[TopCoder]

Back End Changes for User Profile and Audit Requirements Specification

1.Scope

1.1Overview

The TopCoder registration process to become a member of TopCoder is going to be redesigned. The main goal is to allow for a minimal set of required info that the user needs to enter to sign up (handle, email, and password).

The existing TC web registration process will ask a new user to enter a lot of data when he/she tries to register to TC web-site. The registration process takes a lot of time, so many users can simply break the registration process and leave non-registered

Thus there is a need of an easy to use and user friendly web-application for supporting registration on the new users and management of registered user profiles at TC web-site.

The main goal of this application is to simplify registration of the new users on TC web-site by minimizing the count of mandatory data fields for new account registration, and to improve usability of user profile management for the registered users. Any additional profile information will be requested from the user by the system as it is needed. For example, if a user registers to compete in an assembly contest the site would prompt them for any required info that they have not yet entered.

This component is responsible for updating the DAOs and EJB due to the changes to the entities.

1.2Logic Requirements

1.2.1Scope

This component is responsible for implementing the changes specified on the “TC Registration Process - Back End Class Diagram”.

1.2.2User, UserProfile and AuditRecord Entities

The User entity is broken into two entities, some properties are kept in the User entity, and the other properties are moved in the new UserProfile entity.

Note that getters and setters for the moved properties in the User entity won’t be removed, they will now manage the inner userProfile instance variable instead. By keeping them, we can minimize the needed changes in the existing application code as much as possible.

A new AuditRecord entity is also added for auditing purpose.

All entities should follow the Java Bean convention.

The related database tables and Hibernate mapping files should also be updated (or added)

accordingly.

Note that the association between user and profile db tables will be 1-to-1, so the profile's primary key is user_id, which is also a foreign key to the user's user_id primary key column.

And in User entity, the userProfile property should be loaded eagerly and the cascade attribute of this association should be cascade-all.

1.2.3UserDAO

Due to the changes to the entities above, the UserDAO needs to be updated as below:

find(handle, firstName, lastName, email) – its implementation needs to query the UserProfile now as the firstName/lastName are moved into UserProfile

find(email) - new method to query User by the email.

canChangeHandle – return true if the handle can be changed, return false otherwise.

Note that the handle can be changed only if the user is not involved in any recorded activity on TC website. It would be a very complex query, please ask in the forum for more clarification.

1.2.4AuditDAO

The AuditDAO interface is newly added for the AuditRecord entity, and it has the following methods:

audit - insert the AuditRecord into the corresponding database table.

hasOperation – check is there an AuditRecord with its operationType contains the given value.

A hibernate implementation should be provided to AuditDAO interface.

1.2.5User EJB

Due to the changes to the entities above, the User EJB needs to be updated as below:
The setFirstName, setMiddleName, setLastName, getFirstName, getLastName, and getMiddleName methods should now manage the firstName, lastName, and middleName in the user_profile db table.

1.2.6Guideline

The threading, persistence, transaction guildeline in ADS 1.3.1 – 1.3.3 should be followed.

And note that the DAOs/EJB should be able to be created using Spring.

1.3Required Algorithms

None

1.4Example of the Software Usage

This component provides changes to entities and related DAOs/EJBs.

1.5Future Component Direction

None

2.Interface Requirements

2.1.1Graphical User Interface Requirements

None

2.1.2External Interfaces

Refer to the provided architecture TCUML.

2.1.3Environment Requirements

Development language: Java
Compile Target: Java 1.6

2.1.4Package

For Entities:
com.topcoder.web.common.model

For DAO:
com.topcoder.web.common.dao
com.topcoder.web.common.dao.hibernate

For EJB:
com.topcoder.web.ejb.user

3.Software Requirements

3.1Administration Requirements

3.1.1What elements of the application need to be configurable?

None

3.2Technical Constraints

3.2.1Are there particular frameworks or standards that are required?

EJB
Hibernate

3.2.2TopCoder Software Component Dependencies:

None

3.2.3Third Party Component, Library, or Product Dependencies:

Hibernate 3.6
Spring 2.5.6 – note that the DAOs should be able to be created by Spring

3.2.4QA Environment:

J2EE 1.5
JAVA 1.6.0_04
Apache 2.2.4
jira 4.1.2
wiki 2.7
Jive Professional 4.2.5
Red Hat 4.1.2-46
Informix 11.5
JBoss-4.0.4.GA
HTML/CSS
IE6 - latest version
FireFox - latest version
Safari - latest version
Chrome - latest version
JavaScript (jQuery 1.4.4)
HttpUnit
CVS

3.3Design 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.

3.4Required Documentation

3.4.1Design Documentation

Use-Case Diagram
Class Diagram
Sequence Diagram
Component Specification

3.4.2Help / User Documentation

XML documentation must provide sufficient information regarding component design and usage.