upstate.edu

Lesson 8: LocatorTool | Formats |SUNY Upstate Medical University

24-30 minutes

The `LocatorTool` Object

- Within the Velocity context, there is a `com.hannonhill.cascade.velocity.LocatorTool` object `$_`
- The object provides methods to locate assets of various types
- An asset located is a `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter` object
- Descendant classes of this superclass provides methods to work with these Cascade API adapter objects

`org.jdom.Element` vs. `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter`

- There is a general misunderstanding regarding the nature of `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter` objects
- Cascade users tend to mix up `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter` objects with `org.jdom.Element` objects
- An `org.jdom.Element` object is a representation of an XML element or node within an XML tree
- Methods defined in `org.jdom.Element` are normally related to XML traversal (`getDocument`, `getParent`, `getChild`, `getChildren`, `getDescendants`, etc.), retrieval of node data (`getText`, `getValue`, `getAttributes`, etc.), and node manipulation (`addContent`, `removeChildren`, etc.)
- `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter` objects have nothing to do with XML
- They are objects representing Cascade assets like pages and files
- The confusion between `org.jdom.Element` objects on the one hand and `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter` objects on the other is particularly profound in the following areas:
- When dealing with `com.hannonhill.cascade.api.adapters.StructuredDataNodeAPIAdapter` objects: this class provides methods like `getChild` and `getChildren` to get the child nodes of a group node (when the `isGroup` method of such an object returns true); these methods have nothing to do with `org.jdom.Element.getChild` and `org.jdom.Element.getChildren`
- The `com.hannonhill.cascade.api.adapters.StructuredDataNodeAPIAdapter` also provides a `getTextValueAsXMLElement` method that returns an `org.jdom.Element` object; in this case, we can be mixing up a Cascade API adapter object with the data it contains
- A similar situation can also be found in `com.hannonhill.cascade.api.adapters.XMLBlockAPIAdapter`: an XML block represented by this class is a Cascade API adapter object; but when the `getXMLAsXMLElement` method is called

through this object, we get an `org.jdom.Element` object as data returned by the Cascade API object

- The problem is further complicated by Hannon Hill's reluctance of releasing the full Cascade API documentation; information is always given on a need-to-know basis

`locate` Methods of `com.hannonhill.cascade.velocity.LocatorTool`

- `public com.hannonhill.cascade.api.asset.home.FolderContainedAsset locate( java.lang.String, com.hannonhill.cascade.model.dom.identifier.EntityType, java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.FolderContainedAsset locate( java.lang.String, com.hannonhill.cascade.model.dom.identifier.EntityType )`
- `public com.hannonhill.cascade.api.asset.home.Page locatePage( java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Page locatePage( java.lang.String, java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Folder locateFolder( java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Folder locateFolder( java.lang.String, java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Block locateBlock( java.lang.String, java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Block locateBlock( java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.File locateFile( java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.File locateFile( java.lang.String, java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Symlink locateSymlink( java.lang.String, java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Symlink locateSymlink( java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Reference locateReference( java.lang.String )`
- `public com.hannonhill.cascade.api.asset.home.Reference locateReference( java.lang.String, java.lang.String )`

Locating Assets

- Note that all `locate` come as pairs
- For example, `locate( java.lang.String, com.hannonhill.cascade.model.dom.identifier.EntityType, java.lang.String )` and `locate( java.lang.String, com.hannonhill.cascade.model.dom.identifier.EntityType`

)

- The last `String` parameter, when existent, should be the name of the site where the asset can be found
- When both the Velocity format containing calls to `locate` (or `locateX`) and the asset to be located exist in the same site, the same name is not required
- But if the format and the asset exist in different sites, then the site name must be supplied
- This also means that it is generally a good idea to include site names when writing reusable code
- Examples:

```
#set( $page = $_.locate( "index",
$CASCADE_API_PAGE_TYPE, "cascade-admin" ) )
$page.Class.Name ##=>
com.hannonhill.cascade.api.adapters.PageAPIAdapter

#set( $page = $_.locatePage( "index", "cascade-
admin" ) )
$page.Class.Name ##=>
com.hannonhill.cascade.api.adapters.PageAPIAdapter
```

