Oracle Enterprise Repository

Categorization Types and Categorizations API

Overview

It is important to understand the difference between **Categorizations** and **Categorization Types**.

Categorization Types provide the means to define custom taxonomies. **Categorizations** are subsets of Categorization Types, and are assigned to assets. For example, a Categorization Type called *Technology* might contain *Java*, *.NET* and *COBOL* as its assignable Categorizations. Additionally, sub-categorizations under *Java* might include *Servlet* and *Applet*. So an asset in Oracle Enterprise Repository might be assigned to the *Java* Categorization, or it might be more specifically assigned to the *Servlet* Sub-categorization.

The Categorizations to which a particular asset may be assigned are determined by the Categorization Types defined for that asset's Type. As in the example above, if the *Technology* Categorization Type is defined for Asset Type *XYZ*, assets of type *XYZ* may be assigned to the *Java*, *.NET*, or *COBOL*, Categorizations, or to the Sub-categorizations *Servlet* or *Applet*. This taxonomy structure allows assets to be grouped and viewed in a variety of ways within Oracle Enterprise Repository.

Categorization types can also be associated with projects (if Oracle Enterprise Repository is configured for that feature). Any project-assignable Categorization Type is available to all projects. As with assets, a project can be associated with any of the Categorizations available within its assigned Categorization Type (s).

Rules for Categorization Types and Categorizations include:

- The Repository can contain 0 to *n* Categorization Types with each Categorization Type containing 0 to n Categorizations.
- Each Categorization can contain 0 to n Sub-categorizations.
- Each Categorization Type must have a unique name. This name cannot contain spaces or special characters.

- As an option, Categorizations within a given Categorization Type may be made mutually exclusive. That is, when a list of Categorizations is presented, only one may be selected.

- When the Mutually Exclusive option is selected for a Categorization Type, Oracle Enterprise Repository enforces the rule for future usage only. Existing references to multiple selected categorizations within the Type are unchanged.

- When so configured, Categorization Types can be assigned to projects. This allows projects in Oracle Enterprise Repository to be organized by Categorization Type/Categorization.

- If the configuration of a specific Categorization Type is changed to prevent its assignment to Projects, the change affects only subsequent Project assignment. The change does not affect the Categorization Type's assignment to existing Projects..

- Categorization Types may be deleted from Oracle Enterprise Repository. However, doing so also deletes all categorizations within the deleted Categorization Type. Exercise caution when performing this task.

- Categorizations may be deactivated. Deactivation prevents future use of the Categorization (and all sub-categorizations) but does not delete it from Oracle Enterprise Repository. Existing references to a Categorization are unaffected by deactivation.

- Deactivated Categorizations may be reactivated, reversing the aforementioned process.

- Within a given Categorization Type, all Categorizations must be uniquely named. However, the same name may be shared by multiple Categorizations residing in different Categorization Types.

The following methods provide the ability to create, update, list, query, and delete categorization types.

Use Case: Create a Categorization Type

Description

The element name given to a newly created Categorization Type cannot contain special characters or spaces. There are no restrictions on the characters used for singular and plural display names. The `pExclusiveAssign` Boolean determines whether one or multiple Categorizations can be assigned within the Categorization Type. The `pExclusiveAssign` Boolean determines if the Categorization Type is project-assigned. The method prevents duplication of existing Categorizations, and returns the created Categorization Type.

Sample code:

```
package com.flashline.sample.categorizationtypesandapi;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;

import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.CategorizationType;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

public class CreateCategorizationType {
  public static void main(String pArgs[]) throws java.rmi.RemoteException,
      OpenAPIException {
    try {

      ///////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ///////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);

      // //////////////////////////////
      // Authenticate with OER
      // //////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);


      // //////////////////////////////
      // Create the categorization type
      // //////////////////////////////
      String pName = "Name";
      String pSingularDisplay = "SingularDisplay";
      String pPluralDisplay = "PluralDisplay";
      boolean pExclusiveAssign = true;
      boolean pProjectAssign = true;

      CategorizationType lCategorizationType = repository.categorizationTypeCreate(authToken, pName,
          pSingularDisplay, pPluralDisplay, pExclusiveAssign, pProjectAssign);

      // ----------------------------------------
      // clean up
      repository.categorizationTypeDelete(authToken, lCategorizationType);
    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }

}
```

