[TopCoder]

# Review Application Management 1.0 Component Specification

## 1. Design

TopCoder is improving its review process by collecting the reviewers' performance stats and modifying the review signup process so that the best performing reviewers have higher chances to get a review position. The first part of the system upgrade that added the review rating into the Online Review application has already been implemented, see the details here. The next step is to modify the review signup process: instead of the "first come, first served" approach when members assign themselves to review positions they will apply for review positions and the system will select the best applicants for reviewing that project.

This component defines the data model for review applications and DAO interfaces for managing that objects. The component also provides DB-based implementations of those interfaces.

### 1.1 Design Patterns

**Strategy pattern** – ReviewApplicationManager, ReviewAuctionManager and their implementations can be used in some external strategy context; ReviewApplicationManagerImpl uses pluggable ReviewApplicationPersistence instance; ReviewAuctionManagerImpl uses pluggable ReviewAuctionPersistence instance.

**Delegate pattern** – ReviewApplicationManagerImpl / ReviewAuctionManagerImpl directly delegate certain operations to the pluggable ReviewApplicationPersistence / ReviewAuctionPersistence implementation instance.

**DAO/DTO pattern** – ReviewApplicationManager, ReviewAuctionManager, ReviewApplicationPersistence and ReviewApplicationPersistence are DAOs for ReviewApplication, ReviewAuction and other review application or review auction related Data Transfer Objects (DTOs).

### 1.2 Industry Standards

SQL, JDBC, Java Beans, XML

### 1.3 Required Algorithms

#### *1.3.1 Logging*

This component must perform logging in all public business methods of ReviewApplicationManagerImpl, ReviewAuctionManagerImpl, BaseDatabasePersistence, DatabaseReviewApplicationPersistence and DatabaseReviewAuctionPersistence.

All information described below must be logged using log:Log attribute. If log attribute is null, then logging is not required to be performed.

In all mentioned methods method entrance with input argument, 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 thrown exceptions and errors must be logged at ERROR level.

#### *1.3.2 Database Schema*

There're a few new database tables introduced to existing database, the scripts for new tables are provided in create.sql (DDL) and data.sql (preloaded data), developers are expected to execute the database scripts to update the existing database.

### 1.4 Component Class Overview

**BaseLookupEntity**

This is a base class for all model classes which have id and name..

**ReviewAuctionCategory**

This class represents Review Auction Category. e.g. "Specification Review", "Contest Review".

**ReviewAuctionType**

This class represents Review Auction Type. e.g. "Regular Contest Review", "Component Development Review".

**ReviewApplicationRole**

This class represents Review Application Role. e.g. "Primary Reviewer", "Secondary Reviewer".

**ReviewApplicationResourceRole**

This class represents Online Review Application Resource Role. e.g. "Primary Screener", "Reviewer", etc.

**ReviewAuction**

This class represents Review Auction, which is a review application campaign for a project.

Each project will normally have two review auctions: one for the specification review and one for the contest review, but some projects may have less than two auctions (e.g. Component Developments usually don't have spec review).

**ReviewApplicationStatus**

This class represents Review Application Status. e.g. "Pending", "Cancelled", "Approved", "Rejected".

**ReviewApplication**

This class represents Review Application, which is a member's application for a review role within an auction.

**ReviewApplicationManager [Interface]**

This interface defines the contract to manage review applications.

It provides methods to create review application, update review application, get review application statuses and search review applications.

**ReviewApplicationPersistence [Interface]**

This interface defines contract for accessing review application related data from persistence.

It provides methods to create review application, update review application and get review application statuses.

**ReviewApplicationManagerImpl**

This class is an implementation of ReviewApplicationManager that uses Search Builder component to search review applications (by auction ID, user ID, application status ID) in persistence and pluggable ReviewApplicationPersistence instance to create/update review applications in persistence and retrieve review application status lookup values.

This class uses Logging Wrapper component to log errors and debug information.

**BaseDatabasePersistence [Abstract Class]**

This class is base class for persistence implementations that access data in database persistence using JDBC and DB Connection Factory component.

**DatabaseReviewApplicationPersistence**

This class is an implementation of ReviewApplicationPersistence that accesses review application data in database persistence using JDBC and DB Connection Factory component.

This class uses Logging Wrapper component to log errors and debug information.

**ReviewAuctionManager [Interface]**

This interface defines the contract to manage review auctions.

It provides methods to create review auction, search open review auctions, retrieve review auction by ID, retrieve auction categories and auction types lookup values.

**ReviewAuctionPersistence [Interface]**

This interface defines contract for accessing review auction related data from persistence.

