Kevin Bates
Persistence Team
October 20, 2014

Product Implementation Training (PIT)

FileNet Content Manager 5.2.1 GA

Background Search

Introduction

- Course Overview
  5.2.1 Background Search
- Target Audience
  Support Teams and Lab Services
- Suggested Prerequisites
  CPE Search and Sweep frameworks, metadata concepts
- Version Release Date
  November 2014

Course Objectives

After this course you will be able to:

- Describe what a Background Search is
- Perform the necessary steps to create and run a Background Search
- List the results of Background Search

Course Roadmap

- ➔ Background Search Purposes
- Functional Overview
- Extensions
- Demonstration
- Best Practices
- Course Summary

Background Search Purposes

Two primary purposes for implementing background search...

- Long-standing issues with time-consuming queries. Background Search enables queries to be started and run until completion – allowing users to get other work done – and view the results after completion
- Provides the basis for a reporting framework that will then post-process the background search results

Course Roadmap

- Background Search Purposes
- ➔ Functional Overview
- Extensions
- Demonstration
- Best Practices
- Course Summary

Functional Overview

- Background search relies heavily on metadata.
  - Two system classes have been added, CmBackgroundSearch & CmAbstractSearchResult

- CmBackgroundSearch
  - Subclass of CmSweep (i.e., an instance of CmBackgroundSearch is a sweep request)
  - A SearchExpression property identifies the search (sweep) and a sweep executor/handler produces search results of a concrete class of CmAbstractSearchResult
  - CmBackgroundSearch is non-instantiable. I.e., instances can only be created from subclasses of CmBackgroundSearch
  - Queries performed by sweep will impersonate the Background Search owner
  - **Deletion of the CmBackgroundSearch instance will cascade-delete ALL results for that search! Background Search instances must be retained as long as search results are required.**

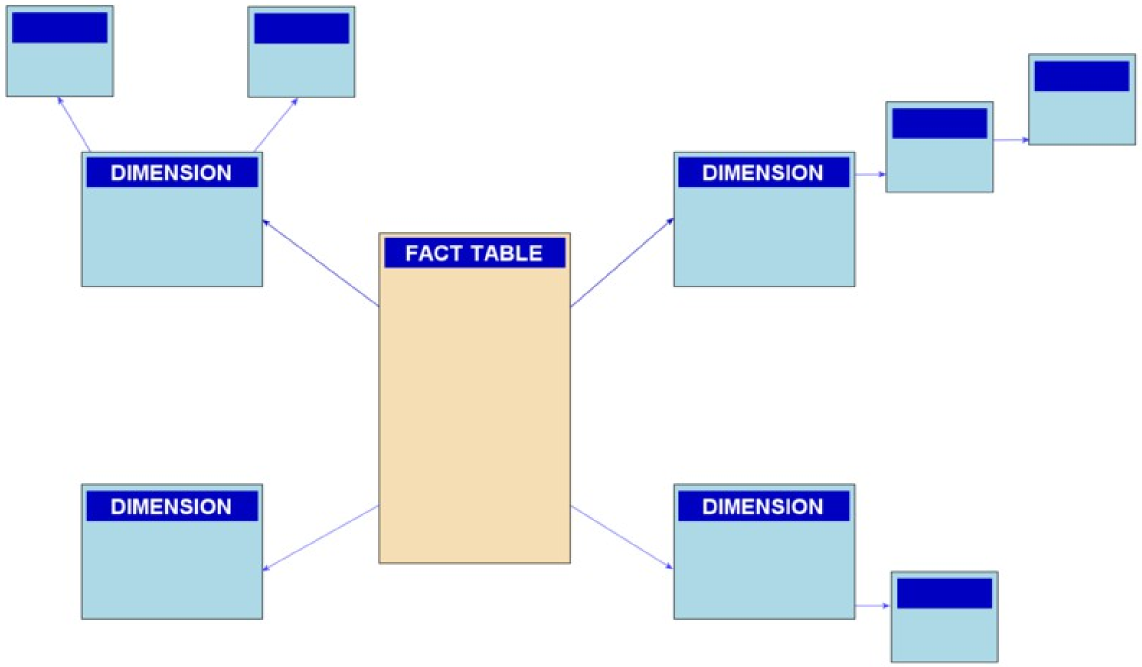
- Properties
  - SearchExpression is a class-constant. i.e., it is set on the subclass definition and is read-only on the instance.
    - Supports parameterized queries. Syntax: @<symbolicName>@ (e.g., WHERE Store IN @Stores@). This enables multiple variations of a search to take place using the same query and search result definitions.
  - SearchResults is an enumeration containing the results of the search (sweep). Because the required class must point to a concrete subclass of CmAbstractSearchResult, the best practice is to create the abstract search result root class first.

