1. Scope

1.1 Overview

This component provides all necessary entities and DAO interfaces and implementations for the Client and Project Entities architecture. This component will define a number of entity classes, along with the necessary annotations for those entities to be used with Hibernate JPA extensions. In addition, DAO interfaces will be provided and basic EJB3 implementations will be part of the design. The basic architecture of the DAO pieces is modeled on the CaveatEmptor example provided here: http://www.hibernate.org/400.html .

1.1.1 Version

1.0

1.2 Logic Requirements

1.2.1 Entities

All entities in the com.topcoder.clients.model package in the interface diagram should be provided. They are just basic data holders that represent the entities used by the framework. They should all implement the Serializable interface. Also, each entity must be serializable to XML, for use with the Client and Project Services component. This should be done through the use of class annotations, like ""@XmlAccessorType(XmlAccessType.FIELD)", "@XmlType(name = "..." propOrder = { "..." })" etc...

1.2.2 Entities and Hibernate

In addition to the XML serialization annotations mentioned above, each entity should also contain the necessary annotations to be used with the Hibernate EntityManager, outlined here: http://www.hibernate.org/397.html The provided database schema should be used to generate the necessary table and column information.

1.2.3 DAO Interfaces

The DAO interfaces found in com.topcoder.clients.model.dao must be provided as part of the design. No modifications are allowed unless approved in the forum.

1.2.4 DAO EJB3 Implementations

In addition to the interfaces mentioned above, implementations must be provided that use the Hibernate EntityManager class to perform persistence to and from the database.

1.2.5 DAO Searching

Search Builder 1.4 supports generating HQL using a provided Filter instance, so that component can be used to perform searching against the Hibernate data store.

1.2.6 Entity Deletion

As part of the Client and Project Entities application requirements, entities are not to be permanently deleted from the database. Because of this, each entity has an "isDeleted" flag that is set to "true" when the entity is deleted by calling DAO.delete(entity). When retrieving information, all rows where is_deleted

is true should be ignored. When deleting, the entity should be updated with that flag set to true, and then saved to the database.

1.2.7 Dependent Updates

When updating Projects, the contained children do not have to be part of the update, only the Project instance provided. It is expected the user will make separate calls to update the children. This same practice applies to all entities with complex dependent fields.

1.2.8 Thread Safety

All DAO implementations should be reasonably thread-safe. It can be assumed that the entities being persisted won't change during an operation, but with that exception, the rest of the code should be able to be called from multiple threads.

1.2.9 Transactions

All access to the database that updates, creates, or retrieves from multiple tables must happen in a transaction. If an error occurs, the transaction should be rolled back gracefully.

1.3 Required Algorithms

None.

1.4 Example of the Software Usage

This component will be used to represent the clients and projects for TopCoder customers.

1.5 Future Component Direction

Further DAO implementations will be added.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Designs must adhere to the interface diagram definition found in the architecture TCUML file provided. Changes to the interfaces should be approved in the forum.

2.1.3 Environment Requirements

- Development language: Java1.5
- Compile target: Java1.5 and Java1.6

2.1.4 Package Structure

com.topcoder.clients.model
com.topcoder.clients.dao
com.topcoder.clients.dao.ejb3

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None

3.2.2 TopCoder Software Component Dependencies:

- Base Exception 2.0
- Search Builder 1.4

**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:

None.

3.2.4 QA Environment:

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

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

3.4.2 Help / User Documentation

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