It provides methods to create review auction, get auction types and auction categories lookup values, get auction category ID for a given auction ID, and get assigned project resource IDs for given project IDs.

**ReviewAuctionManagerImpl**

This class is an implementation of ReviewAuctionManager that uses Search Builder component to search review auctions in persistence and pluggable ReviewAuctionPersistence instance to create review auction in persistence and retrieve review auction categories/types lookup values, get auction category ID for a given auction ID, and get assigned project resource IDs for given project IDs.

This class uses Logging Wrapper component to log errors and debug information.

**DatabaseReviewAuctionPersistence**

This class is an implementation of ReviewAuctionPersistence that accesses review auction data in database persistence using JDBC and DB Connection Factory component.\

This class uses Logging Wrapper component to log errors and debug information.

### 1.5 Component Exception Definitions

**ReviewApplicationManagementConfigurationException**

This exception is thrown by manager implementations and persistence implementations when some error occurs while initializing an instance using the given configuration.

**ReviewApplicationManagementException**

This is the base exception for all non-runtime exceptions in this component.

**ReviewApplicationManagerException**

This exception is thrown by implementations of ReviewApplicationManager when some not expected error occurred.

**ReviewAuctionManagerException**

This exception is thrown by implementations of ReviewAuctionManager when some not expected error occurred.

**ReviewApplicationPersistenceException**

This exception is thrown by implementations of ReviewApplicationPersistence when some not expected error occurred.

**IllegalArgumentException [System]**

This exception is thrown by various methods to indicate illegal argument such as null/empty string where not permitted.

**IllegalStateException [System]**

This exception is thrown by BaseDatabasePersistence#getConnection if the DB Connection Factory isn't initialized properly.

### 1.6 Thread Safety

This component is thread safe.

Implementations of ReviewApplicationManager, ReviewAuctionManager, ReviewApplicationPersistence and ReviewAuctionPersistence are required to be thread safe when entities passed to them are used by the caller in thread safe manner. Additionally it’s assumed that configure() method of ReviewApplicationPersistence and ReviewAuctionPersistence implementations will be called just once right after instantiation.

ReviewApplicationManagerImpl and ReviewAuctionManagerImpl are mutable because they hold cached lookup data are mutable variables (synchronization isn't applied as it is discouraged as per http://apps.topcoder.com/forums/?module=Thread&threadID=768185&start=0), however it is assumed that they will be only initialized once and won't be reloaded, hence those classes can be

considered as thread safe when entities passed to them are used by the caller in thread safe manner. They use thread safe SearchBundle, ReviewApplicationPersistence / ReviewAuctionPersistence and Log instances.

DatabaseReviewApplicationPersistence / DatabaseReviewAuctionPersistence is mutable, but thread safe when configure() method is called just once right after instantiation and entities passed to it are used by the caller in thread safe manner. It uses thread safe DBConnectionFactory and Log instances.

DatabaseReviewApplicationPersistence / DatabaseReviewAuctionPersistence uses auto-commit transaction when updating data in the database.

Lookup model classes (ReviewAuctionCategory, ReviewAuctionType, ReviewApplicationResourceRole, ReviewApplicationRole, ReviewApplicationStatus) are immutable and thread safe. All other model classes are mutable and not thread safe.

ReviewApplicationFilterBuilder is immutable and thread safe static utility class.

## 2. Environment Requirements

### 2.1 Environment

Development language: Java 1.5
Compile target: Java 1.5, Java 1.6
QA Environment: Java 1.5, RedHat Linux 4, Windows 2000, Windows 2003
Database: Informix v11

### 2.2 TopCoder Software Components

**Base Exception 2.0** – is used by custom exceptions defined in this component.

**Configuration API 1.1.0** – is used for initializing classes from this component.

**Configuration Persistence 1.0.2** – is used for reading configuration from file.

**Search Builder 1.4.1** – is used for searching for review applications, review auctions in persistence.

**Database Abstraction 2.0** – defines CustomResultSet class used in this component.

**DB Connection Factory 1.1** – is used for creating database connections.

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

**Object Factory 2.0.1** – is used for creating pluggable object instances.

**Object Factory Configuration API Plugin 1.0** – allows to use Configuration API for creating Object Factory.

**Topcoder Commons Utility 1.0** – provides ParameterCheckUtility, LoggingWrapperUtility and ValidationUtility used by 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

None

## 3. Installation and Configuration

### 3.1 Package Name

com.topcoder.management.review.application
com.topcoder.management.review.application.impl
com.topcoder.management.review.application.impl.persistence
com.topcoder.management.review.application.search