An Overview of the Cascade API

- We will look at library code related to the Cascade API later
- Let us look at some of the classes briefly
- The classes we want to visit are:
- `com.hannonhill.cascade.api.adapters.PermissionsCapableAssetAPIAdapter`
- `com.hannonhill.cascade.api.adapters.FolderContainedAssetAPIAdapter`
- `com.hannonhill.cascade.api.adapters.MetadataAwareAssetAPIAdapter`
- `com.hannonhill.cascade.api.adapters.PublishableAssetAPIAdapter`
- `com.hannonhill.cascade.api.adapters.IdentifierImpl`
- `com.hannonhill.cascade.api.adapters.PathImpl`
- `com.hannonhill.cascade.api.adapters.PathIdentifierImpl`
- `com.hannonhill.cascade.api.adapters.FolderAPIAdapter`
- `com.hannonhill.cascade.api.adapters.PageAPIAdapter`
- `com.hannonhill.cascade.api.adapters.XHTMLDataDefinitionBlockAPIAdapter`
- `com.hannonhill.cascade.api.adapters.MetadataAPIAdapter`
- `com.hannonhill.cascade.api.adapters.DynamicMetadataFieldImpl`
- `com.hannonhill.cascade.api.adapters.StructuredDataNodeAPIAdapter`
- We will only look at the most important methods defined in these classes and ignore everything else
- In the Velocity context, only non-mutative methods can be called; mutative methods like `setName`, when called, will have no effect (if a mutative method is invoked through reflection, an exception will be thrown)
- The relationship between the com.hannonhill.cascade.api.adapters package and the com.hannonhill.cascade.api.asset.common package:
- The com.hannonhill.cascade.api.asset.common package contains interfaces to be implemented
- The com.hannonhill.cascade.api.adapters package contains classes implementing interfaces in the com.hannonhill.cascade.api.asset.common package
- For details, see API Documentation

`com.hannonhill.cascade.api.adapters.PermissionsCapableAssetAPIAdapter`

- This class is subclass of `com.hannonhill.cascade.api.adapters.NamedAssetAPIAdapter`, which is a subclass of `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter` (in the following diagrams, I will omit package names to avoid long strings)

```
BaseAssetAPIAdapter
|__NamedAssetAPIAdapter
   |__PermissionsCapableAssetAPIAdapter
```

- This class inherits two important methods from its ancestors:
- `getIdentifer()`: returns a `com.hannonhill.cascade.api.asset.common.Identifier` object, representing the identifier of an asset
- `getName()`: returns the name (`java.lang.String`) of an asset

com.hannonhill.cascade.api.adapters.FolderContainedAssetAPIAdapter

- This class is a subclass of `com.hannonhill.cascade.api.adapters.PermissionsCapableAssetAPIAdapter`

```
BaseAssetAPIAdapter
|__NamedAssetAPIAdapter
   |__PermissionsCapableAssetAPIAdapter
      |__FolderContainedAssetAPIAdapter
```

- This class is the superclass of classes representing folder contained assets
- This class provides the following important methods:
- `getPath()`: returns the path string
- `getParentFolder()`
- `getSiteName()`

com.hannonhill.cascade.api.adapters.MetadataAwareAssetAPIAdapter

- This class is a subclass of `com.hannonhill.cascade.api.adapters.FolderContainedAssetAPIAdapter`

```
BaseAssetAPIAdapter
|__NamedAssetAPIAdapter
   |__PermissionsCapableAssetAPIAdapter
      |__FolderContainedAssetAPIAdapter
         |__MetadataAwareAssetAPIAdapter
```

- This class is the superclass of classes representing assets that can be associated with a metadata set
- This class provides a single important method:
- `getMetadata()`

com.hannonhill.cascade.api.adapters.com.hannonhill.cascade.api.adapters.P

- This class is a subclass of `com.hannonhill.cascade.api.adapters.MetadataAwareAssetAPIAdapter`

```
BaseAssetAPIAdapter
|__NamedAssetAPIAdapter
   |__PermissionsCapableAssetAPIAdapter
      |__FolderContainedAssetAPIAdapter
         |__MetadataAwareAssetAPIAdapter
            |__PublishableAssetAPIAdapter
```

- This class is the superclass of classes representing assets that can be published
- This class provides two important methods:
- `getShouldBeIndexed()`
- `getShouldBePublished()`

com.hannonhill.cascade.api.adapters.IdentifierImpl

- This class implements `com.hannonhill.cascade.api.asset.common.Identifier`
- Two important methods:
- `getId()`: returns the ID string
- `getType()`: returns the type (a `com.hannonhill.cascade.model.dom.identifier.EntityType` object)

