[TopCoder]

UserProfile and Audit Back End 1.0 Component Specification

1. Design

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 website. The registration process takes a lot of time, so many users can simply break the registration process and leave nonregistered

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 website.

The main goal of this application is to simplify registration of the new users on TC website 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.

This component provides User, UserProfile, AuditRecord entities, UserDAO and AuditDAO together with their Hibernate implementations, EJB UserBean with its remote interface.

1.1 Design Patterns

Strategy pattern – this component defines UserDAO and AuditDAO interfaces together with their implementations that can be possibly used in some external strategy context.

DAO/DTO pattern – User interface, UserBean and UserDAOHibernate are DAOs for User and UserProfile DTOs; AuditDAO is a DAO for AuditRecord DTO.

Delegate pattern – most of User methods delegate execution to the namesake methods of the underlying UserProfile instance.

1.2 Industry Standards

EJB, JDBC, HQL, SQL, JavaBeans

1.3 Required Algorithms

1.3.1 Changes to DDL

1.3.1.1 Update user table

All fields corresponding to properties moved from User entity to UserProfile entity must be removed from the user table (changes are marked with red):

```
create table 'informix'.user (
    user_id DECIMAL(10,0) not null,
    first_name VARCHAR(64),
    last_name VARCHAR(64),
    create_date DATETIME YEAR TO FRACTION default CURRENT YEAR TO FRACTION,
    modify_date DATETIME YEAR TO FRACTION default CURRENT YEAR TO FRACTION,
    handle VARCHAR(50) not null,
    last_login DATETIME YEAR TO FRACTION,
    status VARCHAR(3) not null,
    password VARCHAR(30),
    activation_code VARCHAR(32),
    middle_name VARCHAR(64),
    handle_lower VARCHAR(50),
    timezone_id DECIMAL(5,0),
    last_site_hit_date DATETIME YEAR TO FRACTION
);
```

1.3.1.2 Add user_profile table

A new table that corresponds to UserProfile entity must be added to the database:

```
create table 'informix'.user_profile (
    user_id DECIMAL(10,0) not null,
    first_name VARCHAR(64),
    last_name VARCHAR(64),
    middle_name VARCHAR(64),
    timezone_id DECIMAL(5,0),
);
```

Additionally user_id field in this table must be primary key and at the same time a foreign key to the user table.

1.3.1.3 Add audit_record table

A new table that corresponds to AuditRecord entity must be added to the database:

```
create table 'informix'.audit_record (
   id SERIAL8 NOT NULL,
   operation_type VARCHAR(100) NOT NULL,
   handle VARCHAR(20) NOT NULL,
   ip_address VARCHAR(20) NOT NULL,
   previous_value VARCHAR(4096),
   new_value VARCHAR(4096),
   timestamp DATETIME YEAR TO FRACTION NOT NULL
);
```

1.3.2 Hibernate mapping

It's required to update mapping file for User entity and provide two new mapping files for UserProfile and AuditRecord entities.

1.3.2.1 User.hbm.xml

Changes to the existing file are marked with red.

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.topcoder.web.common.model">

  <class name="User" table="user">
    <id name="id" column="user_id">
      <generator class="com.topcoder.web.common.model.IdGenerator">
        <param name="sequence_name">USER_SEQ</param>
      </generator>
    </id>
    <property name="firstName" column="first_name" access="field"/>
    <property name="middleName" column="middle_name" access="field"/>
    <property name="lastName" column="last_name" access="field"/>
    <property name="handle" access="field"/>
    <property name="status" access="field"/>
    <property name="activationCode" column="activation_code" access="field"/>
    <many-to-one name="timeZone" column="timezone_id" class="TimeZone" access="field"
                 cascade="none"/>
    <one-to-one name="coder" class="Coder" cascade="save-update" access="field"
                fetch="select"/>
    <one-to-one name="contact" class="Contact" cascade="save-update" access="field"
                fetch="select"/>
    <one-to-one name="secretQuestion" class="SecretQuestion" cascade="save-update"
                access="field" fetch="select"/>
    <one-to-one name="professor" class="com.topcoder.web.common.model.educ.Professor"
                cascade="save-update"
                access="field" fetch="select"/>
    <one-to-one name="userSecurityKey" class="UserSecurityKey" cascade="save-update"
                access="field" fetch="select"/>

    <set name="securityGroups" table="user_group_xref" cascade="none" inverse="true"
         access="field">
```

