage of the faster search in only that part of the document.

The search engine sees hyphenated words as two separate words.

Content-based retrieval

## Autonomy K2 query language



- Autonomy K2 query language can be entered in the Words/Phrases field.
  - Choose the query language as the modifier.
- The text is passed to the Autonomy K2 search engine without checking or modification.

© Copyright IBM Corporation 2011

Figure 7-75. Autonomy K2 query language

F1432.0

### Notes:

**Reference:** On your IBM FileNet P8 Linux server system, the documentation is located at /opt/IBM/FileNet/ContentEngine/verity/data/docs/pdf/VerityQueryLanguage.pdf

## Query option

- TimeLimit
  - The time limit for the query, in seconds
- FullTextRowLimit
  - Indicates the number of rows to pull from the full-text index prior to executing the remainder of the query.
  - If a value is not supplied, the value of the ObjectStore.FullTextRowDefault property (stored in the GCD) is used.

© Copyright IBM Corporation 2011

Figure 7-76. Query option

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Reference > SQL Syntax Reference > CBR Queries > CBR Queries using IBM Content Search Services > Query Result Row Limits

FullTextRowLimit: There is one row for each content element that is indexable, plus one more row for all CBR-enabled strings of an object. Data from the full-text index is copied to a temporary table, and that table is then joined with the remainder of the object store database to execute the query.

The FullTextRowLimit option enables you to control the maximum number of rows pulled from the full text index.

For example, suppose a query has the WHERE clause "WHERE color = 'red' and CONTAINS(\*, 'blue')". This query might have 1000 rows that match "CONTAINS(\*, 'blue')", but only 10 rows that match "color = 'red'" and "CONTAINS(\*, 'blue')". If FullTextRowLimit is

set to 500, then only 500 rows are pulled from the full-text index and written to the temporary table. Although only 10 rows match the complete WHERE clause, not all 10 rows might be found because not all 10 are necessarily in the 500 rows fetched from the full text index.

If a value for FullTextRowLimit is supplied and this value is greater than the value of ObjectStore.FullTextRowMax, then the value of ObjectStore.FullTextRowMax is used instead. The FullTextRowMax property is present on the ObjectStore class to prevent queries from using too much processing time and pulling an inordinate number of rows from the full-text index.

Content-based retrieval

## Steps to return object properties

- 
1. Instantiate a SearchScope object.

```
new SearchScope(...)
```

2. Instantiate a SearchSQL object with an SQL statement.

```
new SearchSQL(...)
```

3. Retrieve the objects with selected properties.

```
searchScope.fetchRows(...)
```

4. Iterate through resulting object collection.

```
repositoryRowSet.iterator()
(RepositoryRow) iterator.next
```

5. Retrieve the properties.

```
repositoryRow.getProperties()
```

© Copyright IBM Corporation 2011

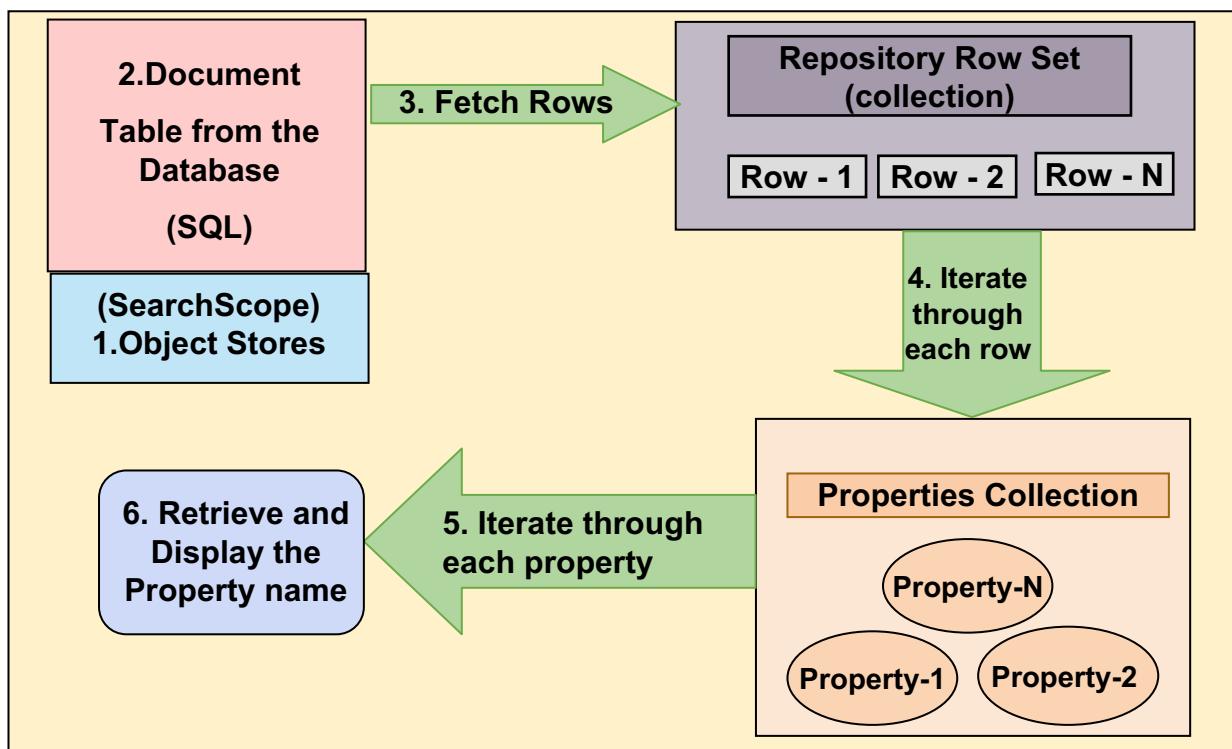
Figure 7-77. Steps to return object properties

F1432.0

### Notes:

Content-based retrieval

## Steps to retrieve properties of the objects



© Copyright IBM Corporation 2011

Figure 7-78. Steps to retrieve properties of the objects

F1432.0

### Notes:

Steps to retrieve properties of the objects:

1. SearchScope object defines the object store or object stores to search from.
2. SQL statement that is passed in as a parameter defines which table and criteria to retrieve.
3. SearchScope.fetchrows(...) method retrieves the rows collection.
4. Iterate through each row. Each row contains a collection of properties for a document.
5. Iterate through a Properties collection to retrieve each property.
6. Retrieve and display the name and value of each property.

Content-based retrieval

## searchScope.fetchRows(...)

```
public RepositoryRowSet fetchRows(
 SearchSQL searchSQL,
 Integer pageSize,
 PropertyFilter filter
 Boolean continuable)
```

- Parameters

- Continuable If false or null, the query is not paged.
  - If true, when the end of the first page is reached, a request for the next page of Properties objects is issued.
  - Page requests iterate until all of the Properties objects satisfying the query are retrieved.

- Returns

- RepositoryRowSet object Rows of properties from object store or stores

© Copyright IBM Corporation 2011

Figure 7-79. searchScope.fetchRows(...)

F1432.0

### Notes:

Parameters:

SearchSQL – The selection list in the searchSQL parameter determines which top-level properties are returned.

Content-based retrieval

## repositoryRow.getProperties()

**Properties getProperties()**

- Returns
  - A Properties collection object containing a row of properties.
- Notes
  - This method gets a single row of properties from the repository database.
  - All properties returned have the names as specified in the SQL statement used for the query.
  - Any properties that are not defined in a particular repository are returned with null values.

© Copyright IBM Corporation 2011

Figure 7-80. repositoryRow.getProperties()

F1432.0

### **Notes:**

Content-based retrieval

## searchSQL.setContainsRestriction(...)

```
public void setContainsRestriction(String
 symbolicClassName, String searchExpression)
```

- Parameters

- symbolicClassName - A String containing the symbolic name of the class
- searchExpression - A String containing the search text to use for the CONTAINS operator

- Notes

- This method restricts the query to return only items where the text in the content element or elements matches the specified string.

© Copyright IBM Corporation 2011

Figure 7-81. searchSQL.setContainsRestriction(...)

F1432.0

### Notes:

The CONTAINS operator can perform the search on CBR-enabled properties for the supported content elements in any of the following ways:

- Search a single property.
- Search all properties within a specified zone.
- Search all properties.

Content-based retrieval

## searchSQL.setFreetextRestriction(...)

```
public void setFreetextRestriction(String
symbolicClassName, String searchExpression)
```

- Parameters

- symbolicClassName - A String containing the symbolic name of the class
- searchExpression - A String containing the search text to use for the FREETEXT operator

- Notes

- This method restricts the query to return only items where the text in the content elements matches the specified string.

© Copyright IBM Corporation 2011

Figure 7-82. searchSQL.setFreetextRestriction(...)

F1432.0

### Notes:

This method uses the FREETEXT operator for CBR queries. The FREETEXT operator searches all content on all CBR-enabled properties on the supported content elements. To search content on a single CBR-enabled property, use the setContainsRestriction(...) method.

Content-based retrieval

## Sample code for content-based document searching

```
SearchScope search = new SearchScope(os) ;
String mySQLString = "SELECT d.this, DocumentTitle,
Name FROM Product d INNER JOIN
ContentSearch v ON v.QueriedObject = d.This
WHERE d.IsCurrentVersion = TRUE AND
CONTAINS(d.*,'" + keyWord + "')";
SearchSQL sql = new SearchSQL(mySQLString);
DocumentSet documents = (DocumentSet)
search.fetchObjects(sql, Integer.valueOf("50"),
null, Boolean.valueOf(true));
com.filenet.api.core.Document doc;
Iterator it = documents.iterator();
while (it.hasNext()) {
 doc = (Document)it.next();
 System.out.println("document="+doc.getName());
}
```

© Copyright IBM Corporation 2011

Figure 7-83. Sample code for content-based document searching

F1432.0

### Notes:

Content-based retrieval

## Sample code for retrieving the properties

```

RepositoryRowSet rowSet = search.fetchRows (sql,
Integer.valueOf("50"),null,Boolean.valueOf(true));
Iterator itRow = rowSet.iterator();
RepositoryRow row;
com.filenet.api.property.Properties props;
com.filenet.api.property.Property prop;
while (itRow.hasNext()) {
 row = (RepositoryRow)itRow.next();
 props = row.getProperties();
 Iterator itProp = props.iterator();
 while (itProp.hasNext()){
 prop =(com.filenet.api.property.Property)
 itProp.next();
 if(prop.getPropertyName().equalsIgnoreCase("Rank")){
 System.out.println(prop.getPropertyName() +
 " = " + prop.getFloat64Value().toString());
 } else
 System.out.println(prop.getPropertyName() +
 " = " + prop.getStringValue());
 }
}

```

© Copyright IBM Corporation 2011

Figure 7-84. Sample code for retrieving the properties

F1432.0

### Notes:

SQL string used for the code on the page:

```

String rowSQLString = "SELECT DocumentTitle, Rank FROM Product d ";
rowSQLString = rowSQLString + "INNER JOIN VerityContentSearch v ON
v.QueriedObject = d.This ";
rowSQLString = rowSQLString + "WHERE d.IsCurrentVersion = TRUE AND
CONTAINS(d.*,'" + keyWord + "')";
SearchSQL sql = new SearchSQL(rowSQLString);

```

Content-based retrieval

## Additional sample VQL statements for content-based search



- Search for documents that contain the two key words in a sentence (use with <SENTENCE>).

```
"SELECT d.This, DocumentTitle, Name FROM Product d
INNER JOIN VerityContentSearch v ON v.QueriedObject =
= d.This WHERE d.IsCurrentVersion = TRUE AND
CONTAINS(d.* , 'model <SENTENCE> Deluxe')"
```

- Search for documents that contain the two key words located close to each other (use with <NEAR>).

```
"SELECT d.This, DocumentTitle, Name FROM Product d
INNER JOIN VerityContentSearch v ON v.QueriedObject =
d.This WHERE d.IsCurrentVersion = TRUE AND
CONTAINS(d.* , tested <NEAR/25> passed')"
```

© Copyright IBM Corporation 2011

Figure 7-85. Additional sample VQL statements for content-based search

F1432.0

### Notes:

**Reference:** On your IBM FileNet P8 Linux server system, the documentation is located at /opt/IBM/FileNet/ContentEngine/verity/data/docs/pdf/VerityQueryLanguage.pdf

SQL statements for the IBM Content Search Services - Proximity Search similar to NEAR with Autonomy :

```
SELECT This, DocumentTitle, Name FROM Loan D INNER JOIN ContentSearch CS ON
CS.QueriedObject = D.This WHERE CONTAINS(D.* , '\"loan payments\" ~3')
```

Note: Loan and Product in the queries are document classes.

Content-based retrieval

# IBM Content Search Services– XML Support



- Xpath-based searches can be used to find XML content
  - Can be used to search an explicit XML node or within the XML hierarchy
  - Only supported when one content element exists with a mime-type of text/xml or application/xml
  - An XML search is always slower than a property- or content-based search
- Example of a SQL statement for searching a XML document:

```
SELECT This, DocumentTitle, Name FROM Loan D INNER
JOIN ContentSearch CS ON CS.QueriedObject=D.This
WHERE CONTAINS(D.* , '@xmlxp: \' '/Loan/customer_name
[.contains(\"Jane\")]' ')
```

© Copyright IBM Corporation 2011

Figure 7-86. IBM Content Search Services- XML Support

F1432.0

## Notes:

Example XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<ListOfElements>
 <Element received_date="20100601121000">foo</Element>
</ListOfElements>
```

The contains clause can be used to:

- Search exact node: @xmlxp:"/ListOfElements/Element[. contains("foo")]"
- Search everything in a hierarchy: @xmlxp:"/ListOfElements[. contains("foo")]"
- Search in all node occurrences: @xmlxp:"//Element[. contains("foo")]"
- Search a date range: @xmlxp:"/ListOfElements/Element[@received\_date >= 20100601000000 and @received\_date <= 20100602000000]"
  - Must be an attribute
  - Must be in this format: YYYYMMDDHHMMSS

Content-based retrieval

## Demonstrations



- Content-based retrieval
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 7-87. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *contentSearchEDU(...)* method in the *SearchEDU.java* file.
3. Observe the SQL queries used for the legacy search engine and for the IBM Content Search Services.
4. Run the code and verify the results as instructed in the *Searches* unit in the Student Exercises book.

Content-based retrieval

## Activities

In your Student Exercises book

- Unit: Searches
- Lesson: Content-based retrieval
- Activities
  - Search for documents based on content.

© Copyright IBM Corporation 2011

Figure 7-88. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.



# Unit 8. Document Versions

## What this unit is about

This unit describes how to retrieve versionable objects, how to check out and check in a document and how to promote and demote a document.

## What you should be able to do

After completing this unit, you should be able to:

- Retrieve versionable objects.
- Check out and check in a document.
- Promote and demote a document.

## How you will check your progress

- Successful completion of lesson exercises.

## References

[www.ibm.com/support/documentation/us/en](http://www.ibm.com/support/documentation/us/en) (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

## Document Versions

# Unit lessons

- 
- Retrieve versionable objects
  - Check out and check in a document
  - Promote and demote a document

© Copyright IBM Corporation 2011

Figure 8-1. Unit lessons

F1432.0

### Notes:

## **Lesson 8.1. Retrieve versionable objects**

## Lesson: Retrieve versionable objects

- Why is this lesson important to you?
  - Amy is a member of the Product Development team. Her documents go through many steps from draft, to revised, to released. All of these documents are stored on the Content Engine as different versions. Amy needs to work with previous versions of some documents. As the team programmer, you are going to write code to retrieve prior versions of a document.

© Copyright IBM Corporation 2011

Figure 8-2. Lesson: Retrieve versionable objects

F1432.0

### Notes:

Retrieve versionable objects

## Activities that you need to complete



- Create a document and document versions.
- Retrieve versionable objects.

© Copyright IBM Corporation 2011

Figure 8-3. Activities that you need to complete

F1432.0

### Notes:

Retrieve versionable objects

## Document versions



- Document versions
  - Save multiple versions of a document.
  - Assign a unique ID number to each document version.
  - Assign a version status to each document version.
  - Record the user ID and time stamp for a version.
- Document versions allow the following:
  - Access to document content of prior versions
  - Separate default security for major and minor versions
  - Change to version status programmatically or by an end user
    - Example: Reinstate a prior version as the current version.
- Operate on versions through the following:
  - Versionable object instance itself
  - VersionSeries object

© Copyright IBM Corporation 2011

Figure 8-4. Document versions

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and Folders > Concepts > About Versioning

The Document object (that inherits the versionable interface) is the one that you are likely to work with the most.

However, other object types inherit from the Versionable object as well:

- CodeModule
- PublishTemplate
- WorkflowDefinition

Retrieve versionable objects

## Three levels of versioning



- System configuration sets three levels of versioning
  - No versioning, single-level versioning, two-level versioning
  - Set versioning using Content Engine Enterprise Manager
- If versions are not enabled
  - Document objects are permanently non-versionable.
- Single-level versioning
  - Each checkin of a document creates a major (released) version.
- Two-level versioning
  - Supports major and minor versions:
    - Minor for drafts, limited user access. Numbered 0.1, 0.2, 1.1, ...
    - Major for final, general user access. Numbered 1.0, 2.0, 3.0, ...
    - Major and minor version numbers are stored separately.  
They are not stored as #.#

© Copyright IBM Corporation 2011

Figure 8-5. Three levels of versioning

F1432.0

### **Notes:**

#### **Help path**

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and Folders > Concepts > About Versioning > Two level

#### **When versions are not enabled**

Version enabling or disabling is handled at the document class level, not at the individual document level.

When a new document is created using a versioning disabled class, that document is permanently nonversionable. It can never be checked out, and it cannot be converted into a versioning-enabled document.

Only one document version can exist in a document version series.

## **Single-level versioning (no enforcement on the server)**

The full versioning scheme supported by the Content Engine includes the ability to create both major and minor versions.

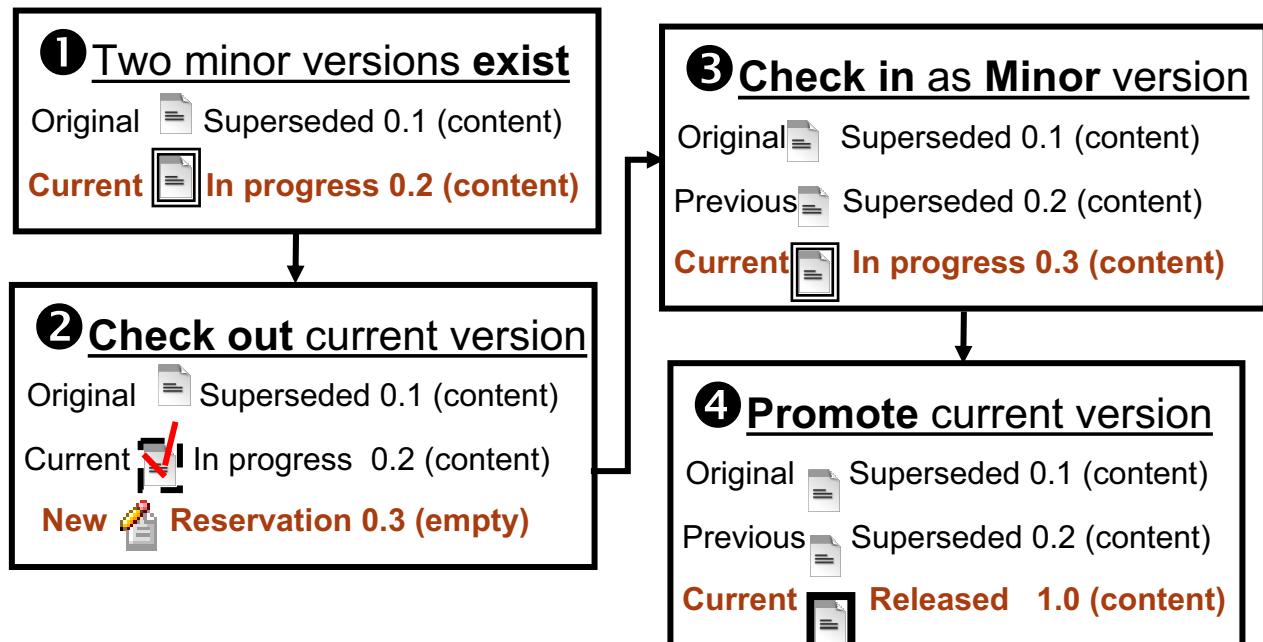
However, an application can be designed so that only major versions are created.

## **Two-level versioning**

Programmatically, two-level versioning behavior is enabled only if the application provides the "Check In as Minor Version" option on Check In.

Retrieve versionable objects

## Two-level versioning (1)

- Check in as a minor version and promote.

© Copyright IBM Corporation 2011

Figure 8-6. Two-level versioning (1)

F1432.0

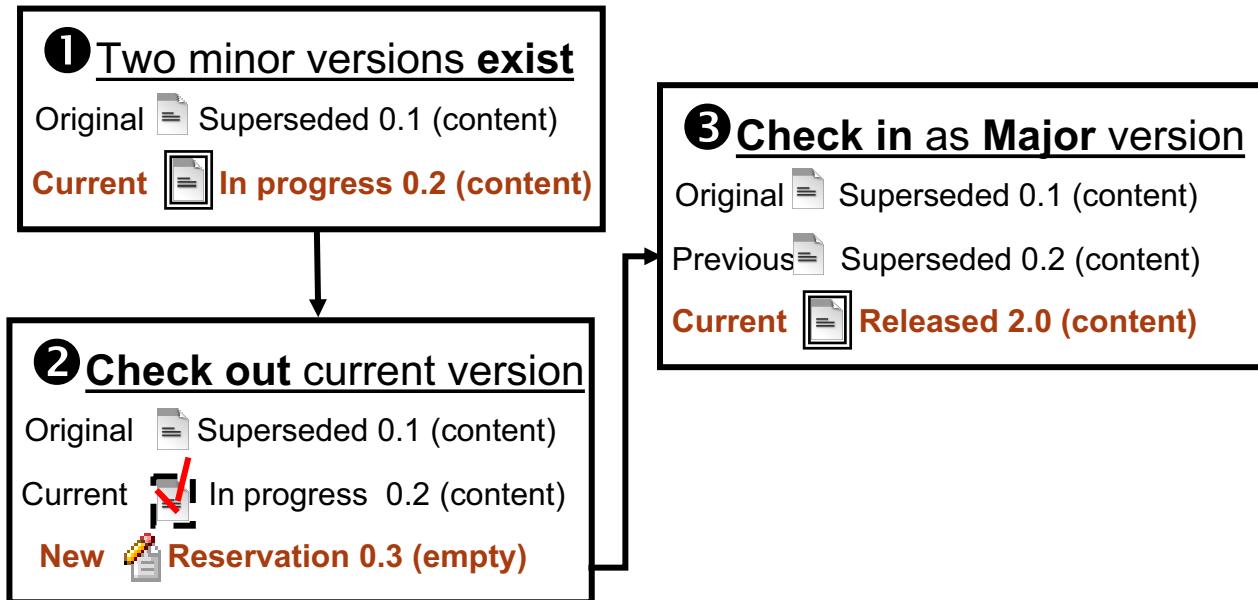
### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and Folders > Concepts > About Versioning > Two level

Retrieve versionable objects

## Two-level versioning (2)

- Check in as a major version.

© Copyright IBM Corporation 2011

Figure 8-7. Two-level versioning (2)

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and Folders > Concepts > About Versioning > Two level

Retrieve versionable objects

## Document version states



- In Process
  - A checked-in minor version
  - Generally made available to a restricted set of users.
  - Only one version in a series can be in this state at a time.
- Reservation
  - Shows checked-out version. Other users cannot check it out.
  - Only one version in a series can be in this state at a time .
- Released
  - A major version, generally made available to all users.
  - Only one version in a series can be in this state at a time.
- Superseded
  - A major or minor version that is not the most recent version
  - There can be many Superseded Majors and Superseded Minors in a version series.

© Copyright IBM Corporation 2011

Figure 8-8. Document version states

F1432.0

### **Notes:**

#### **Help path**

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and Folders > Concepts > About Versioning > Properties

IBM FileNet Content Manager uses the following two terms, which have different meanings:

**Reservation:** A document that is in the Reservation state means that the content of that document is currently being edited. The author who is editing that document has reserved it for his or her use.

**Reserved:** Reserved is a property that is set on the current version of a document when it is checked out. The current version is no longer available to be checked out. When the Reserved property of the current version is set to True, there must also be another version that is the Reservation version.

Retrieve versionable objects

## Reservation object



- A reservation object
  - Is the unchecked-in version of the document.
  - Is a limited use of Document object, not a separate Object class.
  - Is always a minor version.
  - Only one Reservation can exist at a time for a document.
- A reservation object is created
  - When a document is created or checked out.
- A reservation object is deleted
  - When a document is checked in or a check-out is canceled.
- No content in a reservation object when the document is checked out
  - You must set the content when the document is checked in.

© Copyright IBM Corporation 2011

Figure 8-9. Reservation object

F1432.0

### Notes:

A reservation object has VersionStatus.RESERVATION as its VersionStatus property value.

Retrieve versionable objects

## Versionable object



- The Versionable object
  - Represents the base class for versionable objects, including Document.
  - Can be checked out, edited, and checked in as a major or a minor version.
- If versioning is enabled on a versionable object
  - The IsVersioningEnabled property is true.
  - The object can have multiple versions associated with it in an object store.
- All of the versions of a versionable object can be accessed
  - Through its associated VersionSeries object
  - OR
  - From the collection returned by its Versions property

© Copyright IBM Corporation 2011

Figure 8-10. Versionable object

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Getting Started > Concepts > Base Classes and Interfaces

Retrieve versionable objects

## VersionSeries object



- The VersionSeries object
  - Is a utility object for getting information on the versions of a versionable object.
  - Instances constructed by the server in real time
    - Not persistable
    - Read-only
- A VersionSeries object instance has no security attached.
  - It is secured by the current document version associated with it.
  - You can access a specific document version by navigating through the collection returned by the Versions property.
- Deleting a VersionSeries object
  - Deletes all of its associated document versions.

© Copyright IBM Corporation 2011

Figure 8-11. VersionSeries object

F1432.0

### Notes:

The VersionSeries object represents the complete set of versions for a versionable object or document (that is, an object instantiated from a subinterface of the Versionable interface).

Because a VersionSeries object is constructed by the server from versionable objects and does not have its state stored in a database, it is read-only, so you cannot modify any of its properties.

Note that a VersionSeries object is not a collection of versions. When you retrieve a VersionSeries object, you have a single object.

However, certain operations that you perform on the VersionSeries object, such as calling Delete(), affect all versions.

Retrieve versionable objects

## Versionable versus VersionSeries object (1)



- The VersionSeries and Versionable objects share a few methods and properties.

Methods	Versionable	VersionSeries
CancelCheckout	X	X
Checkout	X	X
<b>Properties</b>		
IsReserved	X	X
IsVersioningEnabled	X	X
Reservation	X	X
Versions	X	X

© Copyright IBM Corporation 2011

Figure 8-12. Versionable versus VersionSeries object (1)

F1432.0

### **Notes:**

Retrieve versionable objects

## Versionable versus VersionSeries object (2)



- They also have distinct methods and properties.

Versionable Methods	VersionSeries Methods
ChangeClass CreateAnnotation DemoteVersion Freeze PromoteVersion	MoveContent
Versionable Properties	VersionSeries Properties
FoldersFiledIn IsCurrentVersion IsFrozenVersion MajorVersionNumber MinorVersionNumber ReservationType VersionSeries VersionStatus	AuditedEvents Id ReleasedVersion WorkflowSubscriptions set_Id

© Copyright IBM Corporation 2011

Figure 8-13. Versionable versus VersionSeries object (2)

F1432.0

### Notes:

Retrieve versionable objects

## Document versions example

- The collection of versions of a document

The screenshot shows the IBM FileNet Workplace interface. The top navigation bar includes links for 'Administrator | Friday, February 27, 2009 | Help | Home | Preferences | Sign Out'. Below the navigation bar is a toolbar with various icons. The main content area displays a table of document versions for 'APIVersions.doc'.

**Information Panel:**

- Properties
- Security
- Parent Documents
- Versions** (selected)
- Folders Filed In
- History

**Actions Panel:**

- Download
- Check Out
- Cancel Checkout
- Check In
- Quick Check In
- Save Content
- File

**Table Headers:**

	Title	Version Status	Major Version	Minor Version	Modified By	Modified On
--	-------	----------------	---------------	---------------	-------------	-------------

**Table Data:**

	APIVersions.doc	Released	4	0	Administrator	2/26/09 11:32 PM
	APIVersions.doc	Superseded	3	0	Administrator	2/26/09 8:00 PM
	APIVersions.doc	Superseded	2	1	CEadmin	2/26/09 3:18 PM
	APIVersions.doc	Superseded	2	0	CEadmin	2/26/09 3:18 PM
	APIVersions.doc	Superseded	1	0	CEadmin	2/26/09 3:17 PM

[View Current](#) | [View Release](#)

© Copyright IBM Corporation 2011

Figure 8-14. Document versions example

F1432.0

### Notes:

Retrieve versionable objects

## Steps to retrieve versions of a document

- 
1. Retrieve the versions.

```
document.getVersions()
```

2. Get the individual version.

```
versionableSet.iterator()
iterator.next()
```

3. Retrieve the version numbers and status for each version.

```
versionable.getMajorVersionNumber()
versionable.getMinorVersionNumber()
versionable.getVersionStatus().toString()
```

© Copyright IBM Corporation 2011

Figure 8-15. Steps to retrieve versions of a document

F1432.0

### Notes:

Retrieve versionable objects

## versionable.get\_Versions()



```
VersionableSet get_Versions()
```

- Returns
  - A VersionableSet object that contains all of the document versions associated with this document or version series
- Notes
  - If this object is checked out, the last element in the returned collection is the reservation object.

© Copyright IBM Corporation 2011

Figure 8-16. versionable.get\_Versions()

F1432.0

### Notes:

Retrieve versionable objects

## versionable.get\_MajorVersionNumber()



```
Integer get_MajorVersionNumber()
```

- Returns
  - The major version number of this document version
- Notes
  - The major version number of a document is set to 1 the first time that you check it in as a major version.
  - The version number is incremented by 1 for each additional time that you check it in as a major version.

© Copyright IBM Corporation 2011

Figure 8-17. versionable.get\_MajorVersionNumber()

F1432.0

### Notes:

Retrieve versionable objects

## versionable.get\_MinorVersionNumber()



```
Integer get_MinorVersionNumber()
```

- Returns
  - The minor version number of this document version

- Notes
  - The minor version number of a document is set to 1 the first time that you check it in as a minor version.
  - The version number is incremented by 1 for each additional time that you check it in as a minor version.
  - If you check in a document as a major version, its minor version number is reset to 0.

© Copyright IBM Corporation 2011

Figure 8-18. versionable.get\_MinorVersionNumber()

F1432.0

### Notes:

Retrieve versionable objects

## versionable.get\_VersionStatus()



```
VersionStatus get_VersionStatus()
```

- Returns
  - A constant that indicates the current version status of this document version
- The VersionStatus property can have one of the following values:
  - IN\_PROCESS
  - RELEASED
  - RESERVATION
  - SUPERSEDED

© Copyright IBM Corporation 2011

Figure 8-19. versionable.get\_VersionStatus()

F1432.0

### Notes:

Retrieve versionable objects

## Sample code to retrieve versions for a document

```
...
VersionableSet versions =
 document.get_Versions();
Versionable versionable;
Iterator it = versions.iterator();
while (it.hasNext()){
 versionable = (Versionable)it.next();
 Integer majorNum =
 versionable.get_MajorVersionNumber();
 Integer minorNum =
 versionable.get_MinorVersionNumber();
 String versionStatus =
 versionable.get_VersionStatus().toString();
 System.out.println(", Version Status =" +
versionStatus);
}
```

© Copyright IBM Corporation 2011

Figure 8-20. Sample code to retrieve versions for a document

F1432.0

### Notes:

Retrieve versionable objects

## Demonstrations



- Retrieve versions for a document
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 8-21. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *showVersionsEDU(...)* method in the *VersionsEDU.java* file.
3. Run the code and verify the results as instructed in the *Document Versions* unit in the Student Exercises book.

Retrieve versionable objects

## Steps to retrieve version series for a document

- Retrieve the VersionSeries object.