`com.hannonhill.cascade.api.adapters.PathImpl`

- This class implements `com.hannonhill.cascade.api.asset.common.Path`
- Two important methods:
- `getPathAsString()`: returns the path string
- `getPathSegments()`: returns a list of strings

`com.hannonhill.cascade.api.adapters.PathIdentifierImpl`

- This class implements `com.hannonhill.cascade.api.asset.common.PathIdentifier`
- Relationship between identifier and path:
- The interface `com.hannonhill.cascade.api.asset.common.PathIdentifier` implements `com.hannonhill.cascade.api.asset.common.Identifier` (which has two methods) and adds a `getPath()` method, returning a `com.hannonhill.cascade.api.asset.common.Path` object
- That means that a class implementing `com.hannonhill.cascade.api.asset.common.PathIdentifier` will have three methods: `getId()`, `getType()` and `getPath()`
- Although `com.hannonhill.cascade.api.asset.common.PermissionsCapableAsset.getIdentifier()` returns a `com.hannonhill.cascade.api.asset.common.Identifier` object (that has no path information), the `getIdentifier()` method of its subclass `com.hannonhill.cascade.api.adapters.FolderContainedAssetAPIAdapter` returns a `com.hannonhill.cascade.api.asset.common.PathIdentifier` object, including information of ID, type, and path
- `com.hannonhill.cascade.api.adapters.FolderContainedAssetAPIAdapter` also provides a `getPath()` method, returning the path string , bypassing the `getIdentifier().getPath().getPathAsString()` chain of method calls
- Therefore, all folder-contained asset objects contain these three pieces of information, and provide two ways to access the path string
- On the other hand, `com.hannonhill.cascade.api.adapters.AssetFactoryAPIAdapter`, for example, is a subclass of `com.hannonhill.cascade.api.asset.common.PermissionsCapableAsset`, and its `getIdentifier()` only returns a `com.hannonhill.cascade.api.asset.common.Identifier` object (which has two methods, not three)

`com.hannonhill.cascade.api.adapters.FolderAPIAdapter`

- This class is a subclass of `com.hannonhill.cascade.api.adapters.PublishableAssetAPIAdapter`:

```
BaseAssetAPIAdapter
|__NamedAssetAPIAdapter
   |__PermissionsCapableAssetAPIAdapter
      |__FolderContainedAssetAPIAdapter
         |__MetadataAwareAssetAPIAdapter
            |__PublishableAssetAPIAdapter
               |__FolderAPIAdapter
```

- Important methods:
- `getChildren()`: returns a list of `com.hannonhill.cascade.api.asset.home.FolderContainedAsset` objects
- `getChildrenIdentifiers()`: returns a list of `com.hannonhill.cascade.api.asset.common.PathIdentifier` objects

`com.hannonhill.cascade.api.adapters.PageAPIAdapter`

- This class is a subclass of `com.hannonhill.cascade.api.adapters.PublishableAssetAPIAdapter`:

```
BaseAssetAPIAdapter
|__NamedAssetAPIAdapter
   |__PermissionsCapableAssetAPIAdapter
      |__FolderContainedAssetAPIAdapter
         |__MetadataAwareAssetAPIAdapter
            |__PublishableAssetAPIAdapter
               |__FolderAPIAdapter
               |__PageAPIAdapter
```

- Important methods:
- `getStructuredData()`: returns an array of `com.hannonhill.cascade.api.asset.common.StructuredDataNode` objects
- `getStructuredDataNodes( java.lang.String )`: returns an array of `com.hannonhill.cascade.api.asset.common.StructuredDataNode` objects with that name
- `getStructuredDataNode( java.lang.String )`: returns the first `com.hannonhill.cascade.api.asset.common.StructuredDataNode` object with that name

`com.hannonhill.cascade.api.adapters.XHTMLDataDefinitionBlockAPIAdapter`

- This class is a subclass of `com.hannonhill.cascade.api.adapters.BlockAPIAdapter`, which in turn is a subclass of `com.hannonhill.cascade.api.adapters.MetadataAwareAssetAPIAdapter`:

```
BaseAssetAPIAdapter
|__NamedAssetAPIAdapter
   |__PermissionsCapableAssetAPIAdapter
      |__FolderContainedAssetAPIAdapter
         |__MetadataAwareAssetAPIAdapter
            |__PublishableAssetAPIAdapter
            |  |__FolderAPIAdapter
            |  |__PageAPIAdapter
            |
            |__BlockAPIAdapter


|__XHTMLDataDefinitionBlockAPIAdapter
```

