[TopCoder]

Resource Management Persistence 1.1 Requirements Specification

1. Scope

1.1 Overview

The Resource Management component provides resource management functionalities. This component provides a persistence layer to the Resource Management component. A resource can be optionally associated with a project, phase and submission. Each resource will have a role which identifies the resource's responsibilities for the associated scope. A set of resources can be created, updated or searched for a project. Also notifications can be assigned and unassigned to users.

The version 1.0 DAO implementation manages its own transaction. Every DAO call commits any changes made and closes the database connection.

Version 1.1 of this component must provide a DAO implementation capable of running inside a transaction that is managed externally, possibly by a J2EE container. This new version must not break backward compatibility and both new and old implementations must be interchangeable in terms of the database schema.

1.2 Logic Requirements

1.2.1 Resource Operations

This component will support the following operations.

1.2.1.1 Update Resource

An existing resource can be updated.
The operator needs to be provided for auditing purpose.

1.2.1.2 Get Resource

A resource can be retrieved by specified identifier.

1.2.1.3 All Resource Roles

There should be a way to retrieve all resource roles in the system.

1.2.1.4 Add Notifications

A notification entry will associate a user external ID with a project and notification type. The operation should be able to add multiple external ID's to the same project and notification type.
The operator needs to be provided for auditing purpose.

1.2.1.5 Remove Notifications

The operation should be able to remove multiple external ID's from the same project and notification type.
The operator needs to be provided for auditing purpose.

1.2.1.6 Get Notifications

The operation should be able to retrieve all external ID's associated with a project and notification type.

1.2.1.7 All Notification Types

There should be a way to retrieve all notification types in the system.

1.2.2 Informix Persistence

An Informix plug-in will be developed. The SQL scripts will be provided.

1.2.3 Unmanaged Transaction Persistence

For Version 1.1, another Informix plug-in must be added. This one must not manage its own transaction.

1.2.3.1 It must get the connection in every method and never store it in an instance variable.

1.2.3.2 It must never call close(), commit() or rollback() on the connection.

1.2.3.3 It must throw a RuntimeException (or subclass) in order to rollback the transaction.

1.2.3.4 It must comply with EJB development restrictions (http://java.sun.com/blueprints/qanda/ejb_tier/restrictions.html).

1.2.4 Backward compatibility

In terms of external use, version 1.1 must not break backward compatibility. The existing DAO implementation must behave exactly the same.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

A project management application can use the component as a model layer. Customer specific information can be stored as extended properties.
J2EE applications will be able to use the Resource Management component from an EJB Session Bean. Persistence must be configured to use the new persistence implementation. The J2EE application server will manage the JTA transaction.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Design must adhere to the ResourcePersistence interface definition from the Resource Management Component. Designer can choose to add more methods to the classes, but must keep the ones defined as a minimum.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.4

2.1.4 Package Structure

com.topcoder.management.resource.persistence.sql

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Database connection

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JDBC

3.2.2 TopCoder Software Component Dependencies:

- Resource Management
- Configuration Manager
- DB Connection Factory
- ID Generator

**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:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003
- Informix 10.0

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines.

3.3.1 Database Connections

Database connections must not be cached within the component. Connections should be created for each operation and closed afterwards.

3.3.2 Component Scalability

The component needs to be scalable. Running multiple instances in the same JVM or in multiple JVM's concurrently should not cause any problem.
Version 1.1 implementation of added plug-in should not use thread synchronization primitives to synchronize access with other enterprise bean instances (see EJB restrictions).

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.