Functional Overview

- CmBackgroundSearch Properties (continued)...
  - EffectiveSQL is a variation to the SearchExpression indicating the query used by the sweep with parameter values. Is not necessarily the actual query string submitted for the search.
  - SearchObjectStore is an OVP to an ObjectStore against which the Background Search query is performed. This enables users to dedicate a separate ObjectStore for search results/reporting.
  - MaximumExaminedRowCount is an Integer property that can be used to specify how many rows to return from the underlying query. This property enables the ability to test what a background search will return before producing millions of query results.
  - AllowStringTruncation is a class-constant boolean (default = False) which will cause the background search sweep executor to truncate selected string values if their lengths exceed the maximum string length of the corresponding property on the search result class.
  - OrderByGroupedProperties is a class-constant boolean (default = False) which can be used to make the processing of search results involving aggregation more efficient.
  - Subclass definitions will hold custom properties pertaining to parameters specified in the SearchExpression (e.g., Stores)
  -
  - Various sweep-related properties: SweepTimeslots, SweepStartDate, SweepEndDate, ExaminedObjectCount, etc.

CmAbstractSearchResult

- Custom root class with single OVP (BackgroundSearch) pointing back to referencing CmBackgroundSearch subclass
- Contains property definitions for each item selected in the query that warrants representation in the search results
- Results (instances) are cascade-deleted when CmBackgroundSearch instance is deleted.
- This table will correspond to the FACT TABLE used by reporting framework



Background search authoring and initiation steps...

1) Identify search targets, selection list, and where clause (with parameters)
2) Create concrete subclass of CmAbstractSearchResult,
   a) Add properties for each of the selected items identified in step 1
3) Create subclass of CmBackgroundSearch,
   a) Refine the SearchResults required class to that of the concrete subclass created in step 2,
   b) Add properties for each of the parameters identified in step 1,
   c) Set the default value for the SearchExpression property corresponding to the query identified in step 1
4) Create an instance of CmBackgroundSearch subclass
   a) Assign any parameter-based properties appropriate values
   b) Upon successful creation, progress can be monitored by checking EffectiveEndDate, ProcessedCount, FailureCount and various sweep-related properties
5) Oh, and don't worry, ACCE makes this all very simple!

CmSweep

CmBackgroundSearch

CmBackgroundSearch Subclass

Parameter-based Properties:
Genres (List of String), **Store** (String), **Price** (Double)

CmAbstractSearchResult

CmAbstractSearchResult Subclass

Selected Properties:
BookTitle, ISBN, Author, Pages, Genre, Region, Store, Price

SearchExpression:
SELECT DocumentTitle AS ***BookTitle***, ***ISBN, Author, Pages, Genre, Region, Store, Price***
FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book
WHERE Genre IN **@Genres@** AND Store = **@Store@** AND Price >= **@Price@**

CmBackgroundSearch Instance (Sweep Table)

Genres = 'Humor',...,'Drama',
Store = 'W-04782',
Price = 12.99

Search Results
(Produced by sweep framework)

CmAbstractSearchResult Instance(s) (Custom Root Table)

