TopCoder Project Service Requirements Specification

1. Scope

1.1 Overview

TopCoder Project Service component provides web service interface to allow user to perform CRUD operations on project. Project is used to group different multiple competitions. Competition can be one of standard TopCoder competitions (studio, component, testing, and assembly) or can be custom one. CRUD operations are performed depends on user permissions. Each project associated with user who creates it. Application has two level of permissions – administrative level and non-administrative level. Administrators can update, delete and retrieve any project; the other users can manage only associated with them. Note that is accessible only project id, name and description through the service API. The rest attributes of project have internal usage.

1.2 Logic Requirements

1.2.1 Provide CRUD operations on the project

Execution of CRUD operations depends on user permissions. There are two kinds of user – simple user and administrator. Simple user can manage only own projects while administrator can manage all projects. Check section 1.3 to understand how get the role of user.

1.2.1.1 Create project

Create new project in database. Create new Project entity where name and description should be gotten from given project data. User data should be selected from session context and create date should be set to current Date. Method has to return project data with set id.

1.2.1.2 Update project

Check authorization. User can update only projects associated with him/her. Administrator can update any project.

1.2.1.3 Delete project

Check authorization. User can delete projects associated with him/her. Administrator can delete any project. Note that it is impossible to delete project which have competitions.

1.2.1.4 Retrieve project by id

Check authorization. User can retrieve projects associated with him/her. Administrator can retrieve any project.

1.2.1.5 Retrieve project by user

This is administrative function as only administrator can get projects of some user.

1.2.1.6 Retrieve all projects

Check authorization. User can retrieve projects associated with him/her. Administrator can retrieve all projects.

1.2.2 Service implementation

The component will be built using EJB3 Stateless Session Bean web service endpoint implementation. The EJB must provide both local and remote interfaces. Methods will use REQUIRED transaction attribute.

1.2.3 Database access

Hibernate is used to provide access to database in EJB. Designer is responsible to create full Hibernate mapping for given database schema. Id of entities should be generated by using Hibernate id generation mechanism. All Hibernate mapping should be done by using XML instead of annotation, because Project class should be left as POJO. Note that Hibernate 3.2 is completely compatible with JPA so it should be used as standard JPA provider. All its methods should have security role access – “User”, except methods which gets documents for user. Its role should be “Administrator”

1.2.4 Service description

Created service should correspond given WSDL description. Any change or update of WSDL should be confirmed by PM. Custom exception should correspond fault messages defined in WSDL.

1.2.5 Logging

All logic methods of service should be logged. Methods of JBoss Login Module 1.1 should not be logged. The logging mechanism should be pluggable e.g. it should ability to disable logging.
Logging strategy:

- Entrance and exit of methods should be logged at the INFO level
- Exception should be logged at the ERROR level

1.3 Required Algorithms

- Get user principal by calling getCallerPrincipal() method on the EJBContext
- It should be returned UserProfilePrincipal instance defined in JBoss Login Module.
- User roles will be found in the “roles” attribute of the Profile as a map, where keys are Long instances - role id and values are strings – role names.
- User id will be found in the “id” attribute of the Principal

1.4 Example of the Software Usage

Project can be used to group different competition posted by some client. For example client can post at first some studio competition, then component competitions and at the end assembly competitions. All they can be grouped by one project.

1.5 Future Component Direction

None

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None

2.1.2 External Interfaces

See Interface diagram.

2.1.3 Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5, Java 1.6

2.1.4 Package Structure

com.topcoder.service.project

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Logger name

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

EJB 3.0

JPA 1.0

Hibernate 3.2 and higher

3.2.2 TopCoder Software Component Dependencies:

Base Exception 2.0
Logging Wrapper 2.0
JBoss Login Module 1.1 (use its class UserProfilePrincipal)

3.2.3 Third Party Component, Library, or Product Dependencies:

None

3.2.4 QA Environment:

- RedHat Linux 9
- Informix 10
- JBoss 4.2

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 the TopCoder UML Tool.