Copilot Pool and Profiles DAO 1.0 Component Specification

1. Design

TopCoder is revamping the current copilot selection process to build a more structural one. One big change is that TopCoder brings in the idea of the copilot pool which is consisted of authorized pre-approved members; only members in the copilot pool will have access to the new copilot opportunities (the new copilot opportunities will be posted as copilot selection contests by client/PM). And each copilot will have copilot profile pages which show the statistics of his past copilot performance, such as number of projects managed, number of contests managed, the number of current working on projects and past project histories etc. The copilot profile will be used by the client/PM as a reference when choosing a copilot.

This component provides the DAO layer for the copilot pool and profile module.

1.1 Design Patterns

Strategy pattern – DAO interfaces together with their implementation provided in this component can be possibly used in some external strategy context.

DAO/DTO pattern – GenericDAO is a DAO for DTOs that are subclasses of IdentifiableEntity; CopilotProjectPlanDAO is a DAO for CopilotProjectPlan DTO; CopilotProfileDAO is a DAO for CopilotProfile DTO; CopilotProjectDAO is a DAO for CopilotProject DTO; LookupService is a DAO for CopilotProfileStatus, CopilotProjectStatus and CopilotType DTOs.

Template method pattern – GenericDAOImpl provides template methods create() and update() that use updateAuditTimestamp() method which is overridden in subclasses; here only subclasses know how to update audit timestamps in the aggregated entities.

1.2 Industry Standards

SQL, HQL, IoC

1.3 Required Algorithms

1.3.1 Logging

This component must perform logging in all public methods of GenericDAOImpl<T>, CopilotProfileDAOImpl, CopilotProjectDAOImpl, CopilotProjectPlanDAOImpl, LookupDAOImpl and UtilityDAOImpl (except getters, setters and checkInit() method).

All information must be logged using Log instance obtained by calling BaseDAO#getLog() method. If this getter returns null, then logging is not required to be performed.

In all mentioned methods method entrance with input arguments, method exit with return value and call duration time must be logged at DEBUG level. It's not required to log method exit when method throws an exception.

All errors (for all thrown exceptions) must be logged at ERROR level with exception message and stack trace.

1.3.2 Hibernate entity mapping

The following Hibernate entity mapping should be used in this component:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
        "-//Hibernate/Hibernate Mapping DTD 2.0//EN"
        "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">
<hibernate-mapping package="com.topcoder.direct.services.copilot.model">


    <class name="CopilotProfile" table="copilot_profile">
```

```
        <id name="id" column="copilot_profile_id" type="long">
            <generator class="increment"/>
        </id>
        <property name="createDate" column="create_date" type="timestamp" update="false"/>
        <property name="createUser" column="create_user" type="string" update="false"/>
        <property name="modifyDate" column="modify_date" type="timestamp"/>
        <property name="modifyUser" column="modify_user" type="string"/>
        <property name="userId" column="user_id" type="long"/>
        <many-to-one name="status" class="CopilotProfileStatus" column="
copilot_profile_status_id" cascade="none"/>
        <property name="suspensionCount" column="suspension_count" type="integer"/>
        <property name="reliability" column="reliability" type="float"/>
        <property name="activationDate" column="activation_date" type="timestamp"/>
        <property name="showCopilotEarnings" column="show_copilot_earnings" type="yes_no"/>
        <set name="profileInfos" lazy="false" cascade="all-delete-orphan">
            <key column="copilot_profile_id" not-null="true"/>
            <one-to-many class="CopilotProfileInfo"/>
        </set>
    </class>

    <class name="CopilotProfileStatus" table="copilot_profile_status">
        <id name="id" column="copilot_profile_status_id" type="long">
            <generator class="increment"/>
        </id>
        <property name="createDate" column="create_date" type="timestamp" update="false"/>
        <property name="createUser" column="create_user" type="string" update="false"/>
        <property name="modifyDate" column="modify_date" type="timestamp"/>
        <property name="modifyUser" column="modify_user" type="string"/>
        <property name="name" column="name" type="string"/>
    </class>