'The Girl who Kicked..', 'ISBN 03-4356', 'Larrsson', 793, 'Mystery', 'West', 'W-04782', 17.39

EffectiveSQL:
SELECT DocumentTitle AS BookTitle, ISBN, Author, Pages, Genre, Region, Store, Price
FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book
WHERE Genre IN **('Humor',...,'Drama')** AND Store = **'W-04782'** AND Price >= **12.99**

Course Roadmap

- Background Search Purposes
- Functional Overview
- ➔ Extensions
- Demonstration
- Best Practices
- Course Summary

Extensions

- Because background search doesn't require immediate results, we have the liberty of adding some query behaviors we previously could not support – namely aggregation and grouping clauses.
- The following aggregate functions will be supported (in Background Search only) and will typically be used in conjunction with GROUP BY (unless a single row of accumulated results is desired)...
- COUNT(property)
  - If the property value of the selected row (RepositoryRow) is not null a counter will be incremented
  - COUNT(property) AS myPropertyCounter – increments the value of myPropertyCounter on abstract search result instance
  - If the 'AS' clause is absent, the ExpressionN property value will be incremented where 'N' is the position (1-relative) of the selected item in the selection list.
- SUM(property)
  - Adds the property's value to a summation counter.
  - Same rules for AS or ExpressionN apply
- MIN(property), MAX(property)
  - Tracks the MIN (MAX) value of the selected property in another property.
  - Same rules for AS or ExpressionN apply

Extensions (Cont)

GROUP BY property1, property2, ...

- Aggregates the results based on grouping
- Sweep executor locates existing search results based on product of groupings (e.g., SELECT … From AbstractSearchResultClass WHERE property1 = 'foo' AND property2 = 'bar' AND BackgroundSearch = bsInstance)
-

Aggregation of data (count, sum, min, max) is performed within the background search sweep executor – not the CPE query layer. As a result, it modifies the SQL string by replacing aggregation-related items with known constructs.

SearchExpression:

SELECT Author, SUM(Price), COUNT(Store) AS CopiesSold, MIN(Adjustment) AS MaxDiscount, MAX(Adjustment) AS MaxMarkup FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book WHERE Genre IN **@Genres@** AND Store = **@Store@** GROUP BY Author

EffectiveSQL:

SELECT Author, SUM(Price), COUNT(Store) AS CopiesSold, MIN(Adjustment) AS MaxDiscount, MAX(Adjustment) AS MaxMarkup FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book WHERE Genre IN **('Drama',...'Mystery')** AND Store = '**W-07892**' GROUP BY Author

Actual SQL *(when OrderByGroupedProperties = true):*

SELECT Author, **Price**, **Store**, **Adjustment**, **Adjustment** FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book WHERE Genre IN ('Drama',...'Mystery') AND Store = 'W-07892' **ORDER BY** Author

Extensions (Cont)

Custom Search Functions

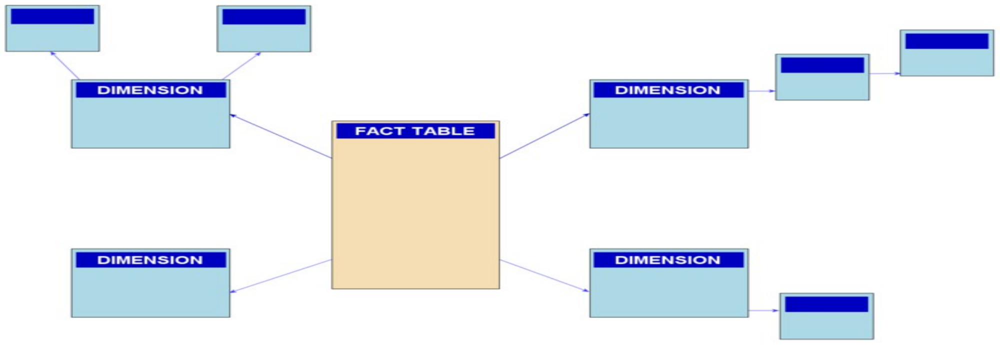
- Will be generally available to all queries
- Enables the use and generation of dimension tables for use in reporting framework