```
document.get_VersionSeries()
```

- Get the released version and its version numbers.

```
versionSeries.get_RetrievedVersion()
releasedDoc.get_MajorVersionNumber()
releasedDoc.get_MinorVersionNumber()
```

- Get the current version and its version numbers.

```
versionSeries.get_CurrentVersion()
CurMajorDoc.get_MajorVersionNumber()
CurMajorDoc.get_MinorVersionNumber()
```

© Copyright IBM Corporation 2011

Figure 8-22. Steps to retrieve version series for a document

F1432.0

### Notes:

Retrieve versionable objects

## versionable.get\_VersionSeries()



```
VersionSeries get_VersionSeries()
```

- Returns
  - A VersionSeries object that contains all of the document versions associated with this document

© Copyright IBM Corporation 2011

Figure 8-23. versionable.get\_VersionSeries()

F1432.0

### Notes:

Retrieve versionable objects

## versionSeries.get\_RetrievedVersion()



```
Versionable get_RetrievedVersion()
```

- Returns
  - A document version that represents the latest released version associated with this document or version series
- Notes
  - A released version is a major version that has a VersionStatus property value of RELEASED.
  - Only one document version at a time in a given version series can be in the released state.

© Copyright IBM Corporation 2011

Figure 8-24. versionSeries.get\_RetrievedVersion()

F1432.0

### Notes:

Retrieve versionable objects

## versionSeries.get\_CurrentVersion()



```
Versionable get_CurrentVersion()
```

- Returns
  - A document version that represents the latest checked-in version associated with this document or version series
- Notes
  - The current version can be either a major version or a minor version.

© Copyright IBM Corporation 2011

Figure 8-25. versionSeries.get\_CurrentVersion()

F1432.0

### Notes:

Retrieve versionable objects

## Sample code to retrieve a version series (1)



```
...
VersionSeries versionSeries =
 document.get_VersionSeries();
Document releasedDoc =
 (Document)versionSeries.get_ReleasedVersion();
Integer majorNum =
 releasedDoc.get_MajorVersionNumber();
Integer minorNum =
 releasedDoc.get_MinorVersionNumber();
...
...
```

© Copyright IBM Corporation 2011

Figure 8-26. Sample code to retrieve a version series (1)

F1432.0

### Notes:

Retrieve versionable objects

## Sample code to retrieve a version series (2)

```
Document CurMajorDoc =
 (Document)versionSeries.get_CurrentVersion();
Integer CurMajorNum =
 CurMajorDoc.get_MajorVersionNumber();
Integer CurMinorNum =
 CurMajorDoc.get_MinorVersionNumber();
System.out.println("Current major version =" +
 CurMajorNum);
System.out.println("Current minor version =" +
 CurMinorNum);
```

© Copyright IBM Corporation 2011

Figure 8-27. Sample code to retrieve a version series (2)

F1432.0

### Notes:

Retrieve versionable objects

## Demonstrations



- Retrieve VersionSeries object for a document
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 8-28. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *showVersionSeriesEDU (...)* method in the *VersionsEDU.java* file.
3. Run the code and verify the results as instructed in the *Document Versions* unit in the Student Exercises book.

Retrieve versionable objects

## Activities



In your Student Exercises book

- Unit: Document Versions
- Lesson: Retrieve versionable objects
- Activities
  - Create a document and document versions.
  - Retrieve versionable objects.

© Copyright IBM Corporation 2011

Figure 8-29. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.

## **Lesson 8.2. Check out and check in a document**

## Lesson:

# Check out and check in a document



- Why is this lesson important to you?
  - Allen and Amanda are two product development authors. Allen writes a draft for a new document and adds it to the Content Engine. Amanda needs to check out Allen's draft, review and edit it, and check it back in. As their programmer, you are going to write code to check out and check in a document.
  - Amanda has checked out Allen's draft. Allen wants to add some more content before Amanda reviews it. Amanda is going to release the document to Allen by canceling her checkout. As their programmer, you are going to write code to cancel the checkout of a document.

© Copyright IBM Corporation 2011

Figure 8-30. Lesson: Check out and check in a document

F1432.0

## Notes:

Check out and check in a document

## Activities that you need to complete



- Check out, check in, and cancel checkout of a document.

© Copyright IBM Corporation 2011

Figure 8-31. Activities that you need to complete

F1432.0

### Notes:

Check out and check in a document

## Steps to check out a document

1. Get the document.

```
Factory.Document.fetchInstance(...)
```

2. If the version is not the current version, retrieve the current version.

```
document.get_IsCurrentVersion().booleanValue()
document.get_CurrentVersion()
```

3. Verify that the document is not reserved.

```
document.get_IsReserved().booleanValue()
```

4. Check out the document and save the changes.

```
document.checkout(...)
document.save(...)
```

© Copyright IBM Corporation 2011

Figure 8-32. Steps to check out a document

F1432.0

### Notes:

Check out and check in a document

## versionable.get\_IsCurrentVersion()



```
Boolean get_IsCurrentVersion()
```



- Returns
  - Whether this document version is the current (latest) version object of a document version series (true) or not (false).
- Notes
  - The current version can be a major version or a minor version.

© Copyright IBM Corporation 2011

Figure 8-33. versionable.get\_IsCurrentVersion()

F1432.0

### Notes:

Check out and check in a document

## document.get\_CurrentVersion()



```
Versionable get_CurrentVersion()
```

- Returns
  - A document version that represents the latest checked-in version associated with this document or version series.
- Notes
  - The current version can be either a major version or a minor version.

© Copyright IBM Corporation 2011

---

Figure 8-34. document.get\_CurrentVersion()

F1432.0

### Notes:

Check out and check in a document

## versionable.get\_IsReserved()



```
Boolean get_IsReserved()
```

- Returns
  - A Boolean value (true or false)
  - For a Document object, this value specifies whether a user has reserved the right to check in the next version following this document version (true) or not (false).
  - For a VersionSeries object, this value specifies whether the current version in this version series is checked out (true) or not (false).

© Copyright IBM Corporation 2011

Figure 8-35. versionable.get\_IsReserved()

F1432.0

### Notes:

Check out and check in a document

## versionable.checkout(...)



```
void checkout(ReservationType type,
 Id reservationId,
 String reservationClass,
 Properties reservationProperties)
```

- Parameters
  - type - Specifies the type of checkout reservation
  - reservationId - Specifies a GUID for the reservation object that is created
  - reservationClass - Specifies the symbolic name of the class
  - reservationProperties - Specifies a Properties object that is carried over to its reservation object

© Copyright IBM Corporation 2011

Figure 8-36. versionable.checkout(...)

F1432.0

### Notes:

#### Parameters

**type** – To specify the type of checkout reservation, use the following ReservationType constants:

- ReservationType.COLLABORATIVE
- ReservationType.EXCLUSIVE
- ReservationType.OBJECT\_STORE\_DEFAULT

### Notes:

To successfully check out a document, the document must

- Be the current version (IsCurrentVersion property is set to true).
- Not be already checked out (IsReserved property is set to false).
- Be version-enabled (IsVersioningEnabled property is set to true).

The user must have the appropriate access rights (MINOR\_VERSION to check out a minor version, or MAJOR\_VERSION to check out a released version).

Check out and check in a document

## Sample code to check out a document

- 
- If the version is not the current version, retrieve the current version.
  - Verify that the document is not reserved.

```
if (document.get_IsCurrentVersion().booleanValue()
 == false)
 document = (Document)document.get_CurrentVersion();
if (document.get_IsReserved().booleanValue() ==
 false) {
 document.checkout(ReservationType.EXCLUSIVE,
 null, null, null);
 document.save(RefreshMode.REFRESH);
}
```

© Copyright IBM Corporation 2011

Figure 8-37. Sample code to check out a document

F1432.0

### Notes:

Check out and check in a document

## Demonstrations



- Check out a document.
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 8-38. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *checkoutEDU(...)* method in the *VersionsEDU.java* file.
3. Run the code and verify the results as instructed in the *Document Versions* unit in the Student Exercises book.

Check out and check in a document

## Steps to check in a document

1. If the version is not the current version, retrieve the current version.

```
document.get_IsCurrentVersion().booleanValue()
document.get_CurrentVersion()
```

2. Verify that the document is reserved.

```
document.get_IsReserved().booleanValue()
```

3. If the version status is not RESERVATION, retrieve the reservation object.

```
document.get_VersionStatus().getValue()
document.get_Reservation()
```

4. Set the content. (Refer to the “Documents” unit.)

5. Check in the document and save the changes.

```
document.checkin(...)
document.save(...)
```

© Copyright IBM Corporation 2011

Figure 8-39. Steps to check in a document

F1432.0

### Notes:

Check out and check in a document

## versionable.get\_Reservation()



```
IndependentObject get_Reservation()
```

- Returns

- The reservation object of version series of this object
- This object has a VersionStatus property with a value of RESERVATION.

- Notes

- A reservation object is created by the server when you check out a document, which you can modify and check in as a new version of that document.

© Copyright IBM Corporation 2011

Figure 8-40. versionable.get\_Reservation()

F1432.0

### Notes:

Check out and check in a document

## document.checkin(...)



```
void checkin(AutoClassify classify,
 CheckinType type)
```

- Parameters

- autoClassify: Specifies whether or not to enable automatic document classification using an AutoClassify constant
- checkinType: Specifies whether to check in the document as a minor or major version using a CheckinType constant

- Notes

- Before a user can check in a document that has content, the user must set the content for the document.

© Copyright IBM Corporation 2011

Figure 8-41. document.checkin(...)

F1432.0

### Notes:

The user must have the appropriate access rights to check in the document:

- Permission.RIGHT\_MINOR\_VERSION to check in a minor version
- Permission.RIGHT\_MAJOR\_VERSION to check in a major version

If the reservation object is an exclusive reservation (RESERVATION\_TYPE\_EXCLUSIVE), the user must be the reservation owner to check in the document.

If the checkin is successful, the reservation object becomes the new current version of the document.

### Parameters

#### Values for AutoClassify constants:

AutoClassify.AUTO\_CLASSIFY - The document is auto-classified during check-in.

AutoClassify.DO\_NOT\_AUTO\_CLASSIFY - The document is not auto-classified during check-in.

**Values for CheckinType constants:**

CheckinType.MAJOR\_VERSION - A document reservation is checked in as a new major version.

CheckinType.MINOR\_VERSION - A document reservation is checked in as a new minor version.

Check out and check in a document

## Sample code to check the version status

```
if (document.get_IsCurrentVersion().booleanVal
ue() == false) {
 document= (Document)
 document.get_CurrentVersion();
}

if (document.get_IsReserved().booleanValue() ==
true{
 if (document.get_VersionStatus().getValue() !=
 VersionStatus.RESERVATION_AS_INT))
{
 Document reservationDoc = (Document)
 document.get_Reservation();
}
```

© Copyright IBM Corporation 2011

Figure 8-42. Sample code to check the version status

F1432.0

### Notes:

Check out and check in a document

## Sample code to check in a document

```
ContentElementList contentList =
 Factory.ContentElement.createList();
ContentTransfer content1 =
 Factory.ContentTransfer.createInstance();
content1.setCaptureSource(new
 FileInputStream("Model 200.GIF"));
content1.set_ContentType("image/jpeg");
contentList.add(content1);
document.set_ContentElements(contentList);
document.checkin(AutoClassify.DO_NOT_AUTO_C
LASSIFY, CheckinType.MINOR_VERSION);
document.save(RefreshMode.REFRESH);
}
```

© Copyright IBM Corporation 2011

Figure 8-43. Sample code to check in a document

F1432.0

### Notes:

Check out and check in a document

## Demonstrations



- Check in a document.
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 8-44. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *checkinEDU(...)* method in the *VersionsEDU.java* file.
3. Run the code and verify the results as instructed in the *Document Versions* unit in the Student Exercises book.

Check out and check in a document

## Steps to cancel a checked-out document

1. If the version is not the current version, retrieve the current version.

```
document.get_IsCurrentVersion().booleanValue()
document.get_CurrentVersion()
```

2. Verify that the document is reserved.

```
document.get_IsReserved().booleanValue()
```

3. Cancel the checkout.

```
document.cancelCheckout()
```

4. Save the changes (to the reservation object).

```
document.save(...)
```

© Copyright IBM Corporation 2011

Figure 8-45. Steps to cancel a checked-out document

F1432.0

### Notes:

Check out and check in a document

## versionable.cancelCheckout()

### Versionable cancelCheckout()

- Returns
  - A reservation object (versionable object) for which a delete pending action has been created

- Notes
  - After a successful call to cancelCheckout, the Content Engine performs the following steps on the reserved document version:
    - Sets the IsReserved property to false.
    - Sets the ReservationType property to null.
    - Deletes its reservation object.

© Copyright IBM Corporation 2011

Figure 8-46. versionable.cancelCheckout()

F1432.0

### Notes:

This method cancels the checkout reservation held on this document or version series by deleting the reservation object associated with it.

Any changes made to the reservation object are lost.

If the reservation object is an exclusive reservation (the ReservationType property of the object is set to EXCLUSIVE), the user must be the reservation owner (value of the reservation object Creator property) or have both WRITE\_OWNER and DELETE access rights.

Regardless of whether the reservation object is an exclusive or a collaborative reservation, the user must also have appropriate access rights on the reservation object (MINOR\_VERSION or MAJOR\_VERSION) to cancel the checkout.

Calling this method is the same as deleting a reserved document reservation object.

Check out and check in a document

## Sample code to cancel a checkout of a document

- Verify that the document
  - Is the current version.
  - Is reserved (is checked out).

```
if(document.get_IsCurrentVersion() .booleanVal
ue ()== false) {
document=(Document)
 document.get_CurrentVersion();
}
if (document.get_IsReserved() .booleanValue ()
== true{
 Document reservationdoc =
 (Document) document.cancelCheckout();
 reservationdoc.save(RefreshMode.REFRESH);
}
```

© Copyright IBM Corporation 2011

Figure 8-47. Sample code to cancel a checkout of a document

F1432.0

### Notes:

Notice that, after cancelling the checkout, the returned reservation object must be saved for the changes to persist.

Check out and check in a document

## Demonstrations



- Cancel a checkout of a document.
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 8-48. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *cancelCheckoutEDU(...)* method in the *VersionsEDU.java* file.
3. Notice that, after cancelling the checkout, the returned reservation object needs to be saved in order for the changes to persist.
4. Run the code and verify the results as instructed in the *Document Versions* unit in the Student Exercises book.

Check out and check in a document

## Activities

In your Student Exercises book

- Unit: Document Versions
- Lesson: Check out and check in a document
- Activities
  - Check out, check in, and cancel checkout of a document.

© Copyright IBM Corporation 2011

Figure 8-49. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.



## **Lesson 8.3. Promote and demote a document**

## Lesson: Promote and demote a document

- 
- Why is this lesson important to you?
    - Tom reviews the documents drafted by the team and releases them to the general audience if they are satisfactory. One of the released documents has an incorrect entry for the product price. He needs to demote that document to draft version status. As their programmer, you are going to write code to promote and demote a version for a document.

© Copyright IBM Corporation 2011

Figure 8-50. Lesson: Promote and demote a document

F1432.0

### **Notes:**

Promote and demote a document

## Activities that you need to complete



- Promote and demote a document.

© Copyright IBM Corporation 2011

Figure 8-51. Activities that you need to complete

F1432.0

### Notes:

Promote and demote a document

## Steps to promote a version for a document

1. If the version is not the current version, retrieve the current version.

```
document.get_IsCurrentVersion().booleanValue()
document.get_CurrentVersion()
```

2. Verify that the document is the minor version.

```
document.get_VersionStatus().getValue()
```

3. Verify that the document is not reserved.

```
document.get_IsReserved().booleanValue()
```

4. Promote the document and save the changes.

```
document.promoteVersion()
document.save()
```

© Copyright IBM Corporation 2011

Figure 8-52. Steps to promote a version for a document

F1432.0

### Notes:

Promote and demote a document

## versionable.promoteVersion()



```
void promoteVersion()
```

- This method promotes an unreleased minor version of the document to a released major version.
- This document must
  - Be the latest minor version (VersionStatus = IN\_PROCESS).
  - Be the current version (IsCurrentVersion = true).
  - Not be reserved (IsReserved = false).
- The user must have appropriate access rights.
- The previous major document version is superseded.

© Copyright IBM Corporation 2011

Figure 8-53. versionable.promoteVersion()

F1432.0

### Notes:

Promote and demote a document

## Sample code to promote a version for a document

- Verify that the document
  - Is the current version.
  - Is the latest minor version.
  - Is not reserved (not checked out).

```
if (document.get_IsCurrentVersion()
 .booleanValue() == false)
 document=(Document)document.get_CurrentVersion();
if ((document.get_VersionStatus().getValue() ==
 VersionStatus.IN_PROCESS_AS_INT) &&
 (document.get_IsReserved().booleanValue() ==
 false))
{
 document.promoteVersion();
 document.save(RefreshMode.REFRESH);
}
```

© Copyright IBM Corporation 2011

Figure 8-54. Sample code to promote a version for a document

F1432.0

### Notes:

Promote and demote a document

## Demonstrations



- Promote a version for a document
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 8-55. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *promoteEDU(...)* method in the *VersionsEDU.java* file.
3. Run the code and verify the results as instructed in the *Document Versions* unit in the Student Exercises book.

Promote and demote a document

## Steps to demote a version for a document

1. If the version is not the current version, retrieve the current version.

```
document.get_IsCurrentVersion().booleanValue()
document.get_CurrentVersion()
```

2. Verify that the document is the released version.

```
document.get_VersionStatus().getValue()
```

3. Verify that the document is not reserved.

```
document.get_IsReserved().booleanValue()
```

4. Demote the document and save the changes.

```
document.demoteVersion()
document.save(...)
```

© Copyright IBM Corporation 2011

Figure 8-56. Steps to demote a version for a document

F1432.0

### Notes:

Promote and demote a document

## versionable.DemoteVersion()



```
void demoteVersion()
```

- This method
  - Demotes the latest major version to an unreleased minor version.
  - Changes the previous major version (if it exists) to the current released major version.
  - Does not delete any document versions (including content) from the document version series.
- This document must
  - Be the latest major version (VersionStatus = RELEASED).
  - Be the current version (IsCurrentVersion = true).
  - Not be reserved (IsReserved = false).
- The user must have appropriate access rights.

© Copyright IBM Corporation 2011

Figure 8-57. versionable.DemoteVersion()

F1432.0

### Notes:

Promote and demote a document

## Sample code to demote a version for a document

- 
- Verify that the document is the current version, the released version, and that it is not reserved.

```
if (document.get_IsCurrentVersion() .booleanValue ()
 == false)

 document =
 (Document) document.get_CurrentVersion () ;

 if ((document.get_VersionStatus () .getValue () ==
 VersionStatus.RELEASED_AS_INT) &&
 (document.get_IsReserved () .booleanValue () ==
 false)) {

 document.demoteVersion () ;
 document.save (RefreshMode.REFRESH) ;
 showVersionsEDU (document) ;
 showVersionSeriesEDU (document) ;

 }
}
```

© Copyright IBM Corporation 2011

Figure 8-58. Sample code to demote a version for a document

F1432.0

### Notes:

Promote and demote a document

## Demonstrations



- Demote a version for a document
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 8-59. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *demoteEDU(...)* method in the *VersionsEDU.java* file.
3. Run the code and verify the results as instructed in the *Document Versions* unit in the Student Exercises book.

Promote and demote a document

## Activities



In your Student Exercises book

- Unit: Document Versions
- Lesson: Promote and demote a document
- Activities
  - Promote and demote a document.

© Copyright IBM Corporation 2011

Figure 8-60. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.

# Unit 9. Security

## What this unit is about

This unit describes how to get and set object permissions, how to apply security template and how to retrieve the security users and groups.

## What you should be able to do

After completing this unit, you should be able to:

- Retrieve object permissions.
- Set object permissions.
- Apply a security template.
- Retrieve users and groups.

## How you will check your progress

- Successful completion of lesson exercises.

## References

[www.ibm.com/support/documentation/us/en](http://www.ibm.com/support/documentation/us/en) (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Security

## Unit lessons



- Retrieve object permissions
- Set object permissions
- Apply a security template
- Retrieve security users and groups

© Copyright IBM Corporation 2011

Figure 9-1. Unit lessons

F1432.0

### Notes:

## **Lesson 9.1. Retrieve object permissions**

## Lesson: Retrieve object permissions

- 
- Why is this lesson important to you?
    - Administrators want to be able to see what security permissions a particular folder or a document has without leaving the application. As the programmer, you are going to add this functionality to the application.

© Copyright IBM Corporation 2011

Figure 9-2. Lesson: Retrieve object permissions

F1432.0

### Notes:

Retrieve object permissions

## Activities that you need to complete



- Retrieve object permissions.

© Copyright IBM Corporation 2011

---

Figure 9-3. Activities that you need to complete

F1432.0

### Notes:

Retrieve object permissions

## Object-level security



- Authorization:
  - Does the requestor have permission to do what he or she is asking to do to the specified object?
- Securable versus nonsecurable objects
  - Most objects are secured by their own set of access rights
    - Example: A document
    - Some objects depend on access rights of the associated object
      - Example: The content element associated with a document
- Security grantees are specified on each object
  - Also known as “security principal”
  - Typically a user or group

© Copyright IBM Corporation 2011

Figure 9-4. Object-level security

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Security overview
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Object Security

Assets managed in the Content Engine are protected and can be seen or modified only based on a user's access rights.

Each Content Engine object has its own security, which controls user (or group) access to that object.

A grantee is either an individual user or a group of users. A user can be a member of zero, one, or more groups, and a group can contain zero, one, or more subgroups. A user account is defined on the Content Engine server using tools provided by the directory service, for example, Microsoft Active Directory or Sun ONE Directory Services.

Retrieve object permissions

## Access control objects and properties



- Access rights for an object are determined by Access Control List (ACL)
  - Made up of one or more Access Control Entries (ACE)
  - An ACE is also called a permission
- Each ACE specifies the following:
  - A security grantee (a user or group)
  - A permission source (Example: Direct or Security Policy)
  - Allow or Deny access type
  - Access level
    - Examples: View Content, Full Control, Custom
  - Apply To
    - Example: This object only or this object and immediate children

© Copyright IBM Corporation 2011

Figure 9-5. Access control objects and properties

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Security overview
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Authorization

Each ACE is represented by a Permission object, which establishes the access granted to a user or group (the grantee), and contains a reference to the user or group being assigned the permission.

A Permissions collection represents the full set of Access Control Entries (ACEs) associated with an object and forms the object Access Control List (ACL).

Retrieve object permissions

## Permission sources



- Default permissions are from
  - The Default Instance Security ACL of its class
  - The parent class to its subclass
- Inherited permissions are from a security parent
- Direct permissions are set directly on the object
  - They override default security
- Template permissions are from a security policy

© Copyright IBM Corporation 2011

Figure 9-6. Permission sources

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 security > Authorization > About access rights
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Authorization

Retrieve object permissions

## Steps to get object permissions

1. Retrieve the object from the object store.

```
Factory.Folder.fetchInstance(...)
```

2. Get permissions collection on the object (Example: a folder).

```
folder.get_Permissions()
```

3. For each permission, get grantee name, grantee type, permission source, access level, and access type.

```
accessPermission.get_GranteeName()
accessPermission.get_GranteeType()
accessPermission.get_PermissionSource()
accessPermission.get_AccessMask()
discretionaryPermission.get_AccessType()
```

© Copyright IBM Corporation 2011

Figure 9-7. Steps to get object permissions

F1432.0

### Notes:

Retrieve object permissions

## containable.get\_Permissions()



```
AccessPermissionList get_Permissions()
```

- Returns
  - A Permissions collection containing the basic object permissions for this object
- Notes
  - The user must have Read permission on the object security to retrieve object permissions
    - AccessLevel.VIEW or
    - AccesRight.READ\_ACL
  - This method filters out Access Control Entries (ACEs) that correspond to deleted grantees (users or groups).

© Copyright IBM Corporation 2011

Figure 9-8. containable.get\_Permissions()

F1432.0

### Notes:

Retrieve object permissions

## accessPermission.get\_GranteeName()



```
String get_GranteeName()
```

- Returns
  - A string representing the name of the grantee associated with this particular permission item
- Notes
  - The returned string is the displayable user or group name (Principal Name) that can appear in a user interface display.

© Copyright IBM Corporation 2011

Figure 9-9. accessPermission.get\_GranteeName()

F1432.0

### Notes:

Retrieve object permissions

## accessPermission.get\_GranteeType()



```
SecurityPrincipalType get_GranteeType()
```

- Returns
  - The object type of the permission grantee
    - SecurityPrincipalType.GROUP
    - SecurityPrincipalType.USER

© Copyright IBM Corporation 2011

Figure 9-10. accessPermission.get\_GranteeType()

F1432.0

### Notes:

Retrieve object permissions

## accessPermission.get\_PermissionSource()

`PermissionSource get_PermissionSource()`

- Returns
  - The source of the permission.
    - PermissionSource.SOURCE\_DIRECT
    - PermissionSource.SOURCE\_DEFAULT
    - PermissionSource.SOURCE\_TEMPLATE
    - PermissionSource.SOURCE\_PARENT
    - PermissionSource.PROXY

© Copyright IBM Corporation 2011

Figure 9-11. accessPermission.get\_PermissionSource()

F1432.0

### Notes:

The source of access rights can be any of the following:

- A security template
- Inheritance from a parent object
- Default access rights from the class from which the object was instantiated
- Direct application (that is, by programmatically setting permissions with a method call)
- Permissions originating from a security proxy

Retrieve object permissions

## accessPermission.get\_AccessMask()



```
Integer get_AccessMask()
```

- Returns
  - An integer representing the type of permission assigned to a user or group

© Copyright IBM Corporation 2011

Figure 9-12. accessPermission.get\_AccessMask()

F1432.0

### Notes:

Retrieve object permissions

## accessPermission.get\_AccessType()

`AccessType get_AccessType()`

- Returns

- An integer that indicates whether a user or group is allowed or denied the associated security access

- Notes

- 1 = AccessType.ALLOW

- 2 = AccessType.DENY

- This method is inherited from the `DiscretionaryPermission` interface.

© Copyright IBM Corporation 2011

Figure 9-13. `accessPermission.get_AccessType()`

F1432.0

### Notes:

Retrieve object permissions

## Sample code to get object permissions (1)

```
...
AccessPermissionList permissions=
 folder.get_Permissions();
Iterator it = permissions.iterator();
while (it.hasNext()){
 AccessPermission permission =
 (AccessPermission)it.next();
 System.out.println("Grantee Name=" +
 permission.get_GranteeName());
 System.out.println("Grantee Type=" +
 permission.get_GranteeType().toString());
 System.out.println("Permission Source=" +
 permission.get_PermissionSource().toString());
...
}
```

© Copyright IBM Corporation 2011

Figure 9-14. Sample code to get object permissions (1)

F1432.0

### Notes:

Retrieve object permissions

## Sample code to get object permissions (2)

```
System.out.println("Access level = " +
AccessLevel.getInstanceFromInt(permission.
get_AccessMask().intValue()).toString());
System.out.println("Access type =" +
permission.get_AccessType().toString());
System.out.println("Inheritable depth=" +
permission.get_InheritableDepth());
}
```

© Copyright IBM Corporation 2011

Figure 9-15. Sample code to get object permissions (2)

F1432.0

### Notes:

Retrieve object permissions

## Demonstrations



- Retrieve permissions of a Content Engine object
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 9-16. Demonstrations

F1432.0

### **Notes:**

#### **DEMONSTRATION —**

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getPermissionsEDU (...)* method in the *SecurityEDU.java* file.
3. Run the code and verify the results as instructed in the *Security* unit in the Student Exercises book.

Retrieve object permissions

## Activities

In your Student Exercises book

- Unit: Security
- Lesson: Retrieve object permissions
- Activities
  - Retrieve object permissions.

© Copyright IBM Corporation 2011

Figure 9-17. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.



## **Lesson 9.2. Set object permissions**

## Lesson: Set object permissions

- 
- Why is this lesson important to you?
    - A number of older documents in the object store do not have the desired security settings. As the programmer, you are going to write code to update the security permissions on these documents so that they are the same as the folder they are filed in.

© Copyright IBM Corporation 2011

Figure 9-18. Lesson: Set object permissions

F1432.0

### Notes:

Set object permissions

## Activities that you need to complete



- Set up the folder and document.
- Set object permissions.

© Copyright IBM Corporation 2011

Figure 9-19. Activities that you need to complete

F1432.0

### Notes:

Set object permissions

## Security inheritance



- An object can inherit permissions from the following:
  - A security parent
  - A security policy
  - Both
    - If conflicts occur, *Deny* takes precedence over *Allow*.
- Inherited parent security example:
  - A document inherits permissions from its parent folder.
- Inherited policy security example:
  - A document inherits permissions from an applied policy.
- Inherited permissions are added to existing permissions.

© Copyright IBM Corporation 2011

Figure 9-20. Security inheritance

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Security overview > How is security applied?
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Security Inheritance

Set object permissions

## Setting security with a security parent



- Example: A folder is a security parent for a document.
- The security folder must be enabled to allow permissions inheritance.
  - Security folder can allow inheritance to immediate children or all children (or no inheritance).
  - Inheritance for each permission (ACE) is set independently.
- The child must also be set to allow permissions to be inherited from the parent.
  - Applies to all the parent ACEs that are set to allow inheritance.

© Copyright IBM Corporation 2011

Figure 9-21. Setting security with a security parent

F1432.0

### **Notes:**

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Security overview > How is security applied?
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Security Inheritance

For documents, do one of the following:

- Use the `setSecurityFolder(...)` API method.
- Select the “Inherit Security from folder” check box in Enterprise Manager.

When an item is added to a folder, the item does not immediately inherit the folder security. The item must also have its security folder set after it has been added to the folder.

Set object permissions

## Steps to set permissions on a folder (1)

1. Create a new Permission object.

```
Factory.AccessPermission.createInstance()
```

2. Set the values for the new permission.

```
accessPermission.set_GranteeName(...)
accessPermission.set_AccessType(...)
accessPermission.set_InheritableDepth(...)
accessPermission.set_AccessMask(...)
```

3. Get the existing Permissions collection from the folder.