    <class name="CopilotProfileInfo" table="copilot_profile_info">
        <id name="id" column="copilot_profile_info_id" type="long">
            <generator class="increment"/>
        </id>
        <property name="createDate" column="create_date" type="timestamp" update="false"/>
        <property name="createUser" column="create_user" type="string" update="false"/>
        <property name="modifyDate" column="modify_date" type="timestamp"/>
        <property name="modifyUser" column="modify_user" type="string"/>
        <many-to-one name="infoType" class="CopilotProfileInfoType"
column="copilot_profile_info_type_id"
                     cascade="none"/>
        <property name="copilotProfileId" column="copilot_profile_id" type="long"
insert="false" update="false"/>
```

```
        <property name="value" column="value" type="string"/>
    </class>

    <class name="CopilotProfileInfoType" table="copilot_profile_info_type">
        <id name="id" column="copilot_profile_info_type_id" type="long">
            <generator class="increment"/>
        </id>
        <property name="createDate" column="create_date" type="timestamp" update="false"/>
        <property name="createUser" column="create_user" type="string" update="false"/>
        <property name="modifyDate" column="modify_date" type="timestamp"/>
        <property name="modifyUser" column="modify_user" type="string"/>
        <property name="name" column="name" type="string"/>
    </class>

    <class name="CopilotProject" table="copilot_project">
        <id name="id" column="copilot_project_id" type="long">
            <generator class="increment"/>
        </id>
        <property name="createDate" column="create_date" type="timestamp" update="false"/>
        <property name="createUser" column="create_user" type="string" update="false"/>
        <property name="modifyDate" column="modify_date" type="timestamp"/>
        <property name="modifyUser" column="modify_user" type="string"/>
        <property name="copilotProfileId" column="copilot_profile_id" type="long"/>
        <property name="name" column="name" type="string"/>
        <property name="tcDirectProjectId" column="tc_direct_project_id" type="long"/>
        <many-to-one name="copilotType" class="CopilotType" column="copilot_type_id"
cascade="none"/>
        <many-to-one name="status" class="CopilotProjectStatus"
column="copilot_project_status_id" cascade="none"/>
        <property name="customerFeedback" column="customer_feedback" type="string"/>
        <property name="customerRating" column="customer_rating" type="float"/>
        <property name="pmFeedback" column="pm_feedback" type="string"/>
        <property name="pmRating" column="pm_rating" type="float"/>
        <property name="privateProject" column="private_project" type="yes_no"/>
        <set name="projectInfos" lazy="false" cascade="all-delete-orphan">
            <key column="copilot_project_id" not-null="true"/>
            <one-to-many class="CopilotProjectInfo"/>
        </set>
        <property name="completionDate" column="completion_date" type="timestamp"/>
    </class>

    <class name="CopilotType" table="copilot_type">
        <id name="id" column="copilot_type_id" type="long">
            <generator class="increment"/>
```

```
    </id>
    <property name="createDate" column="create_date" type="timestamp" update="false"/>
    <property name="createUser" column="create_user" type="string" update="false"/>
    <property name="modifyDate" column="modify_date" type="timestamp"/>
    <property name="modifyUser" column="modify_user" type="string"/>
    <property name="name" column="name" type="string"/>
</class>

<class name="CopilotProjectStatus" table="copilot_project_status">
    <id name="id" column="copilot_project_status_id" type="long">
        <generator class="increment"/>
    </id>
    <property name="createDate" column="create_date" type="timestamp" update="false"/>
    <property name="createUser" column="create_user" type="string" update="false"/>
    <property name="modifyDate" column="modify_date" type="timestamp"/>
    <property name="modifyUser" column="modify_user" type="string"/>
    <property name="name" column="name" type="string"/>
</class>

