Custom CAM Migration

QuestFields Tech Note 025

⇒ *This tech note describes the steps involved in upgrading the Java CAM to the new Custom CAM available in QuestFields Server 2.0.*

1. Prerequisites

qo-cam-custom-configuration

In QuestFields Server 2.0 the Java CAM has been renamed to Custom CAM. The Custom CAM is described in the QuestFields Custom Content Access Module Configuration guide `qo-cam-custom-configuration`. It includes an overview of the API as well.

Java version

QuestFields Server 2.0 requires the use of Java 5. The API has changed to reflect this by the use of generics.

Dependency changes

The Custom CAM has been folded into the main release of QuestFields Server 2.0. The dependencies that are needed are now:

```
log4J-1.2.8.jar
questobjects-base.jar
questobjects-service.jar
```

2. API changes.

The following steps are required to update your Java plugin to support the new server.

a. Update the package

The package for `QueryExecutor` and for `QueryExecutorFactory` has changed from `com.masterobjects.qo.external.javacam` to `com.masterobjects.qo.external.customplugin`.

b. Replace `QoJavaCamException` with `QoCustomException`

Replace occurrences of `QoJavaCamException` with `QoCustomCamException`. `QoCustomCamException` can be found in the `com.masterobjects.qo.external.customplugin` package.

c. Replace `QoJavaCamResults` with `QoCustomResults`

Replace occurrences of `QoJavaCamResults` with `QoCustomResults`. `QoCustomResults` can be found in the `com.masterobjects.qo.external.customplugin` package.

d. Add the `getBuildNumber`, `getName` and `getManufacturer` methods to the `QueryExecutorFactory`

The `QueryFactoryExecutor` has been extended with several methods. Three of these methods are used by the server to determine the version, name and manufacturer of the Java plugin. `getBuildNumber` returns a `String` with the build number, `getName` returns a `String` with the name and `getManufacturer` returns a `String` with the manufacturer (Your company).

e. Add the `getKeys` method to the `QueryExecutorFactory`

One other method that has been added to the `QueryExecutorFactory` is the `getKeys` method. The `getKeys` method receives the Content Query configuration (as a `Hashtable<String,String>`) and returns an `Array` of `String`. This array contains the keys that will be used by the Channel to map the value of the query to. See the `qo-server-administration` guide for more info on how the Channel interacts with the Content Query.

A good way to use this method is to potentially do startup work that was previously done in the `getExecutor`, and to return a `String` array with the `protected static final Strings "query"` and `"qualifier"` as items in the array. This would simplify the rest of the conversion.

f. Change the generic `HashMap` and `Hashtable` to their specific ones

In the signatures of the following methods the generic versions of either `HashMap` or `Hashtable` need to change to the specific versions. The specific versions are always in the form of `<String,String>`:

- In `QueryExecutorFactory.init` change `HashMap factoryProperties` to `HashMap<String,String> factoryPropeties.`
- In `QueryExecutorFactory.getExecutor` change `Hashtable channelProperties` in `Hashtable<String,String> contentQueryProperties` (the rename is optional, but advised since the properties aren't stored in the Channel anymore but in the Content Query configuration).

g. Change the `executeQuery` method in `QueryExecutor` to accept a `HashMap<String,String>` instead of a `query String`, `qualifier String` and a `Map` of `dependencyData`

The `executeQuery` method in the `QueryExecutor` has changed. Instead of receiving a query, a qualifier and a `Map` with dependency data it now receives just one `HashMap`. The `HashMap` contains `key value` pairs. The `keys` are the same as defined in the `getKeys`. Use the `protected static final String` for `query` and for `qualifier` in the `QueryExecutorFactory` to make sure you get the same ones.

3. Configuration changes

The configuration has changed considerably. All the Channel configuration that was passed to the QueryExecutorFactory has been put in the Content Query configuration. The Channel can now call several Content Queries based on the pattern of the incoming query. Both the Channel and the Content Query are now configured in XML files. For an exhaustive explanation of the configuration please review the qo-admin-configuration guide and the qo-cam-custom-configuration guide.

Based on the described migration path a channel configuration file would like this:

```
<channel id="customchannel">

  <name>Custom Channel</name>

  <helpText><![CDATA[This is a custom channel]]></helpText>
```

```
  <copyrightText><![CDATA[The data is retrieved by a Custom CAM plug-in. The data is used for
this demonstration only. The data is &#169; 2007 someone.]]></copyrightText>

  <sortingModeKey>CASE_INSENSITIVE</sortingModeKey>

  <maxResults>1000</maxResults>

  <resultSetLifetime>30</resultSetLifetime>

  <queryTimeout>60000</queryTimeout>

  <minQueryLength>1</minQueryLength>

  <querySelectors>

    <querySelector>

      <contentQueryId>custom-content-query</contentQueryId>

      <selectionCriterion>

        <questerAttribute>INPUT_BUFFER</questerAttribute>

        <groupToVariableMapping group="0" variable="query" defaultValue=""/>

      </selectionCriterion>

         <selectionCriterion>

        <questerAttribute>QUALIFIER</questerAttribute>

        <groupToVariableMapping group="0" variable="qualifier" defaultValue=""/>

      </selectionCriterion>

    </querySelector>

  </querySelectors>

</channel>
```

⇒ *Note that the `INPUT_BUFFER` is mapped to `query` and the `QUALIFIER` to `qualifier` as described in the API changes section.*

The Content Query configuration contains all the QueryExecutor specific configuration (this used to be a part of the Channel configuration). It could look like this:

```
<customContentQuery id="custom-content-query">

  <contentQueryProperties>

    <entry><key>aKey</key><value>Some value</value></entry>

    <entry><key>anotherKey</key><value>another Value</value></entry>

  </contentQueryProperties>

  <camId>custom-cam</camId>
```

```
</customContentQuery>
```

⇒ *Note that there is no need anymore to include a list of all the keys used as was the case with the previous version. All the keys (and their values) are passed to the `getKeys` and `getExecutor` methods.*

The Custom CAM configuration has only changed in the way certain elements are called:

```
<customCam id="custom-cam">

  <factoryClassName>com.masterobjects.qo.external.examples.ConfigTestFactory</factoryClassName>

  <factoryProperties>

        <entry>

            <key>aKey</key>

           <value>a value</value>

        </entry>

  </factoryProperties>

  <pluginDirectory>/questobjects-home/plugins/</pluginDirectory>

</customCam>
```

⇒ *Note that sorting is defined programmatically in the plugin instead of the configuration.*