```
folder.get_Permissions()
```

© Copyright IBM Corporation 2011

Figure 9-22. Steps to set permissions on a folder (1)

F1432.0

### Notes:

### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Working with Security > Setting Permissions

Set object permissions

## Steps to set permissions on a folder (2)



4. Add the Permission object to the collection.

```
permissions.add(...)
```

5. Apply the Permissions collection and save the changes to the folder.

```
Folder.setPermissions(...)
```

```
Folder.save(...)
```

© Copyright IBM Corporation 2011

Figure 9-23. Steps to set permissions on a folder (2)

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Working with Security > Setting Permissions

Set object permissions

## Factory.AccessPermission.createInstance(...)



```
public static AccessPermission createInstance()
```

- Returns
  - An object reference to a new instance of this class
- Notes
  - This method creates a new instance of the AccessPermission class.
  - The created object is a dependent object.
  - It is persisted only when its parent object is persisted.

---

© Copyright IBM Corporation 2011

Figure 9-24. Factory.AccessPermission.createInstance(...)

F1432.0

### Notes:

Set object permissions

## accessPermission.set\_GranteeName()



```
void set_GranteeName(String value)
```

- Parameters

- value - Distinguished name or short name of the user or group whom an access permission is granted

© Copyright IBM Corporation 2011

Figure 9-25. accessPermission.set\_GranteeName()

F1432.0

### Notes:

Set object permissions

## accessPermission.set\_AccessType(...)



```
void set_AccessType(AccessType value)
```

- Parameters

- value - The security access type (Allow or Deny) that a user has for a given AccessPermission object

© Copyright IBM Corporation 2011

Figure 9-26. accessPermission.set\_AccessType(...)

F1432.0

### Notes:

Set object permissions

## accessPermission.set\_InheritableDepth(...)



```
void set_InheritableDepth (Integer value)
```

- Parameters

- value - Specifies the maximum depth to which a permission (ACL) can be inherited

- Notes

- As the ACL gets inherited in a tree of objects, the value is decremented.
  - Values allowed:
    - 0 - No inheritance
    - 1 - Immediate children only.
    - -1 - All children (infinite levels deep)

© Copyright IBM Corporation 2011

Figure 9-27. accessPermission.set\_InheritableDepth(...)

F1432.0

### Notes:

Set object permissions

## accessPermission.set\_AccessMask()



```
void set_AccessMask(Integer value)
```

- Parameters

- value - Bitmask combining bit values representing the security access rights granted on a given object

© Copyright IBM Corporation 2011

Figure 9-28. accessPermission.set\_AccessMask()

F1432.0

### Notes:

Set object permissions

## folder.set\_Permissions(...)



```
void set_Permissions (AccessPermissionList
 value)
```

- Parameters

- value - Specifies a collection of AccessPermission objects in an AccessPermissionList object

© Copyright IBM Corporation 2011

Figure 9-29. folder.set\_Permissions(...)

F1432.0

### Notes:

Set object permissions

## Sample code to set folder permissions

```
...
AccessPermission permission =
 Factory.AccessPermission.createInstance();
permission.set_GranteeName("Managers");
permission.set_AccessType(AccessType.ALLOW);
permission.set_InheritableDepth
 (new Integer(-1));
permission.set_AccessMask(new Integer
 (AccessLevel.FULL_CONTROL_FOLDER_AS_INT));
permissions.add(permission);
AccessPermissionList permissions =
 folder.get_Permissions();
folder.set_Permissions(permissions);
folder.save(RefreshMode.REFRESH);
```

© Copyright IBM Corporation 2011

Figure 9-30. Sample code to set folder permissions

F1432.0

### Notes:

Set object permissions

## Steps to set the security folder to a document

1. Get the document object.

```
Factory.Document.fetchInstance(...)
```

2. Get the containers of the document.

```
document.getContainers(...)
```

3. Set the security folder of the document.

```
document.setSecurityFolder(...)
```

4. Save the changes to the document.

```
document.save(...)
```

© Copyright IBM Corporation 2011

Figure 9-31. Steps to set the security folder to a document

F1432.0

### Notes:

Set object permissions

## document.get\_Containers()



```
public ReferentialContainmentRelationshipSet
getContainers()
```

- Returns

- The ReferentialContainmentRelationshipSet collection object that identifies the containers of this object

© Copyright IBM Corporation 2011

Figure 9-32. document.get\_Containers()

F1432.0

### Notes:

Set object permissions

## document.set\_SecurityFolder(...)

```
void set_SecurityFolder(Folder value)
```

- Parameters

- value - Specifies the object from which this object inherits security

- Notes

- If this ContainableObject object is not filed in the specified folder, then this method also files the object into the specified folder.
  - If this object already has a security parent, then the existing security folder is replaced by the security folder specified in the folder parameter, but the object remains filed in the original folder.

© Copyright IBM Corporation 2011

Figure 9-33. document.set\_SecurityFolder(...)

F1432.0

### Notes:

Set object permissions

## Sample code to set the security folder to a document

```
ReferentialContainmentRelationshipSet rels =
 document.get_Containers();
ReferentialContainmentRelationship rel = null;
Folder parentFolder =null;
Iterator it = rels.iterator();
while (it.hasNext()) {
 rel = (ReferentialContainmentRelationship)
 it.next();
 if(rel.get_Tail().equals(folder)){
 parentFolder =(Folder) rel.get_Tail();
 System.out.println("Parent folder name: "
 + parentFolder.get_FolderName());
 document.set_SecurityFolder(parentFolder);
 document.save(RefreshMode.REFRESH);
 } }
```

© Copyright IBM Corporation 2011

Figure 9-34. Sample code to set the security folder to a document

F1432.0

### Notes:

Set object permissions

## Demonstrations



- Set permissions on a folder
- set the security folder to a document
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 9-35. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *setFolderSecurityEDU(...)* method in the *SecurityEDU.java* file.
3. Run the code and verify the results as instructed in the *Security* unit in the Student Exercises book.
4. Repeat steps 2 and 3 for the *getDocumentEDU(...)* method and the *setSecurityFolderToDocumentEDU(...)* method that sets the security folder to a document.

Set object permissions

## Activities



In your Student Exercises book

- Unit: Security
- Lesson: Set object permissions
- Activities
  - Set up the folder and document.
  - Set object permissions.

© Copyright IBM Corporation 2011

---

Figure 9-36. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.

## **Lesson 9.3. Apply a security template**

## Lesson: Apply a security template

- 
- Why is this lesson important to you?
    - New security measures are being put in place for the application. Access to the documents is restricted to certain users based on the value of a document property. You are going to write code to apply a security template to a document. The code checks the value of a document property and selects the correct template for that document. As a result, the document can be accessed only by the designated users.

© Copyright IBM Corporation 2011

Figure 9-37. Lesson: Apply a security template

F1432.0

### Notes:

Apply a security template

## Activities that you need to complete



- Create a security policy.
- Create documents.
- Apply a security template to an object.

© Copyright IBM Corporation 2011

Figure 9-38. Activities that you need to complete

F1432.0

### Notes:

Apply a security template

## Security policies and templates



- Used for managing object security.
  - Can be applied to a Document, Custom Object, or Folder.
- Security policies consist of one or more security templates.
  - Each security template contains a set of access rights to be applied.
- Security templates can do one of the following:
  - Override direct object security.
  - Add to existing object security.

© Copyright IBM Corporation 2011

Figure 9-39. Security policies and templates

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Authorization > Security policies
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Security Policies

Permissions on an object that originate from a security policy appear on the object ACL with a Source type of Template.

Apply a security template

## Types of security templates



- Versioning security templates
  - Automatically applied to a versionable object as it goes through version state changes
  - Four possible templates:
    - Released
    - Reservation
    - In process
    - Superseded
- Application security templates
  - Must be explicitly applied to an object programmatically.
  - Can apply a list of permissions to an object.

© Copyright IBM Corporation 2011

Figure 9-40. Types of security templates

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Authorization > Security policies
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Security Policies

Apply a security template

## Creating a security template



- Security templates can be created using either of the following:
  - Enterprise Manager
  - Workplace (Advanced Author Tools)
- To create a security template:
  1. Create a security policy.
  2. Add a security template to it.
  3. Add the individual access rights to the template.
  4. Repeat steps 2 and 3 to add additional templates.

© Copyright IBM Corporation 2011

Figure 9-41. Creating a security template

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Authorization > Security policies
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Security Policies

Permissions on an object that originate from a security policy appear on the object ACL with a Source type of Template.

Apply a security template

## Steps to apply a security template



1. Apply a security policy to the object using one of the following:
  - By default for the class
  - Programmatically
2. Assign a specific security template from the policy.

© Copyright IBM Corporation 2011

Figure 9-42. Steps to apply a security template

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Authorization > Security policies
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Security > Concepts > Security Policies

Security policies can be assigned to new instances of a class if the class is set up to have the security policy as the default. In that case, the default security policy is automatically associated with the object instance at the time of creation (unless the default is explicitly overridden).

Apply a security template

## Steps to apply a security policy to a document (1)

1. Retrieve the document collection available in the given folder.

```
folder.get_ContainedDocuments()
```

2. Get the security policies stored in the object store.

```
objectStore.get_SecurityPolicies()
```

3. Get the security policy to use.

```
securityPolicies.iterator();
```

4. Retrieve the security template collection from this policy.

```
securityPolicy.get_SecurityTemplates()
```

© Copyright IBM Corporation 2011

Figure 9-43. Steps to apply a security policy to a document (1)

F1432.0

### Notes:

Apply a security template

## Steps to apply a security policy to a document (2)

- 
5. Retrieve each document and determine its class.

```
document.get_ClassDescription().describedIsOfClass(...)
```

6. Retrieve the document properties collection and assign the security policy.

```
document.getProperties()
properties.putValue(...)
```

7. Determine the value for the document type property, apply the appropriate security template, and save the document.

```
properties.getStringValue(...)
securityTemplate.get_ApplyStateID()
document.applySecurityTemplate(...)
document.save(...)
```

© Copyright IBM Corporation 2011

Figure 9-44. Steps to apply a security policy to a document (2)

F1432.0

### Notes:

Apply a security template

## objectStore.get\_SecurityPolicies()



```
SecurityPolicySet get_SecurityPolicies()
```



- Returns

- Collection of the security policies associated with this ObjectStore object

© Copyright IBM Corporation 2011

Figure 9-45. objectStore.get\_SecurityPolicies()

F1432.0

### Notes:

Apply a security template

## securityPolicy.get\_SecurityTemplates()



```
SecurityTemplateList get_SecurityTemplates()
```



- Returns

- Collection of the security templates associated with a given Security Policy object

© Copyright IBM Corporation 2011

Figure 9-46. securityPolicy.get\_SecurityTemplates()

F1432.0

### Notes:

Apply a security template

## securityTemplate.get\_ApplyStateID()

`Id get_ApplyStateID()`

- Returns
  - An unique identifier (GUID) for the state to which a given security template applies.

- Notes
  - For a VersioningSecurityTemplate, the apply state ID can be one of the following:
    - VersionStatus.IN\_PROCESS
    - VersionStatus.RELEASED
    - VersionStatus.RESERVATION
    - VersionStatus.SUPERSEDED
  - For an ApplicationSecurityTemplate, the apply state ID is defined by your application.

© Copyright IBM Corporation 2011

Figure 9-47. securityTemplate.get\_ApplyStateID()

F1432.0

### Notes:

Apply a security template

## document.applySecurityTemplate(...)

```
void applySecurityTemplate(Id applyStateID)
```

- Parameters

- applyStateID - An Id object representing the unique identifier of the security template to apply

- Notes

- The specified value for the applyStateID parameter must match the value of the ApplyStateID property of one of the templates in the security policy for the object.

© Copyright IBM Corporation 2011

Figure 9-48. document.applySecurityTemplate(...)

F1432.0

### Notes:

## Apply a Security Template

## Sample code to retrieve a security policy

```
DocumentSet documents =
 folder.get_CreatedDocuments();
Iterator documentIt = documents.iterator();
SecurityPolicySet securityPolicies =
 objectStore.get_SecurityPolicies();
SecurityPolicy securityPolicy = null;
Iterator policyIt = securityPolicies.iterator();
while (policyIt.hasNext()){
 securityPolicy = (SecurityPolicy)
 policyIt.next();
 System.out.println("security policy = " +
 securityPolicy.get_Name());
 if(securityPolicy.get_Name().equalsIgnoreCase
 (policyName)){
 SecurityTemplateList securityTemplates =
 securityPolicy.get_SecurityTemplates();
```

© Copyright IBM Corporation 2011

Figure 9-49. Sample code to retrieve a security policy

F1432.0

### Notes:

In the code sample shown, assume the following:

- The folder variable has been assigned a valid folder object in the object store.

## Apply a Security Template

**Sample code to apply a security policy to a document (1)**

```
Iterator securityTemIt =
 securityTemplates.iterator();
Id applyStateId;
SecurityTemplate securityTemplate = null;
while (documentIt.hasNext()) {
 document = (Document)documentIt.next();
 if (document.get_ClassDescription()
 .describedIsOfClass("Product")) {
Properties properties =document.getProperties();
properties.putValue
 (PropertyNames.SECURITY_POLICY,securityPolicy);
System.out.println("Document name: " +
 document.get_Name());
String docStyle =
properties.getStringValue("style");
```

© Copyright IBM Corporation 2011

Figure 9-50. Sample code to apply a security policy to a document (1)

F1432.0

**Notes:**

## Apply a Security Template

**Sample code to apply a security policy to a document (2)**

```
while (securityTemIt.hasNext()) {
 securityTemplate = (SecurityTemplate)
 securityTemIt.next();
 if(docStyle.equalsIgnoreCase("Basic") &&
 securityTemplate.get_DisplayName().equalsIgnoreCase(
 templateName1)){
 System.out.println("Security template name: " +
 securityTemplate.get_DisplayName());
 System.out.println("Security template Id: " +
 securityTemplate.getId().toString());
 applyStateId =
 securityTemplate.get_ApplyStateID();
 System.out.println("Apply state Id: " +
 applyStateId.toString());
```

© Copyright IBM Corporation 2011

Figure 9-51. Sample code to apply a security policy to a document (2)

F1432.0

**Notes:**

## Apply a Security Template

**Sample code to apply a security policy to a document (3)**

```
document.applySecurityTemplate(applyStateId) ;
document.save(RefreshMode.REFRESH) ;}
if(docStyle.equalsIgnoreCase("Deluxe") &&
securityTemplate.get_DisplayName().equalsIgnoreCase(
templateName2)) {
...
document.applySecurityTemplate(applyStateId) ;
document.save(RefreshMode.NO_REFRESH) ;
...}
```

© Copyright IBM Corporation 2011

Figure 9-52. Sample code to apply a security policy to a document (3)

F1432.0

**Notes:**

## Apply a Security Template

# Demonstrations



- Create a security policy.
- Apply a security template to a folder.
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 9-53. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

##### Create a security policy

1. Log on to the Content Engine Enterprise Manager.
2. Create a security policy using the wizard as instructed in the *Security* unit in the Student Exercises book .

##### Apply a security template to a folder

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *setSecurityPolicyEDU (...)* method in the *SecurityEDU.java* file.
3. Run the code and verify the results as instructed in the *Security* unit in the Student Exercises book.

Apply a security template

## Activities

In your Student Exercises book

- Unit: Security
- Lesson: Apply a security template
- Activities
  - Create a security policy.
  - Create documents.
  - Apply a security template to an object.

© Copyright IBM Corporation 2011

Figure 9-54. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.



## **Lesson 9.4. Retrieve security users and groups**

## Lesson: Retrieve security users and groups

- 
- Why is this lesson important to you?
    - The administrators want to be able to find all security users and group names that match a particular pattern, without leaving the application. As their programmer, you are going to add this functionality to the application.

© Copyright IBM Corporation 2011

Figure 9-55. Lesson: Retrieve security users and groups

F1432.0

### **Notes:**

Retrieve security users and groups

## Activities that you need to complete



- Retrieve security users and groups.

© Copyright IBM Corporation 2011

---

Figure 9-56. Activities that you need to complete

F1432.0

### Notes:

Retrieve security users and groups

## What is a realm?



- An abstract entity representing the following:
  - A collection of security principals (users and groups) who have access to objects controlled by the Content Engine
- For Active Directory
  - A realm is the same as the Domain.
- For other directory services
  - A realm is the base object used to search the directory.
  - Details were specified during Content Engine installation.

© Copyright IBM Corporation 2011

Figure 9-57. What is a realm?

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Directory service providers

Retrieve security users and groups

## User and Group objects



- User object
  - Part of the security interface that allows a particular user to access a Content Engine resource
- Group object
  - Reflects a set of user accounts defined by the directory service.
- Creating users, groups, and subgroups
  - Users and groups are created using tools provided by the directory service.
  - You cannot create a new user or group object, but you can instantiate one that has been persisted in the directory service.

© Copyright IBM Corporation 2011

Figure 9-58. User and Group objects

F1432.0

### Notes:

Retrieve security users and groups

## User and group names



- Distinguished Name
  - Uniquely defines a directory entry within an LDAP server.
  - Locates it within the directory tree.
  - Contains a group or user short name plus the name of its domain
  - Example: "CN=Admins,CN=Users,DC=FileNet,DC=com"
- Short Name
  - Represents the unique portion of the distinguished name that does not indicate its location relative to a domain or directory.
  - Example: The short name form of the distinguished name in the previous example is "Admins".
- Principal Name
  - Group or user short name, followed by the "at" sign (@) and the domain name
  - Example: "Administrator@FileNet.com"

© Copyright IBM Corporation 2011

Figure 9-59. User and group names

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Security > IBM FileNet P8 Security > Directory service providers

Retrieve security users and groups

## FileNet User and Group Interfaces



- FileNet P8 Content Java API has interfaces with same name as some Java base interfaces.
- User interface
  - `com.filenet.api.security.User`
- Group interface
  - `com.filenet.api.security.Group`
- Code must give entire name when referenced.

**Examples:**

- `com.filenet.api.security.User user;`
- `com.filenet.api.security.Group group;`

© Copyright IBM Corporation 2011

Figure 9-60. FileNet User and Group Interfaces

F1432.0

### Notes:

Retrieve security users and groups

## Steps to get user names from the directory service



1. Retrieve the EntireNetwork object.

```
Factory.EntireNetwork.fetchInstance(...)
```

2. Retrieve the Realm object.

```
entireNetwork.get_MyRealm()
```

3. Retrieve the users in the realm that match a pattern.

```
realm.findUsers(...)
```

4. Retrieve the name properties.

```
user.get_DistinguishedName()
```

```
user.get_Name()
```

© Copyright IBM Corporation 2011

Figure 9-61. Steps to get user names from the directory service

F1432.0

### Notes:

Retrieve security users and groups

## Factory.EntireNetwork.fetchInstance(...)

```
public static EntireNetwork fetchInstance
(Connection conn, PropertyFilter filter)
```

- Parameters

- conn - Specifies a connection object for establishing the connection to the Content Engine server
- Filter – specify a property filter

- Returns

- An EntireNetwork object

© Copyright IBM Corporation 2011

Figure 9-62. Factory.EntireNetwork.fetchInstance(...)

F1432.0

### Notes:

This method always makes a round-trip to the server. You can optionally include a filter to control which properties to return with the object.

Use the EntireNetwork object to retrieve all of the Realm objects for the FileNet P8 domain, and then retrieve the users and groups associated with a realm.

Retrieve security users and groups

## entireNetwork.get\_MyRealm()



```
public Realm get_MyRealm()
```

- Returns
  - The Realm object in which the current user resides

© Copyright IBM Corporation 2011

Figure 9-63. entireNetwork.get\_MyRealm()

F1432.0

### Notes:

Retrieve security users and groups

## realm.findUsers(...)

```
public UserSet findUsers
 (String searchPattern,
 PrincipalSearchType searchType,
 PrincipalSearchAttribute searchAttribute,
 PrincipalSearchSortType sortType,
 Integer pageSize, PropertyFilter filter)
```

- Parameters
  - **searchPattern** - Specifies a String containing a pattern of characters to search for.
  - **searchType** - Specifies the type of search that you want to conduct.
  - **searchedAttribute** - Specifies the User object property to use.
  - **sortType** - Specifies how to sort the User objects in the returned collection.
  - **pageSize** - Specifies the size of each page of User objects returned.
  - **filter** - Specifies a PropertyFilter object.
- Returns
  - A set of users, a page at a time, from this realm that matches the search parameters.

© Copyright IBM Corporation 2011

Figure 9-64. realm.findUsers(...)

F1432.0

### Notes:

#### Parameters:

**searchPattern:** Case does not matter.

**searchType:** A null value indicates no preference. Possible values:

- PrincipalSearchType.NONE
- PrincipalSearchType.CUSTOM
- PrincipalSearchType.PREFIX\_MATCH
- PrincipalSearchType.SUFFIX\_MATCH
- PrincipalSearchType.CONTAINS
- PrincipalSearchType.EXACT

**searchedAttribute:** A null value indicates none. Possible values:

- PrincipalSearchAttribute.NONE
- PrincipalSearchAttribute.SHORT\_NAME

- PrincipalSearchAttribute.DISPLAY\_NAME

**sortType:** A null value indicates no sorting is to be performed. Possible values:

- PrincipalSearchSortType.NONE
- PrincipalSearchSortType.ASCENDING
- PrincipalSearchSortType.DESCENDING

**pageSize:** If the value is null or zero, page size defaults to 499.

**filter:** If the value is null, this method returns values for all nonobject properties and returns placeholders for all object-valued properties.

Retrieve security users and groups

## Sample code to retrieve the realm

```
...
propertyFilter.addIncludeProperty(0,null,
null, PropertyNames.MY_REALM,1);
EntireNetwork entireNetwork =
Factory.EntireNetwork.fetchInstance
 (connection, propertyFilter);
Realm realm = entireNetwork.get_MyRealm();
String realmName = realm.get_Name();
System.out.println(realmName + "is retrieved");
```

© Copyright IBM Corporation 2011

Figure 9-65. Sample code to retrieve the realm

F1432.0

### Notes:

Retrieve security users and groups

## Sample code to retrieve user names

```
UserSet users = realm.findUsers(pattern,
 PrincipalSearchType.PREFIX_MATCH,
 PrincipalSearchAttribute.SHORT_NAME,
 PrincipalSearchSortType.NONE,
 Integer.valueOf("50"), null);
com.filenet.api.security.User user;
Iterator it = users.iterator();
while (it.hasNext()){
 user = (com.filenet.api.security.User)it.next();
 System.out.println("distinguished name =" +
 user.get_DistinguishedName());
 System.out.println("short name =" +
 user.get_ShortName());
 System.out.println("name =" + user.get_Name());
 System.out.println("display name =" +
 user.get_DisplayName()); }
```

© Copyright IBM Corporation 2011

Figure 9-66. Sample code to retrieve user names

F1432.0

### Notes:

Retrieve security users and groups

## Getting group names from directory service

- Procedure is the same as for getting users except for the following:
  - Get a Groups collection from Realm object instead of a Users collection.

```
realm.findGroups(...)
```

- Use Group objects instead of User objects.
- Same methods apply for Group objects:

```
group.getName()
group.getGroups()
group.getUsers()
```

© Copyright IBM Corporation 2011

Figure 9-67. Getting group names from directory service

F1432.0

### Notes:

Retrieve security users and groups

## realm.findGroups(...)

```
public GroupSet findGroups
 (String searchPattern,
 PrincipalSearchType searchType,
 PrincipalSearchAttribute searchAttribute,
 PrincipalSearchSortType sortType,
 Integer pageSize, PropertyFilter filter)
```

- Parameters
  - searchPattern - Specifies a String containing a pattern of characters to search for.
  - searchType - Specifies the type of search that you want to conduct.
  - searchAttribute - Specifies which attribute to include in the search criteria.
  - sortType - Specifies how to sort the Group objects in the returned collection.
  - pageSize - Specifies the number of elements to return per page.
- Returns
  - A GroupSet collection representing the groups that met the specified criteria

© Copyright IBM Corporation 2011

Figure 9-68. realm.findGroups(...)

F1432.0

### Notes:

Possible values for the searchType parameter:

- 0 = PrincipalSearchType.NONE
- 1 = PrincipalSearchType.CUSTOM
- 2 = PrincipalSearchType.PREFIX\_MATCH
- 3 = PrincipalSearchType.SUFFIX\_MATCH
- 4 = PrincipalSearchType.CONTAINS
- 5 = PrincipalSearchType.EXACT

Possible values for the searchedAttribute parameter:

- 0 = PrincipalSearchAttribute.NONE
- 1 = PrincipalSearchAttribute.SHORT\_NAME
- 2 = PrincipalSearchAttribute.DISPLAY\_NAME

Possible values for the sortType parameter:

- 0 = PrincipalSearchSortType.NONE
- 1 = PrincipalSearchSortType.ASCENDING
- 2 = PrincipalSearchSortType.DESCENDING

For pageSize, if the value is null or zero, the default is 50.

Retrieve security users and groups

## Sample code to retrieve group names

