Kevin Bates
Software Engineer
March 4, 2013

Product Implementation Training (PIT)

IBM FileNet Content Manager 5.2.0
Custom Root Classes

Introduction

- Course Overview:
  - Custom Root Classes - what they are and how they work
- Target Audience:
  - Application Designers, Support Personnel, P8 Administrators
- Prerequisites:
  - P8 Administration, Metadata Authoring and Class Hierarchy concepts
- Version Release Date: March 15, 2013

Course Objectives

After this course you will be able to:

- Describe what a custom root class is and when it is used
- Perform the necessary operations to create a new custom root class hierarchy
- Explain which kind of custom root class is beneficial for a given circumstance

Course Roadmap

- ➔ Custom Root Fundamentals
  - Why Custom Root Classes?
  - Class Hierarchy
  - Flavors of Custom Roots
  - API Overview
- – Demonstration
- – Course Summary

Custom Root Fundamentals: Why Custom Root Classes?

- Use Case: *I, as an application author, need a means of managing instances of disjoint classes where the retrieval requirements for each of the subclasses vary.*
  - *The current mechanism of creating everything as a subclass of Custom Object (stored in the Generic table) does not fulfill this requirement.*

- Resolution: Custom Root Classes

- Common Use Cases: Customer, Account, Policy type objects

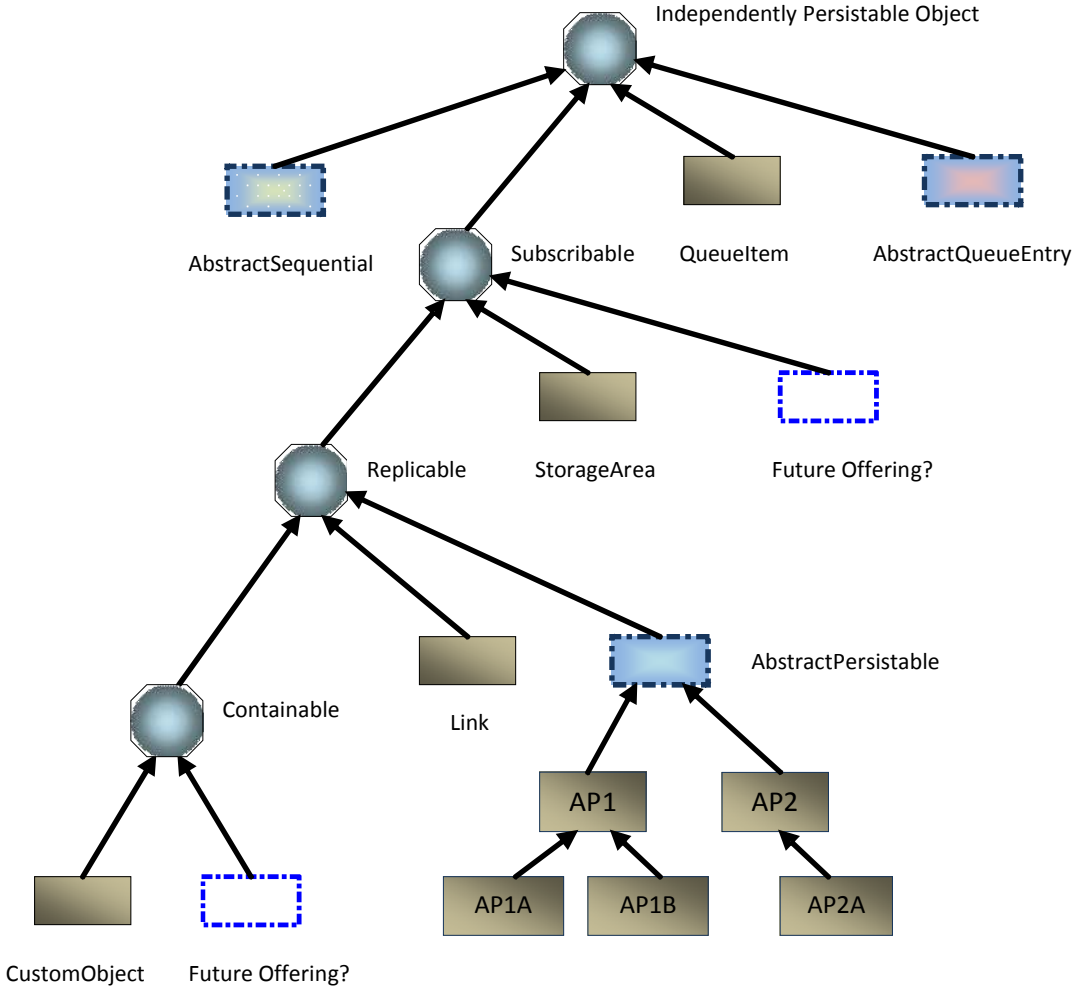
Custom Root Fundamentals: Why Custom Root Classes?

- Benefits:
  - Creation of the root class establishes that all instances of a class and its subclasses be stored in a table used solely for that class hierarchy's persistence
  - Because this set of classes are more closely related, indices can be applied as needed and as makes sense for these classes - leading to much faster retrieval rates
  - Data storage can be better managed. Much less chance of encountering row-length limits (i.e., table overflow on DB2), fewer null columns
  - CE auto-creates and auto-deletes tables (and other related objects) as needed

Custom Root Fundamentals: Why Custom Root Classes?