Use Case: Manipulate Categorization Types

Description

The following operations are demonstrated in the example below

- Retrieve Categorization Types by ID
- Update a Categorization Type
- Delete a Categorization Type
  Excercise Caution! This method deletes the entire Categorization Type and all Categorizations contained therein.
- Query a Categorization Type
  Use various terms, including name, type, and if a Categorization Type is assigned to projects.
- Retrieve *all* Categorization Types

Sample code:

```
package com.flashline.sample.categorizationtypesandapi;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;

import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Categorization;
import com.flashline.registry.openapi.entity.CategorizationType;
import com.flashline.registry.openapi.query.CategorizationTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

public class CategorizationExamples {
  public static void main(String pArgs[]) throws ServiceException, RemoteException,
      OpenAPIException {
    try {

      ///////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ///////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);

      // //////////////////////////////
      // Authenticate with OER
      // //////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);

      // //////////////////////////////
      // Create a Categorization Type
      // //////////////////////////////
      CategorizationType categorizationType = repository
          .categorizationTypeCreate(authToken, "exampleType", "Example Type",
              "Example Types", false, true);

      // //////////////////////////////
      // Find and update a categorization type
      // //////////////////////////////
      CategorizationTypeCriteria categorizationTypeCriteria = new CategorizationTypeCriteria();
      boolean projectAssign = true;
      categorizationTypeCriteria.setProjectAssignCriteria(projectAssign + "");
      CategorizationType[] categorizationTypes = repository
          .categorizationTypeQuery(authToken, categorizationTypeCriteria);

      // Set plural display name
      categorizationType.setDisplayPlural("Updated Example Types");

      // Set singular display name
      categorizationType.setDisplaySingular("Updated Example Type");

      // Set Categorization Type name
      categorizationType.setName("updatedExampleType");

      // Set Categorization Type exclusive assign
      categorizationType.setExclusiveAssign(true);
```

```
// Set Categorization Type project Assignable
categorizationType.setProjectAssignable(false);

// Update Categorization Type
categorizationType = repository.categorizationTypeUpdate(
    authToken, categorizationType);

// Read a Categorization Type
CategorizationType categorizationTypeRead = repository
    .categorizationTypeRead(authToken, categorizationType.getID());

// //////////////////////////////
// Create a Categorization within a Categorization Type
// //////////////////////////////
Categorization categorization = repository.categorizationCreate(
    authToken, "Example Categorization", categorizationType);

// //////////////////////////////
// Create a Categorization within a Categorization (a.k.a. a
// sub-categorization or child categorization)
//
// method validates that:
//   - no child categorization with the same name exists within the
//     parent categorization.
//   - the categorization type is valid
//   - the parent categorization is valid.
// //////////////////////////////
Categorization childCategorization = repository
    .categorizationChildCreate(authToken, "Example Child Categorization",
        categorizationType, categorization);

childCategorization.setName("Updated Example Child Categorization");

childCategorization = repository.categorizationUpdate(authToken,
    childCategorization, categorizationType);

// //////////////////////////////
// Observe various properties of a categorization. Note that the
// properties are not being assigned to local variables in
// the interest of brevity...
// //////////////////////////////
Categorization categorizationRead = repository.categorizationRead(
    authToken, childCategorization.getID());

// Get Categorization parent id
//categorizationRead.getParentID();

// Get Categorization active status
categorizationRead.getActiveStatus();
```

```
// Get Categorization ID
categorizationRead.getID();

// Get Categorization name
categorizationRead.getName();

// Get Categorization recursive name
categorizationRead.getRecursiveName();

// Get Categorization Categorization Type ID
categorizationRead.getTypeID();

// //////////////////////////////
// Retrieve the full tree of categorizations for a Categorization Type.
// This means that the Categorization entity returned will have children
// which contain their children (if any), and so on.
// //////////////////////////////
// Get active Categorizations full tree
boolean active = true;
boolean fullTree = true;
Categorization[] categorizationArray = repository.categorizationReadByType(
   authToken, categorizationType, active, fullTree);

// Get children categorizations for categorization
Categorization[] childCategorizations = repository.categorizationChildRead(authToken,
   categorizationType, categorization, active);

// //////////////////////////////
// Deactivate a Categorization
// //////////////////////////////
categorization = repository.categorizationDeactivate(authToken,
   categorization, categorizationType);

// pActive is set to "true" so that the method returns
// only active categorizations
Categorization[] activeCategorizations = repository
   .categorizationChildRead(authToken, categorizationType,
      categorization, true);

// Get inactive child Categorizations
Categorization[] inactiveCategorizations = repository.categorizationChildRead(authToken,
   categorizationType, categorization, false);

// /////////////////////////////
// Reactivate Categorizations
// /////////////////////////////
for(int i=0; i<inactiveCategorizations.length; i++){
  categorization = repository.categorizationReactivate(authToken,
     inactiveCategorizations[i], categorizationType);
```

```
      }

      // /////////////////////////////
      // Delete a Categorization Type
      // /////////////////////////////
      repository.categorizationTypeDelete(authToken, categorizationType);

    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }

}
```