- Important methods:
- `getStructuredData()`: returns an array of `com.hannonhill.cascade.api.asset.common.StructuredDataNode` objects

- `getStructuredDataNodes( java.lang.String )`: returns an array of `com.hannonhill.cascade.api.asset.common.StructuredDataNode` objects with that name
- `getStructuredDataNode( java.lang.String )`: returns the first `com.hannonhill.cascade.api.asset.common.StructuredDataNode` object with that name

`com.hannonhill.cascade.api.adapters.MetadataAPIAdapter`

- This class is a subclass of `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter`, implementing `com.hannonhill.cascade.api.asset.common.DynamicMetadataField`
- Important methods:
- For all ten wired fields like `author` and `description`, there are ten corresponding methods like `getAuthor()` and `getDescription()`
- `getDynamicFields()`: returns a list of `com.hannonhill.cascade.api.asset.common.DynamicMetadataField` objects
- `getDynamicField( java.lang.String )`: returns the named `com.hannonhill.cascade.api.asset.common.DynamicMetadataField` object

`com.hannonhill.cascade.api.adapters.DynamicMetadataFieldImpl`

- This class implements `com.hannonhill.cascade.api.asset.common.DynamicMetadataField`
- Important methods:
- `getName()`
- `getValue()`: returns a single string from a plain text, a radio, or a dropdown
- `getLabel()`
- `getValues()`: returns an array of strings from a checkbox or a multiselect
- `isCheckbox()`, `isMultiselect()`, `isDropdown()`, `isRadio()`: used to test the type of a field

`com.hannonhill.cascade.api.adapters.StructuredDataNodeAPIAdapter`

- This class is a subclass of `com.hannonhill.cascade.api.adapters.BaseAssetAPIAdapter`, implementing `com.hannonhill.cascade.api.asset.common.StructuredDataNode`
- Important methods:
- `isText()`, `isAsset()`, `isGroup()`: used to test the type of a node
- `getAsset()`: used with an asset node (a chooser), returns the asset (file, block, page, symlink, or linkable) attached to the chooser
- `getChild( java.lang.String )`: used with a group, returns the named child node
- `getChildren()`, `getChildren( java.lang.String )`: used with a group node, returns the children in the group
- `getTextValue()`: returns the text value of a plain text, a WYSIWYG, a multiline, a calendar, a datetime, a dropdown, or a radio
- `getTextValues()`: returns an array of strings, the text values of a

checkbox or a multiselect

- `getTextNodeOptions()`: returns a `com.hannonhill.cascade.api.asset.common.TextNodeOptions` object, through which eight methods can be called: `isCalendar()`, `isDatetime()`, `isCheckbox()`, `isMultiselect()`, `isWysiwyg()`, `isDropdown()`, `isPlainText()`, `isRadio()`

The `LocatorTool$SearchQuery` Object

- The `com.hannonhill.cascade.velocity.LocatorTool` also defines a `query()` method, which returns a `com.hannonhill.cascade.velocity.LocatorTool$SearchQuery` object
- The `LocatorTool$SearchQuery` object can be used to search for assets based on some predefined search criteria
- Repeated calls of `LocatorTool.query()` will return multiple `LocatorTool$SearchQuery` objects, each of which can be used to search for assets independently
- There are a set of methods of `LocatorTool$SearchQuery` that:
- Return a `LocatorTool$SearchQuery` object
- Can be chained
- Should be called to set the search criteria
- Important methods:
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery byContentType( java.lang.String )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery byMetadataSet( java.lang.String )`
- `public java.util.List<? extends com.hannonhill.cascade.api.asset.common.BaseAsset> execute()`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery indexableOnly( boolean )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery hasMetadata( java.lang.String, java.lang.String )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery hasMetadata( java.lang.String, java.util.Collection<java.lang.String> )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery hasTag( java.lang.String )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery hasAnyTags( java.util.Collection<java.lang.String> )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery includeBlocks( boolean )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery includeFiles( boolean )`
- `public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery`

includeFolders(boolean)

- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery includePages(boolean)
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery includeSymlinks(boolean)
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery indexableOnly(boolean)
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery maxResults(int)
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery publishableOnly(boolean)
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery searchAcrossAllSites()
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery siteName(java.lang.String)
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery sortBy(java.lang.String)
- public com.hannonhill.cascade.velocity.LocatorTool$SearchQuery sortDirection(java.lang.String)
- To display the information of a `LocatorTool$SearchQuery` object, just output the object:

```
#set( $query = $_.query() )
$query
```