- Limitations:
  - Change class between different custom roots (i.e., spanning different tables) is not permitted
  - Searches cannot be performed across custom root hierarchies
  - Custom root instances cannot be contained in Folders – use CustomObject (containment requires custom abstract required class OVPs)
  - No support for content bearing instances – use Document (or Annotation)
  - Properties of custom root instances are not full-text indexable

Custom Root Fundamentals: Class Hierarchy



Custom Root Fundamentals: Flavors of Custom Roots

- Abstract Persistable
  - Provides replication and subscribable functionality (event targets, change preprocessing and auditing)
    - Social Collaboration: Tags, Comments, Recommendations, Download Counts
    - IBM Case Management: Case Business Objects (Mediterranean Release)

- Abstract Queue Entry
  - Provides compatibility with Queue Sweep tasks. Table structure will conform to sweep framework requirements and classes can be set as sweep targets
    - Thumbnails: Thumbnail Request (Thumbnail Request Sweep target)
    - Social Collaboration: Activity Streams (Custom Queue Sweep target)

      Note: Custom Queue Sweep authoring is not publicly available in 5.2.0

- Abstract Sequential
  - Provides ordered rows by implementing a column sequence (identity on SQLServer) for every insert. Allows for applications that require their own form of queue processing
    - Social Collaboration: Seedlists

Custom Root Fundamentals: API Overview

Creating a new custom root class is identical to creating any kind of subclass.

```
ReplicableClassDefinition apCD =
        Factory.ReplicableClassDefinition.fetchInstance(
                objectStore, Constants.Class_CmAbstractPersistable, null);
ReplicableDefinition myCustomRoot = apCD.createSubclass(myClassId);
        … set class properties, displayNames, symbolicName, etc. …
        myCustomRoot.save(RefreshMode.REFRESH);
```

Custom Root Fundamentals: API Overview

Creating or fetching an instance of a custom root class requires the class name or ID (otherwise we can't determine the table in/from which to store/fetch the instance data).

```
CmAbstractPersistable myCustomRootInstance =
        Factory.CmAbstractPersistable.createInstance(
                objectStore, myCustomRoot.get_SymbolicName());
        … set any required properties …
        myCustomRootInstance.save(RefreshMode.REFRESH);

CmAbstractPersistable myCustomRootInstance =
        Factory.CmAbstractPersistable.fetchInstance(
                objectStore, myCustomRoot.get_SymbolicName(), id,
   propertyFilter);
```

Custom Root Fundamentals: API Overview

- Similarly so for queries. The FROM clause must reference a concrete class. Using CmAbstractPersistable (et al.) is not permitted in the FROM clause.
  - Allowed

```
SearchSQL sql = new SearchSQL("SELECT [This] FROM [myCustomRoot] WHERE...");
```

  - Disallowed

```
SearchSQL sql = new SearchSQL("SELECT [This] FROM [CmAbstractPersistable] WHERE...");
```

- Be careful about `instanceof`, use `ClassDescription.describedIsOfClass`...
  - Since all concrete custom roots derive from CmAbstractPersistable (et al.), using `instanceof` to distinguish between two types of objects is not sufficient...

```
if ( myCustomerInstance instanceof CmAbstractPersistable)
```
… returns true

```
if ( myAccountInstance instanceof CmAbstractPersistable)
```
… returns true

  - use `ClassDescription.describedIsOfClass()`

```
ClassDescription cd = myCustomerInstance.get_ClassDescription();
if ( cd.describedIsOfClass("Customer") )
```
… returns true

```
if ( cd.describedIsOfClass("Account") )
```
… returns false

Course Roadmap

- Custom Root Fundamentals
  - Why Custom Root Classes?
  - Class Hierarchy
  - Flavors of Custom Roots
  - API Overview
- ➔ Demonstration
- Course Summary

Demonstration

- Create custom root subclass of Abstract Sequential
  - Show table
  - Show sequence
- Create instances of custom root
  - Show sequence values
- Delete instances and custom root class
  - Show table and sequence objects have been dropped

Course Roadmap

- Custom Root Fundamentals
  - Why Custom Root Classes?
  - Class Hierarchy
  - Flavors of Custom Roots
  - API Overview
- Demonstration
- ➔ Course Summary

Course Summary

You have completed this course and can:

- Describe what a custom root class is and when it is used
  - *Class hierarchies are tied to custom table (UT_xxx) to contain similar pieces of information.*
  - *New roots should be created for disjoint hierarchies*
- Perform the necessary operations to create a new custom root class hierarchy
  - *Very similar to today's methods. Instance creation, gets and fetches, FROM clauses require the custom root class name or a subclass within that root*
- Explain which kind of custom root class is beneficial for a given circumstance
  - *AbstractPersistable for general collections of properties, subscribable, replicable*
  - *AbstractSequential for order-imposed requirements*
  - *AbstractQueueEntry for integration with CE queue sweep framework (future)*

Product Help/Documentation/Resources

- Content Engine Java and .NET Developer's Guide
  - Custom Root Classes Concepts

    http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0/topic/com.ibm.p8.ce.dev.ce.doc/customclass_concepts.htm

  - Working With Custom Root Classes

    http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0/topic/com.ibm.p8.ce.dev.ce.doc/customclass_procedures.htm

Note: Links will work at eGA (3/15/2013); before then, replace http://pic.dhe.ibm.com/infocenter/p8docs/v5r2m0 with http://cmfogbert.usca.ibm.com:7777/p8ic520 to use an internal InfoCenter