Software Documentation : Project Management Persistence v1.2

This page last changed on Oct 30, 2010 by Immortal.

Overview

The component provides persistence capabilities to the Project Management component. It is a pluggable implementation of a DAO interface defined in the later component.

Updated Requirements

added requirements are in red
updated requirements are in blue

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 new 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.

Logic Requirements

Persistence

Persistence interfaces were updated. All methods will be implemented with direct queries to the database. See Studio_Replatforming_Online_Review_Module.tcuml for latest persistence interfaces. Based on Project+Management+v1.2 CRUD functionality for FileType, Prize and ProjectStudioSpecification methods can be decoupled into separate interface. This component must provide implementation for defined interfaces.

Terms of use

Current terms of use logic is present mainly in createRoleTermOfUse method. This method uses client_terms_mapping table to work with a query like this:
"select terms_of_use_id, resource_role_id from client_terms_mapping where client_project_id = ?";
This means that terms are configured for each client in client_terms_mapping table. To avoid frontend hardcoding for clients that do not have terms configured, a default terms table will be introduced.
The persistence methods that work with terms of use will look into this table if no entries are found in client_terms_mapping table.
The new table will have the following columns:
create table 'informix'.default_terms (
project_category_id INT not null,
project_type INT not null,
resource_role_id INT not null,
terms_of_use_id DECIMAL(10,0) not null,
cca BOOL)
The createRoleTermsOfUse() method (and possible other methods that work with terms of use - up to designer) will do the following:

- if no special terms apply, based on billing project, verify if projectCategoryId parameter exists in default_terms.project_category_id table column. If exists use the terms_of_use_id linked by such database entries according to method business logic
- if projectCategory does not exist, check by default_terms.project_type_id and use these terms resource_role_id.

No hardcoding will be done to create role terms.

Old methods and database updates

Other updates like create, update and retrieve operations also need to write/retrieve data from the updated database. Updating these methods to comply with the updated database is also in scope of this component.
Any updates required to the database model are in scope of the design. The designers should provide updated ERD for such changes and explain what the updates are about.

Project Operations

This component will support the following operations.

Create Project

A new project can be created. The identifier will be provided using the ID Generator. The operator needs to be provided for auditing purpose.

Update Project

An existing project can be updated. The operator needs to be provided for auditing purpose. An extra explanation should be provided.

Get Project

A project can be retrieved by specified identifier.

All Categories

There should be a way to retrieve all project categories in the system.

All Statuses

There should be a way to retrieve all project statuses in the system.

Informix Persistence

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

Unmanaged Transaction Persistence

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

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

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

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

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

Backward compatibility

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

Required Algorithms

No specific algorithms are required.

Example of the Software Usage

The Project Management component uses the component as the DAO for the Project Entity. Customer specific information can be stored as extended properties.
J2EE applications will be able to use the Project 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.

Future Component Direction

None.

Interface Requirements

Graphical User Interface Requirements

None.

External Interfaces

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

Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5

Package Structure

com.topcoder.management.project.persistence

Software Requirements

Administration Requirements

What elements of the application need to be configurable?

- Database connection

Technical Constraints

Are there particular frameworks or standards that are required?

JDBC

TopCoder Software Component Dependencies:

- Project 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.

Third Party Component, Library, or Product Dependencies:

None.

QA Environment:

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

Design Constraints

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

Database Connections

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

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).

Reuse of existing code

Both the existing DAO and the new one will share everything but the transaction/connection management code. The 1.1 version of this component must minimize code duplication.

Required Documentation

Design Documentation

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

Help / User Documentation

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