## 3.2 Configuration Parameters

### *3.2.1 Configuration of ReviewApplicationManagerImpl*

The following table describes the structure of ConfigurationObject passed to the constructor of ReviewApplicationManagerImpl class (angle brackets are used for identifying child configuration objects). This ConfigurationObject can be optionally read from a configuration file using Configuration Persistence component.

| Parameter | Description | Values |
|---|---|---|
| loggerName | The name of Logging Wrapper logger to be used for logging errors and debug information.<br>When not provided, logging is not performed. | String. Not empty. Optional. |
| searchBundleManager Namespace | The namespace used when creating an instance of SearchBundleManager. | String. Not empty. Required. |
| reviewApplicationSearchBundleName | The name of the search bundle used by this class when searching for review applications. | String. Not empty. Required. |
| <objectFactoryConfig> | This section contains configuration of Object Factory used by this class for creating pluggable object instances. | ConfigurationObject. Required. |
| persistenceKey | The Object Factory key that is used for creating an instance of ReviewApplicationPersistence to be used by this manager. | String. Not empty. Required. |
| <persistenceConfig> | The configuration for ReviewApplicationPersistence instance. | ConfigurationObject. Required. |

### *3.2.2 Configuration of ReviewAuctionManagerImpl*

The following table describes the structure of ConfigurationObject passed to the constructor of ReviewAuctionManagerImpl class (angle brackets are used for identifying child configuration objects). This ConfigurationObject can be optionally read from a configuration file using Configuration Persistence component.

| Parameter | Description | Values |
|---|---|---|
| loggerName | The name of Logging Wrapper logger to be used for logging errors and debug information.<br>When not provided, logging is not performed. | String. Not empty. Optional. |
| searchBundleManager Namespace | The namespace used when creating an instance of SearchBundleManager. | String. Not empty. Required. |
| specReviewAuctionSearchBundleName | The name of the search bundle used by this class when searching for auctions for specification reviews. | String. Not empty. Required. |
| contestReviewAuctionSearchBundleName | The name of the search bundle used by this class when searching for auctions for regular contest reviews. | String. Not empty. Required. |

| | | |
|---|---|---|
| specReviewAuctionCategoryId | The ID of the review auction category for specification review. | Positive Integer. Required. |
| contestReviewAuctionCategoryId | The ID of the review auction category for regular contest review. | Positive Integer. Required. |
| <objectFactoryConfig> | This section contains configuration of Object Factory used by this class for creating pluggable object instances. | ConfigurationObject. Required. |
| persistenceKey | The Object Factory key that is used for creating an instance of ReviewAuctionPersistence to be used by this manager. | String. Not empty. Required. |
| <persistenceConfig> | The configuration for ReviewAuctionPersistence instance. | ConfigurationObject. Required. |

### 3.2.3 *Configuration of DatabaseReviewApplicationPersistence*

The following table describes the structure of ConfigurationObject passed to the constructor of DatabaseReviewApplicationPersistence class (angle brackets are used for identifying child configuration objects).

| Parameter | Description | Values |
|---|---|---|
| loggerName | The name of Logging Wrapper logger to be used for logging errors and debug information.<br>When not provided, logging is not performed. | String. Not empty. Optional. |
| dbConnectionFactoryConfig | The configuration to be used for creating DBConnectionFactoryImpl instance. | ConfigurationObject. Required. |
| connectionName | The connection name to be passed to the connection factory when establishing a database connection. If not specified, a default connection is used. | String. Not empty. Optional. |

### 3.2.4 *Configuration of DatabaseReviewAuctionPersistence*

The following table describes the structure of ConfigurationObject passed to the constructor of DatabaseReviewAuctionPersistence class (angle brackets are used for identifying child configuration objects).

| Parameter | Description | Values |
|---|---|---|
| loggerName | The name of Logging Wrapper logger to be used for logging errors and debug information.<br>When not provided, logging is not performed. | String. Not empty. Optional. |
| dbConnectionFactoryConfig | The configuration to be used for creating DBConnectionFactoryImpl instance. | ConfigurationObject. Required. |

| | | |
|---|---|---|
| connectionName | The connection name to be passed to the connection factory when establishing a database connection. If not specified, a default connection is used. | String. Not empty. Optional. |

### 3.3 Dependencies Configuration

Please see docs of Logging Wrapper, DB Connection Factory, Search Builder and Object Factory components to configure them properly.

#### *3.3.1 Configuration of Search Builder component*

Please refer to test_files/SearchBundleManager.xml for detailed configurations(including filter configurations and search bundle configurations). The content of SearchBundleManager.xml isn't shown in the component specification for brevity.