```
...
GroupSet groups = realm.findGroups(pattern,
 PrincipalSearchType.PREFIX_MATCH,
 PrincipalSearchAttribute.SHORT_NAME,
 PrincipalSearchSortType.NONE,
 Integer.valueOf("50"), null);
com.filenet.api.security.Group group;
Iterator groupIt = groups.iterator();
while (groupIt.hasNext()){
 group=(com.filenet.api.security.Group)
 groupIt.next();
 System.out.println("name =" + group.getName());
 System.out.println("Distinguished name = " +
 group.getDistinguishedName());
 System.out.println("short name =" +
 group.getShortName());
```

© Copyright IBM Corporation 2011

Figure 9-69. Sample code to retrieve group names

F1432.0

### Notes:

Retrieve security users and groups

## Demonstrations



- Retrieve security user and group names from the directory service.
  - Explore the code sample.
  - Run the solution code.
  - Observe the results.

© Copyright IBM Corporation 2011

Figure 9-70. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getRealmEDU(...)* and *showUserGroupsEDU(...)* methods in the *SecurityEDU.java* file.
3. Run the code and verify the results as instructed in the *Security* unit in the Student Exercises book.

Retrieve security users and groups

## Activities



In your Student Exercises book

- Unit: Security
- Lesson: Retrieve security users and groups
- Activities
  - Retrieve security users and groups.

© Copyright IBM Corporation 2011

Figure 9-71. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.

# Unit 10. Events and Subscriptions

## What this unit is about

This unit describes how to implement a Java event handler to automate the Content Engine actions.

## What you should be able to do

After completing this unit, you should be able to:

- Implement a Java event handler to write to a log file.
- Create subscriptions with event actions on a Document class.
- Test the deployed code modules.
- Optional: Implement a Java event handler to update an external database.

## How you will check your progress

- Successful completion of lesson exercises.

## References

[www.ibm.com/support/documentation/us/en](http://www.ibm.com/support/documentation/us/en) (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

## Events and Subscriptions

### Unit lesson

- 
- Implement a Java event handler

© Copyright IBM Corporation 2011

Figure 10-1. Unit lesson

F1432.0

#### Notes:

## **Lesson 10.1. Implement a Java event handler**

## Lesson: Implement a Java event handler

- Why is this lesson important to you?
  - Your assignment is to automate the Content Engine actions for the following business scenarios:
    - Log information when a document is created.
    - Update an external database when a document is created.
  - You are going to write code implement a Java event handler and work with the system administrator to create event actions and subscriptions to automate these actions.

© Copyright IBM Corporation 2011

Figure 10-2. Lesson: Implement a Java event handler

F1432.0

### Notes:

#### Scenario:

Management wants to include the user who creates the document in the event log every time a document is added. Developers update the code and give the administrator the new jar file to add.

Implement a Java event handler

## Activities that you need to complete



- Implement a Java event handler.
- Optional: Update a database record.

© Copyright IBM Corporation 2011

Figure 10-3. Activities that you need to complete

F1432.0

### Notes:

Implement a Java event handler

## Event-action architecture



- Provides mechanism for launching actions in response to event triggers
- Classes and objects subscribe to event-action sets
  - Action is launched when event occurs to instance of class or to object
- Elements of event-action architecture
  - Event (the trigger)
  - Target (Class or Object subscribing to an event)
  - Event Action (the response)
  - Subscription (rule tying target to an event and event action)

© Copyright IBM Corporation 2011

Figure 10-4. Event-action architecture

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events and subscriptions > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Events and Subscriptions > Events Concepts

An event is something that is triggered on instances of a class or to a specific object (the target). Often, events are initiated by some user action, such as checking a document out or in, or launching a workflow. There are a number of system-defined events for each class type. For example, Creation, Checkin, Checkout, File, and Unfile. Additionally, you can create and trigger custom events. (See the Auditing module for more information on the Event class.)

An **event action** is what you want to do in response to an event, such as update a database or send an email. An event action is instantiated in code or created using the

Content Engine Enterprise Manager. Associated with the event action is custom Java code that performs the desired action. The custom code is the event handler. You are going to implement an event handler and add code to it.

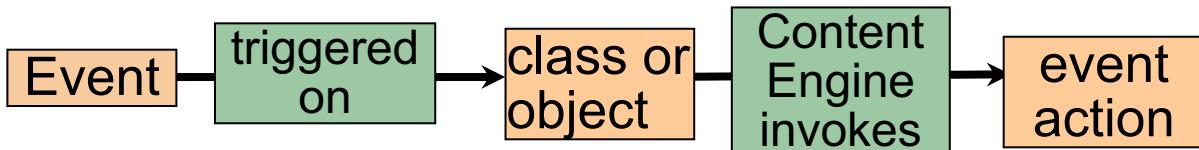
A **subscription** determines that, when a particular event or events happen to a target, start this event action in response.

Implement a Java event handler

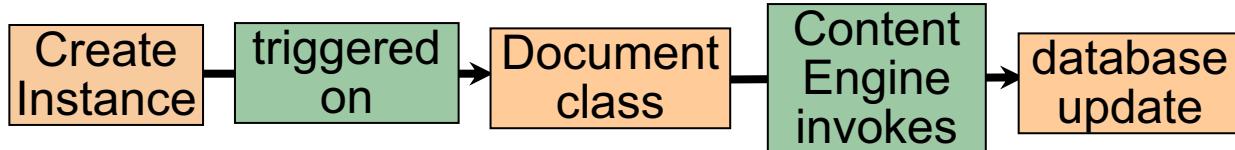
## Subscriptions



A Subscription defines: When an event is triggered on a class or object, Content Engine invokes the event action



Example:



© Copyright IBM Corporation 2011

Figure 10-5. Subscriptions

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events and subscriptions > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Events and Subscriptions > Subscription Concepts

A subscription enables an event action that is implemented to execute when an event is triggered on a target Content Engine object.

Implement a Java event handler

## Subscriptions for classes or instances

- Subscriptions can be applied to entire class
  - Operate when event is triggered on a member of a class
  - Example:
    - Event: Creation of a new folder of class Folder
    - Event action: Create four subfolders in the new folder
- Can be applied to specific object instances
  - Example:
    - Whenever document ID {536C4348-46C2-4E53-97C1-4ACD5FAD77B3} is checked in as a major (released) version, change its security and announce its availability by email.

© Copyright IBM Corporation 2011

Figure 10-6. Subscriptions for classes or instances

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Events and Subscriptions > Subscription Concepts
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events and subscriptions > Concepts

Implement a Java event handler

## Steps to create an event action



1. Write an event handler in Java and compile the code.
2. Use Content Engine Enterprise Manager to create event action that uses your event handler.
  - a. Specify the event handler Java class.
  - b. Optionally, configure the code module with Java class file
3. Assign this event action to a subscription.
  - a. Optionally, apply a filter.

© Copyright IBM Corporation 2011

---

Figure 10-7. Steps to create an event action

F1432.0

### Notes:

After you write and compile an event handler and configure it in the Content Engine, you can configure classes or objects to subscribe to it. Subsequently, when a triggering event occurs, the event handler is invoked.

For example, if the event handler sends an email message to a group of people, you create a subscription on a Product document class that invokes the event handler whenever a new Product document is created. Subsequently, whenever someone adds a Product document, the email message is sent out.

Implement a Java event handler

## Applying a filter to an event subscription

- To restrict the application scope of a subscription, you can create a filter.
  - For example, restrict the scope to documents in class Memorandum with a security level of Confidential:  
SecurityLevel = 'Confidential'
  - You can also filter on filing operations.
- Creation event co-occurs with several other events.
  - Adding a new document triggers a Creation and Checkin event.
  - Checkout and file operations also trigger Creation event.
  - If you want to do something **only** when adding a new document, you have to filter out the Creation event caused by a checkout.

© Copyright IBM Corporation 2011

Figure 10-8. Applying a filter to an event subscription

F1432.0

### Notes:

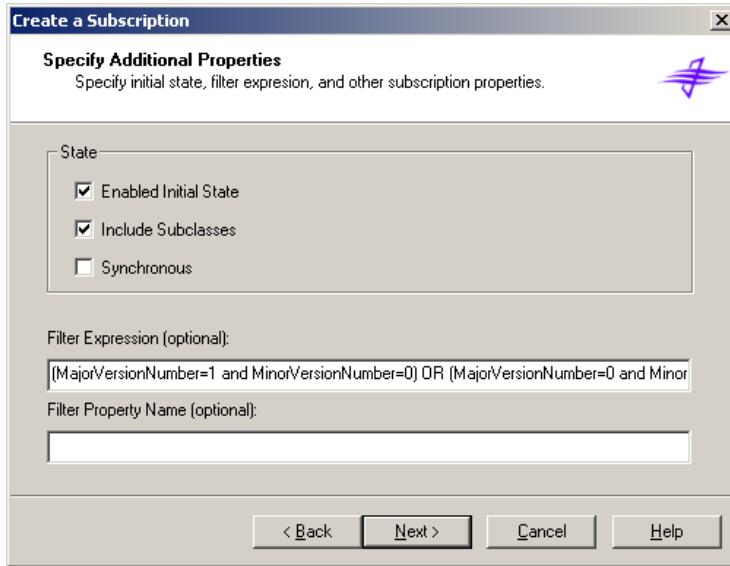
#### Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events and subscriptions > How to > Work with subscriptions > Create a subscription

Implement a Java event handler

## Define subscription filter

- Use the Create a Subscription wizard to specify a filter.
- Filter out Creation events triggered by Checkout operation:  
 $(\text{MajorVersionNumber}=1 \text{ and } \text{MinorVersionNumber}=0)$  OR  
 $(\text{MajorVersionNumber}=0 \text{ and } \text{MinorVersionNumber}=1)$



© Copyright IBM Corporation 2011

Figure 10-9. Define subscription filter

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events and subscriptions > How to > Work with subscriptions > Create a subscription

The document checkout operation triggers the Creation event (a new reservation object is created) in addition to the expected Checkout event. Therefore, if you are subscribing to the Creation event, you might need to create a filter in the subscription so that your event handler doesn't fire when a checkout occurs.

The filter in the example above applies to the new document object (the source object) that is passed into the event handler. As a new document, it has a version number of 1.0 or 0.1.

Implement a Java event handler

## Synchronous and Asynchronous mode



- Synchronous execution

- The call to execute the event action blocks further processing by the subscription processor until the action completes and returns.
- The action runs in the same transaction as the originating activity on the target object.
- If the action fails, the transaction rolls back.

- Asynchronous execution

- The action occurs on a separate execution thread.
- Allows the subscription processor to continue without waiting for the results of the action.
- The action cannot be in the same transaction as the originating activity.

© Copyright IBM Corporation 2011

Figure 10-10. Synchronous and Asynchronous mode

F1432.0

### Notes:

The exercises in this unit uses Asynchronous mode which is the default setting.

Implement a Java event handler

## Event handler requirements



- Your Java-based event handler must do the following:
  - Implement the EventActionHandler interface
  - Implement onEvent(...) method.
- Your implementation of onEvent() can test for event type:  
`if(event.getClassName().equalsIgnoreCase("nnnEvent"))`
  - Where nnn= type of the event
- You can package the class file in a JAR file.
  - Convenient for packaging several event handlers

© Copyright IBM Corporation 2011

Figure 10-11. Event handler requirements

F1432.0

### Notes:

Implement a Java event handler

## Event handler coding guidelines



- Cannot display user interface elements.
- Cannot call save method on source object.
- Coordinate synchronous event procedure execution time with EJB transaction timeout.
- Asynchronous event can operate on the triggering object (if it loads a new copy from the database), while a synchronous event cannot.

© Copyright IBM Corporation 2011

Figure 10-12. Event handler coding guidelines

F1432.0

### Notes:

Because the event handler code runs on the Content Engine server, you cannot use it display interface elements. If you attempt to do so while the code is executing synchronously, the interface element freezes both the server process and the client application process.

You cannot call the Save method on the source object of the event. (The triggered event is passed to the event handler, and you can call the get\_SourceObject method on the event to retrieve the source object.) This restriction is enforced because, when the event handler is called, a Save method is already in progress for the source object. The actual persistence of this object occurs shortly after the completion of any event handlers that must execute. If the system allowed the Save method to be called in the event handler, one of two things might happen:

The Save method might cause another set of subscriptions for the object being saved. This set likely results in another event handler executing recursively, causing a loop that ends only with a stack overflow in the server or a transaction timeout.

For synchronous events, the time required for the event procedure to execute must be short enough not to cause the transaction to time out, because the event procedure is running in the context of an EJB transaction. EJB transactions have a finite time to complete. Otherwise, they are forced to abort as a way to prevent deadlocks. If an event procedure must cause a long-running task to execute, consider changing the timeout setting or use a queued component or another mechanism to defer the work beyond the limit of the transaction.

The EJB transaction timeout is defined in the J2EE application server console. The default value of this timeout is different on each application server. See your application server documentation for information on this default and how to change it.

Implement a Java event handler

## Code modules



- Code modules are versioned documents.
  - Used to store the event action handler.
- Two options to deploy document event action handler:
  - Set the location in the classpath of the application server on which the Content Engine runs.
  - Check it into an object store as a CodeModule object.

© Copyright IBM Corporation 2011

Figure 10-13. Code modules

F1432.0

### Notes:

A code module is automatically available in these situations:

- When you are deploying the Content Engine to multiple application server instances
- When you are moving your content metadata from one system to another

If you reference document event action handler in the classpath of an application server, you must manually distribute them to new systems.

Implement a Java event handler

## Update event action with a new code module version



- If you modify the code for a Java event action handler contained within a code module:
  - You must update any event actions referencing the code module.
- Steps to update:
  1. Check out the code module.
  2. Modify the Java event action handler source and compile.
  3. Check in the code module with the new version of the Java class.
  4. Copy the object reference for the code module.
  5. Update the Code Module property of the event action by pasting the object reference.

© Copyright IBM Corporation 2011

Figure 10-14. Update event action with a new code module version

F1432.0

### Notes:

### Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events and subscriptions > How to > Work with event actions > Modify an action

Implement a Java event handler

## Adding external JAR files to the Content Engine



- Some event actions might require external JAR files.
- These files can be added in one of the following ways:
  - Add as content elements of the code module.
    - Imported with the Content Engine objects
  - Copy to the application server or set in the classpath.
    - Globally available for the other event actions

© Copyright IBM Corporation 2011

Figure 10-15. Adding external JAR files to the Content Engine

F1432.0

### Notes:

Implement a Java event handler

## eventActionHandler.onEvent(...)



```
void onEvent(ObjectChangeEvent event,
 Id SubscriptionID)
```

- Parameters

- event - An event of type ObjectChangeEvent or subclass
- subscriptionId - An ID value that represents the GUID of the subscription that defines the triggered event

- Notes:

- Invoked when a subscribed event occurs on a target object that is set in a subscription.
- Implement this method with the actions to be taken when the event is triggered.

© Copyright IBM Corporation 2011

Figure 10-16. eventActionHandler.onEvent(...)

F1432.0

### Notes:

#### Help path

For a complete list of available Content Engine events, refer to this topic:

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events and subscriptions > Concepts

#### Interfaces that inherit from ObjectChangeEvent

CancelCheckout, ChangeClass, Checkin, Checkout

Creation, Deletion, File, Unfile

Update, ChangeClass, PromoteVersion, DemoteVersion

Implement a Java event handler

## ObjectChangeEvent



- The ObjectChangeEvent interface represents an event.
- This interface is the base for many event interfaces.
  - CancelCheckoutEvent, ChangeClassEvent, FileEvent, CustomEvent, and others
- To Instantiate an ObjectChangeEvent object:
  - Call the getInstance or fetchInstance method on the Factory.ObjectChangeEvent class, or
  - Retrieve an ObjectChangeEvent object from a ObjectChangeEventSet collection

© Copyright IBM Corporation 2011

Figure 10-17. ObjectChangeEvent

F1432.0

### Notes:

Implement a Java event handler

## objectChangeEvent.get\_SourceObject()



`IndependentObject get_SourceObject()`

- Returns
  - The source object of the event at the time the event occurred
- Notes
  - Must typecast to the specific Content Engine object (Document, Folder, or CustomObject).

© Copyright IBM Corporation 2011

Figure 10-18. objectChangeEvent.get\_SourceObject()

F1432.0

### Notes:

Implement a Java event handler

## Sample code to implement an event action handler

```
public class LogEventAction implements
 EventActionHandler{
 public void onEvent(ObjectChangeEvent event, Id
 subscriptionId) throws EngineRuntimeException{
 try {
 if (event.getClassName() .equalsIgnoreCase
 ("CreationEvent")) {
 ...
 }
 } catch (IOException e) {
 throw new EngineRuntimeException(
 ExceptionCode.E_FAILED);
 }
 }
}
```

© Copyright IBM Corporation 2011

Figure 10-19. Sample code to implement an event action handler

F1432.0

### Notes:

Implement a Java event handler

## EngineRuntimeException



- Exception thrown by  
eventActionHandler.onEvent()
  - Defined as `ExceptionCode` object.
  - Supports `ErrorStack`.

© Copyright IBM Corporation 2011

Figure 10-20. EngineRuntimeException

F1432.0

### Notes:

```
catch (IOException e) {
 throw new EngineRuntimeException(ExceptionCode.E_FAILED);
}

Or

catch (IOException e) {
 ErrorRecord er[] = {new ErrorRecord (e)};
 ErrorStack es = new ErrorStack(e.getMessage(),er);
 throw new EngineRuntimeException(es);
}
```

Implement a Java event handler

## Error logging and debugging



- What can go wrong?
  - Missing Java classes
- Debugging
  - <Application server>/FileNet/server1/p8\_server\_error.log
  - Example:  
opt.ibm/WebSphere/AppServer/profiles/APPSrv01/FileNet/se  
rver1/p8\_server\_error.log

© Copyright IBM Corporation 2011

Figure 10-21. Error logging and debugging

F1432.0

### Notes:

Content Engine trace logging is performed by the Apache log4j 1.2.x package. Trace logging is typically implemented to collect and record information about application failures in test or production environments.

When you enable logging, log statements inserted into the IBM FileNet P8 software cause log entries to be written to an output location. As the application (client) makes requests to the server, the logging mechanism captures information about the request and writes it to some output medium, such as a file or a console.

IBM FileNet customer support and development teams use the information provided by trace logging to diagnose and solve problems. The fully qualified path to use as the output location for trace logs value defaults to the working directory path of the application server instance and is appended with "/FileNet". The file name for the trace log is always p8\_server\_trace.log.

When the Content Engine starts, if no error logger has been configured for the system (in other words, no log4j configuration), a default error logger "filenet\_error" is configured to gather logging at the INFO level. The default error logger writes entries to the

p8\_server\_error.log file in the working directory path of the application server instance and is appended with "/ FileNet ".

Implement a Java event handler

## Sample messages in P8\_server\_error.log file

```

com.filenet.api.exception.EngineRuntimeException: EVENT_HANDLER_THROWN: The Event handler threw an
exception.
at com.filenet.engine.queueitem.SubscriptionProcessor.executeHandler(SubscriptionProcessor.java:802)
at com.filenet.engine.queueitem.SubscriptionProcessor.execute(SubscriptionProcessor.java:658)
at com.filenet.engine.queueitem.SubscriptionProcessor.execute(SubscriptionProcessor.java:619)
at com.filenet.engine.queueitem.EventQueueItemHandler.execute(EventQueueItemHandler.java:75)
at com.filenet.engine.queueitem.EventQueueItemHandler$1.run(EventQueueItemHandler.java:48)
at com.filenet.engine.context.CallState.doAs(CallState.java:214)
at com.filenet.engine.context.CallState.doAs(CallState.java:151)
at com.filenet.engine.queueitem.EventQueueItemHandler.executeAs(EventQueueItemHandler.java:39)
at
com.filenet.engine.queueitem.QueueItemExecutor.loadAndExecuteQueuedRow(QueueItemExecutor.java:120)
at com.filenet.engine.queueitem.QueueExecutor.dispatchQueuedRow(QueueExecutor.java:328)
at com.filenet.engine.queueitem.QueueExecutor.dispatchEvent(QueueExecutor.java:183)
at com.filenet.engine.queueitem.QueueExecutor.execute(QueueExecutor.java:123)
at com.filenet.engine.tasks.BackgroundTask.safeExecute(BackgroundTask.java:189)
at com.filenet.engine.tasks.BackgroundTask.access$000(BackgroundTask.java:39)
at com.filenet.engine.tasks.BackgroundTask$1.run(BackgroundTask.java:133)
at com.filenet.engine.context.CallState.doAsSystem(CallState.java:336)
at com.filenet.engine.tasks.BackgroundTask.run(BackgroundTask.java:128)
at com.filenet.engine.jca.workmgr.ThreadPool$WorkerThread.run(ThreadPool.java:85)
Caused by: java.lang.NoClassDefFoundError: com/ibm/filenet/gls/LogOperation
at com.ibm.filenet.gls.LogAction.onEvent(LogAction.java:22)
at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java(Compiled Code))
at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java(Compiled Code))
at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java(Compiled Code))
at java.lang.reflect.Method.invoke(Method.java(Compiled Code))
at com.filenet.engine.queueitem.SubscriptionProcessor.executeHandler(SubscriptionProcessor.java:790)

```

© Copyright IBM Corporation 2011

Figure 10-22. Sample messages in P8\_server\_error.log file

F1432.0

### Notes:

Implement a Java event handler

## Demonstrations



- Implement a Java event handler.
  - Explore the code sample for implementing a Java event handler.
  - Create a subscription with event action on a Document class and test the code.

© Copyright IBM Corporation 2011

Figure 10-23. Demonstrations

F1432.0

### Notes:

#### DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *LogEventActionEDU.java* file. This Java class implements the *EventHandler* and *onEvent(...)* method.
3. Deploy the code module by creating an event action and a subscription on a Document class.
4. Verify the results as instructed in the *Events and Subscriptions* unit in the Student Exercises book.
5. Optional: Repeat the steps 2 through 4 using the *DBActionEDU.java* file.

Implement a Java event handler

## Activities

In your Student Exercises book

- Unit: Events and Subscriptions
- Lesson: Implement a Java event handler
- Activities
  - Implement a Java event handler.
  - Optional: Update a database record.

© Copyright IBM Corporation 2011

Figure 10-24. Activities

F1432.0

### Notes:

Use your Student Exercises book to perform the activities listed.



# Unit 11. Auditing

## What this unit is about

This unit describes how to retrieve the audit history for existing objects and deleted objects, and how to raise custom events.

## What you should be able to do

After completing this unit, you should be able to:

- Retrieve the audit history for existing objects
- Retrieve the audit history for deleted objects
- Raise custom events

## How you will check your progress

- Successful completion of lesson exercises.

## References

[www.ibm.com/support/documentation/us/en](http://www.ibm.com/support/documentation/us/en) (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Auditing

## Unit lessons



- Retrieve the audit history for existing objects
- Retrieve the audit history for deleted objects
- Work with custom events

© Copyright IBM Corporation 2011

Figure 11-1. Unit lessons

F1432.0

### Notes:

## **Lesson 11.1. Retrieve the audit history for existing objects**

## Lesson:

# Retrieve the audit history for existing objects



- Why is this lesson important to you?
  - The administrator of your company has configured a document class for the auditing. Management wants to retrieve the auditing history of a document of that class.
  - An object has been updated. Management wants to have audit data values before and after modification.
  - As their programmer, you are going to write code to perform these tasks.

© Copyright IBM Corporation 2011

Figure 11-2. Lesson: Retrieve the audit history for existing objects

F1432.0

## Notes:

Retrieve the audit history for existing objects

## Activities that you need to complete



- Configure auditing.
- Retrieve the audit history for existing objects.

© Copyright IBM Corporation 2011

Figure 11-3. Activities that you need to complete

F1432.0

### Notes:

Retrieve the audit history for existing objects

## Auditing concepts



- Content Engine provides audit logging to monitor event activity of FileNet P8 objects.
- The result of auditing is the automatic logging of audit entries.
- Audit definitions represent information describing how to audit an event.
- Most server events can be audited.
- Client events cannot be audited through this mechanism.

© Copyright IBM Corporation 2011

Figure 11-4. Auditing concepts

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Auditing > Auditing overview > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Auditing > Auditing Concepts

Audit entries are instances of one of the subclasses of the Event class.

For example, a document class can be configured to automatically create audit entries whenever documents of that class are checked in, provided that auditing has been enabled for the object store and belong to the Checkin Event class.

#### Audit entries contain the following information (properties):

- The event, method, or action that occurred, along with applicable parameters (for example, a Change State event includes that type of change: promote or demote)

- The name of the user who performed the action
- The date and time of the event
- The class and ID of the associated object
- Whether the event was a success ("Success Audit") or failure ("Failure Audit")
- The names of any changed properties
- For security updates, a statement that the permissions (permissions object of the document) were modified

Retrieve the audit history for existing objects

## Enable and configure auditing



- Enabled at the object store level.
- Configured on individual classes for specific events.
- Enabled and configured using either of these:
  - Content Engine API
  - Content Engine Enterprise Manager
- To determine if auditing is enabled system-wide, retrieve the AuditLevel property of the ObjectStore object.
  - If auditing is disabled, audit events are not recorded.

© Copyright IBM Corporation 2011

Figure 11-5. Enable and configure auditing

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Auditing > Auditing Concepts > Setup Requirements
- To find out all the objects and events are auditable, refer to

IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Auditing > Audit overview > Auditable objects and events

The following are examples of Content Engine classes that can be audited:

Annotation

ChoiceList

ClassDefinition

ComponentRelationship

Containable  
CustomObject  
Document  
EventAction  
Folder  
Publish Request  
SecurityPolicy  
Subscribable  
Version Series

**Note:** Subclasses of these classes can also be audited.

Retrieve the audit history for existing objects

## Configure auditing for a specific property

- Audit the changes to a specific property of a class
  - Auditing specific properties reduces the size of the audit log.
- An audited property must be mapped to a corresponding event property.
  - Map a custom property to the corresponding property template on the event class.
  - Create a corresponding property template for the system property you want to audit.
  - System properties do not have preexisting property templates.

© Copyright IBM Corporation 2011

Figure 11-6. Configure auditing for a specific property

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Auditing > Auditing Concepts > Property Auditing
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Auditing > Configuring auditing > Configuring auditing for a property

When you configure an event class to audit a specific property, Content Engine extends the audit log database table with a column corresponding to the property. When an audited event is triggered on its source object, Content Engine stores an instance of the event in the table and copies the value of the property being audited from the source object to the applicable column in the event instance.

## To configure a specific property for auditing

- In addition to the basic setup requirements for auditing a class, property auditing requires configuration of the event class and the class of the source object.
- Create a custom property.
- Assign the custom property, that you created, to an event of the Object Change Event class or one of its subclasses.
  - The Object Change Event class is located in the Other Classes > Event folder.
- Assign this custom property to a class that you want to audit.
  - On the Property Definitions tab of the class you want to audit, select this property definition and click **Edit**. On the More tab, set the **Audit As** property to the custom property template that you assigned to the event class.

Retrieve the audit history for existing objects

## Audit event logs



- The audit event logs exist as a table in the database of the object store.
- The log can be viewed or exported to an XML file for reporting and other purposes.
  - Must run a query for the events that you want.
- Audit events remain in the audit log even if the audited object is deleted.

© Copyright IBM Corporation 2011

Figure 11-7. Audit event logs

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Auditing > Auditing overview > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Auditing > Auditing Concepts

This lesson is about the retrieval of audit history for existing objects.

#### How the data is stored in the audit logs when a specific property of a class is monitored?

When property templates are added to an event, it adds columns to the Audit Log – physically the Event table in the object store database.

In previous releases, if we wanted to capture any property information, we had to record the states of the objects concerned in the form of binary objects (before and after

snapshots). We then had to extract the information we needed from those objects – this usually required custom code.

In P8 5.0 we can expose the properties we need directly as columns on the Event table which makes these properties easily searchable and thus makes the Auditing system more flexible.

When you want to monitor properties of two or more Content Engine classes at the same time, instead of exposing both properties (for example - FolderName for a folder class and DocumentTitle for a Document class) directly on the Audit Log table, mapping the two properties to a single generic property of the event ( for example - Audit Title) allows to conserve the use of columns in the Audit Log table.

If we have exposed both properties in the above scenario, then update event records originating from a Document class object will not use the FolderName value. Likewise, a Folder class will not use the DocumentTitle property.

#### **Removing audit records:**

Audit records remain in the Audit Log table until they are either manually deleted or removed by a Disposition Policy. Audit disposition is a subsystem for automatic deletion of audited events in the Content Engine audit log (Event table). Audit disposition runs as a background thread within the Content Engine.

**Notes:** The lab exercises have instructions for both auditing an object and a specific property of a class.

Retrieve the audit history for existing objects

## Event object



- Is the parent object interface through which audit-related functionality is exposed.
- Represents a system-defined or custom event that can be triggered on an object class or instance.
- Use Event object and its subclasses to retrieve the following:
  - Information about the event
  - Source object of an event

© Copyright IBM Corporation 2011

Figure 11-8. Event object

F1432.0

### Notes:

#### Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Auditing > Auditing Concepts > Auditable Classes and Events

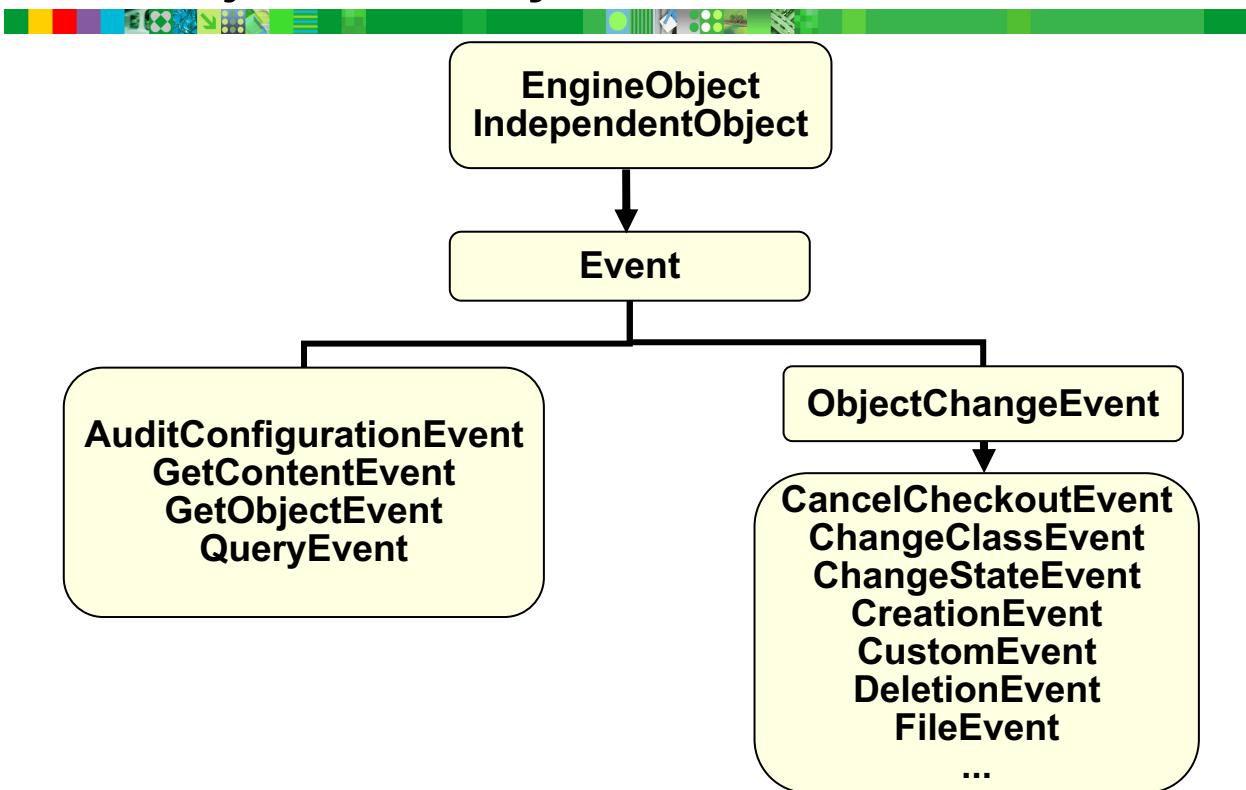
When an audited event occurs, the Content Engine API creates an Event object that is stored in the object store database to form an audit record.

All events are derived from the Event interface. An Event-based object is one of the following subobject types:

- **AuditConfigurationEvent**, which occurs when the auditing configuration of an object store is changed. This event is not auditable.
- **ObjectChangeEvent**-derived objects, which occur when the state of an object changes. These events are auditable.
- **RetrievalEvent**-derived objects, which occur when a Content Engine object or its content is retrieved or queried. These events are auditable.

Retrieve the audit history for existing objects

## Event object hierarchy



© Copyright IBM Corporation 2011

Figure 11-9. No title

F1432.0

### Notes:

#### Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Auditing > Auditing Concepts > Auditable Classes and Events
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Auditing > Audit overview > Auditable objects and events

The following are the examples of auditable event types defined by the system:

ChangeStateEvent

CheckoutEvent

CreationEvent

CustomEvent

DeletionEvent

DemoteVersionEvent

GetContentEvent

GetObjectEvent

PromoteVersionEvent

PublishRequestEvent

QueryEvent

UpdateEvent

CheckinEvent

Retrieve the audit history for existing objects

## Steps to retrieve audited events

1. Retrieve audited events for the given object.

```
containable.get_AuditedEvents()
```

2. Retrieve the event details.

```
event.get_InitiatingUser()
event.getClassName()
event.get_DateCreated()
event.get_EventStatus()
```

3. Retrieve the updated properties, original, and the source objects from the Update Event.

```
updateEvent.get_ModifiedProperties()
updateEvent.get_OriginalObject()
updateEvent.get_SourceObject()
```

© Copyright IBM Corporation 2011

Figure 11-10. Steps to retrieve audited events

F1432.0

### Notes:

Retrieve the audit history for existing objects

## containable.get\_AuditedEvents()



```
EventSet get_AuditedEvents ()
```

- Returns
  - EventSet collection of the Event objects
    - Collection contains the audited events that have occurred for the object.

© Copyright IBM Corporation 2011

Figure 11-11. containable.get\_AuditedEvents()

F1432.0

### Notes:

Retrieve the audit history for existing objects

## event.get\_InitiatingUser()



```
String get_InitiatingUser()
```

- Parameters
  - None
- Returns
  - The authentication provider's LDAP Distinguished Name (DN) of the user that initiated the event
- Notes:
  - This property is available for asynchronous events only.
  - For synchronous events, get the current user from the server context.

© Copyright IBM Corporation 2011

Figure 11-12. event.get\_InitiatingUser()

F1432.0

### Notes:

Retrieve the audit history for existing objects

## engineObject.getClassName()



```
String get_ClassName()
```

- Returns
  - A String specifying the name of the class from which this object is instantiated.
- Notes
  - EngineObject is a top-level interface from which Event interface is derived.

© Copyright IBM Corporation 2011

Figure 11-13. engineObject.getClassName()

F1432.0

### Notes:

Retrieve the audit history for existing objects

## event.get\_DateCreated()



```
java.util.Date get_DateCreated()
```

- Parameters
  - None
- Returns
  - The date and time that the object was created.
- Notes
  - The Content Engine stores dates and times using Universal Time Coordinates (UTC).
  - After initial object creation, this value is read-only for all users.

© Copyright IBM Corporation 2011

Figure 11-14. event.get\_DateCreated()

F1432.0

### Notes:

Retrieve the audit history for existing objects

## event.get\_EventStatus()



```
Integer get_EventStatus()
```

- Returns

- An integer value indicating whether the operation that caused the creation of this event was successful for a system event.
  - Zero (0) indicates a successful operation.
  - Otherwise, an error code is returned.

- Notes

- For a custom event, this value can be set by a client application to convey state to the event action handler.

© Copyright IBM Corporation 2011

Figure 11-15. event.get\_EventStatus()

F1432.0

### Notes:

For a custom event, this property can be set by a client application to convey state to the event action handler.

Therefore, client application and event action handler define and interpret the value of this property.

In the event action handler, this property can be retrieved from the event object passed to the handler.

Retrieve the audit history for existing objects

## updateEvent.get\_ModifiedProperties()



```
StringList get_ModifiedProperties()
```

- Returns
  - A list of the symbolic names of the properties modified during the update event
- Notes
  - This property is available only when the event object is retrieved.

© Copyright IBM Corporation 2011

Figure 11-16. updateEvent.get\_ModifiedProperties()

F1432.0

### Notes:

This property is not available from the event object passed to the event action handler.

Retrieve the audit history for existing objects

## updateEvent.get\_OriginalObject()

`IndependentObject get_OriginalObject()`

- Returns
  - Object state (original object) for the event prior to the update
- Notes
  - To compare the two states of the source object, fetch the properties of the objects returned by OriginalObject and SourceObject.
  - To get the source object at the time the event occurred, use get\_SourceObject. (See next page.)

© Copyright IBM Corporation 2011

Figure 11-17. updateEvent.get\_OriginalObject()

F1432.0

### Notes:

The original object can be obtained from the following objects:

ChangeClassEvent

ChangeStateEvent

CheckinEvent, CheckoutEvent

ClassifyCompleteEvent

CustomEvent

DemoteVersionEvent, PromoteVersionEvent

FreezeEvent

LockEvent, UnlockEvent

PublishCompleteEvent

TakeFederatedOwnershipEvent

UpdateEvent, UpdateSecurityEvent

Retrieve the audit history for existing objects

## updateEvent.get\_SourceObject()



```
IndependentObject get_SourceObject()
```

- Returns
  - Object state for the event upon completion of the update.

© Copyright IBM Corporation 2011

Figure 11-18. updateEvent.get\_SourceObject()

F1432.0

### Notes:

The source object is the state of the object immediately after the event happens. You can use this property to identify an object if it no longer exists in the object store (see next lesson).

The sourceObject can be obtained from the following classes or their subclasses (such as UpdateEvent):

- EventQueueItem
- ObjectChangeEvent
- GetContentEvent
- GetObjectEvent

Retrieve the audit history for existing objects

## Sample code to retrieve audited events from an object

```
EventSet auditedEvents =
 CEOBJECT.get_AuditedEvents();

Iterator eventIt = auditedEvents.iterator();

while (eventIt.hasNext()) {

 Event event = (Event) eventIt.next();

 System.out.println("The event name: " +
 event.getClassName());

 System.out.println("Initiating user: " +
 event.get_InitiatingUser());

 System.out.println("Time created: " +
 event.get_DateCreated());

 System.out.println("Event status: " +
 event.get_EventStatus().toString());

}
```

© Copyright IBM Corporation 2011

Figure 11-19. Sample code to retrieve audited events from an object

F1432.0

### Notes:

Retrieve the audit history for existing objects

## Sample code to retrieve the modified properties

- Retrieve the modified properties for an object from an update event.