This will output:

```
[metadataFieldName = null
[displayName|title|summary|teaser|keywords|description|author|startDate|endDate|reviewD
 metadataFieldValues = [null]
 * Assign metadataFieldName and metadataFieldValue
by calling hasMetadata($name, $value)
 tags = [null]
 * Assign tags by calling hasAnyTags(['tag1',
'tag2']) or hasTag('tag')
 * Assign contentTypeLink by calling
byContentType($link) or metadataSetLink by calling
byMetadataSet($link)
 sortBy =
[summary|keywords|endDate|author|created|displayName|description|title|path|reviewDate|
 sortDirection = asc [asc|desc]
 maxResults = 100 [anything above 2000 will be
trimmed down to 2000]
 siteName = _chan_core
 * Call searchAcrossAllSites() to null out
 indexableOnly = true
 publishableOnly = false]

WARNING: Neither contentTypeLink nor
metadataSetLink nor tags have been provided.
Please provide one of these for this query to be
executable.
```

- To search for assets, either `byContentType` or `byMetadataSet` must be called; both methods accept an asset link:

```
#set( $query = $_.query() )
#set( $query = $query.byContentType(
"site://_brisk/Page" ) )
## or
#set( $query = $query.byMetadataSet(
"site://_brisk/Block" ) )
```

- When `byContentType` is called, only assets of type `com.hannonhill.cascade.api.adapters.PageAPIAdapter` will be returned
- All other search criteria have default settings; see the output of the `LocatorTool$SearchQuery` object for these default settings
- After calling all relevant methods to set up the search criteria, call `execute` to perform the search
- Examples:

```
## Ex 1: get all pages from a site
#set( $pages = $_.query().byMetadataSet(
"site://_brisk/Page" ).siteName(
    "formats" ).maxResults( 2000 ).sortBy(
"startDate" ).sortDirection( "desc" ).execute() )
##
#foreach( $page in $pages )
    $_EscapeTool.xml( $page.Metadata.Title )
#end

## Ex 2: get pages with certain metadata
#set( $pages1 = $_.query().byMetadataSet(
"site://_brisk/Page" ).hasMetadata(
    "exclude-from-menu", "yes" ).siteName(
"formats" ).maxResults( 2000 ).execute() )
#set( $pages2 = $_.query().byMetadataSet(
"site://_brisk/Page" ).hasMetadata(
    "tree-picker", "center" ).siteName( "formats"
).maxResults( 2000 ).execute() )
$pages1.size()
$pages2.size()

#set( $map1 = {} )
#set( $map2 = {} )

#foreach( $page in $pages1 )
    #set( $map1[ $page.Path ] = $page )
#end
#foreach( $page in $pages2 )
    #set( $map2[ $page.Path ] = $page )
#end
## and
#set( $intersection = {} )
#foreach( $key in $map1.keySet() )
    #if( $map2[ $key ].Class.Name )
        #set( $intersection[ $key ] = $map1[ $key
] )
    #end
#end
## or
#set( $union = {} )
#foreach( $key in $map1.keySet() )
    #set( $union[ $key ] = $map1[ $key ] )
#end
#foreach( $key in $map2.keySet() )
```

```
    #if( !$union[ $key ].Class.Name )
        #set( $union[ $key ] = $map2[ $key ] )
    #end
#end

Map 1:        $map1
Map 2:        $map2
Intersection: $intersection
Union:        $union

## Ex 3: find all multimedia blocks of all
functionality in a site
#import( "site://_brisk/core/library/velocity
/chanw/chanw-initialization" )
#set( $siteName = "cancer" )

#set( $blocks = $_.query().byMetadataSet(
"site://_brisk/Block" ).includePages(
    false ).includeFiles( false ).includeFolders(
false ).includeSymlinks(
    false ).siteName( $siteName ).maxResults( 2000
).execute() )
##$query.searchAcrossAllSites()
#set( $contentBlocks = {
    "single-image"  : [],
    "gallery"       : [],
    "random-image"  : [],
    "slide-show"    : [],
    "youtube-video" : []
} )

#foreach( $block in $blocks )
    #if( $block.Class.Name ==
$COM_CASCADE_DATA_BLOCK_API_CLASS_NAME &&
$block.DataDefinitionPath == "Block Multimedia" )
        #set( $dummy = $contentBlocks[
$block.getStructuredDataNode( "choose-type"
).TextValues[ 0 ] ].add( $block.Identifier.Id ) )
    #end
#end

$contentBlocks
```