## 4. Usage Notes

### 4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Setup Informix database tcs_catalog:

  a. Run test_files/create.sql to create the tables.

  test_files/drop.sql can be used to drop tables.

  b. Update "jdbc_url", "user" and "password" properties in test_files/SearchBundleManager.xml and test_files/ReviewApplicationManagement.xml if needed.

- 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 Configuration File*

```
<?xml version="1.0"?>
<CMConfig>
  <Config name="com.topcoder.management.review.application.impl.ReviewApplicationManagerImpl">
    <Property name="loggerName">
      <Value>myLogger</Value>
    </Property>
    <Property name="objectFactoryConfig">
      <Property name="DatabaseReviewApplicationPersistence">
        <Property name="type">
          <Value>com.topcoder.management.review.application.impl.persistence.DatabaseReviewApplicationPersistence</Value>
        </Property>
      </Property>
    </Property>
    <Property name="searchBundleManagerNamespace">
      <Value>ReviewApplicationManagement.SearchBuilderManager</Value>
    </Property>
    <Property name="reviewApplicationSearchBundleName">
      <Value>reviewApplicationSearchBundle</Value>
    </Property>
    <Property name="persistenceKey">
      <Value>DatabaseReviewApplicationPersistence</Value>
    </Property>
    <Property name="persistenceConfig">
      <Property name="loggerName">
        <Value>myLogger</Value>
      </Property>
```

```
    <Property name="dbConnectionFactoryConfig">
      <Property name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
        <Property name="connections">
          <Property name="default">
            <Value>myConnection</Value>
          </Property>
          <Property name="myConnection">
            <Property name="producer">
              <Value>com.topcoder.db.connectionfactory.producers.JDBCConnectionProducer</Value>
            </Property>
            <Property name="parameters">
              <Property name="jdbc_driver">
                <Value>com.informix.jdbc.IfxDriver</Value>
              </Property>
              <Property name="jdbc_url">
                <Value>jdbc:informix-sqli://localhost:1526/tcs_catalog:informixserver=ol_topcoder</Value>
              </Property>
              <Property name="SelectMethod">
                <Value>cursor</Value>
              </Property>
              <Property name="user">
                <Value>informix</Value>
              </Property>
              <Property name="password">
                <Value>123456</Value>
              </Property>
            </Property>
          </Property>
        </Property>
      </Property>
      <Property name="connectionName">
        <Value>myConnection</Value>
      </Property>
    </Property>
  </Config>

  <Config name="com.topcoder.management.review.application.impl.ReviewAuctionManagerImpl">
    <Property name="loggerName">
      <Value>myLogger</Value>
    </Property>
    <Property name="objectFactoryConfig">
      <Property name="DatabaseReviewAuctionPersistence">
        <Property name="type">
          <Value>com.topcoder.management.review.application.impl.persistence.DatabaseReviewAuctionPersistence</Value>
        </Property>
      </Property>
    </Property>
    <Property name="searchBundleManagerNamespace">
      <Value>ReviewApplicationManagement.SearchBuilderManager</Value>
    </Property>
    <Property name="specReviewAuctionSearchBundleName">
      <Value>specReviewAuctionSearchBundle</Value>
    </Property>
    <Property name="contestReviewAuctionSearchBundleName">
      <Value>contestReviewAuctionSearchBundle</Value>
    </Property>
    <Property name="contestReviewAuctionCategoryId">
      <Value>1</Value>
    </Property>
    <Property name="specReviewAuctionCategoryId">
      <Value>2</Value>
    </Property>
    <Property name="persistenceKey">
      <Value>DatabaseReviewAuctionPersistence</Value>
    </Property>
    <Property name="persistenceConfig">
      <Property name="loggerName">
        <Value>myLogger</Value>
      </Property>
```

```
    <Property name="dbConnectionFactoryConfig">
      <Property name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
        <Property name="connections">
          <Property name="default">
            <Value>myConnection</Value>
          </Property>
          <Property name="myConnection">
            <Property name="producer">
              <Value>com.topcoder.db.connectionfactory.producers.JDBCConnectionProducer</Value>
            </Property>
            <Property name="parameters">
              <Property name="jdbc_driver">
                <Value>com.informix.jdbc.IfxDriver</Value>
              </Property>
              <Property name="jdbc_url">
                <Value>jdbc:informix-sqli://localhost:1526/tcs_catalog:informixserver=ol_topcoder</Value>
              </Property>
              <Property name="SelectMethod">
                <Value>cursor</Value>
              </Property>
              <Property name="user">
                <Value>informix</Value>
              </Property>
              <Property name="password">
                <Value>123456</Value>
              </Property>
            </Property>
          </Property>
        </Property>
      </Property>
      <Property name="connectionName">
        <Value>myConnection</Value>
      </Property>
    </Property>
  </Config>
</CMConfig>
```