<class name="CopilotProjectPlan" table="copilot_project_plan">
    <id name="id" column="copilot_project_plan_id" type="long">
        <generator class="increment"/>
    </id>
    <property name="createDate" column="create_date" type="timestamp" update="false"/>
    <property name="createUser" column="create_user" type="string" update="false"/>
    <property name="modifyDate" column="modify_date" type="timestamp"/>
    <property name="modifyUser" column="modify_user" type="string"/>
    <property name="copilotProjectId" column="copilot_project_id" type="long"/>
    <property name="plannedDuration" column="planned_duration" type="integer"/>
    <property name="plannedBugRaces" column="planned_bug_races" type="integer"/>
    <set name="plannedContests" lazy="false" cascade="all-delete-orphan">
        <key column="copilot_project_plan_id" not-null="true"/>
        <one-to-many class="PlannedContest"/>
    </set>
</class>

<class name="PlannedContest" table="planned_contest">
    <id name="id" column="planned_contest_id" type="long">
        <generator class="increment"/>
    </id>
    <property name="createDate" column="create_date" type="timestamp" update="false"/>
    <property name="createUser" column="create_user" type="string" update="false"/>
    <property name="modifyDate" column="modify_date" type="timestamp"/>
```

```
        <property name="modifyUser" column="modify_user" type="string"/>
        <property name="name" column="name" type="string"/>
        <property name="description" column="description" type="string"/>
        <property name="projectCategoryId" column="project_category_id" type="long"/>
        <property name="copilotProjectPlanId" column="copilot_project_plan_id" type="long"
insert="false"
                  update="false"/>
        <property name="startDate" column="start_date" type="timestamp"/>
        <property name="endDate" column="end_date" type="timestamp"/>
    </class>


    <class name="CopilotProjectInfo" table="copilot_project_info">
        <id name="id" column="copilot_project_info_id" type="long">
            <generator class="increment"/>
        </id>
        <property name="createDate" column="create_date" type="timestamp" update="false"/>
        <property name="createUser" column="create_user" type="string" update="false"/>
        <property name="modifyDate" column="modify_date" type="timestamp"/>
        <property name="modifyUser" column="modify_user" type="string"/>
        <many-to-one name="infoType" class="CopilotProjectInfoType"
column="copilot_project_info_type_id"
                     cascade="none"/>
        <property name="copilotProjectId" column="copilot_project_id" type="long"
insert="false" update="false"/>
        <property name="value" column="value" type="string"/>
    </class>


    <class name="CopilotProjectInfoType" table="copilot_project_info_type">
        <id name="id" column="copilot_project_info_type_id" type="long">
            <generator class="increment"/>
        </id>
        <property name="createDate" column="create_date" type="timestamp" update="false"/>
        <property name="createUser" column="create_user" type="string" update="false"/>
        <property name="modifyDate" column="modify_date" type="timestamp"/>
        <property name="modifyUser" column="modify_user" type="string"/>
        <property name="name" column="name" type="string"/>
    </class>