```
if(event.getClassName().equalsIgnoreCase("UpdateEvent")) {
 UpdateEvent updateEvent = (UpdateEvent)event;
 StringList modifiedProperties =
 updateEvent.get_ModifiedProperties();
 String props[] = new
 String[modifiedProperties.size()];
 Iterator it = modifiedProperties.iterator();
 PropertyFilter pf = new PropertyFilter();
 while (it.hasNext()) {
 pf.addIncludeProperty(0, null, null,
 (String)it.next());
 }
}
```

© Copyright IBM Corporation 2011

Figure 11-20. Sample code to retrieve the modified properties

F1432.0

### Notes:

Retrieve the audit history for existing objects

## Sample code to retrieve the each property value (1)

- Retrieve the source object and original object.

```
IndependentObject sourceObj =
 updateEvent.get_SourceObject();

IndependentObject originalObj =
 updateEvent.get_OriginalObject();
```

- Instantiate the class that you wrote in the "Properties" unit.

```
PropertiesEDU myInstance = new PropertiesEDU();
```

© Copyright IBM Corporation 2011

Figure 11-21. Sample code to retrieve the each property value (1)

F1432.0

### Notes:

Retrieve the audit history for existing objects

## Sample code to retrieve each property value (2)

- Retrieve and display the value for each property.
- ```
it = modifiedProperties.iterator();
while (it.hasNext()){
    String propertyName = (String)it.next();
    System.out.println("new value");
    displayPropertyEDU(sourceObj,
    sourceObj.getProperties().get(propertyName));
    System.out.println("old value");
    displayPropertyEDU(originalObj,
    originalObj.getProperties().get(propertyName));
}
```

© Copyright IBM Corporation 2011

Figure 11-22. Sample code to retrieve each property value (2)

F1432.0

Notes:

Refer to the *Properties* unit for the details about *displayPropertyEDU(...)* method.

Retrieve the audit history for existing objects

Sample code to retrieve audited events for document versions



```
Document doc = (Document) CEOObject;
VersionableSet Docversions =
    doc.get_Versions();
Document Docversion;
Iterator it = Docversions.iterator();
while (it.hasNext()) {
    Docversion = (Document) it.next();
    getEventsEDU(Docversion);
}
```

Note: The getEventsEDU(DocVersion) method refers to the code that you are going to write as shown in the previous four pages.

© Copyright IBM Corporation 2011

Figure 11-23. Sample code to retrieve audited events for document versions

F1432.0

Notes:

Retrieve the audit history for existing objects

Sample code to audit the changes to a specific property

```
...
System.out.println("New product ID:" +
    updateEvent.getProperties().getStringValue("P
    roduct_id"));
System.out.println("New product style:" +
    updateEvent.getProperties().getStringValue("s
    tyle"));
...
...
```

Note: Only the lines of code that retrieves the property values from an event are shown here.

© Copyright IBM Corporation 2011

Figure 11-24. Sample code to audit the changes to a specific property

F1432.0

Notes:

Retrieve the audit history for existing objects

Demonstrations



- Configure auditing
- Retrieve the audit events for a document and its versions
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 11-25. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

Configure auditing

1. Sign in to the Content Engine Enterprise Manager.
2. Enable auditing, set audit definition, and test the configuration as instructed in the "Auditing" unit in the Student Exercises book.

Retrieve the audit events for a document and its versions:

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getEventsEDU(...)* and *getVersionsEventsEDU(...)* methods in the *AuditEDU.java* file.
3. Run the code and verify the results as instructed in the *Auditing* unit in the Student Exercises book.

4. Configure the auditing for a specific property of a class and edit the code for the *getEventsEDU(...)* method as instructed in the *Auditing* unit in the Student Exercises book.

Retrieve the audit history for existing objects

Activities



In your Student Exercises book

- Unit: Auditing
- Lesson: Retrieve the audit history for existing objects
- Activities
 - Configure auditing.
 - Retrieve the audit history for existing objects.

© Copyright IBM Corporation 2011

Figure 11-26. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 11.2. Retrieve the audit history for deleted objects

Lesson:

Retrieve the audit history for deleted objects

- 
- Why is this lesson important to you?
 - From time to time, objects get deleted, and management wants to get information about who deletes the objects and when. As their programmer, you are going to write code to do this task.

© Copyright IBM Corporation 2011

Figure 11-27. Lesson: Retrieve the audit history for deleted objects

F1432.0

Notes:

Retrieve the audit history for deleted objects

Activities that you need to complete



- Retrieve the audit history for deleted objects.

© Copyright IBM Corporation 2011

Figure 11-28. Activities that you need to complete

F1432.0

Notes:

Retrieve the audit history for deleted objects

Using SQL queries to retrieve audit data



- Deleted objects are not accessible through the API objects.
- Audit history for deleted objects does exist in a log table on the object store database.
- Retrieve audit history information on instance of classes.
 - Perform an SQL query on audit log in object store.

© Copyright IBM Corporation 2011

Figure 11-29. Using SQL queries to retrieve audit data

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Auditing > Viewing audit entries > Querying the audit log
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Auditing > Working with Auditing-related Objects > Querying the Audit Event Log

Retrieve the audit history for deleted objects

Steps to retrieve audit events for deleted objects



1. Retrieve the audited events for the deleted object.

```
searchScope.fetchObjects(...)
```

2. Retrieve the event details.

```
event.get_InitiatingUser()  
event.getClassName()  
event.get_DateCreated()  
event.get_EventStatus()
```

3. Retrieve the source object from the event.

```
event.getProperties().getObjectValue(...)
```

4. Retrieve the values for the properties and other events of the object by reusing the code from the previous lesson.

© Copyright IBM Corporation 2011

Figure 11-30. Steps to retrieve audit events for deleted objects

F1432.0

Notes:

Retrieve the audit history for deleted objects

Sample code to retrieve the delete events (1)



```
String sql = "SELECT * FROM DeletionEvent";
sql += "WHERE (InitiatingUser like";
sql += " '%" + user + "%')";
SearchScope search = new SearchScope(os);
SearchSQL searchSQL = new SearchSQL(sql);
EventSet events =(EventSet)search.fetchObjects
    (searchSQL, Integer.valueOf("50"),
     null, Boolean.valueOf(true));
Event event;
Iterator it = events.iterator();
while (it.hasNext()){
    event = (Event)it.next();
```

© Copyright IBM Corporation 2011

Figure 11-31. Sample code to retrieve the delete events (1)

F1432.0

Notes:

Retrieve the audit history for deleted objects

Sample code to retrieve the delete events (2)

```
System.out.println("The event name: " +
    event.getClassName());
System.out.println("Initiating user: " +
    event.get_InitiatingUser());
System.out.println("Time created: " +
    event.get_DateCreated());
System.out.println("Event status: " +
    event.get_EventStatus().toString());
System.out.println("Object name: " +
    event.get_Name());
Containable CEOObject = (Containable)
event.getProperties().getObjectValue
    (PropertyNames.SOURCE_OBJECT);
getEventsEDU(CEOObject);
}
```

© Copyright IBM Corporation 2011

Figure 11-32. Sample code to retrieve the delete events (2)

F1432.0

Notes:

The getEventsEDU(CEOObject) method refers to the code that you wrote in the previous lesson. This method is called in this sample to retrieve the values for the properties and other events of the object.

Retrieve the audit history for deleted objects

Demonstrations



- Retrieve the audited events for deleted objects
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 11-33. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getDeletionEventsEDU(...)* method in the *AuditEDU.java* file.
3. Run the code and verify the results as instructed in the *Auditing* unit in the Student Exercises book.

Retrieve the audit history for deleted objects

Activities

In your Student Exercises book

- Unit: Auditing
- Lesson: Retrieve the audit history for deleted objects
- Activities
 - Retrieve the audit history for deleted objects.

© Copyright IBM Corporation 2011

Figure 11-34. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 11.3. Work with custom events

Lesson: Work with custom events

- 
- Why is this lesson important?
 - You have been instructed to audit custom events. As the programmer, you are going to write code to raise the custom event.

© Copyright IBM Corporation 2011

Figure 11-35. Lesson: Work with custom events

F1432.0

Notes:

Work with custom events

Activities that you need to complete



- Create a custom event subclass.
- Raise a custom event for a document.

© Copyright IBM Corporation 2011

Figure 11-36. Activities that you need to complete

F1432.0

Notes:

Work with custom events

Raising and auditing custom events



- Custom events are raised programmatically on an object.
- You configure events on individual classes for auditing by adding properties to the CustomEvent class.

© Copyright IBM Corporation 2011

Figure 11-37. Raising and auditing custom events

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Auditing > Configuring auditing > Creating a custom event

Work with custom events

Steps to raise a custom event on a subscribable object

1. Create an instance of custom event object.

```
Factory.CustomEvent.createInstance(...)
```

2. Set event status.

```
customEvent.set_EventStatus(...)
```

3. Raise the event on the Document object.

```
document.raiseEvent(...)
```

4. Save the document.

```
document.save(...)
```

© Copyright IBM Corporation 2011

Figure 11-38. Steps to raise a custom event on a subscribable object

F1432.0

Notes:

Work with custom events

Factory.CustomEvent.CreateInstance(...)

```
public static CustomEvent createInstance  
    (ObjectStore os, String classId)
```

- Parameters
 - os – Specifies the object store where this object will be persisted.
- Returns
 - An object reference to a new instance of this class
- Notes
 - To persist the created object to the object store, you must explicitly call the save method, or commit the object using a batch operation.
 - Both approaches cause a round-trip to the server.

© Copyright IBM Corporation 2011

Figure 11-39. Factory.CustomEvent.CreateInstance(...)

F1432.0

Notes:

Work with custom events

customEvent.set_EventStatus(...)

```
void set_EventStatus(Integer value)
```

- Parameter
 - value – Specify a integer value for the event status.
- Notes
 - For a custom event, this value can be set by a client application to convey state to the event action handler.
 - The client application and event action handler define and interpret the value of this property.
 - In the event action handler, this property can be retrieved from the event object passed to the handler.

© Copyright IBM Corporation 2011

Figure 11-40. customEvent.set_EventStatus(...)

F1432.0

Notes:

Settability of this property is read-only for most users.

For users who have been granted privileged write access
(AccessRight.PRIVILEGED_WRITE), this property is settable only on create.

After initial object creation, this property is read-only for all users.

Work with custom events

subscribable.raiseEvent(...)



```
public void raiseEvent(CustomEvent  
                      customEvent)
```

- Parameter

- customEvent - Specifies the CustomEvent object to be raised.

- Notes

Before you can use this method, the following conditions must be met:

- An appropriate CustomEvent object must exist.
 - A subscription subscribes to the Subscribable object.
 - A subscription subscribes to the custom event to be raised.

© Copyright IBM Corporation 2011

Figure 11-41. subscribable.raiseEvent(...)

F1432.0

Notes:

The SubscriptionTarget property of the subscription has been set to this Subscribable object.

The SubscribedEvents property of the subscription has been set to a SubscribedEventList collection that includes the custom event.

Work with custom events

Sample code to raise custom event



```
CustomEvent customEvent =  
    Factory.CustomEvent.createInstance(objectSt  
ore, eventName);  
customEvent.set_EventStatus(new Integer(99));  
document.raiseEvent(customEvent);  
document.save(RefreshMode.REFRESH);  
System.out.println("CustomAPIEvent is raised");
```

© Copyright IBM Corporation 2011

Figure 11-42. Sample code to raise custom event

F1432.0

Notes:

Work with custom events

Demonstrations



- Create a CustomEvent subclass
- Raise a custom event for a document
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 11-43. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

Create a custom event subclass:

1. Sign in to the Content Engine Enterprise Manager.
2. Create a custom event subclass and set audit definition to a document class using the instructions in the "Auditing" unit in the Student Exercises book.

Raise a custom event for a document:

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *raiseCustomEventEDU(...)* method in the *AuditEDU.java* file.
3. Run the code and verify the results as instructed in the *Auditing* unit in the Student Exercises book.

Work with custom events

Activities

In your Student Exercises book

- Unit: Auditing
- Lesson: Work with custom events
- Activities
 - Create a custom event subclass.
 - Raise a custom event for a document.

© Copyright IBM Corporation 2011

Figure 11-44. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 12. Batches

What this unit is about

This unit describes how to do a batch update operation and how to do a batch retrieval.

What you should be able to do

After completing this unit, you should be able to:

- Perform a batch update operation
- Perform a batch retrieval operation

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Batches

Unit lessons

- 
- Batch update
 - Batch retrieval

© Copyright IBM Corporation 2011

Figure 12-1. Unit lessons

F1432.0

Notes:

Lesson 12.1. Batch update

Lesson: Batch update

- 
- Why is this lesson important to you?
 - The Product Development team creates content, adds it as a document, and files it in a folder. As their programmer, you are going to write code to perform a batch update operation for these tasks.

© Copyright IBM Corporation 2011

Figure 12-2. Lesson: Batch update

F1432.0

Notes:

Batch update

Activities that you need to complete



- Change security on the folder.
- Perform a batch update operation.

© Copyright IBM Corporation 2011

Figure 12-3. Activities that you need to complete

F1432.0

Notes:

Batch update

Batch definition



- A batch
 - Accumulates and packages multiple operations (method calls) on objects.
 - Is then executed in a single operation.
 - Can significantly improve performance.
- Use a batch when
 - Executing a series of operations that can be completed (or be in progress) independently, without depending on
 - The state of another object included in the batch
 - The result of another operation in the batch
- Example of a batch processing:
 - Checking in a collection of Document objects

© Copyright IBM Corporation 2011

Figure 12-4. Batch definition

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Batches > Concepts

Batch update

Batch class and BatchItemHandle interface



- Each operation included in a batch is referenced as a BatchItemHandle instance.
- A Batch
 - Is a subclass of the Batch abstract class.
 - Contains the list of BatchItemHandle instances.
- Whether a batch is a transactional operation depends on its type:
 - Batch updates
 - Batch retrievals

© Copyright IBM Corporation 2011

Figure 12-5. Batch class and BatchItemHandle interface

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Batches > Concepts

Batch update

Batch updates



- Batch update operation
 - Updates or deletes the persisted objects.
 - Is executed transactionally.
 - Does not return a value.
- Operation steps:
 - IndependentlyPersistableObject references are accumulated.
 - Then, an instance of the UpdatingBatch class is executed as a single transaction (the updateBatch method).
- All of the pending commits succeed or all fail.
 - If the batch operation fails, the transaction is then rolled back and an exception is thrown.

© Copyright IBM Corporation 2011

Figure 12-6. Batch updates

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Batches > Concepts

Batch update

Steps to execute a batch

1. Create an instance of an UpdatingBatch class.

```
updatingBatch.createUpdatingBatchInstance(...)
```

2. Create a first update to be included in the batch (create a document).

```
Factory.Document.createInstance(...)
```

...

3. Create the other updates to be included in the batch.

```
folder.file(...)
```

...

4. Add all updates to the UpdatingBatch instance and execute the batch.

```
updatingBatch.add(...)
```

```
updatingBatch.updateBatch()
```

© Copyright IBM Corporation 2011

Figure 12-7. Steps to execute a batch

F1432.0

Notes:

Batch update

objectStore.get_Domain



Domain get_Domain()

- Returns
 - The FileNet P8 domain to which a given object belongs

© Copyright IBM Corporation 2011

Figure 12-8. objectStore.get_Domain

F1432.0

Notes:

Batch update

updatingBatch.createUpdatingBatchInstance(...)

```
public static UpdatingBatch  
createUpdatingBatchInstance(Domain domain,  
RefreshMode refresh)
```

- Parameters

- **domain** - Specifies the Domain object for the FileNet P8 domain.
- **Refresh** - Specifies a RefreshMode object.

- Returns

- The UpdatingBatch object created

© Copyright IBM Corporation 2011

Figure 12-9. updatingBatch.createUpdatingBatchInstance(...)

F1432.0

Notes:

Parameters

domain - Specifies the Domain object for the FileNet P8 domain to use as the scope of this UpdatingBatch instance.

Refresh - Specifies A RefreshMode object indicating whether the UpdatingBatch object returned is to contain refreshed data from the server.

Batch update

UpdatingBatch.add(...)

```
public BatchItemHandle add  
    (IndependentlyPersistableObject object,  
     PropertyFilter filter)
```

- Parameters
 - object – Specifies an IndependentlyPersistableObject instance to update.
 - Filter – Specifies a PropertyFilter object.
- Returns
 - A BatchItemHandle instance representing the object added to the batch

© Copyright IBM Corporation 2011

Figure 12-10. UpdatingBatch.add(...)

F1432.0

Notes:

This method adds an item to the batch to be updated by this instance.

The item is referenced by a BatchItemHandle instance, and the batch is a list of BatchItemHandle instances.

Parameters

filter - A PropertyFilter object that represents information for controlling which property values (and with what level of detail and recursion) to return.

If the filter is null, this method returns values for all nonobject properties and returns placeholders for all object-valued properties.

Batch update

updatingBatch.updateBatch()

```
public void updateBatch()
```

- Notes

- This method updates all of the items in this batch.
- Calling this method commits the pending batch update operation.
- The batch operation is executed within a single transaction.

© Copyright IBM Corporation 2011

Figure 12-11. updatingBatch.updateBatch()

F1432.0

Notes:

Batch update

Sample code for batch update (1)

```
Domain domain = objectStore.get_Domain();  
UpdatingBatch ub =  
    UpdatingBatch.createUpdatingBatchInstance  
        (domain, RefreshMode.REFRESH);
```

© Copyright IBM Corporation 2011

Figure 12-12. Sample code for batch update (1)

F1432.0

Notes:

The code for adding the document, setting the content, setting the properties, and checking in the document is discussed in detail in “Documents” unit.

Batch update

Sample code for batch update (2)

First update is included in the batch.

```
Document myDoc = Factory.Document.createInstance  
    (objectStore, "Product");  
  
...  
myDoc.set_ContentElements(contentList);  
myDoc.checkin(AutoClassify.DO_NOT_AUTO_CLASSIFY  
, CheckinType.MAJOR_VERSION);  
com.filenet.api.property.Properties properties  
    = myDoc.getProperties();  
properties.putValue("DocumentTitle",  
                    "BatchAPI.gif");  
properties.putValue("product_id", "PDT101");  
myDoc.set_MimeType("image/gif");
```

© Copyright IBM Corporation 2011

Figure 12-13. Sample code for batch update (2)

F1432.0

Notes:

Batch update

Sample code for batch update (3)

```
ReferentialContainmentRelationship rel =  
    folder.file(myDoc, AutoUniqueName.AUTO_UNIQUE,  
    "BatchAPI.gif", DefineSecurityParentage.  
        DO_NOT_DEFINE_SECURITY_PARENTAGE);
```

Add all the updates to the UpdatingBatch instance and execute the batch.

```
ub.add(myDoc, null)  
ub.Add(rel, null)  
updatingBatch.updateBatch()
```

© Copyright IBM Corporation 2011

Figure 12-14. Sample code for batch update (3)

F1432.0

Notes:

Batch update

Demonstrations



- Change security on the folder
- Execute a batch update
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 12-15. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

Change security on the folder

1. Sign in to the Content Engine Enterprise Manager.
2. Change the security on the folder as instructed in the *Batches* unit in the Student Exercises book.

Execute a batch update

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *executeBatchEDU(...)* method in the *BatchEDU.java* file.
3. Run the code and verify the results as instructed in the *Batches* unit in the Student Exercises book.

Batch update

Activities



In your Student Exercises book

- Unit: Batches
- Lesson: Batch update
- Activities
 - Change security on the folder.
 - Perform a batch update operation.

© Copyright IBM Corporation 2011

Figure 12-16. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 12.2. Batch retrieval

Lesson: Batch retrieval

- 
- Why is this lesson important to you?
 - The billing department in your company wants to retrieve information for many customers. As their programmer, you are going to write code to retrieve this information in a batch operation to increase the performance.

© Copyright IBM Corporation 2011

Figure 12-17. Lesson: Batch retrieval

F1432.0

Notes:

Batch retrieval

Activities that you need to complete



- Perform a batch retrieval operation.

© Copyright IBM Corporation 2011

Figure 12-18. Activities that you need to complete

F1432.0

Notes:

Batch retrieval

Batch retrievals



- Batch retrieval operations retrieve independent objects
 - Not executed transactionally.
 - Each object retrieval succeeds or fails individually.
- Operation steps:
 - IndependentObject references are accumulated.
 - Then, each included object is either refreshed (retrieved) or gets its own exception.
 - Any subsequent changes to the objects retrieved in the batch are done in place (only references to the objects in the batch are retrieved).

© Copyright IBM Corporation 2011

Figure 12-19. Batch retrievals

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Batches > Concepts

Batch retrieval

Steps to execute a batch

1. Create an instance of an RetrievingBatch class.

```
retrievingBatch.createRetrievingBatchInstance(...)
```

2. Retrieve the objects to be added to the batch.

```
Factory.Document.getInstance(...)
```

```
...
```

3. Add the retrievals to the batch.

```
retrievingBatch.add(...)
```

```
...
```

4. Execute the batch.

```
retrievingBatch.retrieveBatch()
```

© Copyright IBM Corporation 2011

Figure 12-20. Steps to execute a batch

F1432.0

Notes:

Batch retrieval

Steps to retrieve batch item handles and objects

1. Retrieve batch item handles.

```
retrievingBatch.getBatchItemHandles(...)
```

2. Retrieve the objects from the batch item handle.

```
batchItemHandle.getObject()
```

3. Retrieve values of the properties from the object.

```
document.getProperties().getStringValue(...)
```

4. Retrieve and display any exceptions thrown.

```
batchItemHandle.getException().getMessage()
```

© Copyright IBM Corporation 2011

Figure 12-21. Steps to retrieve batch item handles and objects

F1432.0

Notes:

Batch retrieval

retrievingBatch.createRetrievingBatchInstance(...)

```
public static RetrievingBatch  
createRetrievingBatchInstance(Domain domain)
```

- Parameters
 - domain - Specifies the Domain object for the FileNet P8 domain.
- Returns
 - The RetrievingBatch object created

© Copyright IBM Corporation 2011

Figure 12-22. retrievingBatch.createRetrievingBatchInstance(...)

F1432.0

Notes:

Batch retrieval

retrievingBatch.add(...)

```
public BatchItemHandle add(IndependentObject  
    object, PropertyFilter filter)
```

- Parameters

- object – Specifies an IndependentlyPersistableObject instance to add.
- Filter – Specifies a PropertyFilter object.

- Returns

- A BatchItemHandle instance representing the object added to the batch

© Copyright IBM Corporation 2011

Figure 12-23. retrievingBatch.add(...)

F1432.0

Notes:

This method adds an object reference to the batch to be retrieved by this instance.

The object is referenced by a BatchItemHandle instance, and the batch contains a list of BatchItemHandle instances.

Batch retrieval

retrievingBatch.retrieveBatch()

```
public void retrieveBatch()
```

- Notes

- This method retrieves all of the objects referenced in this batch.
- Calling this method executes the pending batch retrieval operation.

© Copyright IBM Corporation 2011

Figure 12-24. retrievingBatch.retrieveBatch()

F1432.0

Notes:

Batch retrieval

batch.getBatchItemHandles(...)

```
public java.util.List getBatchItemHandles  
    (IndependentObject object)
```

- Parameters
 - object – Specifies an IndependentlyPersistableObject instance or null.
- Returns
 - A list of BatchItemHandle instances referencing the specified object
- Notes
 - If the specified object is not in the batch, an empty list is returned.
 - If the object parameter is null, a list of all BatchItemHandle instances is returned.

© Copyright IBM Corporation 2011

Figure 12-25. batch.getBatchItemHandles(...)

F1432.0

Notes:

RetrievingBatch is derived from Batch class.

Batch retrieval

batchItemHandle.hasException()

```
boolean hasException()
```

- Returns

- A Boolean value of True if there is an exception associated with the batch item
- Otherwise, the value is False.

© Copyright IBM Corporation 2011

Figure 12-26. batchItemHandle.hasException()

F1432.0

Notes:

Batch retrieval

batchItemHandle.getException()



`EngineRuntimeException getException()`

- Returns
 - An EngineRuntimeException object containing the exception assigned to this batch item

© Copyright IBM Corporation 2011

Figure 12-27. batchItemHandle.getException()

F1432.0

Notes:

Batch retrieval

Sample code for batch retrieval (1)

Get the domain and create an Instance of an RetrieveBatch class.

```
Domain domain = objectStore.get_Domain();  
RetrievingBatch rb = RetrievingBatch.  
    createRetrievingBatchInstance(domain);
```

Retrieve the objects to be added to the batch.

```
Document doc1 = Factory.Document.getInstance  
    (objectStore, null, "/Research/Research  
Information Basic Model A.doc");  
  
Document doc2 = Factory.Document.getInstance  
    (objectStore, null, "/Research/Research  
Information Basic Model C.doc");
```

© Copyright IBM Corporation 2011

Figure 12-28. Sample code for batch retrieval (1)

F1432.0

Notes:

Batch retrieval

Sample code for batch retrieval (2)

```
rb.add(doc1, null)
rb.add(doc2, null)
rb.retrieveBatch()
```

Retrieve each batch item handle and the objects.

```
java.util.List batchItemHandles =
    rb.getBatchItemHandles(null);
Iterator it = batchItemHandles.iterator();
BatchItemHandle batchItemHandle;
Document document;
while (it.hasNext()) {
    ...
}
```

© Copyright IBM Corporation 2011

Figure 12-29. Sample code for batch retrieval (2)

F1432.0

Notes:

Batch retrieval

Sample code for batch retrieval (3)

Retrieve values of the properties from the object.

```
...
batchItemHandle = (BatchItemHandle)it.next();
document=(Document)batchItemHandle.getObject();
System.out.println("Document name: " +
    document.get_Name + "Document class: " +
    document.getClassName() + "Product Id: " +
    document.getProperties().getStringValue("produ-
    ct_id"));
if(batchItemHandle.hasException()){
    // Displays the exception
    EngineRuntimeException thrown =
    batchItemHandle.getException();
    System.out.println("Exception: " +
        thrown.getMessage());
...
}
```

© Copyright IBM Corporation 2011

Figure 12-30. Sample code for batch retrieval (3)

F1432.0

Notes:

Batch retrieval

Demonstrations



- Perform a batch retrieval
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 12-31. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *retrieveBatchEDU(...)* method in the *BatchEDU.java* file.
3. Run the code and verify the results as instructed in the *Batches* unit in the Student Exercises book.

Batch retrieval

Activities

In your Student Exercises book

- Unit: Batches
- Lesson: Batch retrieval
- Activities
 - Perform a batch retrieval operation.

© Copyright IBM Corporation 2011

Figure 12-32. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 13. Annotations

What this unit is about

This unit describes how to create annotations, and how to retrieve annotations and copy them to a different version of a document.

What you should be able to do

After completing this unit, you should be able to:

- Create Annotation objects.
- Retrieve annotations.
- Copy annotations.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Annotations

Unit lessons



- Create Annotation objects
- Retrieve annotations
- Optional: Copy annotations

© Copyright IBM Corporation 2011

Figure 13-1. Unit lessons

F1432.0

Notes:

Lesson 13.1. Create Annotation objects

Lesson: Create Annotation objects

- 
- Why is this lesson important to you?
 - The sales team is reviewing documents and wants to add comments and footnotes as annotations. As their programmer, you are going to write code to create Annotation objects, set content, and associate them with the documents.
 - .

© Copyright IBM Corporation 2011

Figure 13-2. Lesson: Create Annotation objects

F1432.0

Notes:

Create Annotation objects

Activities that you need to complete



- Work with graphical annotations in Image Viewer.
- Create Annotation objects.

© Copyright IBM Corporation 2011

Figure 13-3. Activities that you need to complete

F1432.0

Notes:

Create Annotation objects

Image Viewer



- What is an Image Viewer ?
 - A feature provided with IBM FileNet Workplace to view image documents available through Workplace
- What can you do with Image Viewer?
 - View, zoom, magnify, scroll, pan, rotate, and print.
 - View and add annotations to image documents.
- File types supported by Image Viewer:
 - TIFF 6.0
 - GIF, JPEG, and JPG
 - BMP and COLD
 - By default, each of these file types opens automatically in Image Viewer.
- Documents with multiple content elements
 - Open as a single image with multiple pages in Image Viewer.

© Copyright IBM Corporation 2011

Figure 13-4. Image Viewer

F1432.0

Notes:

Create Annotation objects

Demonstrations



- Using Image Viewer

© Copyright IBM Corporation 2011

Figure 13-5. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Sign in to Workplace.
2. Add annotations to a document using Image Viewer. Follow the instructions in the *Annotations* unit in the Student Exercises book.

Create Annotation objects

Working with annotations in Image Viewer



- Image Viewer annotations can include the following:
 - Lines and arrows
 - Colored highlights
 - Text with a transparent or filled background
 - Sticky notes
 - Stamps
- These annotations can be associated with identified portions of the document image.
- The graphic effects of annotations
 - Are graphically visible in Image Viewer.
 - Are not graphically visible in Enterprise Manager.

© Copyright IBM Corporation 2011

Figure 13-6. Working with annotations in Image Viewer

F1432.0

Notes:

Create Annotation objects

Image Viewer with annotations

The screenshot shows a Microsoft Internet Explorer window displaying a "Physician's/Surgeon's Statement" form. The form includes fields for patient information (Name: June A. Brewster, Date of Birth: 1/17/63), examination date (December 12, 1998), family doctor (Dr. Alfred Gordahl), surgeon (Dr. Michael Blumenthal), and symptoms (Shortness of breath, tired). It also lists a surgical procedure (Caged bypass) and a CPT4 code. Annotations include a red arrow pointing from the "Caged bypass" text to a red box containing the text "Need approval". A red circle highlights the handwritten note "Cat Scan" in the "In this section please list separately all of your charges" field. The status bar at the bottom right of the browser window says "Internet".

© Copyright IBM Corporation 2011

Figure 13-7. Image Viewer with annotations

F1432.0

Notes:

Create Annotation objects

Annotation objects



- **Annotation objects**

- Represent incidental information that is attached to an object in order to be available with that object.
- Are associated with a document, folder, or custom object.
- Have zero or more content elements.
- Are defined by the base Annotation class or a subclass.
- Are independently securable with default class security or security from the annotated object .

- **Actions on Annotation objects**

- Search for them and retrieve them using a query.
- Subscribe to server-side events.
- Audit operations on them.

© Copyright IBM Corporation 2011

Figure 13-8. Annotation objects

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Annotations > Concepts

An Annotation object derives its default security from its parent class or from the object that it is annotating, depending on configuration. Default security can be overridden.

Create Annotation objects

Annotated objects and annotations



- Annotated objects
 - Are the objects that can be given associated annotations.
 - Can have zero or more annotations.
- Annotations and document versions
 - Each annotation associated with single document version
 - Annotations are not automatically copied to next document version
- How annotations and annotated objects relate
 - Creating an annotation requires association with an annotated object.
 - Deleting an annotated object also deletes its annotations.
 - Annotations can be modified or deleted independently of annotated object.

© Copyright IBM Corporation 2011

Figure 13-9. Annotated objects and annotations

F1432.0

Notes:

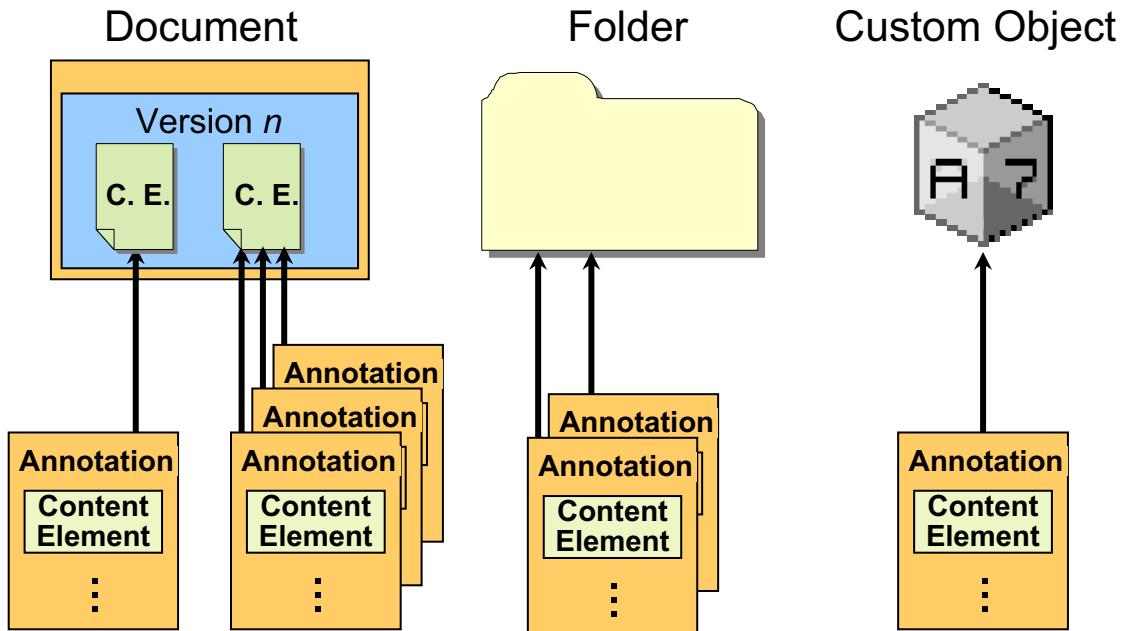
Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Annotations > Concepts

A custom application must copy an annotation from one document version to the next version if annotations must be preserved from one document version to the next version of that document. This topic is covered in Lesson 3.

Create Annotation objects

Annotated objects and annotations



© Copyright IBM Corporation 2011

Figure 13-10. Annotated objects and annotations

F1432.0

Notes:

Create Annotation objects

Steps to create an Annotation object

1. Retrieve the object (example: document) to be annotated.