### 4.3.2 Review Application Manager Demo

```
// Create an instance of ReviewAuctionManagerImpl using default configuration
ReviewAuctionManager reviewAuctionManager = new ReviewAuctionManagerImpl();
// Get Review Auction Types
List<ReviewAuctionType> types = reviewAuctionManager.getAuctionTypes();

// Create Auction
ReviewAuction auction = new ReviewAuction();
auction.setProjectId(100000);
auction.setAuctionType(types.get(0));

auction = reviewAuctionManager.createAuction(auction);
long auctionId = auction.getId();

// Create an instance of ReviewApplicationManagerImpl using default configuration
ReviewApplicationManager reviewApplicationManager = new ReviewApplicationManagerImpl();

// Create an instance of ReviewApplicationManagerImpl using custom configuration
reviewApplicationManager = new ReviewApplicationManagerImpl(
    ReviewApplicationManagerImpl.DEFAULT_CONFIG_FILENAME,
    ReviewApplicationManagerImpl.DEFAULT_CONFIG_NAMESPACE);

// Create an instance of ReviewApplicationManagerImpl using custom configuration
ConfigurationObject configuration = getConfig(ReviewApplicationManagerImpl.DEFAULT_CONFIG_NAMESPACE);
reviewApplicationManager = new ReviewApplicationManagerImpl(configuration);

// Get Review Application Statuses
List<ReviewApplicationStatus> statuses = reviewApplicationManager.getApplicationStatuses();

// Create Review Application
```

```
ReviewApplication application = new ReviewApplication();
application.setUserId(123);
application.setAuctionId(auctionId);
application.setApplicationRoleId(4);
application.setStatus(statuses.get(0));
application.setCreateDate(new Date());

application = reviewApplicationManager.createApplication(application);

long applicationId = application.getId();

// Update Review Application
application.setStatus(statuses.get(1));

reviewApplicationManager.updateApplication(application);

// Search by auction ID
Filter auctionIdFilter = ReviewApplicationFilterBuilder.createAuctionIdFilter(auctionId);
List<ReviewApplication> applications = reviewApplicationManager.searchApplications(auctionIdFilter);

// Search by user ID
Filter userIdFilter = ReviewApplicationFilterBuilder.createUserIdFilter(123);
applications = reviewApplicationManager.searchApplications(userIdFilter);

// Search by application status ID
Filter applicationStatusIdFilter =
    ReviewApplicationFilterBuilder.createApplicationStatusIdFilter(statuses.get(1).getId());
applications = reviewApplicationManager.searchApplications(applicationStatusIdFilter);

// Search by combination of filters
Filter filter = new AndFilter(auctionIdFilter, new AndFilter(userIdFilter, applicationStatusIdFilter));
applications = reviewApplicationManager.searchApplications(filter);
```

### *4.3.3 Review Auction Manager Demo*

```
// Create an instance of ReviewAuctionManagerImpl using default configuration
ReviewAuctionManager reviewAuctionManager = new ReviewAuctionManagerImpl();

// Create an instance of ReviewApplicationManagerImpl using custom configuration
reviewAuctionManager = new ReviewAuctionManagerImpl(ReviewAuctionManagerImpl.DEFAULT_CONFIG_FILENAME,
    ReviewAuctionManagerImpl.DEFAULT_CONFIG_NAMESPACE);

// Create an instance of ReviewAuctionManagerImpl using custom configuration
ConfigurationObject configuration = getConfig(ReviewAuctionManagerImpl.DEFAULT_CONFIG_NAMESPACE);
reviewAuctionManager = new ReviewAuctionManagerImpl(configuration);

// Get Review Auction Categories
List<ReviewAuctionCategory> categories = reviewAuctionManager.getAuctionCategories();

// Get Review Auction Types
List<ReviewAuctionType> types = reviewAuctionManager.getAuctionTypes();

// Create Auction
ReviewAuction auction = new ReviewAuction();
auction.setProjectId(100000);
auction.setAuctionType(types.get(0));

auction = reviewAuctionManager.createAuction(auction);

long auctionId = auction.getId();

// Search Auctions by auction category ID
List<ReviewAuction> auctions = reviewAuctionManager.searchOpenAuctions(1);

// Search Auctions by aution category ID and project category IDs
auctions = reviewAuctionManager.searchOpenAuctions(1, Arrays.asList(1L, 2L, 3L));

// Get Auction
auction = reviewAuctionManager.getAuction(auctionId);
```

## 5. Future Enhancements

None