Catalog Services 1.0 Component Specification

1. Design

The Catalog stores information about TopCoder components and applications.

This component will provide services for the Catalog. The services implementation is pluggable, and provides an implementation based on Catalog Entities component, which defines entities and the O/R mapping.

This design uses EJB 3.0 and EntityManager to fulfill requirements.

1.1 Design Patterns

None.

1.2 Industry Standards

XML

JPA

Hibernate

JBoss

Informix Database

1.3 Required Algorithms

1.3.1Component to AssetDTO

This is the mapping of the DTO fields:

- name <---> Component.name
- clientIds <---> List<Component.clients.clientId> [optional]
- versionText <---> CompVersion.versionText
- shortDescription <---> Component.shortDesc
- detailedDescription <---> Component.description
- functionalDescription <---> Component.functionalDesc
- rootCategory <---> Component.rootCategory
- categories <---> Component.categories
- technologies <---> CompVersion.technologies
- productionDate <---> CompVersion.versionDates(phase.id).productionDate [optional]
- link <---> CompVersion.link [optional]
- forum <---> CompVersion.forum [optional]
- currentLatest <---> whether the currentVersion is also the latest [optional]
- id <---> Component.id [optional]
- versionId <---> CompVersion.id [optional]
- userIds <---> List<Component.users.userId> [optional]

- informationComplete:boolean <---> wheter the information in the DTO is complete or partial [optional]

The user can choose to retrieve the current or the latest version. To get the current version access Component.currentVersion. To get the latest one just p ick in Component.versions the one with the highest version field. This can be done either via query or by getting Component.versions and comparing dates. It is left to developer the choice of which approach to take, keeping in mind that the goal is performance.

1.3.2 AssetDTO to Component

The mapping of course is the same as in 1.3.1. ClientId, productionDate, link and forum When doing the translation, take into account the following:

Component entity

- Set status to Status.REQUESTED

CompVersion Entity

- Set phaseTime to 1976-05-05
- Set phasePrice to 0

CompVersionDates entity

- Just create an entry for the collaboration phase (phase_id=111)
- Set totalSubmissions to 0
- Set price to 0
- Set productionDate to the specified value, or null if not specified
- Set postingDate to 1976-05-05
- Set all the other dates to 2000-01-01
- Set levelId to 100
- Set status to Status.NEW_POST
- Leave all the comments fields in null

1.3.3Find components

When creating the query, keep in mind that only non-null and not empty parameters are considered, so the actual query String is created at runtime.

It's implemented in a single query with various number of joins and dynamically built 'where' clause. The example of the queries:

1. When only SearchCriteria.description is set (for the current version):

select A.component_id, A.component_name, A.short_desc, A.root_category_id, B.version_text, B.comp_vers_id from comp_catalog A

join comp_versions B on *A.current_version=B.comp_vers_id*

where A.component_id in (**select distinct C.component_id from comp_catalog C where (lower(C.description) like '%description%')**)

The bold part is dynamically build part which selects Component.componentIds eligible for the retrieval.

Then main query which takes only necessary properties filters Cmponent's ids using 'in'.

In this case shown join between comp_versions and comp_catalog by current_version field (when assets for the current version demanded).

For example, here more complex query with filtering by clientId and name (for the latest version):

```
select A.component_id, A.component_name, A.short_desc, A.root_category_id,
B.version_text, B.comp_vers_id from comp_catalog A
        join comp_versions B on A.component_id=B.component_id
         where A.component_id in (select distinct C.component_id from comp_catalog C
                                        join comp_client CL on
CL.component_id=C.component_id
                                          where (CL.client_id=?) and
(lower(C.component_name) like '%some service%'))
        and version=(select max(version) from comp_versions where
component_id=A.component_id)
```

Here added a join to the comp_client table, and the taken version is the max for each component (**bold** selection).

Join to versions is performed by 'component_id' from 'comp_version' table (*italic* selection).

In the most complex case (when all search criterias are entered):

```
select A.component_id, A.component_name, A.short_desc, A.root_category_id,
B.version_text, B.comp_vers_id from comp_catalog A
        join comp_versions B on A.current_version=B.comp_vers_id
         where A.component_id in (select distinct C.component_id from comp_catalog C
                left outer join comp_user U on C.component_id=U.component_id
                left outer join comp_client CL on CL.component_id=C.component_id
                left outer join user_client UC on UC.client_id=CL.client_id
                 where (U.user_id=1011 OR UC.user_id=1011)
                        AND (CL.client_id=562)
                        AND (lower(C.component_name) like '%a service%')
                        AND (lower(C.description) like '%ejb%')
                        AND (C.root_category_id in (12,14)))
```

All possible variations of parameters tested by method of full enumeration.

The algorithm of building the query is the following:

1. Build query for extracting IDs of eligible components

    1.1. If searchCriteria.userId is set add three joins to comp_user, comp_client and user_client tables and one where clause comp_user.user_id=? OR user_client.user_id=?

    1.2. If searchCriteria.clientId is set then if no userId set add join to comp_client (otherwise it was added in 1.1). Add where clause comp_client..client_id=?

