[TopCoder]

Catalog Services Requirements Specification

1. Scope

1.1 Overview

The Catalog stores information about TopCoder components and applications.

This component will provide services for the Catalog. The services implementation will be pluggable, and it will provide an implementation based on Catalog Entities component, which defines entities and the O/R mapping. Since that component is under development, the file Entities.zuml describes the interface with this component.

Also, a DB diagram (DataModel.png) and DDL for creating the DB tables (Catalog.ddl) are provided together with this specification

1.2 Logic Requirements

1.2.1 Component Data Transfer Object

This component must define a Data Transfer Object in order to provide services that are closer to business requirements. This component will translate from Data Transfer Objects to persistent entities and vice-versa.

In order to synchronize DTO's with the database, the designer might want to store ids in the DTOs.

The Component DTO will store the following fields:

- Name
- Client id (not mandatory)
- Version text
- Short Description
- Detailed description
- Functional description
- Root Category
- List of categories
- List of technologies
- Production Date (not mandatory)
- Link (not mandatory)
- Forum (not mandatory)

1.2.2 Service Implementation

The component will provide a service implementation based on Catalog Entities.

The services will be exposed as EJB 3.0 methods.

The EJB(s) must provide both local and remote interfaces. Methods will use REQUIRED transaction attribute.

The following services must be provided:

1.2.2.1 Get Active Categories

Run named query `getActiveCategories`, that will return a list of `Category` objects.

1.2.2.2 Get Active Technologies

Run named query `getActiveTechnologies`, that will return a list of `Technology` objects.

1.2.2.3 Get Phases

Run named query `getAllPhases`, that will return a list of `Phase` objects.

1.2.2.4 Get component by id

Given a `component_id` value, retrieve the component with that id and return a Component DTO object.

When calling this method, it must be possible to choose whether to retrieve:

- the current version
- the latest version (Version entity with the highest version field)

This service must also communicate to the user whether the current version is the latest. This may involve adding a field in the DTO.

1.2.2.5 Get component by version id

Given a `comp_version_id` value, retrieve the component having that version and return a Component DTO object

1.2.2.6 Create new component (from DTO)

Given a Component DTO, it must translate it into persistent entities and save it.

In order to do 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 satusId to Status.NEW_POST
  - Leave all the comments fields in null

1.2.2.7 Create new component version (from DTO)

Given a Component DTO and component_id, it must update the entities related to the Component and create new entities related to CompVersion.

For example, if the component DTO has a short description of “manages catalog” and a version text “1.1”, then it must retrieve the component with the specified component_id, change its short description, and create a new CompVersion that will contain the version text “1.1” (and of course, fill the other fields with appropriate values).

1.2.2.8 Update component (from DTO)

Given a Component DTO the information for the component, version and related tables will be updated.

1.2.2.9 Find Components

Given some criteria, it must return a list of Component DTOs containing just this information:

- Name
- Version (text)
- Root Category
- Short Description

It must be indicated in the DTO that the information is not complete.

As in 1.2.2.4, the user must be able to choose whether to retrieve the current version or the last.

The search criteria must include at least:

- User id. If an user id is specified, the component must meet one or both of those conditions:
  - In Component entity, users list must contain a CompUser object with that user id
  - In Component entity, clients list must contain a CompClient object that contains in its users list that user id.
- Client id: in Component entity, clients list must contain a CompClient object with that client id.
- List of categories: given a list of category ids, the root category of the component must be one of those.
- Component name: allow for partial matches.
- Descriptions: it must search in all the description fields, allowing for partial matches

1.2.2.10 Assign user to component

Given a user id and component id, assign the user to the component.

1.2.2.11 Remove user from component

Given a user id and component id, remove the user from the component.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

An application can use this component to manage TopCoder catalog.

1.5 Future Component Direction

More services can be provided to meet additional needs.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

See file Entities.zuml

2.1.3 Environment Requirements

- Development language: Java 5.0
- Compile target: Java 5.0

2.1.4 Package Structure

com.topcoder.catalog.service

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

- EJB 3.0
- Hibernate

3.2.2 TopCoder Software Component Dependencies:

- Catalog Entities 1.0

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

JBoss

3.2.4 QA Environment:

- RedHat Enterprise Linux 4
- JBoss 4.2 GA
- Java 1.5
- Informix 10.00.UC 5

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in

the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.