```
      <key column="login_id" not-null="true"/>
      <one-to-many class="UserGroup"/>
    </set>
    <set name="emailAddresses" cascade="save-update" inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="Email"/>
    </set>
    <set name="phoneNumbers" cascade="save-update" inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="Phone"/>
    </set>
    <set name="addresses" table="user_address_xref" cascade="save-update"
         access="field">
      <key column="user_id" not-null="true"/>
      <many-to-many column="address_id" class="Address"/>
    </set>
    <set name="notifications" table="user_notify_xref" cascade="save-update"
         access="field">
      <key column="user_id" not-null="true"/>
      <many-to-many column="notify_id" class="Notification"/>
    </set>
    <set name="userPreferences" table="user_preference" inverse="true"
         cascade="save-update" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="UserPreference"/>
    </set>
    <set name="demographicResponses" cascade="all-delete-orphan" inverse="true"
         access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="DemographicResponse"/>
    </set>
    <set name="terms" table="user_terms_of_use_xref" cascade="none" access="field">
      <key column="user_id" not-null="true"/>
      <many-to-many column="terms_of_use_id" class="TermsOfUse"/>
    </set>
    <set name="eventRegistrations" table="event_registration" cascade="save-update"
         inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="EventRegistration"/>
    </set>
    <set name="responses" table="response" cascade="save-update" inverse="true"
         access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="Response"/>
    </set>
    <set name="ballots" cascade="save-update" inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.voting.RankBallot"/>
    </set>
    <set name="compPrizes" table="tcs_catalog:user_contest_prize" cascade="save-update"
         inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.model.comp.UserContestPrize"/>
    </set>

    <set name="schools" cascade="none" inverse="true">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.model.UserSchool"/>
    </set>
    <set name="createdSchools" cascade="save-update" inverse="true">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.model.School"/>
    </set>

    <one-to-one name="userProfile" class="UserProfile" cascade="all"
                access="field" property-ref="user" fetch="join" lazy="false"/>
  </class>

</hibernate-mapping>
```

1.3.2.2 UserProfile.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.topcoder.web.common.model">

  <class name="UserProfile" table="user_profile">
    <id name="id" column="user_id">
      <generator class="foreign">
        <param name="property">user</param>
      </generator>
    </id>
    <one-to-one name="user" class="User" access="field"/>
    <property name="firstName" column="first_name" access="field"/>
    <property name="middleName" column="middle_name" access="field"/>
    <property name="lastName" column="last_name" access="field"/>
    <many-to-one name="timeZone" column="timezone_id" class="TimeZone" access="field"
                 cascade="none"/>
    <one-to-one name="coder" class="Coder" cascade="save-update" access="field"
                fetch="select"/>
    <one-to-one name="contact" class="Contact" cascade="save-update" access="field"
                fetch="select"/>
    <one-to-one name="secretQuestion" class="SecretQuestion" cascade="save-update"
                access="field" fetch="select"/>
    <one-to-one name="professor" class="com.topcoder.web.common.model.educ.Professor"
                cascade="save-update"
                access="field" fetch="select"/>

    <set name="phoneNumbers" cascade="save-update" inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="Phone"/>
    </set>
    <set name="addresses" table="user_address_xref" cascade="save-update"
         access="field">
      <key column="user_id" not-null="true"/>
      <many-to-many column="address_id" class="Address"/>
    </set>
    <set name="notifications" table="user_notify_xref" cascade="save-update"
         access="field">
      <key column="user_id" not-null="true"/>
      <many-to-many column="notify_id" class="Notification"/>
    </set>
    <set name="userPreferences" table="user_preference" inverse="true"
         cascade="save-update" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="UserPreference"/>
    </set>
    <set name="demographicResponses" cascade="all-delete-orphan" inverse="true"
         access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="DemographicResponse"/>
    </set>
    <set name="terms" table="user_terms_of_use_xref" cascade="none" access="field">
      <key column="user_id" not-null="true"/>
      <many-to-many column="terms_of_use_id" class="TermsOfUse"/>
    </set>
    <set name="eventRegistrations" table="event_registration" cascade="save-update"
         inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="EventRegistration"/>
    </set>
    <set name="responses" table="response" cascade="save-update" inverse="true"
         access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="Response"/>
    </set>
    <set name="ballots" cascade="save-update" inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.voting.RankBallot"/>
```

```
    </set>
    <set name="compPrizes" table="tcs_catalog:user_contest_prize" cascade="save-update"
         inverse="true" access="field">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.model.comp.UserContestPrize"/>
    </set>

    <set name="schools" cascade="none" inverse="true">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.model.UserSchool"/>
    </set>
    <set name="createdSchools" cascade="save-update" inverse="true">
      <key column="user_id" not-null="true"/>
      <one-to-many class="com.topcoder.web.common.model.School"/>
    </set>
  </class>

</hibernate-mapping>
```