```
Factory.Document.fetchInstance(...)
```

2. Create an Annotation object.

```
Factory.Annotation.createInstance(...)
```

3. Set content to the Annotation object (steps are similar to setting content for a Document object).

```
annotation.set_ContentElements(...)
```

4. Set annotated object and save the Annotation object.

```
annotation.set_AnnotatedObject(...)
```

```
annotation.save(...)
```

© Copyright IBM Corporation 2011

Figure 13-11. Steps to create an Annotation object

F1432.0

Notes:

Create Annotation objects

Factory.Annotation.createInstance(...)

```
public static Annotation createInstance  
    (ObjectStore os, String classId)
```

- Parameters

- os - Specifies an object store where annotation is created.
- classId - Specifies an identifier for the type of class to be created.

- Returns

- An object reference to a new instance of this class

© Copyright IBM Corporation 2011

Figure 13-12. Factory.Annotation.createInstance(...)

F1432.0

Notes:

To persist the created object to the object store, you must explicitly call the save method, or commit the object using a batch operation.

Both approaches cause a round-trip to the server.

Create Annotation objects

annotation.set_ContentElements(...)



```
void set_ContentElements  
    (ContentElementList value)
```

- Parameters

- value – Specifies an ContentElementList object containing the list of content elements associated with this annotation.

© Copyright IBM Corporation 2011

Figure 13-13. annotation.set_ContentElements(...)

F1432.0

Notes:

Create Annotation objects

annotation.set_AnnnotatedObject(...)



```
void set_AnnnotatedObject  
    (IndependentObject value)
```

- Parameters
 - value – Specifies an IndependentObject object to which this annotation has been applied.

© Copyright IBM Corporation 2011

Figure 13-14. annotation.set_AnnnotatedObject(...)

F1432.0

Notes:

Create Annotation objects

Sample code to create an Annotation object

```
Annotation myAnno =
    Factory.Annotation.createInstance
        (objectStore, ClassNames.ANNOTATION);
ContentElementList contentList =
    Factory.ContentElement.createList();
ContentTransfer content =
    Factory.ContentTransfer.createInstance();
content.setCaptureSource(new
    FileInputStream("annotationText.txt"));
content.set_ContentType("text/plain");
contentList.add(content);
myAnno.set_ContentElements(contentList);
myAnno.set_AnnotatedObject(document);
myAnno.save(RefreshMode.REFRESH);
```

© Copyright IBM Corporation 2011

Figure 13-15. Sample code to create an Annotation object

F1432.0

Notes:

Create Annotation objects

Demonstrations



- Create an Annotation object
 - Set the content to a Annotation object.
 - Associate the Annotation object to a target object.

© Copyright IBM Corporation 2011

Figure 13-16. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *createAnnotationEDU(...)* method in the *AnnotationsEDU.java* file.
3. Run the code and verify the results as instructed in the *Annotations* unit in the Student Exercises book.

Create Annotation objects

Activities

In your Student Exercises book

- Unit: Annotations
- Lesson: Create Annotation objects
- Activities
 - Work with graphical annotations in Image Viewer.
 - Create Annotation objects.

© Copyright IBM Corporation 2011

Figure 13-17. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 13.2. Retrieve annotations

Lesson: Retrieve annotations

- 
- Why is this lesson important to you?
 - The Sales team adds their comments to documents in the form of annotations. Anna, a Product Development team member, needs to retrieve their comments and incorporate them into the documents. As the programmer, you are going to write code to retrieve the annotation content when Anna accesses the comments.

© Copyright IBM Corporation 2011

Figure 13-18. Lesson: Retrieve annotations

F1432.0

Notes:

Retrieve annotations

Activities that you need to complete



- Retrieve annotations.

© Copyright IBM Corporation 2011

Figure 13-19. Activities that you need to complete

F1432.0

Notes:

Retrieve annotations

Steps to retrieve annotations



1. Retrieve annotations for a given document.

```
document.get_Annotations(...)
```

2. Retrieve the name of each annotation in collection.

```
annotationSet.iterator()  
annotation.get_Name()
```

3. Retrieve the creator and the date created for each annotation in collection.

```
annotation.get_Creator()  
annotation.get_DateCreated()
```

© Copyright IBM Corporation 2011

Figure 13-20. Steps to retrieve annotations

F1432.0

Notes:

Retrieve annotations

document.get_Annotations()



```
AnnotationSet get_Annotations()
```

- Returns
 - An AnnotationSet object that contains the annotations associated with this object

© Copyright IBM Corporation 2011

Figure 13-21. document.get_Annotations()

F1432.0

Notes:

Retrieve annotations
annotation.get_Name()



```
String get_Name()
```

- Returns
 - The name for this object

© Copyright IBM Corporation 2011

Figure 13-22. annotation.get_Name()

F1432.0

Notes:

Retrieve annotations

annotation.get_Creator()



```
String get_Creator()
```

- Returns
 - The name of the user assigned as the creator of the object

© Copyright IBM Corporation 2011

Figure 13-23. annotation.get_Creator()

F1432.0

Notes:

Retrieve annotations
annotation.get_DateCreated()



```
java.util.Date get_DateCreated()
```

- Returns
 - The date and time that the object was created
- Notes
 - The Content Engine stores dates and times using Universal Time Coordinates (UTC).

© Copyright IBM Corporation 2011

Figure 13-24. annotation.get_DateCreated()

F1432.0

Notes:

Retrieve annotations

Sample code to retrieve annotations



```
AnnotationSet annos =  
    document.get_Annotations();  
Iterator it = annos.iterator();  
while (it.hasNext())  
{  
    Annotation anno = (Annotation)it.next();  
    System.out.println ("Anno =" +  
                       anno.get_Name() );  
    System.out.println(", created by " +  
                       anno.get_Creator() + " on " +  
                       anno.get_DateCreated().toString());  
}
```

© Copyright IBM Corporation 2011

Figure 13-25. Sample code to retrieve annotations

F1432.0

Notes:

Retrieve annotations

Demonstrations



- Retrieve annotations
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 13-26. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getAnnotationsEDU(...)* method in the *AnnotationsEDU.java* file.
3. Run the code and verify the results as instructed in the *Annotations* unit in the Student Exercises book.

Retrieve annotations

Activities

In your Student Exercises book

- Unit: Annotations
- Lesson: Retrieve Annotation objects
- Activities
 - Retrieve annotations.

© Copyright IBM Corporation 2011

Figure 13-27. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 13.3. Copy annotations

Lesson: Optional: Copy annotations

- 
- Why is this lesson important to you?
 - Anna, on the Product Development team, has modified documents based on the sales team's suggestions that were in the annotations. When Anna checks in the documents, her manager needs to view the sales recommendations as well as her revisions. Annotations are not automatically carried over to the next version of the document. As the programmer, you are going to write code to copy annotations from one version to the another.

© Copyright IBM Corporation 2011

Figure 13-28. Lesson: Optional: Copy annotations

F1432.0

Notes:

Copy annotations

Activities that you need to complete



- Create a new version for the document.
- Copy annotations.

© Copyright IBM Corporation 2011

Figure 13-29. Activities that you need to complete

F1432.0

Notes:

Copy annotations

Steps to copy annotations



1. Retrieve the versions for the given document.
2. Select the version from which to copy the annotation.
3. Retrieve the annotations from that document version.
4. Create a new Annotation object for each annotation to be copied.
5. Use the content of the old annotation to set the content to the new annotation.
6. Set the document as the annotated object to each of the new annotation.
7. Save the new annotations.
8. Save the document.

© Copyright IBM Corporation 2011

Figure 13-30. Steps to copy annotations

F1432.0

Notes:

Copy annotations

Sample code to copy annotations (1)

```
VersionableSet versions=document.get_Versions();  
Versionable documentVersion;  
Document docCopy = null;  
Iterator versionableIt;  
versionableIt = versions.iterator();  
while (versionableIt.hasNext()) {  
    documentVersion =  
        (Versionable)versionableIt.next();  
    Integer majorNum =  
        documentVersion.get_MajorVersionNumber();  
    if (versionNumber == majorNum) {  
        docCopy = (Document)documentVersion;  
        System.out.println("Document Version: " +  
            majorNum + " is retrieved");  
        AnnotationSet annotations =  
            docCopy.get_Annotations();
```

© Copyright IBM Corporation 2011

Figure 13-31. Sample code to copy annotations (1)

F1432.0

Notes:

Copy annotations

Sample code to copy annotations (2)

```
if (annotations != null){  
    it = annotations.iterator();  
    while (it.hasNext()) {  
        Annotation annotation =  
            (Annotation) it.next();  
        ContentTransfer oldContent =  
            (ContentTransfer) annotation.get_ContentElemen  
ts().get(0);  
        Annotation newAnnotation =  
            Factory.Annotation.createInstance(os, ClassNam  
es.ANNOTATION);  
        ContentElementList contentList =  
            Factory.ContentElement.createList();
```

© Copyright IBM Corporation 2011

Figure 13-32. Sample code to copy annotations (2)

F1432.0

Notes:

Copy annotations

Sample code to copy annotations (3)

```
ContentTransfer content =
Factory.ContentTransfer.createInstance();
content.setCaptureSource(annotation.accessContentStream(0));
content.set_RetrievalName(oldContent.get_RetrievalName());
content.set_ContentType(oldContent.get_ContentType());
contentList.add(content);
newAnnotation.set_ContentElements
                (contentList);
newAnnotation.set_AnnotatedObject(document);
newAnnotation.save(RefreshMode.REFRESH);
} // while
} // if
document.Save(RefreshMode.NO_REFRESH)
```

© Copyright IBM Corporation 2011

Figure 13-33. Sample code to copy annotations (3)

F1432.0

Notes:

Copy annotations

Demonstrations



- Copy an annotation from one version of the document to another version
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 13-34. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *copyAnnotationsEDU(...)* method in the *AnnotationsEDU.java* file.
3. Run the code and verify the results as instructed in the *Annotations* unit in the Student Exercises book.

Copy annotations

Activities

In your Student Exercises book

- Unit: Annotations
- Lesson: Optional: Copy annotations
- Activities
 - Create a new version for the document.
 - Copy annotations.

© Copyright IBM Corporation 2011

Figure 13-35. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 14. Document Lifecycle

What this unit is about

This unit describes how to create a document lifecycle policy, how to retrieve a lifecycle policy and how to change a document lifecycle state.

What you should be able to do

After completing this unit, you should be able to:

- Create a lifecycle policy and assign it to a Document class.
- Retrieve a document lifecycle policy.
- Change a document lifecycle state.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Document Lifecycle

Unit lesson



- Change a document lifecycle state
- Create lifecycle actions

© Copyright IBM Corporation 2011

Figure 14-1. Unit lesson

F1432.0

Notes:

Lesson 14.1. Change a document lifecycle state

Lesson: Change a document lifecycle state

- 
- Why is this lesson important to you?
 - The Product Development team in your company wants to create its content as document objects, and then move these objects through four states: Draft, Approved, Published, and Archived. You, as their programmer, are going to write code to change a document lifecycle state.

© Copyright IBM Corporation 2011

Figure 14-2. Lesson: Change a document lifecycle state

F1432.0

Notes:

Change a document lifecycle state

Activities that you need to complete



- Create a lifecycle policy and assign it to a Document class.
- Change a document lifecycle state.

© Copyright IBM Corporation 2011

Figure 14-3. Activities that you need to complete

F1432.0

Notes:

Change a document lifecycle state

Document lifecycles



- A document lifecycle is
 - The set of user-defined states through which a document moves from its creation to final disposition
 - These states are specified in a policy that defines the lifecycle.
 - Example: Planned, Scheduled, In Progress, and Completed
- Two Content Engine objects define a lifecycle
 - Document lifecycle policy
 - Lifecycle action
- Administrator configures document lifecycles
 - Uses Content Engine Enterprise Manager to create the actions and policy.
- CURRENT_STATE Property of a document
 - Represents the current lifecycle state of the document.

© Copyright IBM Corporation 2011

Figure 14-4. Document lifecycles

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Lifecycles > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Document Lifecycle Policies > Lifecycle Policy Concepts

Change a document lifecycle state

Sample document lifecycle state

The screenshot shows a software interface for managing document lifecycle states. At the top, there's a decorative header bar with various icons. Below it is a navigation bar with links: My Workplace, Tasks, Shortcuts, Browse, Search, and Author. On the left, a sidebar titled 'Information' contains links for Properties, Security, Parent Documents, Versions, Folders Filed In, and History. Under 'Actions', there are links for Download, Check Out, Cancel Checkout, Check In, Quick Check In, and Save Content. The main content area shows a document named 'APILifecycle.jpg (Version 1.0, Released)' with a class of 'LifeCycleDoc'. A table lists properties: Document Title (APILifecycle.jpg) and Life Cycle State. The 'Life Cycle State' dropdown menu is open, showing options: Approved (no change), Draft (demote), Approved (no change), Published (promote), set exception, and reset. The 'Approved (no change)' option is highlighted. To the right of the table are 'Printable View', 'Apply', and 'Exit' buttons.

© Copyright IBM Corporation 2011

Figure 14-5. Sample document lifecycle state

F1432.0

Notes:

This screen capture shows a document with lifecycle states.

The Clear Exception option is available in the menu only when the document is in the Exception state.

Change a document lifecycle state

Document lifecycle policy and lifecycle action



- A document lifecycle policy
 - Defines a set of states that a document moves through in a business process.
 - Has linear transitions.
 - Specifies allowed forward and backward movement for each state (promotion and demotion).
 - Identifies the action that is executed when a document lifecycle state changes.
- A document lifecycle action
 - Is an action that the system performs when a document moves from one state to another.
 - Executes a Java class defined in the lifecycle policy.

© Copyright IBM Corporation 2011

Figure 14-6. Document lifecycle policy and lifecycle action

F1432.0

Notes:

Help paths

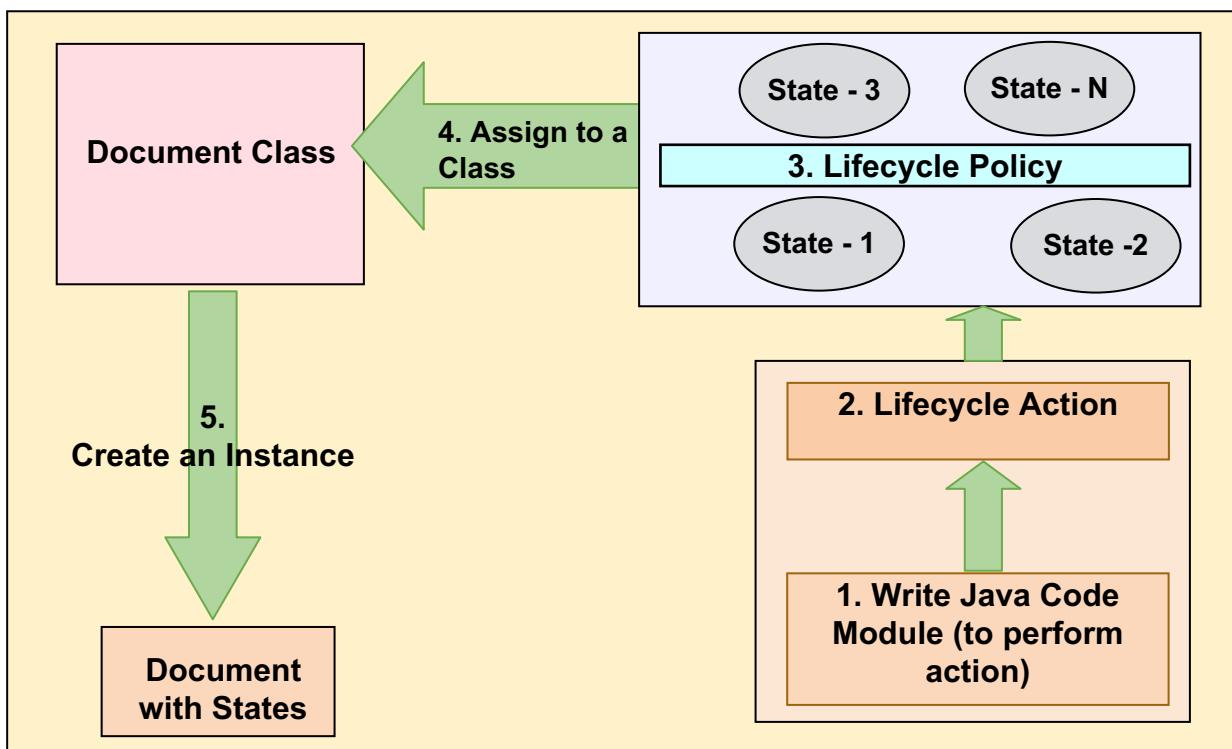
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Lifecycles > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Document Lifecycle Policies > Lifecycle Policy Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Document Lifecycle Policies > Lifecycle Policy Concepts > Lifecycle Policy Setup Requirements

The lifecycle actions are developed as Java classes.

The next lesson provides information about creating lifecycle actions.

Change a document lifecycle state

Setup process for lifecycle policy and action



© Copyright IBM Corporation 2011

Figure 14-7. Setup process for lifecycle policy and action

F1432.0

Notes:

Setup process for lifecycle policy and action

- **Write a Java class:** This class implements the DocumentLifecycleActionHandler interface. This Java class includes methods for the actions to be taken when a document lifecycle state is changed. Examples: Update a database, write to a log file, file or unfile a document, and so on.
- **Create a lifecycle action object** with the Java code module using the FileNet Enterprise Manager.
- **Assign the lifecycle action to a lifecycle policy:** The lifecycle policy defines the states for a Document class and optionally includes a lifecycle action.
- **Assign the lifecycle policy to a Document class:** When the policy is applied to a Document class, the document can be moved through states as defined in the policy.
- **Create an instance of Document class with a policy:** This action adds a document that has the defined states. When the state of the document is changed, the lifecycle action as defined in the Java class is triggered.

Change a document lifecycle state

Steps to change a document lifecycle state



1. Retrieve the lifecycle policy associated with a document.

```
document.get_DocumentLifecyclePolicy()
```

2. Get the current lifecycle state.

```
document.getProperties()  
properties.getStringValue(...)
```

3. Get the possible lifecycle states.

```
documentLifecyclePolicy.get_DocumentStates()  
documentState.get_StateName()
```

4. Promote or demote the document (next page).

© Copyright IBM Corporation 2011

Figure 14-8. Steps to change a document lifecycle state

F1432.0

Notes:

Change a document lifecycle state

Promote or demote the document

- Promote the document.

```
document.changeState(LifecycleChangeFlags.PROMOTE)
```

Or

- Verify that the document can be demoted and demote it.

```
documentState.get_CanBeDemoted.booleanValue()  
document.changeState(LifecycleChangeFlags.DEMOTE)
```

© Copyright IBM Corporation 2011

Figure 14-9. Promote or demote the document

F1432.0

Notes:

Change a document lifecycle state

document.get_DocumentLifecyclePolicy()



```
DocumentLifecyclePolicy  
    get_DocumentLifecyclePolicy()
```

- Returns
 - The DocumentLifecyclePolicy object

© Copyright IBM Corporation 2011

Figure 14-10. document.get_DocumentLifecyclePolicy()

F1432.0

Notes:

Change a document lifecycle state

document.getProperties()

Properties getProperties()

- Returns
 - A Properties collection representing the cached properties of this object
- Notes
 - An object reference does not have values in its property collection.
 - If the properties of an object reference are required, you must refresh the object before calling this method.

© Copyright IBM Corporation 2011

Figure 14-11. document.getProperties()

F1432.0

Notes:

Change a document lifecycle state

properties.getStringValue(...)



```
String getStringValue(String propertyName)
```

- Parameters
 - propertyName - Specify the property name
 - Example: PropertyNames.CURRENT_STATE
- Returns
 - The value of the named property in a String
 - Null if the property value was not set

© Copyright IBM Corporation 2011

Figure 14-12. properties.getStringValue(...)

F1432.0

Notes:

Change a document lifecycle state

documentLifecyclePolicy.get_DocumentStates()



```
DocumentStateList get_DocumentStates()
```

- Returns
 - A collection of DocumentState objects that specify the lifecycle state names available for this Document object

© Copyright IBM Corporation 2011

Figure 14-13. documentLifecyclePolicy.get_DocumentStates()

F1432.0

Notes:

Change a document lifecycle state

documentState.get_StateName()



```
String get_StateName()
```

- Returns
 - The name of this lifecycle state as defined by its lifecycle policy
- Notes
 - This method is for information only.
 - It is not needed to promote or demote a document.
 - If this lifecycle state becomes a current lifecycle state of a document, the server sets CurrentState property of the document to the value of this property.

© Copyright IBM Corporation 2011

Figure 14-14. documentState.get_StateName()

F1432.0

Notes:

Change a document lifecycle state

documentState.get_CanBeDemoted()



```
Boolean get_CanBeDemoted()
```

- Returns
 - True if the current lifecycle state of this Document object can be demoted (changed to the previous state) to the previous state
- Notes
 - The lifecycle policy specifies whether a state can be demoted.

© Copyright IBM Corporation 2011

Figure 14-15. documentState.get_CanBeDemoted()

F1432.0

Notes:

Change a document lifecycle state

document.changeState(...)

```
void changeState(LifecycleChangeFlags flags)
```

- Parameters

- flags - Specify an action to perform (Examples: promote, demote)
- Use a LifecycleChangeFlags constant.

Examples:

- LifecycleChangeFlags.PROMOTE
- LifecycleChangeFlags.DEMOTE

- Notes

- This method changes the current lifecycle state of this document.
- Cannot change the lifecycle state of a document if it is a reservation object.

© Copyright IBM Corporation 2011

Figure 14-16. document.changeState(...)

F1432.0

Notes:

Valid LifecycleChangeFlags constants are as follows:

PROMOTE: Promotes the current lifecycle state to the next state in its document lifecycle policy and sets the CurrentState property of the document to the name of the next state.

DEMOTE: Demotes the current lifecycle state to the previous state in its document lifecycle policy and sets the CurrentState property of the document to the name of the previous state, unless the CanBeDemoted property of the document current state is set to false. Throws an error if the document is in the exception state or in the first state of its lifecycle.

SET_EXCEPTION: Places the document into the exception state and sets its IsInExceptionState property to true. Throws an error if the document IsInExceptionState property is already set to true.

CLEAR_EXCEPTION: Removes the document from the exception state and sets its IsInExceptionState property to false. Throws an error if the document IsInExceptionState property is already set to false.

RESET: Resets the lifecycle state to the initial state in its document lifecycle policy and sets the document CurrentState property to the name of the initial state. Throws an error if the document is in the exception state.

Change a document lifecycle state

Sample code to get document lifecycle states

Verify whether the document has an associated lifecycle policy.

```
DocumentLifecyclePolicy lifeCyclePolicy =  
    document.get_DocumentLifecyclePolicy();
```

Get the document lifecycle states and display the names.

```
if !(lifeCyclePolicy == null) {  
    DocumentStateList lifeCycleStates =  
        lifeCyclePolicy.get_DocumentStates();  
    Iterator stateIt = lifeCycleStates.iterator();  
    while (stateIt.hasNext()) {  
        DocumentState state =  
            (DocumentState) stateIt.next();  
        String stateName = state.get_StateName();  
        System.out.println("state = " + stateName);  
    }  
}
```

© Copyright IBM Corporation 2011

Figure 14-17. Sample code to get document lifecycle states

F1432.0

Notes:

Change a document lifecycle state

Sample code to get current lifecycle state



```
Properties properties = document.getProperties()
String currentState = properties.getStringValue
(PropertyNames.CURRENT_STATE);
System.out.println("current state=" +
currentState);
```

© Copyright IBM Corporation 2011

Figure 14-18. Sample code to get current lifecycle state

F1432.0

Notes:

Change a document lifecycle state

Sample code to promote or demote a document

Determine if the current state can be demoted and demote the document.

```
if (documentState.get_CanBeDemoted().booleanValue()) {  
  
    document.changeState(LifecycleChangeFlags.DEMOTE)  
}
```

Promote the document.

```
document.changeState(LifecycleChangeFlags.PROMOTE)
```

© Copyright IBM Corporation 2011

Figure 14-19. Sample code to promote or demote a document

F1432.0

Notes:

Change a document lifecycle state

Demonstrations



- Create a lifecycle policy and assign it to a Document class
- Change a document lifecycle state
 - Retrieve a document lifecycle policy.
 - Retrieve document lifecycle states.
 - Change a document lifecycle state.

© Copyright IBM Corporation 2011

Figure 14-20. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

Create a lifecycle policy and assign it to a Document class

1. Open the Content Engine Enterprise Manager.
2. Create a lifecycle policy using the wizard as instructed in the *Document Lifecycle* unit in the Student Exercises book.
3. Assign the policy to a Document class.

Change a document lifecycle state

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *changeLifeCycleEDU (...)* method in the *DocumentLifecycleEDU.java* file.
3. Run the code and verify the results as instructed in the *Document Lifecycle* unit in the Student Exercises book.

Change a document lifecycle state

Activities



In your Student Exercises book

- Unit: Document Lifecycle
- Lesson: Change a document lifecycle state
- Activities
 - Create a lifecycle policy and assign it to a Document class.
 - Change a document lifecycle state.

© Copyright IBM Corporation 2011

Figure 14-21. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 14.2. Create lifecycle actions

Lesson: Create lifecycle actions

- 
- Why is this lesson important to you?
 - You are going to write Java classes that will be used in lifecycle actions to do the following:
 - File a document to a folder when the document is promoted.
 - Unfile the document from that folder when the document is demoted.
 - Log information to a file when the document state changes.

© Copyright IBM Corporation 2011

Figure 14-22. Lesson: Create lifecycle actions

F1432.0

Notes:

Create lifecycle actions

Activities that you need to complete



- Create a code module for a lifecycle action.
- Deploy the code module and test the lifecycle policy.

© Copyright IBM Corporation 2011

Figure 14-23. Activities that you need to complete

F1432.0

Notes:

Create lifecycle actions

DocumentLifecycleActionHandler interface



- DocumentLifecycleActionHandler
 - Represents the interface used by Java application clients in order to implement document lifecycle action handlers.
- To implement code that the server executes during a given lifecycle state change, do the following:
 1. Create a class to implement this interface.
 2. Populate the specific method for each lifecycle state change for which you want an action to be taken.

© Copyright IBM Corporation 2011

Figure 14-24. DocumentLifecycleActionHandler interface

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Document Lifecycle Policies > Working with Lifecycle-related Objects > Creating a Lifecycle Action Handler

To create a document lifecycle action handler, you must implement all of the methods in the Java DocumentLifecycleActionHandler interface.

Create lifecycle actions

Steps to create a lifecycle action



1. Create a Java class that does the following:
 - Implements DocumentLifecycleActionHandler interface.
 - Implements required methods.
2. Compile the Java code to get the class file.
 - Optionally, archive the class files in a JAR file.
3. Use Content Engine Enterprise Manager to create the code module (optional) and lifecycle action.
4. Associate the lifecycle action with a lifecycle policy.
5. Associate the lifecycle policy with a document class.

© Copyright IBM Corporation 2011

Figure 14-25. Steps to create a lifecycle action

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Lifecycles > Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Document Lifecycle Policies > Working with Lifecycle-related Objects > Creating a Lifecycle Action Handler

Create lifecycle actions

documentLifecycleActionHandler.onDocumentPromote(...)



