[TopCoder]

Software Documentation : Java Custom Terms Of Use 1.1

This page last changed on Nov 06, 2011 by ywu.

1. Scope

1.1 Overview

The component defines data model and functionality for managing Terms Of Use and their relations to users and contest roles. The existing Terms Of Use functionality is implemented in EJBs, which brings unnecessary complications to the applications using them. Additionally several enhancements are to be done to the existing functionality.

The existing implementation is provided as a reference. Designers are expected to study the code to get familiar with how things are done. Designers are not required to follow the same interfaces and are encouraged design a new consistent and easy to use API. On the other hand the amount of changes required for the applications to switch to the new component should be kept minimal (ideally just configuration update and method names changes). For example, no DB schema change should be required other than the ones explicitly mentioned in this document, the primary keys should be generated in the same way etc.

The 1.1 version enhances this component to add the support for terms dependencies. It also redesigns the management of agreeability.

1.2 Logic Requirements

1.2.1 DB schema

The existing DB schema is provided in the attachments. The tables of interest are:

1. terms_of_use. Contains name, text and properties of the terms of use.
2. terms_of_use_type. Contains types of the terms of use.
3. user_terms_of_use_xref. Links users to terms of use (i.e. if a user agreed to a term, there would be a corresponding entry in this table).
4. user_terms_of_use_ban_xref. Contains [user, terms of use] pairs that are forbidden to be added to user_terms_of_use_xref table.
5. project_role_terms_of_use_xref. Links resource roles for a specific project to a set of terms of use (e.g. for each project that requires Standard Submitter Terms Of Use for registering the project there would be a record in this table for the Submitter resource role).

There are two new fields to be added to the tables:

1. "group_ind" field should be added to the project_role_terms_of_use_xref table. This is an integer field that is used to group links by groups. The concept of groups will be explained later in this document. The field can take null values, records with null value are considered to be in the same group.
2. ~~"member_agreeable" field should be added to the terms_of_use table. This is a field of the same type as the "electronically_signable" field and indicates whether a user can agree to the terms of use (either electronically or by sending a hard copy).~~ The two fields 'member_agreeable' and 'electronically_signable' should be removed in the version 1.1.
3. There should be a new table named **terms_of_use_agreeability_type_lu**, which contains the types of TOUs agreeabilities. The schema will be *terms_of_agreeability_type_lu (terms_of_use_agreeability_type_id, name, description)*. The values (names) of the types will be something like 'Non-agreeable', 'Non-electronically-agreeable', 'Electronically-agreeable' etc. Then we need to update the **terms_of_use** table to remove the *electronically_signable*' and *member_agreeable* columns. Instead, we now refer to the new **terms_of_use_agreeability_type_lu** table by adding a new column terms_of_use_agreeability_type_id (FK).
4. There should be a new table named **terms_of_use_dependency**, which contains *[dependency_terms_of_use_id, dependent_terms_of_use_id]* pair to specify a dependency relationship between two TOUs.

This is the designer's responsibility to provide the updated DB schema.

1.2.2 Terms Of Use

The component will provide the entity classes for the Terms Of Use and its type, as well as CRUD functionality for them. The component will also provide method(s) to retrieve all terms of use in the system of a specific type, or simply all terms of use.

There should be a new entity class for the agreeability type.
The **TermsOfUse** class should have its *electronicallySignable* and *memberAgreeable* attributs removed, along with the getters/setters. The agreeability type should be added to the **TermsOfUse** entity. The **TermsOfUseDao** interface and its implementation should be updated accordingly.

1.2.3 User <-> Terms Of Use links

The component will provide functionality to manage the User <-> Terms Of Use links. At least the following methods should be provided:

1. Adding a new terms of use for a user. The method should ensure the user does not have a ban for this terms of use.
2. Removing a terms of use from a user.
3. Getting list of all terms of use associated with a specific user.
4. Getting list of all users associated with a specific terms of use.
5. Checking whether a user has the specific terms of use.
6. Checking whether a user has a ban for the specific terms of use.

The **UserTermsOfUseDao** interface and its implementation should be updated too for the changes in the DB schema.

1.2.4 Project Role <-> Terms Of Use links

The existing implementation supports operations such as adding a new Project <-> Terms Of Use link, removing a link, checking whether a link exists and retrieving all links for a specific project and a set of resource roles.