</hibernate-mapping>
```

1.4 Component Class Overview

BaseDAO [abstract]

This class is base for all DAO implementations provided in this component. It holds a Logging Wrapper logger and Hibernate session factory for TCS Catalog database to be used by subclasses. Also it provides a protected method for retrieving entities of specific type from persistence. It's assumed that subclasses of this class will be initialized using Spring IoC.

CopilotProfile

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

CopilotProfileDAO [interface]

This interface represents a copilot profile DAO. It extends GenericDAO<CopilotProfile> and provides an additional method for retrieving copilot profile by the given copilot user ID.

CopilotProfileDAOImpl

This class is an implementation of CopilotProfileDAO that uses Hibernate session to access entities in persistence. It extends GenericDAOImpl<CopilotProfile> that provides common functionality for CopilotProfileDAO, CopilotProjectDAO and CopilotProjectPlanDAO implementations provided in this component. This class uses Logging Wrapper logger to log errors and debug information.

CopilotProfileInfo

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

CopilotProfileInfoType

This class represents a "copilot profile info type" lookup entity. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

CopilotProfileStatus

This class represents a "copilot profile status" lookup entity. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

CopilotProject

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

CopilotProjectDAO [interface]

This interface represents a copilot project DAO. It extends GenericDAO<CopilotProject> and provides an additional method for retrieving a list of copilot projects for the specified copilot.

CopilotProjectDAOImpl

This class is an implementation of CopilotProjectDAO that uses Hibernate session to access entities in persistence. It extends GenericDAOImpl<CopilotProject> that provides common functionality for CopilotProfileDAO, CopilotProjectDAO and CopilotProjectPlanDAO implementations provided in this component. This class uses Logging Wrapper logger to log errors and debug information.

CopilotProjectInfo

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

CopilotProjectInfoType

This class represents a "copilot project info type" lookup entity. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

CopilotProjectPlan

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

CopilotProjectPlanDAO [interface]

This interface represents a copilot project plan DAO. It extends GenericDAO<CopilotProjectPlan> and provides an additional method for retrieving copilot project plan by the given copilot project ID.

CopilotProjectPlanDAOImpl

This class is an implementation of CopilotProjectPlanDAO that uses Hibernate session to access entities in persistence. It extends GenericDAOImpl<CopilotProjectPlan> that provides common functionality for CopilotProfileDAO, CopilotProjectDAO and CopilotProjectPlanDAO implementations provided in this component. This class uses Logging Wrapper logger to log errors and debug information.

CopilotProjectStatus

This class represents a "copilot project status" lookup entity. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

CopilotType

This class represents a "copilot type" lookup entity. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

GenericDAO<T extends IdentifiableEntity> [interface]

This interface represents a generic DAO for managing entities in persistence. It defines methods for creating, updating, deleting and retrieving entities to/from persistence.

GenericDAOImpl<T extends IdentifiableEntity> [abstract]

This class is an implementation of GenericDAO that uses Hibernate session to access entities in persistence. It's assumed that this class will be initialized using Spring IoC. This class uses Logging Wrapper logger to log errors and debug information.

IdentifiableEntity [abstract]

This is a base class for all entities that have ID and auditing fields. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

LookupDAO [interface]

This interface represents a lookup DAO. It provides method for retrieving all copilot profile statuses, copilot project statuses and copilot types from persistence.

LookupDAOImpl

This class is an implementation of LookupDAO that uses Hibernate session to access entities in persistence. This class uses Logging Wrapper logger to log errors and debug information.

LookupEntity [abstract]

This is a base class for all lookup entities. It's assumed that each lookup entity has a name field. This class is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters.

PlannedContest