Pitfalls

- **Methods to Avoid**

  The methods listed below are for internal Oracle Enterprise Repository use only. Incorrect use of these methods may disrupt the functionality of Categorization Types (though permanent damage is unlikely) . The functionality provided by these methods is incorporated in the Oracle Enterprise Repository `CategorizationType` methods.

  - getActiveStatus() int - CategorizationType
  - getCategorizations() Categorizations[] - CategorizationType
  - GetEntityType() String - CategorizationType
  - getKey() String - CategorizationType
  - getTypeDesc() TypeDesc - CategorizationType
  - hashCode() int - CategorizationType
  - isCustom() boolean - CategorizationType
  - isFlat() boolean - CategorizationType
  - isSystemOnly() boolean - CategorizationType
  - setActiveStatus(int activeStatus) void - CategorizationType
  - setAssetAssignable(boolean assetAssignable) void - CategorizationType
  - setCategorizations(Categorizations[] categorizations) void - CategorizationType
  - setCustom(boolean custom) void - CategorizationType
  - setEntityType(String entityType) void - CategorizationType
  - setFlat(boolean flat) void - CategorizationType

- setID(long ID) void - CategorizationType
- setKey(String key) void - CategorizationType
- setSystemOnly(boolean systemOnly) void - CategorizationType

Use Case: Manipulate Categorizations

The following code sample illustrates creation, updating, listing, and deactivation/ reactivation of Categorizations. As stated above, Categorizations are subordinate to Categorization Types. That is, a Categorization *belongs to* a Categorization Type.