```
void onDocumentPromote(Document doc,  
    DocumentLifecyclePolicy policy) throws  
    EngineRuntimeException
```

- Parameters

- doc - Specifies a Document object whose lifecycle state is being changed.
- policy – Specifies a DocumentLifecyclePolicy object that is associated with the document.

- Notes

- This action executes when the lifecycle state of a document is promoted.

© Copyright IBM Corporation 2011

Figure 14-26. documentLifecycleActionHandler.onDocumentPromote(...)

F1432.0

Notes:

Create lifecycle actions

documentLifecycleActionHandler.onDocumentDemote(...)

```
void onDocumentDemote(Document doc,  
                      DocumentLifecyclePolicy policy) throws  
                      EngineRuntimeException
```

- Parameters

- doc - Specifies a Document object whose lifecycle state is being changed.
- policy – Specifies a DocumentLifecyclePolicy object that is associated with the document.

- Notes

- This action executes when the lifecycle state of a document is demoted.

© Copyright IBM Corporation 2011

Figure 14-27. documentLifecycleActionHandler.onDocumentDemote(...)

F1432.0

Notes:

Create lifecycle actions

document.get_CurrentState()



```
String get_CurrentState()
```

- Returns
 - The current lifecycle state of this document as defined by its document lifecycle policy
- Notes
 - Each document lifecycle policy defines a set of states through which a document can move during its lifecycle.

© Copyright IBM Corporation 2011

Figure 14-28. document.get_CurrentState()

F1432.0

Notes:

Create lifecycle actions

EngineRuntimeException raised by the lifecycle handler

- Used for all unchecked exceptions.
- Defined as ExceptionCode objects.
- Support the use of an ErrorStack.

```
catch (IOException e) {
    throw new EngineRuntimeException
        (ExceptionCode.E_FAILED, e);
}

Or

catch (IOException e) {
    ErrorRecord er[] = {new ErrorRecord (e)};
    ErrorStack es = new
    ErrorStack(e.getMessage(), er);
    throw new EngineRuntimeException(es);
}
```

© Copyright IBM Corporation 2011

Figure 14-29. EngineRuntimeException raised by the lifecycle handler

F1432.0

Notes:

Exceptions raised by the lifecycle handler support the exception chaining (constructing a chain of exceptions by wrapping additional exceptions).

Create lifecycle actions

Error logging and debugging

- What can go wrong?
 - Errors in Java classes
 - Missing Java classes
- Debugging: Error messages appear during actions.

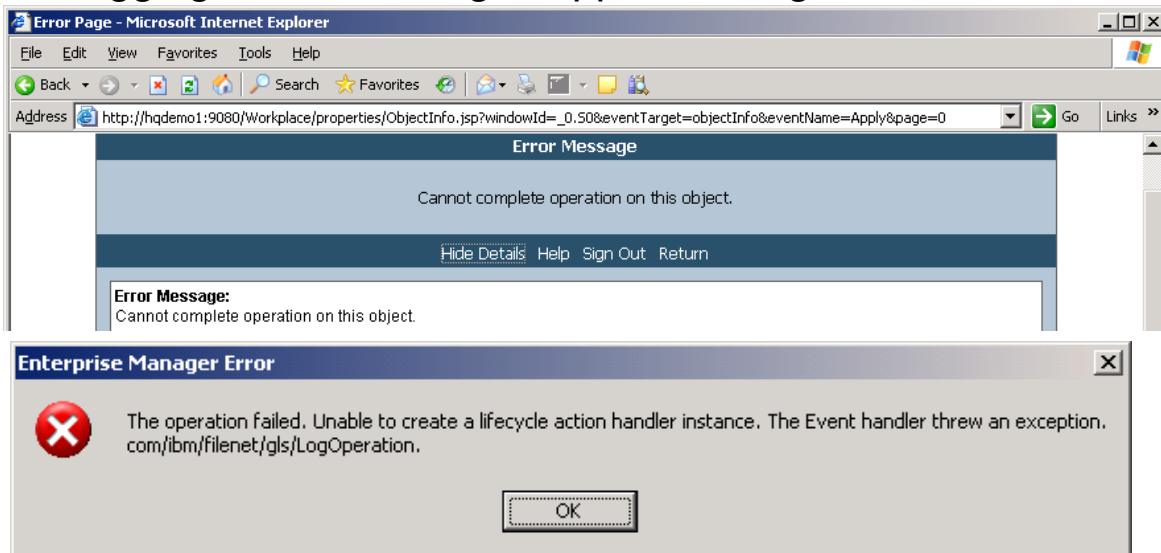


Figure 14-30. Error logging and debugging

F1432.0

Notes:

Create lifecycle actions

Code modules



- Code modules are versioned documents.
 - Used to store the lifecycle action handler.
- Two options to deploy document lifecycle action handler:
 - Set the location in the classpath of the application server on which the Content Engine runs.
 - Check it into an object store as a CodeModule object.

© Copyright IBM Corporation 2011

Figure 14-31. Code modules

F1432.0

Notes:

A code module is automatically available in these situations:

- When you are deploying the Content Engine to multiple application server instances
- When you are moving your content metadata from one system to another

If you reference document lifecycle action handler in the classpath of an application server, you must manually distribute them to new systems.

Create lifecycle actions

Update lifecycle action



- If you modify the code for a Java lifecycle action handler contained within a code module:
 - You must update any lifecycle actions referencing the code module.
- Steps to update:
 1. Check out the code module.
 2. Modify the Java lifecycle action handler source and compile.
 3. Check in the code module with the new version of the Java class.
 4. Copy the object reference for the code module.
 5. Update the Code Module property of the lifecycle action by pasting the object reference.

© Copyright IBM Corporation 2011

Figure 14-32. Update lifecycle action

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Events Subscription > How to > Work with event actions > Modify an action

Create lifecycle actions

Adding external JAR files to the Content Engine

- Some lifecycle actions might require external JAR files.
 - Example: For accessing a Database
- These files can be added in one of the following ways:
 - Add as content elements of the code module.
 - Imported with the Content Engine objects
 - Copy to the application server or set in the classpath.
 - Globally available for the other lifecycle actions

© Copyright IBM Corporation 2011

Figure 14-33. Adding external JAR files to the Content Engine

F1432.0

Notes:

- Add JAR files as content elements of the code module:

Advantage: When importing an object store from one Content Engine to the other, the library files are also imported, so it is not necessary to set the classpath.
- Copy JAR files to the application server or set in the classpath:

Example: Copy JAR files to the Websphere folder: C:\Program Files\IBM\WebSphere\AppServer\lib\ext

Advantage: Globally available for other lifecycle actions

 - When referencing external JAR files directly (that is, not as code modules), a best practice is to copy them to the application server external library directory. If they are left in their original location, the resulting CLASSPATH variable might be too long.

Create lifecycle actions

Sample code to use lifecycle action handler (1)

```
public class LifeCycleActionEDU implements  
    DocumentLifecycleActionHandler {  
  
    public void onDocumentPromote(Document doc,  
        DocumentLifecyclePolicy policy) throws  
        EngineRuntimeException{  
  
        if(doc.get_CurrentState() .equalsIgnoreCase  
            ("Approved"))  
  
        { ... }  
    }  
  
    ...  
  
    public void onDocumentDemote(Document doc,  
        DocumentLifecyclePolicy policy) throws  
        EngineRuntimeException  
  
    { ... }  
}
```

© Copyright IBM Corporation 2011

Figure 14-34. Sample code to use lifecycle action handler (1)

F1432.0

Notes:

Create lifecycle actions

Sample code to use lifecycle action handler (2)

- Implement other methods.

```
public void onDocumentSetException( Document doc,
    DocumentLifecyclePolicy policy )
    throws EngineRuntimeException
{...}

public void onDocumentClearException( Document doc,
    DocumentLifecyclePolicy policy )
    throws EngineRuntimeException
{...}

public void onDocumentResetLifecycle(Document doc,
    DocumentLifecyclePolicy policy )
    throws EngineRuntimeException
{...}
```

© Copyright IBM Corporation 2011

Figure 14-35. Sample code to use lifecycle action handler (2)

F1432.0

Notes:

Create lifecycle actions

Sample code to add custom methods

```
public void updateProductEDU(String product_id,
    String style) throws Exception{
    try {Class.forName("com.ibm.db2.jcc.DB2Driver");
        java.sql.Connection connection =
        DriverManager.getConnection
            ("jdbc:db2://ccv01123:50000/PRODUCTD", "db2inst1",
            "acm4ever");
        CallableStatement cs = connection.prepareCall
            ("'{call InsertProduct(?,?)}'");
        cs.setString(1, product_id);
        cs.setString(2, style);
        cs.execute();
    } //try
    catch (Exception e)
    {...}
```

© Copyright IBM Corporation 2011

Figure 14-36. Sample code to add custom methods

F1432.0

Notes:

Create lifecycle actions

Demonstrations



- Create a code module for a lifecycle action

© Copyright IBM Corporation 2011

Figure 14-37. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

Start Eclipse and open the *CMJavaAPISolution* project.

1. Review the code for the *LifeCycleActionEDU.java* file. It implements the DocumentLifecycleActionHandler and its methods.
2. Deploy the code module by creating a lifecycle action and a lifecycle policy.
3. Associate the policy to a Document class.
4. Verify the results as instructed in the *Document Lifecycle* unit in the Student Exercises book.

Create lifecycle actions

Activities



In your Student Exercises book

- Unit: Document Lifecycle
- Lesson: Create lifecycle actions
- Activities
 - Create a code module for a lifecycle action.
 - Deploy the code module and test the lifecycle policy.

© Copyright IBM Corporation 2011

Figure 14-38. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 15. Publishing

What this unit is about

This unit describes how to create a publish template and how to publish and republish a document.

What you should be able to do

After completing this unit, you should be able to:

- Create a publish template.
- Publish documents.
- Republish documents.

How you will check your progress

- Successful completion of lesson exercises.

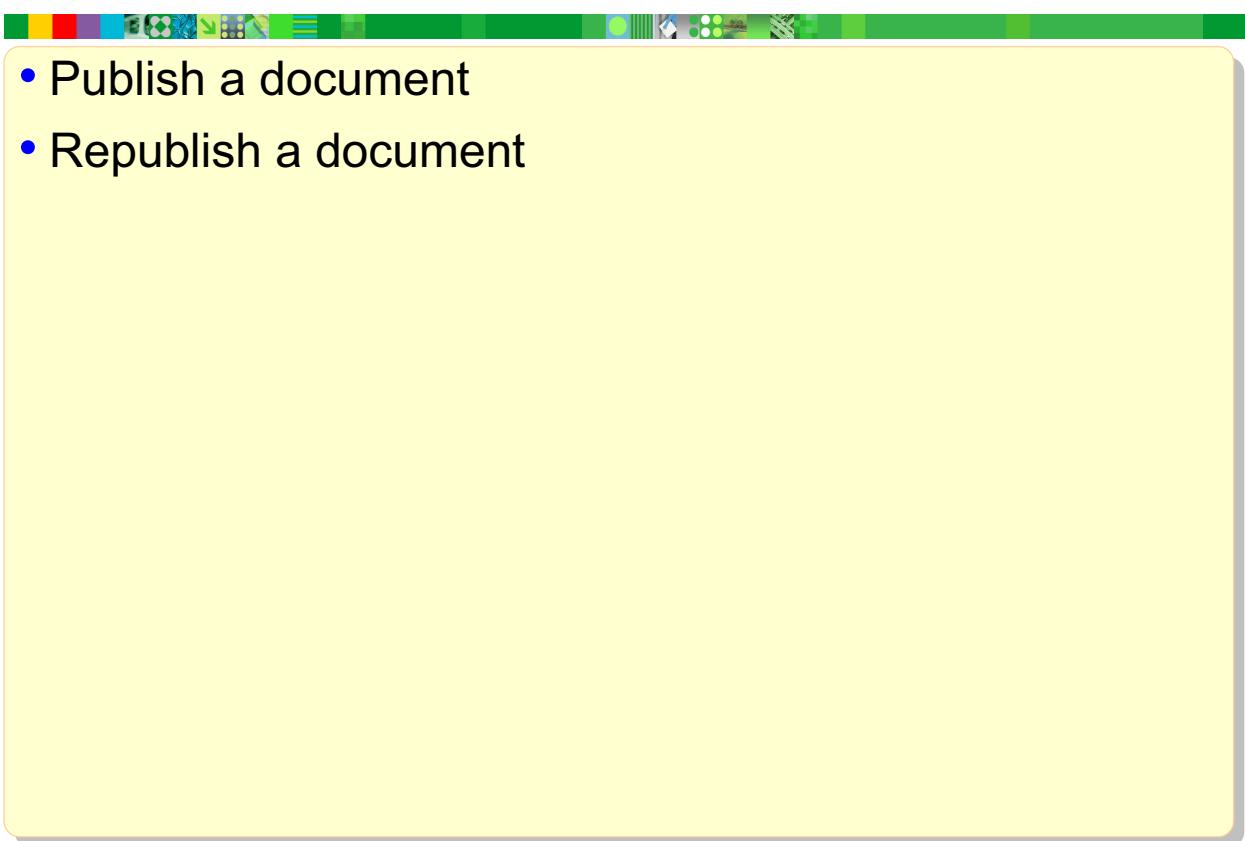
References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Publishing

Unit lessons



© Copyright IBM Corporation 2011

Figure 15-1. Unit lessons

F1432.0

Notes:

Lesson 15.1. Publish a document

Lesson: Publish a document

- 
- Why is this lesson important to you?
 - The Marketing department creates documents that describe their products. When the documents are ready to be put on the company Web site, the Marketing department needs to publish the documents so that they are filed in the proper folder, with the proper security. As the programmer, you are going to write code to publish a document.

© Copyright IBM Corporation 2011

Figure 15-2. Lesson: Publish a document

F1432.0

Notes:

Publish a document

Activities that you need to complete



- Publish a document.

© Copyright IBM Corporation 2011

Figure 15-3. Activities that you need to complete

F1432.0

Notes:

Publish a document

Publishing overview



- Enables a replica of a document to be made.
- The replica – publication document
 - Can have its own securities.
 - Can have its own properties.
 - Can exist in different folder from original (source) document.
 - Can have different format from the original document (PDF or HTML).
 - Is not changed when the source document is changed.

© Copyright IBM Corporation 2011

Figure 15-4. Publishing overview

F1432.0

Notes:

Help paths

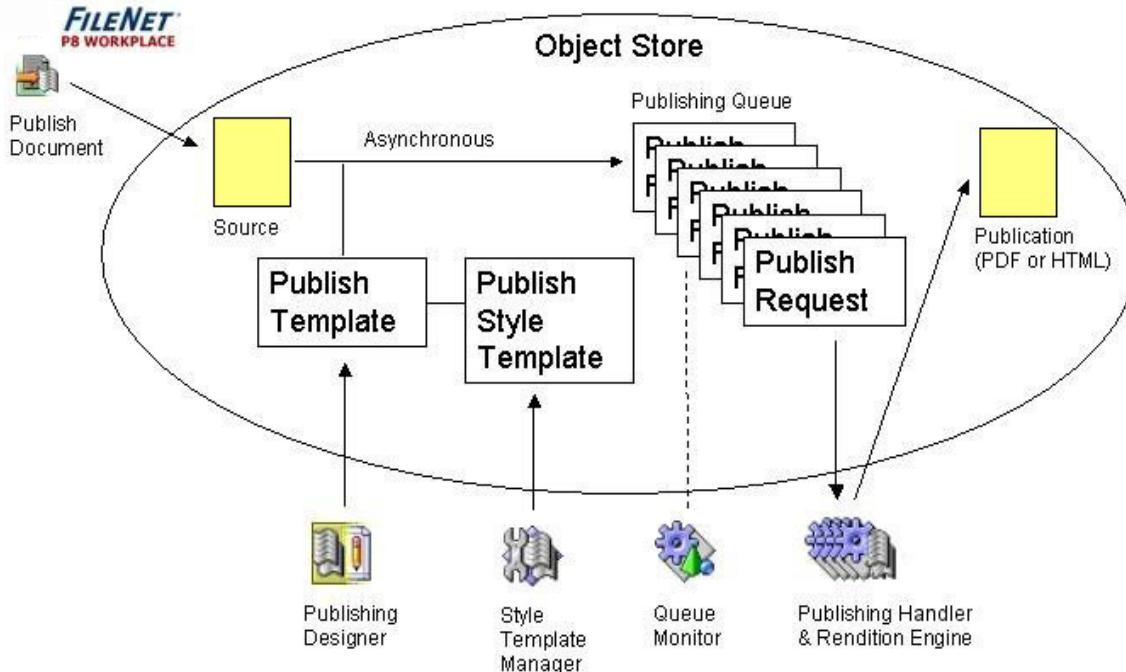
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Publishing Concepts
- IBM FileNet P8 Version 5.0 information center > Working with documents > Publishing documents > Publishing overview

You can choose to publish a document many times.

You can also set different options on each of the generated publication documents.

Publish a document

Publishing components



© Copyright IBM Corporation 2011

Figure 15-5. Publishing components

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Working with documents > Publishing documents > Publishing overview > Publishing environment diagram
- IBM FileNet P8 Version 5.0 information center > Working with documents > Publishing documents > Publishing overview

Publishing consists of the following components:

Publishing Designer - A Workplace applet for designing publish templates.

Publishing Handler (collocated on Content Engine server):

A software application for handling publishing requests for file extension of a specific publication. The IBM FileNet P8 Platform offers PDF and HTML publishing.

Liquid RenderPerfect and DCOM Business Service Manager must be installed on a machine that has Content Engine installed.

Rendition Engine

A software server (RenderPerfect) for rendering source documents into another format (FileNet P8 Platform formats are PDF and HTML).

The Rendition Engine components (Liquent RenderPerfect, DCOM Business Service Manager) are installed on both the Content Engine machine and Rendition Engine server.

Publish a document

Publishing actions



- What publishing does:
 - Makes a document available to a different audience from the original audience.
 - Creates a new document object:
 - Automatically files this new document in a specified folder.
 - Security can be automatically set.
 - Document class can be changed.
- What publishing does not do:
 - Affect the original document.
 - Transform the content (called “rendering”).

© Copyright IBM Corporation 2011

Figure 15-6. Publishing actions

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Publishing Concepts
- IBM FileNet P8 Version 5.0 information center > Working with documents > Publishing documents > Publishing overview

The Rendition Engine is not required in order to publish a document.

The Rendition Engine is required only if the document format needs to be changed (example: transform Word to HTML).

Publish a document

Publish templates



- Applied when a document is published.
- Defined using Workplace Publishing Designer.
- Publish templates can specify:
 - Publishing options
 - Example: Delete a published document if the source is deleted
 - Destination folder to publish to
 - Security for the published document
 - Document class for the published document
 - A Style Template to apply to the published document

© Copyright IBM Corporation 2011

Figure 15-7. Publish templates

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Publishing Concepts > Publish Template

Style Template

- Is used only when transforming content (rendering).
- Specifies which MIME types can be transformed.
- Requires the Rendition Engine.

The security applied to the new publication is specified in the publish template as one of four options:

- Default security
- Source document security

- Destination folder security
- Publish template-specified security

Publish a document

MIME types and publish templates



- MIME type matters only if a publish template specifies a style template
 - Only an issue when renditions are done
- Publish templates that do not specify rendering
 - Can be used with any MIME type document
- Style templates can be specified to work only on documents with certain MIME types
 - Ensures that the style template works on a particular document.

© Copyright IBM Corporation 2011

Figure 15-8. MIME types and publish templates

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Publishing Concepts > Publish Template
- IBM FileNet P8 Version 5.0 information center > Working with documents > Publishing documents > Publishing Style Template Manager > Specify MIME types

Publish a document

Steps to publish a document



1. Get the publish template associated with the document.

```
objectStore.fetchObject(...)
```

2. Create the publish options XML string.

```
String options = "<publishoptions>..."
```

3. Publish the document and save the changes.

```
document.publish(...)  
publishRequest.save(...)
```

4. Get the publishing status (optional).

```
publishRequest.get_PublishingStatus(...)
```

© Copyright IBM Corporation 2011

Figure 15-9. Steps to publish a document

F1432.0

Notes:

Publish a document

Publish options XML



- Contains information used to automate the selection of options for a publishing request.
 - Properties to apply to the destination document
 - Security to apply to the destination document
 - Instructions for republishing a document, such as whether to version the existing destination document or replace it
- The values that you specify in the publish options XML for a publishing request override the publish options that are set in the associated publish template.

© Copyright IBM Corporation 2011

Figure 15-10. Publish options XML

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Publishing Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Working with Publishing Objects > Publishing and Republishing a Document > Publish Options XML

Publish a document

Creating the publish options XML string

- When you use the Content Engine API to publish, you can override some options specified in the publish template.
 - Examples: Published document name, folder to publish to
 - You must specify a <publishoptions> xml, which might be empty.
 - Example: "<publishoptions></publishoptions>"
- Sample publish options XML string:

```
"<publishoptions>" +
  "<publicationname>" + docName +
  "</publicationname>" +
  "<outputfoldername>" + folderName +
  "</outputfoldername>" +
"</publishoptions>"
```

© Copyright IBM Corporation 2011

Figure 15-11. Creating the publish options XML string

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Publishing Concepts
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Working with Publishing Objects > Publishing and Republishing a Document > Publish Options XML

Publish a document

instantiatingScope.fetchObject(...)

```
IndependentObject fetchObject(String  
    classIdent, String objectIdent,  
    PropertyFilter filter)
```

- Parameters

- classIdent - Specifies the class name:
`ClassNames.PUBLISH_TEMPLATE`
- objectIdent - Specifies folder path for the publish template.
- Filter - Specifies a PropertyFilter object.

- Returns

- An object of the type requested. In this case, it is PublishTemplate object.

© Copyright IBM Corporation 2011

Figure 15-12. instantiatingScope.fetchObject(...)

F1432.0

Notes:

Throughout this course, fetchInstance methods of the Factory object are most commonly used to retrieve Content Engine objects.

However, the fetchObject method of Objectstore (inherited from InstantiatingScope interface) is used in this lesson to retrieve the publish template because the Objectstore method allows path-based retrieval.

The following factory method requires the ID for the publish template that is being retrieved.

```
public static PublishTemplate fetchInstance(ObjectStore os, Id objectId, PropertyFilter filter)
```

Publish a document

document.publish(...)

```
PublishRequest publish
    (PublishTemplate publishTemplate,
     String publishOptions)
```

- Parameters
 - publishTemplate - Specifies the PublishTemplate object to use for this publish operation.
 - publishOptions - Specifies a string containing XML that specifies the publish options for the published document.
- Returns
 - The PublishRequest object that is queued for this request

© Copyright IBM Corporation 2011

Figure 15-13. document.publish(...)

F1432.0

Notes:

This method publishes this Document object according to the specifications included in the specified publish template as modified by the contents of the publishOptions parameter.

This method returns the PublishRequest object that is queued for this request.

All publish requests are asynchronous, including copy operations, which do not transform the document.

The publish template that you specify, as well as any objects specified in the publish template itself, must reside on the same object store as this Document object.

Note that publishing to a document class containing a required binary- or object-valued property is not supported because there is no way to set these property values during publishing.

After a successful call to publish, call the save method of the returned PublishRequest object to submit the request to the publish queue.

Publish a document

publishRequest.get_PublishingStatus()



PublishingStatus get_PublishingStatus ()

- This method returns the PublishingStatus enumeration value.

| PublishingStatus Enumeration | Specifies that the request... |
|------------------------------|----------------------------------|
| PUBLISHING_COMPLETE | Has been successfully processed. |
| PUBLISHING_IN_QUEUE | Has not yet been processed. |
| PUBLISHING_IN_WORK | Is currently being processed. |
| PUBLISHING_DELETING | Is pending deletion. |
| PUBLISHING_IN_ERROR | Was unsuccessfully processed. |

© Copyright IBM Corporation 2011

Figure 15-14. publishRequest.get_PublishingStatus()

F1432.0

Notes:

When the sample code is run, you most commonly receive the PUBLISHING_IN_QUEUE response. This response is returned because this API is called immediately after the document.publish() call. However, if there is a time delay between these two calls, you receive the PUBLISHING_COMPLETE response if the publish operation is successful.

Publish a document

Sample code to get the document for publishing

```
PropertyFilter pf = new PropertyFilter();
pf.addIncludeProperty(0, null, null,
                      PropertyNames.NAME,1);
pf.addIncludeProperty(0, null, null,
                      PropertyNames.DEPENDENT_DOCUMENTS,1);
pf.addIncludeProperty(0, null, null,
                      PropertyNames.DESTINATION_DOCUMENTS,1);
Document document=Factory.Document.fetchInstance
(store,"/SourceDocuments/APIPublishDoc", pf);
String documentName = document.get_Name();
System.out.println(documentName + " Document
is retrieved");
return document;
```

© Copyright IBM Corporation 2011

Figure 15-15. Sample code to get the document for publishing

F1432.0

Notes:

Publish a document

Sample code to get the publish template

- Get the publish template.

```
PublishTemplate publishTemplate =
    (PublishTemplate) objectStore.fetchObject
    (ClassNames.PUBLISH_TEMPLATE,
    "/PublishTemplates/APIPublishTemplate", null);
```

- Construct an XML string to define the name for the published document.

```
String option =
    "<publishoptions><publicationname>" +
        document.get_Name () + "_Publication
    </publicationname></publishoptions>";
```

Continued on the next slide...

© Copyright IBM Corporation 2011

Figure 15-16. Sample code to get the publish template

F1432.0

Notes:

Publish a document

Sample code to publish a document

- Publish the document.

```
PublishRequest pr = doc.publish(pubTemplate,  
                                option);
```

- Save the PublishRequest object and display the publishing status.

```
PropertyFilter pf = new PropertyFilter();  
pf.addIncludeProperty(0, null, null,  
                     PropertyNames.PUBLISHING_STATUS,1);  
pr.save(RefreshMode.REFRESH, pf);  
System.out.println("Publishing status is" +  
pr.get_PublishingStatus().toString());
```

© Copyright IBM Corporation 2011

Figure 15-17. Sample code to publish a document

F1432.0

Notes:

Publish a document

Demonstrations



- Create a publish template
- Publish a document
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 15-18. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

Create a publish template:

1. Sign in to Workplace.
2. Create a publish template using the Publishing Designer as instructed in the *Publishing* unit in the Student Exercises book.

Publish a document:

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *publishDocumentEDU(...)* method in the *PublishEDU.java* file.
3. Run the code and verify the results as instructed in the *Publishing* unit in the Student Exercises book.

Publish a document

Activities

In your Student Exercises book

- Unit: Publishing
- Lesson: Publish a document
- Activities
 - Publish a document.

© Copyright IBM Corporation 2011

Figure 15-19. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 15.2. Republish a document

Lesson: Republish a document



- Why is this lesson important to you?
 - The Marketing department has revised a document that describes their latest product update. Now they need to publish the document again to the company Web site. The document must be filed in the same folder and must have the same security applied as the previous publication.
As the programmer, you are going to write code to republish the document and use the same publish template.

© Copyright IBM Corporation 2011

Figure 15-20. Lesson: Republish a document

F1432.0

Notes:

Republish a document

Activities that you need to complete



- Republish a document.
- Optional: Retrieve publish templates.

© Copyright IBM Corporation 2011

Figure 15-21. Activities that you need to complete

F1432.0

Notes:

Republish a document

Steps to republish a document



1. Get the publications associated with the document.

```
document.get_DestinationDocuments(...)
```

2. Create the publish options XML string.

```
Options = "<publishoptions>..."
```

3. Republish the document and save the changes.

```
document.republish(...)  
publishRequest.save(...)
```

4. Get the publishing status (optional).

```
publishRequest.get_PublishingStatus(...)
```

© Copyright IBM Corporation 2011

Figure 15-22. Steps to republish a document

F1432.0

Notes:

Republish a document

Creating the publish options XML string

When republishing using the Content Engine API

- You can override some options specified in the publish template.
 - Examples: Republished document name, folder to republish
- You must specify a <publishoptions> value, which can be empty.
 - Example: "<publishoptions></publishoptions>"

© Copyright IBM Corporation 2011

Figure 15-23. Creating the publish options XML string

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Publishing > Working with Publishing Objects > Publishing and Republishing a Document > Publish Options XML

Republish a document

Document.get_DestinationDocuments(...)



```
DocumentSet get_DestinationDocuments ()
```

- Returns
 - Documents that are publications of this document

© Copyright IBM Corporation 2011

Figure 15-24. Document.get_DestinationDocuments(...)

F1432.0

Notes:

Republish a document

document.republish(...)

```
PublishRequest republish
    (Document publishedDocument, String
     publishOptions)
```

- Parameters

- publishedDocument - Specifies the Document object (that is, the published document) to republish.
- publishOptions - Specifies a string containing XML that indicates the publish options for the republished document.

- Returns

- The PublishRequest object that is queued for this request

© Copyright IBM Corporation 2011

Figure 15-25. document.republish(...)

F1432.0

Notes:

If the publishOptions parameter is null, the values for the publication name, output folder, and event action are used as specified in the associated publish template.

This method returns the PublishRequest object that is queued for this request.

If no transform is involved (the document is not rendered in a different format), the PublishRequest object reflects a completed status and does not represent a persisted publish request.

The specified published document must have been published with a publish template that still exists and has this Document object as its source document.

Republish a document

Sample code to get the publication document

- Get the publications associated with this document.

```
DocumentSet pubDocs =  
    sourceDoc.get_DestinationDocuments();
```

- Get the publication document to be republished.

```
Iterator it = publications.iterator();  
Document publication = null;  
while(it.hasNext()) {  
    publication = (Document)it.next();  
    System.out.println("publication = " +  
        publication.get_Name());  
    if (publicationName.equalsIgnoreCase  
        (publication.get_Name()))  
        break;  
}
```

© Copyright IBM Corporation 2011

Figure 15-26. Sample code to get the publication document

F1432.0

Notes:

Republish a document

Sample code to republish a document

- Construct an XML string to define the name of the republished document.

```
String publishOpts =  
    "<publishoptions>  
        <publicationname>" +  
        sourceDoc.getName() + "Republished  
    </publicationname>  
    </publishoptions>");
```

- Republish the source document.

```
sourceDoc.republish(publication, publishOpts);
```

© Copyright IBM Corporation 2011

Figure 15-27. Sample code to republish a document

F1432.0

Notes:

Republish a document

Sample code to get the republishing status

- Save the IPublishRequest object.

```
PropertyFilter pf = new PropertyFilter();
pf.addIncludeProperty(0, null, null,
    PropertyNames.PUBLISHING_STATUS);
pr.save(RefreshMode.REFRESH, pf);
```

- Display the republishing status.

```
System.out.println(pr.get_PublishingStatus().
    toString());
```

© Copyright IBM Corporation 2011

Figure 15-28. Sample code to get the republishing status

F1432.0

Notes:

Republish a document

Sample SQL statement to retrieve publish templates

- Build an SQL statement.