1.3.2.3 AuditRecord.hbm.xml

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.topcoder.web.common.model">
  <class name="AuditRecord" table="audit_record">
    <id name="id" column="id">
      <generator class="native"/>
    </id>
    <property name="operationType" column="operation_type" access="field"/>
    <property name="handle" access="field"/>
    <property name="ipAddress" column="ip_address" access="field"/>
    <property name="previousValue" column="previous_value" access="field"/>
    <property name="newValue" column="new_value" access="field"/>
    <property name="timestamp" access="field"/>
  </class>
</hibernate-mapping>
```

1.4 Component Class Overview

AuditDAO [interface]

This interface defines methods for persisting audit records and checking whether some operation was previously performed by specific user.

UserDAO [interface]

This interface represents a DAO for User entities. It defines methods for searching user by ID, handle, email, first name and/or last name, and also methods for saving user to persistence and checking whether user handle can be changed.

AuditDAOHibernate

This class is an implementation of AuditDAO that manages AuditRecord entities in persistence using Hibernate.

UserDAOHibernate

This class is an implementation of UserDAO that manages User entities in persistence using Hibernate. Currently change of handle is not allowed for all users, thus canChangeHandle() always returns false.

AuditRecord

This class is a container for information about a single audit record. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

User

This class is a container for information about a single user. It is a POJO with some additional method that simplify access to the stored properties. Note that most of the methods simply delegate to the namesake method of the inner UserProfile instance.

UserProfile

This class is a container for user profile information. It is a POJO with some additional method that simplify access to the stored properties.

Note: initially all fields and methods of this class can be copied from the existing source code of the User class.

User [interface]

This is a remote interface to the UserBean stateless session bean.

UserBean

This class represents a stateless EJB bean that performs managing of user data in persistence.

1.5 Component Exception Definitions

None

1.6 Thread Safety

This component is thread safe when used properly.

Implementations of UserDAO and AuditDAO are required to be thread safe when entities passed to them are used by the caller in thread safe manner.

UserDAOHibernate and AuditDAOHibernate are immutable and thread safe when entities passed to them are used by the caller in thread safe manner.

Implementations of User EJB interface must be thread safe.

UserBean is immutable and thread safe (not taking into account the SessionContext field in the base class that is actually never used).

AuditRecord, User and UserProfile are mutable and not thread safe DTOs.

Transaction management is not performed in this component. It's assumed that transactions will be managed externally using Spring declarative transactions approach.

2. Environment Requirements

2.1 Environment

Development language: Java 1.6
Compile Target: Java 1.6
QA Environment: J2EE 1.5, JAVA 1.6.0_04, Apache 2.2.4, Red Hat 4.1.2-46, Informix 11.5, JBoss-4.0.4.GA, Spring 2.5.6

2.2 TopCoder Software Components

ID Generator 3.0.2 – defines IDGenerationException used in this component.

NOTE: The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

Hibernate 3.6 (http://www.hibernate.org/)

NOTE: The default location for 3rd party packages is ../lib relative to this component installation. Setting the ext_libdir property in topcoder_global.properties will overwrite this default location.

3. Installation and Configuration

3.1 Package Name

com.topcoder.web.common.dao
com.topcoder.web.common.dao.hibernate
com.topcoder.web.common.model
com.topcoder.web.ejb.user

3.2 Configuration Parameters

None

3.3 Dependencies Configuration

Please check the existing source code to see how dependencies are configured.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Please see the demo.

4.3 Demo

4.3.1 Usage of UserBean EJB

```
// Get user EJB
Context context = new InitialContext();
com.topcoder.web.ejb.user.User userBean =
    (com.topcoder.web.ejb.user.User) context.lookup("java:comp/env/ejb/UserBean");

// Create new user "test_user"
long userId = userBean.createNewUser("test_user", 'U', "TEST_DATA_SOURCE",
"TEST_DATA_SOURCE");

// Set first name of the user
userBean.setFirstName(userId, "John", "TEST_DATA_SOURCE");