SELECT BSB::GetAuthorBio(Author) AS Bio, BSB::GetStoreInfo(Store) AS StoreInfo, …

AdHoc query framework knows (based on syntax and registry) that BSB::GetAuthorBio() and BSB::GetStoreInfo() are custom functions, what they require as input and what they produce as output

The results will typically be an object-valued property (essentially pointing to the dimension table where more information can be obtained), but could also be simple base types

For example, BSB::GetStoreInfo(Store) reads a row from the table referenced by the required class of the StoreInfo OVP. That row can contain address, web site, phone number and possibly pointers to other dimension tables. If the row is not there (based on the input parameters) GetStoreInfo adds the appropriate row and returns what it just committed.

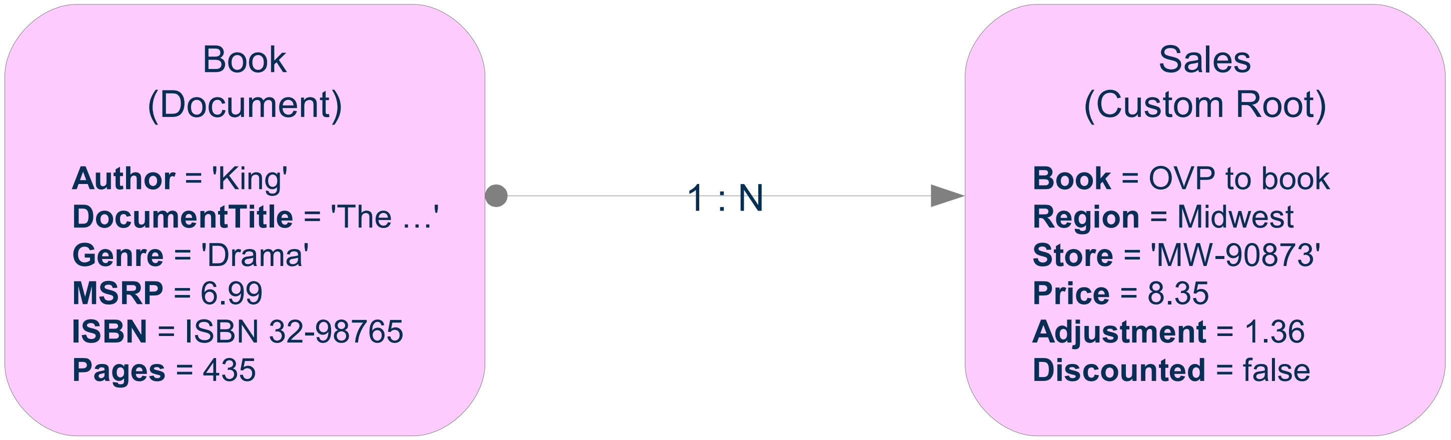


Course Roadmap

- Background Search Purposes
- Functional Overview
- Extensions
- ➔ Demonstration
- Best Practices
- Course Summary

Demonstration – BS Books

- BS Books, a fledgling book seller, currently sells 100 books across 10 authors, 6 genres and 6 price-points (excluding markups and discounts)
- They manage their sales across 5 regions of the US where each region has multiple stores (some regions are expanding faster than others)
- BS Books wants to optimize profits and doesn't know where to begin – so they purchase CPE 5.2.1 and perform some basic background searches



Demonstration (Cont) – BS Books – Search Expressions

Genre Sales in a set of Stores...

```
SELECT Genre, SUM(Price) AS TotalSales, COUNT(Store) AS CopiesSold, MIN(Adjustment) AS MaxDiscount, MAX(Adjustment),
    SUM(Adjustment) AS TotalAdjustments
            FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book
            WHERE Store IN @Stores@
            GROUP BY Genre
```