```
String sql = "SELECT pt.Name, pt.Id,  
pt.DateCreated, pt.StyleTemplate FROM  
PublishTemplate AS pt ";  
  
    sql += " WITH INCLUDESUBCLASSES LEFT OUTER  
JOIN PublishStyleTemplate AS st " ;  
  
    sql += " WITH INCLUDESUBCLASSES ON  
pt.StyleTemplate = st.This " ;  
  
    sql += " WHERE (pt.IsCurrentVersion = true)  
ORDER BY pt.DateCreated";
```

Continued on the next slide...

© Copyright IBM Corporation 2011

Figure 15-29. Sample SQL statement to retrieve publish templates

F1432.0

Notes:

Republish a document

Sample code to retrieve publish templates

```
SearchScope search = new SearchScope(os);
SearchSQL searchSQL = new SearchSQL(sql);
DocumentSet documents = (DocumentSet)
search.fetchObjects(searchSQL, Integer.valueOf("5
0"), null, Boolean.valueOf(true));
com.filenet.api.core.Document doc;
Iterator it = documents.iterator();
while (it.hasNext()){
    doc = (com.filenet.api.core.Document)it.next();
    System.out.println("publish template = "+
doc.get_Name());
}
```

© Copyright IBM Corporation 2011

Figure 15-30. Sample code to retrieve publish templates

F1432.0

Notes:

Republish a document

Demonstrations



- Republish a document
- Retrieve publish templates:
 - Explore the code sample.
 - Run the solution code.
 - Observe the results.

© Copyright IBM Corporation 2011

Figure 15-31. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *rePublishDocumentEDU(...)* method in the *PublishEDU.java* file.
3. Run the code and verify the results as instructed in the *Publishing* unit in the Student Exercises book.
4. Repeat steps 2 and 3 for the *getPublishTemplatesEDU (...)* method that retrieves the publish templates available for the given object store.

Republish a document

Activities



In your Student Exercises book

- Unit: Publishing
- Lesson: Republish a document
- Activities
 - Republish a document.
 - Optional: Retrieve publish templates.

© Copyright IBM Corporation 2011

Figure 15-32. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Unit 16. Compound Documents

What this unit is about

This unit describes how to create and how to retrieve compound documents.

What you should be able to do

After completing this unit, you should be able to:

- Create compound documents.
- Retrieve compound documents.

How you will check your progress

- Successful completion of lesson exercises.

References

www.ibm.com/support/documentation/us/en (Support & downloads page)

Note: IBM FileNet products belong to the Information Management product set.

Compound Documents

Unit lessons

- 
- Create compound documents
 - Retrieve compound documents

© Copyright IBM Corporation 2011

Figure 16-1. Unit lessons

F1432.0

Notes:

Lesson 16.1. Create compound documents

Lesson: Create compound documents

- 
- Why is this lesson important to you?
 - Your company wants to store several files with product information as a single document. You, as the programmer, are going to write code to create compound documents on the Content Engine.

© Copyright IBM Corporation 2011

Figure 16-2. Lesson: Create compound documents

F1432.0

Notes:

Create compound documents

Activities that you need to complete



- Create compound documents.

© Copyright IBM Corporation 2011

Figure 16-3. Activities that you need to complete

F1432.0

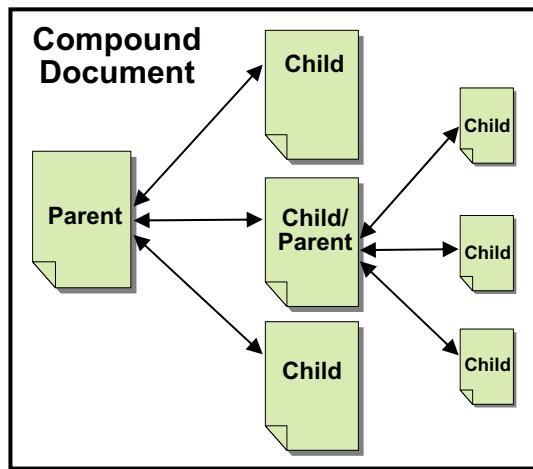
Notes:

Create compound documents

Compound documents



- Hierarchical collection of documents assembled into a single logical document



- Assembly and presentation are done by a custom application.

© Copyright IBM Corporation 2011

Figure 16-4. Compound documents

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and folders > Concepts > About compound documents
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content Engine Overview > Features > Compound documents
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Compound Documents > Concepts

Create compound documents

Components of compound documents

- A compound document consists of three components:
 - A parent document (content is optional)
 - One or more child document components
 - Component relationship objects that link a parent to each child

© Copyright IBM Corporation 2011

Figure 16-5. Components of compound documents

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and folders > Concepts > About compound documents
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content Engine Overview > Features > Compound documents
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Compound Documents > Concepts

Nesting

- A child can be the parent of another compound document.
- A parent can be a child of another compound document.

The compound relationship introduces complexity that is managed in your application. Presentation, navigation among components, checking out and in, and so on, are handled in your application.

Create compound documents

Features of compound documents



- Provide for
 - Independent modification of individual components
 - Sharing of component documents in other documents
- Separate security for each component
 - Each component document is secured independently.
 - A given user might or might not be able to see and update.
- Settable deletion behaviors
 - Cascade deletion – deleting parent deletes all children
 - Deletion prevention – cannot delete parent or children while linked

© Copyright IBM Corporation 2011

Figure 16-6. Features of compound documents

F1432.0

Notes:

Help paths

- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Documents and folders > Concepts > About compound documents
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content Engine Overview > Features > Compound documents
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Compound Documents > Concepts

Cascade Deletion settings

- Deleting parent deletes all children
- Deleting parent has no effect on children

Deletion Prevention settings

- Allow both parent and child deletion
- Prevent both parent and child deletion
- Prevent parent deletion
- Prevent child deletion

Create compound documents

Managing and viewing compound documents

- Enterprise Manager

- Displays documents in compound relationship with the  icon and with a check box on the General tab of the Properties page.
- Has QueryBuilder feature to search for component relationship objects.

- Workplace

- Displays documents in compound relationship with the  icon.
- Has wizards for creating and modifying compound documents.

- Custom applications

- Create a custom applications using Content Engine API.

© Copyright IBM Corporation 2011

Figure 16-7. Managing and viewing compound documents

F1432.0

Notes:

Help paths

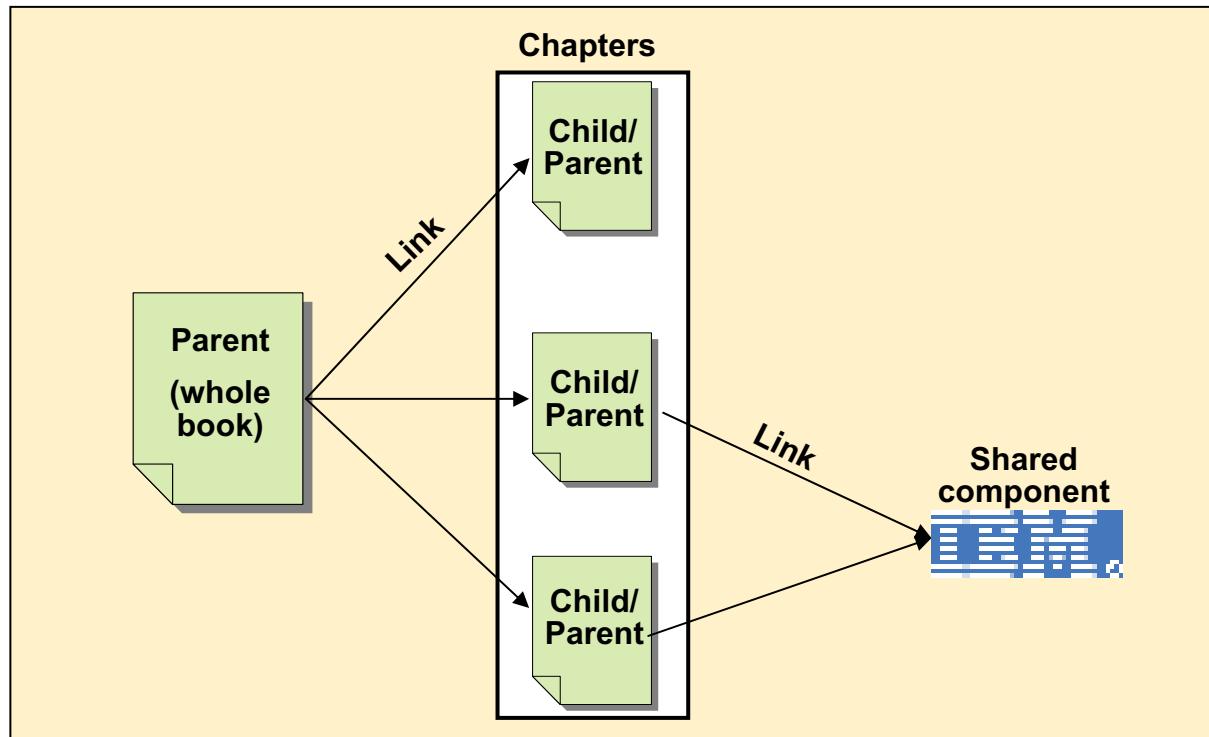
- IBM FileNet P8 Version 5.0 information center > Administering IBM FileNet P8 > Administering Content Engine > Content Engine Overview > Features > Compound documents
- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Compound Documents > Concepts

Content Engine Enterprise Manager and Workplace provide basic capabilities for creating, viewing, and managing compound documents.

A custom application is required for more complex viewing and manipulation, including integration with document editing tools.

Create compound documents

Compound document – chapter book



© Copyright IBM Corporation 2011

Figure 16-8. Compound document - chapter book

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Compound Documents > Concepts > Example use cases

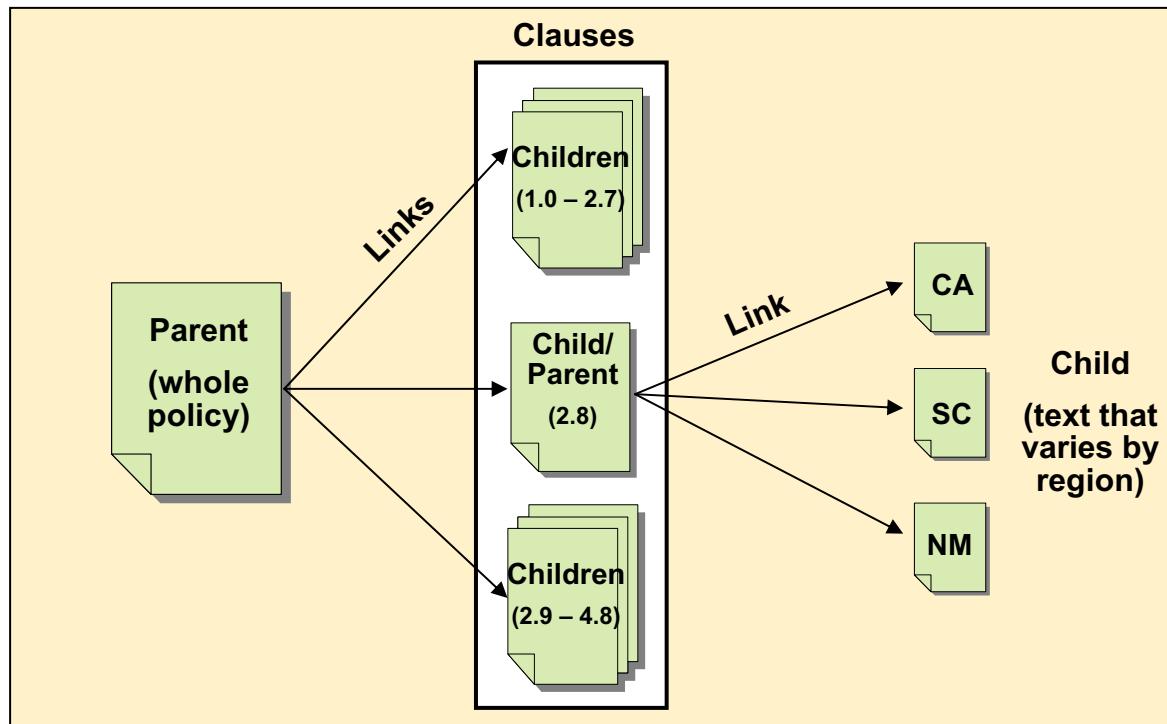
Components of a chapter book

The diagram shows a parent document with no text, which represents the book as a whole. It has links to the following:

- Separate child document for each chapter
- Two child documents, which have links to the following:
 - Shared child document that contains a logo (image)

Create compound documents

Compound document – insurance policy



© Copyright IBM Corporation 2011

Figure 16-9. Compound document - insurance policy

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Compound Documents > Concepts > Example use cases

Components of an insurance policy The diagram shows a parent with no text, which represents the policy as whole. It has links to the following:

- Individual child documents representing clauses (or series of clauses) in policy body
- The center document, which is another child or parent that is dynamically pointed to another child document with text for the region where the policy is being sold. (This text is variable.) This center document is linked to the following:
 - Final set of child documents with remainder of policy language.

Create compound documents

Building a compound document



- Connect existing documents into a structure.
 - Identify parent and children and their relationships.
 - Set compound document status on parent document.
 - Set up component relationships, taking versioning into account.
- Build a structure and add content.
 - Create parent and child document instances without content.
 - Set compound document status and set up relationships
 - Add content to the component documents.

© Copyright IBM Corporation 2011

Figure 16-10. Building a compound document

F1432.0

Notes:

When a compound document is built, the component documents are linked together in a hierarchical relationship consisting of a parent and children. A child document can in turn be a parent in another compound document or in a subhierarchy in the original compound document. Similarly, a parent can be a child document.

You can build documents using the following methods:

- Connecting existing documents.
- Creating new documents and connecting them together.
- Combining both methods.

One aspect of building a compound document is defining what happens when a component document is updated: Does the compound document continue to point to the original version or does it point to a new one?

Create compound documents

ComponentRelationship class



- Defines version relationships:
 - Static or dynamic
 - Binding to major versus minor versions
- Sets update behavior for entire compound document .
- Defines sort order for child documents.
- Sets deletion behaviors: prevention dependencies and cascading

© Copyright IBM Corporation 2011

Figure 16-11. ComponentRelationship class

F1432.0

Notes:

Sort order

You can specify the sort order of child documents in order to ensure that a book is assembled with the chapters in the correct order.

Deletion behavior of the child component document

The ComponentCascadeDeleteAction enumeration has constants defined for the component cascade delete settings.

Valid settings are as follows:

CASCADE_DELETE: Causes the child component document to be automatically deleted when the parent component document is deleted. Document deletion succeeds if you have FN_ACCESS_DELETE rights for all child components involved in the cascade delete.

NO CASCADE_DELETE (default setting): Causes no special action to occur when the parent component document is deleted.

Create compound documents

Document class



- Discover children of a parent, and parents of a child.
- Retrieve all parent relationships for a child.
- Retrieve all child relationships for a parent.
- Set the compound document state.
- Set the ComponentBindingLabel.

© Copyright IBM Corporation 2011

Figure 16-12. Document class

F1432.0

Notes:

Create compound documents

Steps to create a compound document

1. Create or retrieve a parent document object.

```
Factory.Document.createInstance (...) Or  
Factory.Document.fetchInstance (...)
```

2. Set parent document compound document status.

```
document.set_CompoundDocumentState (...)
```

3. Retrieve the child document.

4. Set up component relationship between child and parent.

```
Factory.ComponentRelationship.createInstance (...)  
componentRelationship.set_ChildComponent (...)  
componentRelationship.set_VersionBindType (...)  
componentRelationship.save (...)
```

© Copyright IBM Corporation 2011

Figure 16-13. Steps to create a compound document

F1432.0

Notes:

For creating a document, refer to the steps in the "Create Document objects" lesson in the "Documents" unit.

Create compound documents

document.set_CompoundDocumentState(...)

```
void set_CompoundDocumentState  
    (CompoundDocumentState value)
```

- Parameters
 - value - Specifies the value for the CompoundDocumentState
- Notes
 - The CompoundDocumentState property determines whether a document can be a parent component in a compound document.

© Copyright IBM Corporation 2011

Figure 16-14. document.set_CompoundDocumentState(...)

F1432.0

Notes:

Valid CompoundDocumentState enumeration constants:

COMPOUND_DOCUMENT: Designates the document as the parent component in a compound document. Initially, this parent document has no child components. A document must be designated as a compound document to become a parent component for a ComponentRelationship object.

STANDARD_DOCUMENT(default setting): Designates the document as a standard document. This setting prevents the document from being a parent component within a compound document. A document cannot be designated as a standard document when it is referenced as a parent component by a ComponentRelationship object.

null (default setting - 3.5.x upgraded to 4.x): Designates the document as a standard document. By default, pre-existing documents in an object store upgraded to 4.0.x return a null value instead of STANDARD_DOCUMENT. Note that, although a null value can be returned, you cannot set this property to null.

Create compound documents

Factory.ComponentRelationship.createInstance(...)

```
public static ComponentRelationship  
createInstance (ObjectStore os, String  
classId, ComponentRelationshipType type,  
Document parentComponent, String name)
```

- Parameters

- os – Specifies the ObjectStore object in which this class instance is located.
- classId – Specifies a String identifier for the type of class to be created.
- type - Specifies a ComponentRelationshipType.
- parentComponent - Specifies the parent document.
- name - Specifies a String for the name of the class.

- Returns

- An object reference to a new instance of ComponentRelationship class

© Copyright IBM Corporation 2011

Figure 16-15. Factory.ComponentRelationship.createInstance(...)

F1432.0

Notes:

To persist the created object to the object store, you must explicitly call the save method after setting properties.

- Parameters

classId - The identifier can be specified as the class name constant or the symbolic name for the class. It can also be null, in which case an object of the base class type is returned.

type - Specifies the type of compound document relationship between the child document and the parent document.

Another form of this method is the following:

```
public static ComponentRelationship createInstance (ObjectStore os, String  
classId)
```

If this method is used, the parent component, ComponentRelationshipType, and the name need to be set separately.

Create compound documents

componentRelationship.set_ChildComponent(...)



```
void set_ChildComponent(Document value)
```

- Parameters

- value - Specifies the Document object to be designated as the child in this component relationship.

- Notes

- If the relationship type is DYNAMIC_CR, the version of the child document must be eligible-for-binding.

Example: A major version of the document must exist when the version bind rule specifies that only major versions can be bound.

© Copyright IBM Corporation 2011

Figure 16-16. componentRelationship.set_ChildComponent(...)

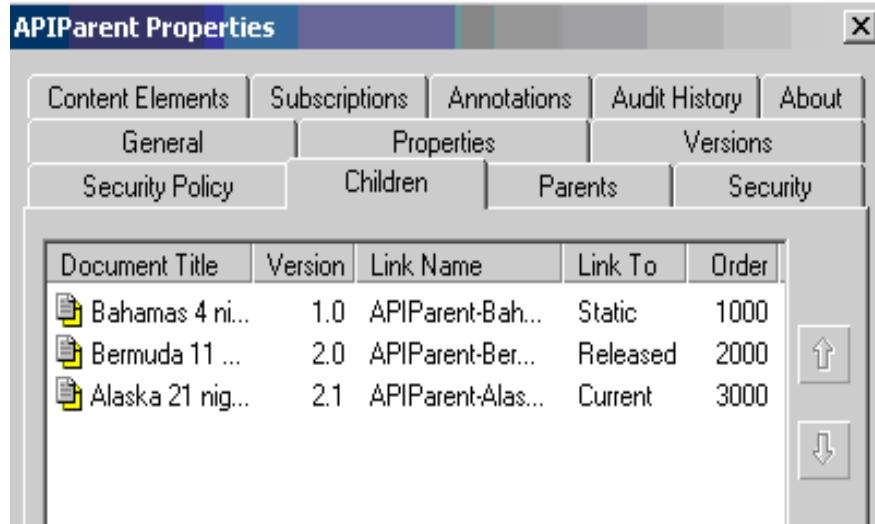
F1432.0

Notes:

Create compound documents

Setting child version relationships

- Static_CR links to the child document version at the time that the relation created and remains static.
- DYNAMIC_CR links to the child document version that is the latest.



© Copyright IBM Corporation 2011

Figure 16-17. Setting child version relationships

F1432.0

Notes:

Help path

- IBM FileNet P8 Version 5.0 information center > Developing IBM FileNet P8 applications > Content Engine Development > Content Engine Java and .NET Developer's Guide > Compound Documents > Concepts > Component Relationship Types

Create compound documents

componentRelationship.set_VersionBindType(...)



```
void set_VersionBindType(VersionBindType  
    value)
```

- Parameters

- value - Specifies the The VersionBindType Object that determines the child component document version.

© Copyright IBM Corporation 2011

Figure 16-18. componentRelationship.set_VersionBindType(...)

F1432.0

Notes:

VersionBindType determines the child component document version bound with the parent component for component relationship types of DYNAMIC_CR.

The VersionBindType class has constants defined for the bind type settings. Valid settings are the following:

LATEST_VERSION: All document versions, major and minor, are considered candidates for binding

LATEST_MAJOR_VERSION: Only the major document versions are considered candidates for binding

Create compound documents

componentRelationship.save(...)



```
void save(RefreshMode refreshMode)
```



- Parameters
 - refreshMode - Specifies the property name.
- Notes
 - This method saves the changes made to this object.
 - You can optionally refresh all of the object properties.
 - This method is inherited from
IndependentlyPersistableObject interface.

© Copyright IBM Corporation 2011

Figure 16-19. componentRelationship.save(...)

F1432.0

Notes:

Create compound documents

Sample code to set compound document state

- 
- Set compound document status to the document to be a parent.

...

```
parentDoc.set_CompoundDocumentState  
    (CompoundDocumentState.COMPOUND_DOCUMENT) ;  
parentDoc.save(RefreshMode.REFRESH) ;
```

The variable **parentDoc** in the code sample refers to the Document object that was created earlier.

Continued on the next slide...

© Copyright IBM Corporation 2011

Figure 16-20. Sample code to set compound document state

F1432.0

Notes:

Create compound documents

Sample code to retrieve child documents



```
Document staticChild =  
    Factory.Document.fetchInstance  
        (os, "/APIFolder/APIChild1", null);  
  
Document dynamicChild =  
    Factory.Document.fetchInstance(os,  
        "/APIFolder/APIChild2", null);
```

Continued on the next slide...

© Copyright IBM Corporation 2011

Figure 16-21. Sample code to retrieve child documents

F1432.0

Notes:

Create compound documents

Sample code to set up static component relationship

```
ComponentRelationship componentRelationship1 =  
    Factory.ComponentRelationship.createInstance  
(os, null, ComponentRelationshipType.DYNAMIC_CR,  
     parentDoc, "ComponentRelation1");  
  
componentRelationship1.set_ChildComponent  
    (staticChild);  
  
componentRelationship1.save  
    (RefreshMode.REFRESH);  
  
System.out.println("staticChild is added");
```

© Copyright IBM Corporation 2011

Figure 16-22. Sample code to set up static component relationship

F1432.0

Notes:

Create compound documents

Sample code to set up dynamic component relationship

- All the steps are similar to the static component relationship, as shown in the previous slide, except for the following:
 - An additional step for the version binding type has been added.
 - The value for the component relationship type is different.

```
...
ComponentRelationship componentRelationship2 =
Factory.ComponentRelationship.createInstance
(os,null,ComponentRelationshipType.DYNAMIC_CR,par
entDoc,"ComponentRelation2");
componentRelationship2.set_VersionBindType
(VersionBindType.LATEST_MAJOR_VERSION);
...
```

© Copyright IBM Corporation 2011

Figure 16-23. Sample code to set up dynamic component relationship

F1432.0

Notes:

Create compound documents

Demonstrations



- Create compound documents:
 - Set compound document status to the document.
 - Retrieve documents and set component relationship with the parent document.

© Copyright IBM Corporation 2011

Figure 16-24. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *createCompoundDocumentEDU(...)* method in the *CompoundDocumentsEDU.java* file.
3. Run the code and verify the results as instructed in the *Compound Documents* unit in the Student Exercises book.

Create compound documents

Activities

In your Student Exercises book

- Unit: Compound Documents
- Lesson: Create compound documents
- Activities
 - Create compound documents.

© Copyright IBM Corporation 2011

Figure 16-25. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

Lesson 16.2. Retrieve compound documents

Lesson: Retrieve compound documents

- 
- Why is this lesson important to you?
 - Your company employees access compound documents to review their content. As their programmer, you are going to write code to retrieve a compound document and its child documents when an employee accesses it.

© Copyright IBM Corporation 2011

Figure 16-26. Lesson: Retrieve compound documents

F1432.0

Notes:

Retrieve compound documents

Activities that you need to complete



- Retrieve compound documents.

© Copyright IBM Corporation 2011

Figure 16-27. Activities that you need to complete

F1432.0

Notes:

Retrieve compound documents

Steps to retrieve component relationships

1. Verify that the parent document has compound document status.

```
document.get_CompoundDocumentState()  
CompoundDocumentState.getValue()
```

2. Retrieve the name and type of the component relationship and version binding type.

```
document.get_ChildRelationships()  
componentRelationship.getName()  
componentRelationship.getVersionBindType()
```

© Copyright IBM Corporation 2011

Figure 16-28. Steps to retrieve component relationships

F1432.0

Notes:

Retrieve compound documents

Steps to retrieve the child documents

1. Retrieve the child documents collection.

```
document.get_ChildDocuments()
```

2. Retrieve the individual child document and its information.

```
document.get_Name()  
document.get_MajorVersionNumber()  
document.get_MinorVersionNumber()
```

© Copyright IBM Corporation 2011

Figure 16-29. Steps to retrieve the child documents

F1432.0

Notes:

Retrieve compound documents

document.get_CompoundDocumentState()



```
CompoundDocumentState get_CompoundDocumentState()
```

- Returns
 - The CompoundDocumentState
- Notes
 - The CompoundDocumentState property determines whether a document can be a parent component in a compound document.

© Copyright IBM Corporation 2011

Figure 16-30. document.get_CompoundDocumentState()

F1432.0

Notes:

Retrieve compound documents

compoundDocumentState.getValue()

```
public int getValue()
```

- Returns
 - The internal integer value associated with a specific instance of this class

© Copyright IBM Corporation 2011

Figure 16-31. compoundDocumentState.getValue()

F1432.0

Notes:

Retrieve compound documents

document.get_ChildRelationships()



ComponentRelationshipSet

get_ChildRelationships ()

- Returns
 - A ComponentRelationshipSet collection object
- Notes
 - The collection contains the ComponentRelationship objects referencing this document as the parent component document.

© Copyright IBM Corporation 2011

Figure 16-32. document.get_ChildRelationships()

F1432.0

Notes:

Retrieve compound documents

componentRelationship.get_Name()



```
String get_Name()
```

- Returns
 - The Name of the object
- Notes
 - For most classes, this property is read-only, but for this object, it is read/write.

© Copyright IBM Corporation 2011

Figure 16-33. componentRelationship.get_Name()

F1432.0

Notes:

Retrieve compound documents

componentRelationship.get_VersionBindType()



```
VersionBindType get_VersionBindType()
```

- Returns
 - The VersionBindType object
- Notes
 - The VersionBindType Object determines the child component document version bound with the parent component.

© Copyright IBM Corporation 2011

Figure 16-34. componentRelationship.get_VersionBindType()

F1432.0

Notes:

Retrieve compound documents

document.get_ChildDocuments()

DocumentSet get_ChildDocuments()

- Returns
 - DocumentSet collection object
- Notes
 - The collection contains the child Document objects
 - That are bound to this parent document
 - For which the user has read access
 - With the version that is bound to this parent document
 - The same child document can be in the collection more than one time.

© Copyright IBM Corporation 2011

Figure 16-35. document.get_ChildDocuments()

F1432.0

Notes:

Retrieve compound documents

Sample code to retrieve the parent document

- Create a Property Filter object with the required properties.

```
propFilter.addIncludeProperty(0, null, null,  
    PropertyNames.NAME,1);  
propFilter.addIncludeProperty(0, null, null,  
    PropertyNames.COMPOUND_DOCUMENT_STATE,1);  
propFilter.addIncludeProperty(0, null, null,  
    PropertyNames.COMPONENT_RELATIONSHIP_TYPE,1);  
propFilter.addIncludeProperty(0, null, null,  
    PropertyNames.CHILD_RELATIONSHIPS,1);  
propFilter.addIncludeProperty(0, null, null,  
    PropertyNames.CHILD_DOCUMENTS,1);
```

- Retrieve the parent document.

```
Document parentDoc =  
    Factory.Document.fetchInstance(objectstore,  
        "/APIFolder/APIParent",propFilter);
```

© Copyright IBM Corporation 2011

Figure 16-36. Sample code to retrieve the parent document

F1432.0

Notes:

Retrieve compound documents

Sample code to retrieve component relationships (1)

```
CompoundDocumentState compDocState =
    doc.get_CompoundDocumentState();
int compDocStatevalue = compDocState.getValue();
if(compDocStatevalue == CompoundDocumentState.
    COMPOUND_DOCUMENT_AS_INT) {
    ComponentRelationshipSet comRelationshipSet =
        doc.get_ChildRelationships();
    Iterator it = comRelationshipSet.iterator();
    ComponentRelationship comRelation;
    ...
}
```

© Copyright IBM Corporation 2011

Figure 16-37. Sample code to retrieve component relationships (1)

F1432.0

Notes:

Retrieve compound documents

Sample code to retrieve component relationships (2)

```
while (it.hasNext())
{
    comRelation=(ComponentRelationship)it.next();
    String comName = comRelation.get_Name();
    VersionBindType vbType =
        comRelation.get_VersionBindType();
    System.out.println("Name = " + comName);
    System.out.println("Version bind type = " +
        vbType);
}
```

© Copyright IBM Corporation 2011

Figure 16-38. Sample code to retrieve component relationships (2)

F1432.0

Notes:

Retrieve compound documents

Sample code to retrieve child documents

```
DocumentSet childDocs =  
    parentDoc.get_ChildDocuments();  
Iterator childIt = childDocs.iterator();  
Document childDoc;  
While (childIt.hasNext()) {  
    childDoc = (Document) childIt.next();  
    System.out.println("Child Document= " +  
        childDoc.get_Name());  
    System.out.println("Major version= " +  
        childDoc.get_MajorVersionNumber());  
    System.out.println("Minor version= " +  
        childDoc.get_MinorVersionNumber());
```

© Copyright IBM Corporation 2011

Figure 16-39. Sample code to retrieve child documents

F1432.0

Notes:

Retrieve compound documents

Demonstrations



- Retrieve compound documents
 - Retrieve a compound document.
 - Retrieve document component relationships.
 - Retrieve the child documents.

© Copyright IBM Corporation 2011

Figure 16-40. Demonstrations

F1432.0

Notes:

DEMONSTRATION —

1. Start Eclipse and open the *CMJavaAPISolution* project.
2. Review the code for the *getCompoundDocumentEDU(...)* method in the *CompoundDocumentsEDU.java* file.
3. Run the code and verify the results as instructed in the *Compound Documents* unit in the Student Exercises book.

Retrieve compound documents

Activities



In your Student Exercises book

- Unit: Compound Documents
- Lesson: Retrieve compound documents
- Activities
 - Retrieve compound documents.

© Copyright IBM Corporation 2011

Figure 16-41. Activities

F1432.0

Notes:

Use your Student Exercises book to perform the activities listed.

IBM
®