// Set middle name of the user
userBean.setMiddleName(userId, "A", "TEST_DATA_SOURCE");

// Set last name of the user
userBean.setLastName(userId, "Smith", "TEST_DATA_SOURCE");

// Set status of the user to active
userBean.setStatus(userId, 'A', "TEST_DATA_SOURCE");

// Set activation code of the user
userBean.setActivationCode(userId, "12345", "TEST_DATA_SOURCE");

// Set password of the user
userBean.setPassword(userId, "test_pass", "TEST_DATA_SOURCE");

// Change handle of the user
userBean.setHandle(userId, "first_user", "TEST_DATA_SOURCE");

// Retrieve the first name of the user
String firstName = userBean.getFirstName(userId, "TEST_DATA_SOURCE");
// firstName must be "John"

// Retrieve the middle name of the user
String middleName = userBean.getMiddleName(userId, "TEST_DATA_SOURCE");
// middleName must be "A"
```

```
// Retrieve the last name of the user
String lastName = userBean.getLastName(userId, "TEST_DATA_SOURCE");
// lastName must be "Smith"

// Retrieve the status of the user
char status = userBean.getStatus(userId, "TEST_DATA_SOURCE");
// status must be 'A'

// Retrieve the activation code of the user
String activationCode = userBean.getActivationCode(userId, "TEST_DATA_SOURCE");
// activationCode must be "12345"

// Retrieve the handle of the user
String handle = userBean.getHandle(userId, "TEST_DATA_SOURCE");
// handle must be "first_user"

// Retrieve the password of the user
String password = userBean.getPassword(userId, "TEST_DATA_SOURCE");
// password must be "test_pass"

// Check if user with the specified ID exists
boolean exists = userBean.userExists(userId, "TEST_DATA_SOURCE");
// exists must be true

// Check if user with the specified handle exists
exists = userBean.userExists("test_user", "TEST_DATA_SOURCE");
// exists must be false

exists = userBean.userExists("first_user", "TEST_DATA_SOURCE");
// exists must be true
```

4.3.2 *Usage of UserDAOHibernate*

```
// It's assumed that "first_user" with ID=1 is created using the code from 4.3.1

// Create an instance of UserDAOHibernate
UserDAO userDAO = new UserDAOHibernate();

// Create an instance of UserDAOHibernate using Hibernate session
Session session = ...
userDAO = new UserDAOHibernate(session);

// Find user by ID
User user = userDAO.find(1);
// user.getHandle() must be "first_user"
// user.getId() must be 1
// user.getUserProfile().getId() must be 1
// user.getUserProfile().getFirstName() must be "John"

// Find active user by handle, case insensitive
user = userDAO.find("first_USER", true, true);
// user.getHandle() must be "first_user"
// user.getId() must be 1
// user.getUserProfile().getId() must be 1
// user.getUserProfile().getFirstName() must be "John"

// Specify email address for the user and save it
Email email = new Email();
email.setAddress("john_smith@topcoder.com");
email.setPrimary(true);
email.setEmailTypeId(Email.TYPE_ID_PRIMARY);
email.setStatusId(Email.STATUS_ID_ACTIVE);
user.addEmailAddress(email);
userDAO.saveOrUpdate(user);

// Find user by the primary email address
user = userDAO.find("john_smith@topcoder.com");
// user.getHandle() must be "first_user"
// user.getId() must be 1
// user.getUserProfile().getId() must be 1
// user.getUserProfile().getFirstName() must be "John"
```

```
// Check if user handle can be changed
boolean canBeChanged = userDAO.canChangeHandle("first_user");
// canBeChanged must be false
```

4.3.3 *Usage of AuditDAOHibernate*

```
// Create an instance of AuditDAOHibernate
AuditDAO auditDAO = new AuditDAOHibernate();

// Create an instance of AuditDAOHibernate using Hibernate session
Session session = ...
auditDAO = new AuditDAOHibernate(session);

// Check if user "first_user" has performed login operation
boolean performedLogin = auditDAO.hasOperation("first_user", "login");
// performedLogin must be false

// Audit user login operation
AuditRecord record = new AuditRecord();
record.setNew(true):
record.setHandle("first_user");
record.setIpAddress("111.222.111.222");
record.setOperationType("login");
record.setTimestamp(new Date());
auditDAO.audit(record);

// Check if user "first_user" has performed login operation
performedLogin = auditDAO.hasOperation("first_user", "login");
// performedLogin must be true
```

5. Future Enhancements

None