Sample code:

```
package com.flashline.sample.categorizationtypesandapi;

import java.net.MalformedURLException;
import java.net.URL;
import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;

import com.flashline.registry.openapi.base.OpenAPIException;
import com.flashline.registry.openapi.entity.AuthToken;
import com.flashline.registry.openapi.entity.Categorization;
import com.flashline.registry.openapi.entity.CategorizationType;
import com.flashline.registry.openapi.query.CategorizationTypeCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

public class CategorizationExamples {
  public static void main(String pArgs[]) throws ServiceException, RemoteException,
      OpenAPIException {
    try {

      ///////////////////////////////////////////////////////////
      // Connect to Oracle Enterprise Repository
      ///////////////////////////////////////////////////////////
      URL lURL = null;
      lURL = new URL(pArgs[0]);
      FlashlineRegistry repository = new FlashlineRegistryServiceLocator()
          .getFlashlineRegistry(lURL);

      // //////////////////////////////
      // Authenticate with OER
      // //////////////////////////////
      AuthToken authToken = repository.authTokenCreate(pArgs[1],
          pArgs[2]);
```

```
// //////////////////////////////
// Create a Categorization Type
// //////////////////////////////
CategorizationType categorizationType = repository
    .categorizationTypeCreate(authToken, "exampleType", "Example Type",
        "Example Types", false, true);

// //////////////////////////////
// Find and update a categorization type
// //////////////////////////////
CategorizationTypeCriteria categorizationTypeCriteria = new CategorizationTypeCriteria();
boolean projectAssign = true;
categorizationTypeCriteria.setProjectAssignCriteria(projectAssign + "");
CategorizationType[] categorizationTypes = repository
    .categorizationTypeQuery(authToken, categorizationTypeCriteria);

// Set plural display name
categorizationType.setDisplayPlural("Updated Example Types");

// Set singular display name
categorizationType.setDisplaySingular("Updated Example Type");

// Set Categorization Type name
categorizationType.setName("updatedExampleType");

// Set Categorization Type exclusive assign
categorizationType.setExclusiveAssign(true);

// Set Categorization Type project Assignable
categorizationType.setProjectAssignable(false);

// Update Categorization Type
categorizationType = repository.categorizationTypeUpdate(
    authToken, categorizationType);

// Read a Categorization Type
CategorizationType categorizationTypeRead = repository
    .categorizationTypeRead(authToken, categorizationType.getID());

// //////////////////////////////
// Create a Categorization within a Categorization Type
// //////////////////////////////
Categorization categorization = repository.categorizationCreate(
    authToken, "Example Categorization", categorizationType);

// //////////////////////////////
// Create a Categorization within a Categorization (a.k.a. a
// sub-categorization or child categorization)
//
// method validates that:
```

```
//   - no child categorization with the same name exists within the
//     parent categorization.
//   - the categorization type is valid
//   - the parent categorization is valid.
// ///////////////////////////////
Categorization childCategorization = repository
    .categorizationChildCreate(authToken, "Example Child Categorization",
        categorizationType, categorization);

childCategorization.setName("Updated Example Child Categorization");

childCategorization = repository.categorizationUpdate(authToken,
    childCategorization, categorizationType);

// ///////////////////////////////
// Observe various properties of a categorization. Note that the
// properties are not being assigned to local variables in
// the interest of brevity...
// ///////////////////////////////
Categorization categorizationRead = repository.categorizationRead(
    authToken, childCategorization.getID());

// Get Categorization parent id
//categorizationRead.getParentID();

// Get Categorization active status
categorizationRead.getActiveStatus();

// Get Categorization ID
categorizationRead.getID();

// Get Categorization name
categorizationRead.getName();

// Get Categorization recursive name
categorizationRead.getRecursiveName();

// Get Categorization Categorization Type ID
categorizationRead.getTypeID();

// ///////////////////////////////
// Retrieve the full tree of categorizations for a Categorization Type.
// This means that the Categorization entity returned will have children
// which contain their children (if any), and so on.
// ///////////////////////////////
// Get active Categorizations full tree
boolean active = true;
boolean fullTree = true;
Categorization[] categorizationArray = repository.categorizationReadByType(
    authToken, categorizationType, active, fullTree);
```

```
      // Get children categorizations for categorization
      Categorization[] childCategorizations = repository.categorizationChildRead(authToken,
        categorizationType, categorization, active);

      // ////////////////////////////
      // Deactivate a Categorization
      // ////////////////////////////
      categorization = repository.categorizationDeactivate(authToken,
        categorization, categorizationType);

      // pActive is set to "true" so that the method returns
      // only active categorizations
      Categorization[] activeCategorizations = repository
        .categorizationChildRead(authToken, categorizationType,
          categorization, true);

      // Get inactive child Categorizations
      Categorization[] inactiveCategorizations = repository.categorizationChildRead(authToken,
        categorizationType, categorization, false);

      // ////////////////////////////
      // Reactivate Categorizations
      // ////////////////////////////
      for(int i=0; i<inactiveCategorizations.length; i++){
       categorization = repository.categorizationReactivate(authToken,
         inactiveCategorizations[i], categorizationType);
      }

      // ////////////////////////////
      // Delete a Categorization Type
      // ////////////////////////////
      repository.categorizationTypeDelete(authToken, categorizationType);

    } catch (OpenAPIException lEx) {
      System.out.println("ServerCode = " + lEx.getServerErrorCode());
      System.out.println("Message    = " + lEx.getMessage());
      System.out.println("StackTrace:");
      lEx.printStackTrace();
    } catch (RemoteException lEx) {
      lEx.printStackTrace();
    } catch (ServiceException lEx) {
      lEx.printStackTrace();
    } catch (MalformedURLException lEx) {
      lEx.printStackTrace();
    }
  }

}
```

Pitfalls

- **Methods to Avoid**

  The methods listed below are for internal Oracle Enterprise Repository use only and should not be used. Incorrect use of these methods could cause improper functioning of categorizations. The functions provided by these methods provide are incorporated in the Oracle Enterprise Repository categorization methods.

  - getDescription() String - Categorization
  - GetEntityType() String - Categorization
  - getKey() String - Categorization
  - getLevel() long - Categorization
  - getType() CategorizationType - Categorization
  - getTypeDesc() TypeDesc - Categorization
  - hashCode() int - Categorization
  - set_super(Categorization _super) void - Categorization
  - setActiveStatus(int activeStatus) void - Categorization
  - setDeleted(boolean deleted) void - Categorization
  - setDescription(String description) void - Categorization
  - setEntityType(String entityType) void - Categorization
  - setID(long ID) void - Categorization
  - setKey(String key) void - Categorization
  - setRecursiveName(String recursiveName) void - Categorization
  - setType(CategorizationType type) void - Categorization
  - setTypeID(long ID) void - Categorization