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

UtilityDAO [interface]

This interface represents a helper utility DAO. It provides methods for retrieving the number of bug races for specific contest, the latest bug resolution date for specific contest, the earnings of specific copilot user and contest IDs for specific copilot user and TC direct project.

UtilityDAOImpl

This class is an implementation of UtilityDAO that uses Hibernate sessions to access databases. It uses 3 data sources for the following databases: TCS Catalog (session factory for this DB is stored in BaseDAO), TopCoder DW (contains payments data) and JIRA DB. Data from these databases are accessed by this class using native SQL queries. This class uses Logging Wrapper logger to log errors and debug information.

1.5 Component Exception Definitions

CopilotDAOException

This exception is thrown by BaseDAO, implementations of LookupDAO, UtilityDAO and GenericDAO when some unexpected error occurred. Also this exception is used as a base class for other specific custom exceptions.

CopilotDAOInitializationException

This exception is thrown by BaseDAO when the class was not initialized properly (e.g. while required properly is not specified or property value has invalid format).

EntityNotFoundException

This exception is thrown by implementations of UtilityDAO and GenericDAO when entity with the given ID doesn't exist in the persistence.

1.6 Thread Safety

This component is thread safe (when used correctly).

Implementations of GenericDAO, CopilotProfileDAO, CopilotProjectDAO and CopilotProjectPlanDAO must be thread safe when entities passed to them are used by the caller in thread safe manner.

Implementations of LookupDAO and UtilityDAO must be thread safe.

IdentifiableEntity, LookupEntity, CopilotProfile, CopilotProfileInfoType, CopilotProfileInfo, CopilotProfileStatus, CopilotProjectPlan, PlannedContest, CopilotProjectStatus, CopilotProjectInfoType, CopilotProjectInfo, CopilotProject and CopilotType are mutable and not thread safe entities.

BaseDAO has mutable attributes, thus it's not thread safe. But it's assumed that its subclasses will be initialized via Spring IoC before calling any business method, this way it's always used in thread safe manner. BaseDAO uses thread safe SessionFactory and Session instances.

GenericDAOImpl has mutable attributes, thus it's not thread safe. But it's assumed that it will be initialized via Spring IoC before calling any business method, this way it's always used in thread safe manner. It uses thread safe SessionFactory, Session and Log instances.

CopilotProfileDAOImpl, CopilotProjectDAOImpl, CopilotProjectPlanDAOImpl, LookupDAOImpl and UtilityDAOImpl has mutable attributes, thus it's not thread safe. But it's assumed that it will be initialized via Spring IoC before calling any business method, this way it's always used in thread safe manner. They use thread safe SessionFactory, Session and Log instances.

GenericDAOImpl uses Spring declarative transactions, thus all methods that can modify data in persistence have @Transactional annotation.

2. Environment Requirements

2.1 Environment

Development language: Java1.5
Compile target: Java1.5
QA Environment: Java 1.5, JBoss 4.0.2, Informix 11.0

2.2 TopCoder Software Components

Logging Wrapper 2.0 – is used for logging errors and debug information.

Base Exception 2.0 – is used by custom exception defined 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

Spring 2.5.6 (http://www.springsource.org/)

Hibernate 3.5.4 (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.direct.services.copilot.dao
com.topcoder.direct.services.copilot.dao.impl
com.topcoder.direct.services.copilot.model

3.2 Configuration Parameters

3.2.1 Configuration of BaseDAO subclasses

It's assumed that subclasses of BaseDAO will be initialized by Spring using a setter dependency injection. The following properties must be injected:

Property	Description	Values
loggerName	The name of the logger to be used for logging errors and debug information. If not specified, logging is not performed.	String. Not empty. Optional.
sessionFactory	The Hibernate session factory to be used when retrieving session for accessing entities stored in TCS Catalog database.	SessionFactory. Required.

3.2.2 Configuration of UtilityDAOImpl

It's assumed that UtilityDAOImpl will be initialized by Spring using a setter dependency injection. The following properties must be injected:

Property	Description	Values
jiraSessionFactory	The Hibernate session factory to be used when retrieving session for accessing JIRA database.	SessionFactory. Required.