As per the existing implementation, for each [project, role] pair there's one set of terms of use a user needs to agree to in order to take the role for the project. For example, for a regular CCA project a reviewer usually needs to have two terms: Standard Reviewer terms of use and the CCA terms of use. Therefore a user needs to have both terms in order to signup for the review for this project. The system also won't let add anyone who does not have all of the required terms. In practice however, a client often wants to perform a review for his project along with the member reviewers. This is currently not supported by the system and we have to temporary add necessary terms for the client user. On the other hand all client users have specific Client terms of use. We would like the system to be clever enough to allow a user to be a reviewer if he has both the Standard Reviewer terms of use and the CCA terms ***or*** if he just has the Client terms of use. The member reviewers would still have to agree to the Standard Reviewer terms of use and the CCA in order to review the project while the client users would only need to have the Client terms of use (which they usually already have).

In general, a [project, role] pair can be associated with many groups of terms of use and a user can take the role for the project if and only if he has all of the terms from at least one group. There will be a new "group_ind" column in the project_role_terms_of_use_xref table that will be used to group the links with the same [project, role] pair. The column will be nullable with null being the default value, so that all existing records in the project_role_terms_of_use_xref table for the same [project, role] pair will be considered to belong to the same group.

At least the following methods should be provided:

1. Adding a new link for a specific [project, role] pair and a terms of use.
2. Removing a link for a specific [project, role] pair and a terms of use.
3. Search for the links for a specific [project, role] pair. The result will be a number of lists of terms and within each list the terms will be ordered by the "sort_order" column ascending. ~~The method will also have a search parameter to indicate whether to include the non-member-agreeable terms.~~

~~If these terms are not to be included and some of the resulting lists have such terms then the entire lists should be omitted.~~ We will use the new agreeability entity to filter the terms.

4. Removing all links for a specific project.

The **ProjectTermsOfUseDao** interface and its implementation should be updated to search for TOUs by a given agreeability type, or by a list of agreeability types, instead of the 'includeNonMemberAgreeable' flag.

1.2.5. Terms of Use Dependency

The TOU dependency is introduced for the requirement that a TOU (the dependent) may be required to be accessible to a user if and only if another TOU or a list of TOUs (the dependency) have been agreed by the user. So when users signs up for a project, we need to first present the list of terms that they need to agree to, and the terms that they has access to based on what terms they have agreed to. After they agree to these accessible terms, they are presented with the next dependent terms, and so on. A TOU may have zero, one or many dependencies.
The existing interfaces should be updated to support the creation, update, deletion and retrieval of the dependency relationships. At least the following functions should be provided:

1. Create a dependency relationship between two TOUs. Please make sure there is no dependency loop.
2. Retrieve all the dependencies of a TOU.
3. Retrieve all the dependents of a TOU.
4. Delete a dependency relationship between two given TOUs.
5. Delete all the dependency relationship for a TOU as either dependency or dependent.

1.2.6 Error handling

The CRUD operations defined above should properly define the strategy for handling errors (e.g. when adding a record which already exists, or deleting nonexistent record). It can be either return codes or exceptions, the decision is up to the designer.

1.2.7 Logging

The component should log all activity and errors using the Logging Wrapper component.

1.2.8 Thread-Safety

The component is required to be thread-safe because it will be called from multiple threads.

1.2.9 DB connections

Unlike the existing implementation the component should utilize the DB Connection Factory component for getting the DB connections.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

The component will be used by various applications for managing the terms of use and their relations to users and contest roles.

1.6 Future Component Direction

Any enhancement needs to be approved either in forum or in email with managers to eliminate over-complicating the component with useless functions.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None, only API interface will be provided.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5, Java 1.6

2.1.4 Package Structure

com.cronos.termsofuse

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:

- Base Exception 2.0
- Configuration API 1.0
- Configuration Persistence 1.0.2
- DB Connection Factory 1.1.0
- ID Generator 3.0.0
- Logging Wrapper 1.2.0

**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:

Any third party library needs to be approved.

3.2.4 QA Environment:

- Java 1.5
- RedHat Linux 4
- Windows 2000
- Windows 2003

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
- Updated DB schema

3.4.2 Help / User Documentation

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