1.3 If searchCriteria.name is set then and not empty add where clause (lower(comp_catalog.component_name) like '%?%')

1.4 If searchCriteria.description is set and not empty thenadd where clause (lower(comp_catalog.description) like '%?%') (add OR for each of the others descriptions)

1.5 If searchCriteria.categories is set then add where clause (comp_catalog.root_category_id in (*?,?, ...*)) (? - for each category)

2. if currentVersion then build query with join by comp_catalog.current_version=comp_versions.comp_vers_id, otherwise by comp_catalog.component_id=comp_versions.component_id

3. Build whole query using parts made in 2 and 3.

4. If not 'currentVersion', add where clause 'version=(**select max(version) from comp_versions where component_id=A.component_id**)'

5. Add special hibernate-mapping for the native query:

```
<class name="com.topcoder.catalog.service.AssetDTO" abstract="true">
        <id name="id" column="component_id"/>
        <property name="versionId" column="comp_vers_id"/>
        <property name="name" column="component_name"/>
        <property name="shortDescription" column="short_desc"/>
        <property name="versionText" column="version_text"/>
        <many-to-one name="rootCategory" column="root_category_id"/>
    </class>
```

Execute query with hint query.setHint("org.hibernate.readOnly", true) to indicate that it's a read-only entity.

1.4 Component Class Overview

CatalogServices:

This interface defines the contract for the catalog services. It provides various ways to get Components, Categories, Technologies and Phases and to modify/update Components from/to database.

It uses AssetDTO in order to provide Component information.

Implementations should be thread-safe.

CatalogServicesLocal :

This interface defines the contract for the catalog services for local use. It provides various ways to get Components, Categories, Technologies and Phases and to modify/update Components from/to database.

It uses AssetDTO in order to provide Component information.

Implementations should be thread-safe.

CatalogServicesRemote :

This interface defines the contract for the catalog services for remote use. It provides various ways to get Components, Categories, Technologies and Phases and to modify/update Components from/to database.

It uses AssetDTO in order to provide Component information.

Implementations should be thread-safe.

CatalogServicesImpl :

This stateless session bean realizes both local and remote interfaces for the catalog services. It provides various ways to get Components, Categories, Technologies and Phases and to modify/update Components from/to database.

It uses AssetDTO in order to provide Component information. This implementation uses entity manager to access persistence.

Class is thread-safe.

AssetDTO :

This class is a simple DTO that provide a representation of a component that is closer to business requirements than persistent entities are.

It provides getter and setter for each field.
Class is not thread-safe.

SearchCriteria :

This class represents the search criteria that is used when finding assets.

It should have at least on applicable for search property (non-null and not empty). All fields are initialized in the constructor.

Class is thread-safe.

1.5 Component Exception Definitions

EntityNotFoundException :

This exception is thrown by CatalogServices whenever an entity is not found in persistence.

Class is not thread-safe.

PersistenceException :

This exception is thrown by CatalogServices whenever an error occurs when interacting with persistence.

Class is not thread-safe.

1.6 Thread Safety

This component is not thread safe, because AssetDTO is not thread safe, nor are the entities defined in Catalog Entities. CatalogServicesImpl is thread-safe because it is a

stateless session bean and access to persistence is done within transactions.

AssetDTO should never be used in several threads like the other entities so it should not affect thread safety of services. In such case component is thread safe.

2. Environment Requirements

2.1 Environment

- At minimum, Java1.5 is required for compilation and executing test cases.

2.2 TopCoder Software Components

- **Catalog Entities 1.0:** defines the entities used in this component.
- **Base Exception 2.0:** used to define custom exception.

2.3 Third Party Components

None.

3. Installation and Configuration

3.1 Package Name

com.topcoder.catalog.service

3.2 Configuration Parameters

None.

3.3 Dependencies Configuration

None.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Follow demo.

4.3 Demo

Obtaining CatalogServices interfaces

Whether you need the local interface or the remote, you can obtain looking up the initial context:

public CatalogService getCatalogService() {

```
        try {
            // get object from JNDI
            Context context = new InitialContext();
            return (CatalogService) context.lookup("CatalogService/remote"); // or
"CatalogService/local" if CatalogService is in the same EAR file
```

```
        } catch (NamingException e) {
            throw new IllegalStateException(
              "Cannot lookup 'CatalogService'. Check the configuration (Jboss is running,
"
                  + "id_generator_ejb.jar and catalog_services_ejb.jar are deployed
successfully). " + "The nested exception is: " + e.getMessage(), e);
        }
}
```

Services can be used like normal methods, and it unnecessary to enumerate all of them.

Create a new asset

```
AssetDTO newAsset = new AssetDTO();
newAsset.setName("Catalog Services");
newAsset.setVersionText("1.0");
newAsset.setShortDescription("short");
newAsset.setDetailedDescription("detailed");
newAsset.setFunctionalDescription("functional");
// set the root category (categories are in the database
newAsset.setRootCategory(javaCategory);

// assign categories which this asset belongs to
newAsset.setCategories(Arrays.asList(ejb3Category));

newAsset.setTechnologies(Arrays.asList(
   java15Technology,
   informixTechnology
));
remote.createAsset(newAsset);
```

Create a new version

```
// retrieve asset with current version
AssetDTO asset = remote.getAssetById(assetId, true);
asset.setName("Catalog Service"); // update asset name
asset.setVersionText("1.1"); // sent new text version
asset.setVersionId(null); // reset version's ID
asset.setProductionDate(parseDate("2008/01/10")); // set new production date
remote.createVersion(asset);
```

Updating an asset works in a similar way, except that it doesn't create a new version, but just updates the version referred by DTO.

Find assets

```
List<AssetDTO> assets = remote.findAssets(
   new SearchCriteria(null, null, null, "catalog", null), true);
```

Suppose we have in catalog Catalog Service and Catalog Entities assets. The list will contain them both.

5. Future Enhancements

More services can be provided to meet additional needs.