paymentSessionFactory	The Hibernate session factory to be used when retrieving session for accessing database with payments data.	SessionFactory. Required.
copilotPaymentTypeId	The ID of the copilot payment type.	Long. Must be positive. Required.
copilotResourceRoleId	The ID of the copilot resource role.	Long. Must be positive. Required.
userResourceInfoTypeId	The ID of the user resource info type.	Long. Must be positive. Required.

Also see section 3.2.1 for additional parameters.

3.3 Dependencies Configuration

Please see docs of Logging Wrapper component to configure it properly.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Set up your Informix database, please use dbschema.sql script from test_files directory.
- Modify the database connection strings in applicationContext.xml and demoApplicationContext.xml to reflect your 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 Sample Spring beans configuration

This section shows how to configure DAO implementations provided in this component with Spring beans configuration file. Please see docs of the Spring framework and docs of the dependency components for description on how to configure the dependency beans.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/tx
       http://www.springframework.org/schema/tx/spring-tx-2.0.xsd">


     <!-- configuration of other dependency beans goes here -->


    <!-- This configuration uses only one data source and session factory, all the tables
were created in the same database
         - this is only done during the testing phase - for deployment all the session
factories need to be configured separately -->
    <bean id="tcsCatalogDataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
        <property name="driverClassName" value="com.informix.jdbc.IfxDriver"/>
        <property name="url" value="jdbc:informix-
sqli://localhost:9088/copilot:informixserver=informix_db"/>
        <property name="username" value="informix"/>
        <property name="password" value="adminadmin"/>
    </bean>


    <bean id="tcsCatalogSessionFactory"
class="org.springframework.orm.hibernate3.LocalSessionFactoryBean">
        <property name="dataSource" ref="tcsCatalogDataSource"/>
        <property name="mappingResources">
            <list>
```

```
            <value>mapping.hbm.xml</value>
        </list>
    </property>
    <property name="hibernateProperties">
        <value>
            hibernate.dialect=org.hibernate.dialect.InformixDialect
            hibernate.show_sql=true
        </value>
    </property>
</bean>

<alias name="tcsCatalogSessionFactory" alias="jiraSessionFactory"/>
<alias name="tcsCatalogSessionFactory" alias="paymentSessionFactory"/>

<bean id="copilotProfileDAO"
      class="com.topcoder.direct.services.copilot.dao.impl.CopilotProfileDAOImpl">
    <property name="loggerName">
        <value>myLogger</value>
    </property>
    <property name="sessionFactory" ref="tcsCatalogSessionFactory"/>
</bean>

<bean id="copilotProjectDAO"
      class="com.topcoder.direct.services.copilot.dao.impl.CopilotProjectDAOImpl">
    <property name="loggerName">
        <value>myLogger</value>
    </property>
    <property name="sessionFactory" ref="tcsCatalogSessionFactory"/>
</bean>

<bean id="copilotProjectPlanDAO"
      class="com.topcoder.direct.services.copilot.dao.impl.CopilotProjectPlanDAOImpl">
    <property name="loggerName">
        <value>myLogger</value>
    </property>
    <property name="sessionFactory" ref="tcsCatalogSessionFactory"/>
</bean>

<bean id="lookupDAO"
      class="com.topcoder.direct.services.copilot.dao.impl.LookupDAOImpl">
    <property name="loggerName">
        <value>myLogger</value>
    </property>
```

```
        <property name="sessionFactory" ref="tcsCatalogSessionFactory"/>
    </bean>

    <bean id="utilityDAO"
          class="com.topcoder.direct.services.copilot.dao.impl.UtilityDAOImpl">
        <property name="loggerName">
            <value>myLogger</value>
        </property>
        <property name="sessionFactory" ref="tcsCatalogSessionFactory"/>
        <property name="jiraSessionFactory" ref="jiraSessionFactory"/>
        <property name="paymentSessionFactory" ref="paymentSessionFactory"/>
        <property name="copilotPaymentTypeId">
            <value>20</value>
        </property>
        <property name="copilotResourceRoleId">
            <value>14</value>
        </property>
        <property name="userResourceInfoTypeId">
            <value>1</value>
        </property>
    </bean>

    <tx:annotation-driven transaction-manager="txManager"/>
    <bean id="txManager"
class="org.springframework.orm.hibernate3.HibernateTransactionManager">
        <property name="sessionFactory" ref="tcsCatalogSessionFactory"/>
    </bean>
</beans>
```