Regional Genre Sales...

```
SELECT Genre, SUM(Price) AS TotalSales, COUNT(Store) AS CopiesSold, Region, SUM(Adjustment) AS TotalAdjustments
            FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book
            GROUP BY Region, Genre
```

Total Sales - Nationwide...

```
SELECT COUNT(Store) AS CopiesSold, SUM(Price) AS TotalSales, SUM(Adjustment) AS TotalAdjustments
            FROM sbsSales
```

National Book Sales “The Best Sellers List”...

```
SELECT DocumentTitle AS BookTitle, SUM(Price) AS TotalSales, COUNT(Store) AS CopiesSold, Author, ISBN, MSRP, Pages, Genre
            FROM sbsBook b INNER JOIN sbsSales s ON b.This = s.Book
            GROUP BY ISBN
```

Course Roadmap

- Background Search Purposes
- Functional Overview
- Extensions
- Demonstration

➔ Best Practices

- Course Summary

Best Practices

- Read David Skinner's techdoc!
  - Performance tuning Content Platform Engine background searches.
- Don't use asterisks in COUNT (e.g., COUNT(*)). Use Id instead (e.g., COUNT(Id))
- When using functions (aggregate or custom) use an alias instead of relying on ExpressionN. This makes the results processing more readable and doesn't lead to multiple ExpressionN PropertyTemplates for various data types.
  - Example: SELECT SUM(Price) AS TotalSales
- When building background searches across large data sets, test results first by setting MaximumExaminedRowCount to non-zero value.
  - Once the results are confirmed as working, delete that instance of the CmBackgroundSearch subclass (which cascade deletes the test results)
  - Create a new instance of that same subclass with MaximumExaminedRowCount = 0 to perform the full search
- Name Background Search instances appropriately so that results are easily identified. The name will usually be a function of the class name and parameters in use.

Course Roadmap

- Background Search Purposes
- Functional Overview
- Extensions
- Demonstration
- Best Practices
- ➔ Course Summary

Course Summary

You have completed this course and can:

- Describe what a Background Search is
- Perform the necessary steps to create and run a Background Search
- List the results of Background Search

Product Help/Documentation/Resources

Background Search Topics

Administration - Finding objects with background search queries (parent topic):

http://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1/com.ibm.p8.ce.admin.tasks.doc/p8pcc261.htm

Background search query syntax:

http://www.ibm.com/support/knowledgecenter/api/content/SSNW2F_5.2.1/com.ibm.p8.ce.dev.ce.doc/query_sql_syntax_rel_queries.htm#background_search_query_syntax

Developer's guide – concepts:

http://www.ibm.com/support/knowledgecenter/api/content/SSNW2F_5.2.1/com.ibm.p8.ce.dev.ce.doc/query_concepts.htm#background_searches

Custom Function Topics

Administration - How to create custom search functions via ACCE:

http://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1/com.ibm.p8.ce.admin.tasks.doc/p8pcc263.htm

Custom search function query syntax:

http://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1/com.ibm.p8.ce.dev.ce.doc/query_sql_syntax_rel_queries.htm%23query_sql_syntax_rel_queries__custom_function_query_syntax

Developer's guide – concepts:

http://www.ibm.com/support/knowledgecenter/SSNW2F_5.2.1/com.ibm.p8.ce.dev.ce.doc/query_concepts.htm%23query_concepts__custom_functions

Performance tuning Content Platform Engine background searches

http://www.ibm.com/support/docview.wss?uid=swg27043132

Contacts

- Subject Matter Experts (SME)/Area of Expertise:
  Kevin Bates (framework processing & persistence)
  kevin.bates@us.ibm.com
  David Skinner (query processing & tuning)
  skinneda@us.ibm.com
- Quality Assurance:
  Robert Mariano
  rmariano@us.ibm.com