

Oracle Enterprise Repository

Department API

Overview

Description

Departments can be created, read, queried for, and modified. These operations are described below. Bear in mind that once a Department is created, it cannot be deleted. Only two Department attributes are meaningful to a user: name and description.

Additional Import(s) Required

```
import com.flashline.registry.openapi.entity.Department;
```

Use Case: Manipulate Departments

Description

The following sample code illustrates typical tasks involving the manipulation of departments in Oracle Enterprise Repository. This includes creation, updating, querying, and deleting.

Sample code:

```
package com.flashline.sample.departmentapi;  
  
import java.net.MalformedURLException;  
import java.net.URL;  
import java.rmi.RemoteException;
```

```

import java.util.Calendar;

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.Department;
import com.flashline.registry.openapi.query.DepartmentCriteria;
import com.flashline.registry.openapi.service.v300.FlashlineRegistry;
import com.flashline.registry.openapi.service.v300.FlashlineRegistryServiceLocator;

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

        try {

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

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

            // //////////////////////////////////////
            // Create a new department
            // Each Department requires a unique name. Descriptions are optional.
            // //////////////////////////////////////
            Department dept = repository.departmentCreate(authToken,
                "My Dept "+Calendar.getInstance().getTimeInMillis(), "A New Department");

            // //////////////////////////////////////
            // Read a department
            // In order to read a Department you must have the Department name.
            // //////////////////////////////////////
            Department dept2 = repository.departmentRead(authToken,
                "ADepartment");

            // //////////////////////////////////////
            // Query for a department
            //
            // In order to query for a Department you must fill out a
            // DepartmentCriteria object with an array of SearchTerms. A SearchTerm

```

```

// is a key/value pair. Currently the only valid key is "name".
//
// A query for name is a match if the value for the name term
// occurs anywhere in the name of the department. For example,
// a search for fred matches fred, alfred, and fredrick.
// ///////////////////////////////////
DepartmentCriteria criteria = new DepartmentCriteria();
criteria.setNameCriteria("DepartmentName");
Department[] depts = repository.departmentQuery(authToken,
    criteria);

// ///////////////////////////////////
// Update a department
//
// To update a Department you need only to modify a Department
// reference and call departmentUpdate...
// ///////////////////////////////////
String lOldName = dept.getName();
String lNewName = "New " + dept.getName();
Department dept3 = repository.departmentRead(authToken, lOldName);
dept3.setName(lNewName);
repository.departmentUpdate(authToken, dept3);

} 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();
}
}
}

```