4.3.2 *Usage of CopilotProjectPlanDAOImpl*

Please note that this section shows how to use GenericDAO-specific CRUD methods. These methods can be used similarly with GenericDAOImpl<T>, CopilotProfileDAOImpl and CopilotProjectDAOImpl.

```
// Crates and saves required dependant entities in other to test saving CopilotProjectPlan
CopilotProfile copilotProfile = …;
CopilotProject copilotProject = …;
// Saves the CopilotProfile and CopilotProject in the database


// Note: The CopilotProjectPlanDAO has been already injected into this demo so no need to
retrieve it from
// application context
// Despite using autowiring you can alternatively use the application context.
// Retrieves CopilotProjectPlanDAO from the Spring application context
// ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
// CopilotProjectPlanDAO copilotProjectPlanDAO =
//   (CopilotProjectPlanDAO) context.getBean("copilotProjectPlanDAO");


// Creates copilot project plan
CopilotProjectPlan copilotProjectPlan = new CopilotProjectPlan();
copilotProjectPlan.setPlannedDuration(14);
```

```
copilotProjectPlan.setPlannedBugRaces(2);
copilotProjectPlan.setCopilotProjectId(copilotProject.getId());
copilotProjectPlan.setCreateUser("user");
copilotProjectPlan.setModifyUser("user");
copilotProjectPlan.setCreateDate(new Date());
copilotProjectPlan.setModifyDate(new Date());Set<PlannedContest> plannedContests = new
HashSet<PlannedContest>();
PlannedContest plannedContest = new PlannedContest();
plannedContest.setName("Sample Design Contest");
plannedContest.setDescription("This is a sample design contest");
plannedContest.setProjectCategoryId(1);
plannedContest.setStartDate(new Date(new Date().getTime() + 24 * 3600 * 1000));
plannedContest.setEndDate(new Date(new Date().getTime() + 216 * 3600 * 1000));
plannedContest.setCreateUser("user");
plannedContest.setModifyUser("user");
plannedContest.setCreateDate(new Date());
plannedContest.setModifyDate(new Date());
plannedContests.add(plannedContest);
copilotProjectPlan.setPlannedContests(plannedContests);

long copilotProjectPlanId = copilotProjectPlanDAO.create(copilotProjectPlan);

// Updates the planned project duration
copilotProjectPlan.setPlannedDuration(10);

copilotProjectPlanDAO.update(copilotProjectPlan);

// Retrieves the copilot project plan by its ID
copilotProjectPlan = copilotProjectPlanDAO.retrieve(copilotProjectPlanId);
// copilotProjectPlan.getId() must be equal to copilotProjectPlanId
// copilotProjectPlan.getPlannedDuration() must be 10
// copilotProjectPlan.getPlannedBugRaces() must be 2
// copilotProjectPlan.getCopilotProjectId() must be 1
// copilotProjectPlan.getPlannedContests().size() must be 1

// Retrieves the copilot project plan by copilot project ID
copilotProjectPlan = copilotProjectPlanDAO.getCopilotProjectPlan(1);
// copilotProjectPlan.getId() must be equal to copilotProjectPlanId

// Delete the copilot project plan by copilot project ID
copilotProjectPlanDAO.delete(copilotProjectPlanId);

// Retrieves all copilot project plans
List<CopilotProjectPlan> copilotProjectPlans = copilotProjectPlanDAO.retrieveAll();
// copilotProjectPlans.size() must be 0
```

4.3.3 *Usage of CopilotProfileDAOImpl, CopilotProjectDAOImpl, LookupDAOImpl and UtilityDAOImpl*

```
// Retrieves CopilotProfileDAO from the Spring application context
CopilotProfileDAO copilotProfileDAO =
    (CopilotProfileDAO) context.getBean("copilotProfileDAO");

// Retrieves the copilot profile for copilot user with ID=1
CopilotProfile copilotProfile = copilotProfileDAO.getCopilotProfile(1);

// Retrieves CopilotProjectDAO from the Spring application context
CopilotProjectDAO copilotProjectDAO =
    (CopilotProfileDAO) context.getBean("copilotProjectDAO");

// Retrieves the copilot projects for copilot with profile ID=1
List<CopilotProject> copilotProjects = copilotProjectDAO.getCopilotProjects(1);

// Retrieves LookupDAO from the Spring application context
LookupDAO lookupDAO =
    (LookupDAO) context.getBean("lookupDAO");

// Retrieves all copilot profile statuses
List<CopilotProfileStatus> copilotProfileStatuses = lookupDAO.getAllCopilotProfileStatuses();

// Retrieves all copilot project statuses
List<CopilotProjectStatus> copilotProjectStatuses = lookupDAO.getAllCopilotProjectStatuses();
```

```
// Retrieves all copilot types
List<CopilotType> copilotTypes = lookupDAO.getAllCopilotTypes();

// Retrieves UtilityDAO from the Spring application context
UtilityDAO utilityDAO =
    (UtilityDAO) context.getBean("utilityDAO");

// Retrieves the bug count for contest with ID=1001
int contestBugCount = utilityDAO.getContestBugCount(1001);

// Retrieves the copilot earnings for copilot user with ID=1
double copilotEarnings = utilityDAO.getCopilotEarnings(1);

// Retrieves the latest bug resolution date for contest with ID=1
Date contestLatestBugResolutionDate = utilityDAO.getContestLatestBugResolutionDate(1);

// Retrieves the IDs of OR contests for copilot with user ID=1
// and TC direct project with ID=101
long[] contestIds = utilityDAO.getCopilotProjectContests(1, 101);
```

5. Future Enhancements

None