Using `#chanwInvokeQueryWithMap`

- Using a `LocatorTool$SearchQuery` object can be tedious because we need to configure so many different settings
- We must repeat the following code pattern whenever we use such an object:
- Call `com.hannonhill.cascade.velocity.LocatorTool.query()` to create the object
- Call all relevant methods to provide mandatory information or to override default settings
- Call `com.hannonhill.cascade.velocity.LocatorTool$SearchQuery.execute()` to perform the search
- The library macro named `#chanwInvokeQueryWithMap` can be used to hide away most of the details:
  `#import( "site://_brisk/core/library/velocity /chanw/chanw-library-import" )`

```
#chanwInvokeQueryWithMap( {
    $PARAM_TYPE         :
$PARAM_VALUE_METADATA_SET,
    $PARAM_LINK         : "site://_common_assets
/Page",
    $PARAM_HAS_METADATA : [ "exclude-from-menu", [
"Yes" ] ],
    $PARAM_INDEXABLE    : true
} )

$chanwInvokeQueryWithMap.size()
```

- The map passed into the macro should contain entries related to various settings
- When calling this macro, at least two entries must be provided: `$PARAM_TYPE` (with the value `$PARAM_VALUE_CONTENT_TYPE` or `$PARAM_VALUE_METADATA_SET`) and `$PARAM_LINK` (the asset link)
- Note that all constants related to the `com.hannonhill.cascade.velocity.LocatorTool$SearchQuery` object are defined in `chanw-process-cascade-api`:

```
#set( $PARAM_VALUE_CONTENT_TYPE  = "ContentType" )
#set( $PARAM_VALUE_METADATA_SET  = "MetadataSet" )
#set( $PARAM_VALUE_ASC           = "asc" )
#set( $PARAM_VALUE_DESC          = "desc" )
#set( $PARAM_VALUE_AUTHOR        = "author" )
#set( $PARAM_VALUE_CREATED       = "created" )
#set( $PARAM_VALUE_DESCRIPTION   = "description" )
#set( $PARAM_VALUE_DISPLAY_NAME  = "displayName" )
#set( $PARAM_VALUE_END_DATE      = "endDate" )
#set( $PARAM_VALUE_KEYWORDS      = "keywords" )
#set( $PARAM_VALUE_MODIFIED      = "modified" )
#set( $PARAM_VALUE_LAST_MODIFIED = "modified" )
#set( $PARAM_VALUE_NAME          = "name" )
#set( $PARAM_VALUE_PATH          = "path" )
#set( $PARAM_VALUE_REVIEW_DATE   = "reviewDate" )
#set( $PARAM_VALUE_START_DATE    = "startDate" )
#set( $PARAM_VALUE_SUMMARY       = "summary" )
#set( $PARAM_VALUE_TEASER        = "teaser" )
#set( $PARAM_VALUE_TITLE         = "title" )
#set( $PARAM_TYPE                = "type" )
#set( $PARAM_LINK                = "link" )
#set( $PARAM_ASSETS              = "assets" )
#set( $PARAM_BLOCKS              = "blocks" )
#set( $PARAM_FILES               = "files" )
#set( $PARAM_FOLDERS             = "folders" )
#set( $PARAM_PAGES               = "pages" )
#set( $PARAM_SYMLINKS            = "symlinks" )
#set( $PARAM_HAS_METADATA        = "has-metadata"
)
#set( $PARAM_DIRECTION           = "direction" )
#set( $PARAM_MAX_RESULTS         = "max-results" )
#set( $PARAM_SITE_NAME           = "site-name" )
#set( $PARAM_SORT_BY             = "sort-by" )
#set( $PARAM_INDEXABLE           = "indexable" )
#set( $PARAM_PUBLISHABLE         = "publishable" )
```

- Default values assigned to parameters:

```
{
    $PARAM_ASSETS :
    {
        $PARAM_BLOCKS   : true,
        $PARAM_FILES    : true,
```

```
        $PARAM_FOLDERS  : true,
        $PARAM_PAGES    : true,
        $PARAM_SYMLINKS : true
    },
    $PARAM_DIRECTION   : $PARAM_VALUE_ASC,
    $PARAM_INDEXABLE   : true,
    $PARAM_PUBLISHABLE : false,
    $PARAM_MAX_RESULTS : 100
}
```