Software Documentation : Java Custom Review Application Management

This page last changed on Nov 01, 2012 by volodymyrk.

1. Scope

1.1 Overview

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.

1.2 Logic Requirements

Search Builder

The requirements below assume the designers and developers are familiar with the Search Builder component and the way it works. If you are not please go ahead and familiarize yourself with the Search Builder component before proceeding.

1.2.1 Data model classes

The component defines a bunch of new DB tables that need to be mapped by new entity classes (see create.sql file for the new tables definition). The current DB schema is also provided (see tcs_catalog.zip).

All entity classes and manager interfaces must meet the interface shown on the class diagram (see interface.gif)

1.2.1.1 ReviewAuctionCategory

The ReviewAuctionCategory class maps the review_auction_category_lu table. This entity defines the upper level categories of review types. The table will contains only two records for now (see data.sql file for the data for the new tables):

1. "Specification Review" (for reviewing requirements specifications)
2. "Contest Review" (for regular contest reviews).

1.2.1.2 ReviewAuctionType

The ReviewAuctionType class maps the review_auction_type_lu table. This entity defines review types such as "Regular Contest Review" (1 primary reviewer and 2 secondary reviewers), "Component Development Review" (1 primary failure, 1 accuracy and 1 stress reviewer) etc.

This class also contains a list of ReviewApplicationRole objects associated with this auction type. The list is ordered by the values of review_application_role_lu.order_index DB column.

1.2.1.3 ReviewApplicationRole

The ReviewApplicationRole class maps the review_application_role_lu table. This entity defines a single review role within an auction type. For example, "Primary Reviewer" would be a single review role within the "Regular Contest Review" auction type (even though it breaks down to several resource roles in Online Review). The ReviewApplicationRole#positions field defines the number of available positions for this role. For example, for "Secondary Reviewer" role it will take value 2 or 3 since there can be several secondary reviewers in a contest. For primary reviewer the positions value will be 1 because there can be at most one primary reviewer in a contest.

This class also contains a list of ReviewApplicationResourceRole objects associated with this review role.

This class doesn't map review_application_role_lu.order_index DB column. The ReviewAuctionType#applicationRoles list should already be ordered by that DB column value (in ascending order).

1.2.1.4 ReviewApplicationResourceRole

The ReviewApplicationResourceRole class maps the review_application_role_resource_role_xref table. It defines many-to-many relationship between the review application roles and the actual resource roles in Online Review. For example, the "Primary Reviewer" role will have four resource roles in Online Review: Primary Screener, Reviewer, Aggregator and Final Reviewer. "Secondary Reviewer" role will have only one resource role: Reviewer. The ReviewApplicationResourceRole#uniqueRole field defines whether this resource role has to be unique within the same project. For example, a project can't have more than one Primary Screener resource, so the value will be "true" for Primary Screener resource role. But a project can have multiple Reviewer resources so the value will be "false" for Reviewer resource role.

1.2.1.5 ReviewApplicationStatus

The ReviewApplicationStatus class maps the review_application_status_lu table. This entity defines the review application status. The records will be Pending, Cancelled, Approved and Rejected (may be more in future).

1.2.1.6 ReviewAuction

The ReviewAuction class maps the review_auction table. This entity defines 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).

ReviewAuction class contains three fields that do not directly map to any DB field:

1. ReviewAuction#open field defines whether the auction is currently open. This means that the project is in Active status, certain phases are currently open and there are still open review positions available. The specific conditions depend on the auction category.
2. ReviewAuction#assignmentDate field defines the date when the system will assign applicants to open review positions. If the date is in the past it means that the assignment will happen immediately when someone applies for review.
3. ReviewAuction#openPositions field is a list that defines how many positions for each application role are currently open. The list will have exactly as many elements as ReviewAuction#auctionType.applicationRoles list. i-th element of this list will correspond to i-th element of ReviewAuction#auctionType.applicationRoles list and will be less or equal than ReviewAuction#auctionType.applicationRoles[i].positions.

1.2.1.7 ReviewApplication

The ReviewApplication class maps the review_application table. This entity defines a member's application for a review role within an auction.

1.2.2 ReviewAuctionManager

The component will provide ReviewAuctionManager interface for the ReviewAuction objects and a DB-based implementation of that interface.

1.2.2.1 createAuction method

This method will create a new record in the DB table and will assign the ReviewAuction#id field.

1.2.2.2 searchOpenAuctions methods

The two search methods will return a list of all open auctions for the specified auction category and (optionally) project category. The two methods will share the same implementation with the only difference that one of them uses an additional search filter for the project category ID.

The implementation will only support two existing auction categories (Specification Review and Contest Review) for now and will use different search bundles for the two auction categories.

1. Call Search Builder component to search for all open auctions. The search bundle's context is different for the two auction categories. You need to set a search filter for the project status (project_status_id fields needs to be equal to 1 for Active projects) and optionally for the project category (see the project_category_id field).
2. Loop through the list of returned records and construct ReviewAuction objects. Only open auctions should be included in the resulting list.
   a. ReviewAuction#openPositions list should be constructed individually for each auction. See the algorithm explanation in section 1.3.
   b. ReviewAuction#open is true if the "open" SQL value is true and at least one element of ReviewAuction#openPositions list is greater than 0.

The context for the two search bundles is provided in files contest_review_search.sql and spec_review_search.sql.

1.2.2.3 getAuction method

This method will retrieve the auction record from the DB by its ID. It first needs to query the DB to determine the auction category ID and then perform a search similar to the searchOpenAuctions () methods.

The getAuction and searchOpenAuctions methods will share most of their logic. The only exceptions are that the search methods use the search filter for the Active project status (while getAuction method doesn't) and getAuction method uses a search filter for the auction ID (while seachOpenAuctions method don't). The getAuction method will return auction regardless of its state (open or closed) while the search methods only return the list of open auctions.

1.2.2.4 getAuctionCategories method

This method will return all existing auction categories.

1.2.2.5 getAuctionTypes method

This method will return all existing auction types.

1.2.3 ReviewApplicationManager

The component will provide ReviewApplicationManager interface for the ReviewApplication objects and a DB-based implementation of that interface.

1.2.3.1 createApplication method

This method will create a new record in the DB table and will assign the ReviewApplication#id field.

1.2.3.2 updateApplication method

This method will update an existing record in the DB table.

The method will only update the review_application table (e.g. any changes to review_application_status_lu object are not persisted).

1.2.3.3 searchApplications methods

This method will search all review applications using the Search Builder component. The designer is responsible to provide the search bundle's context for it.

The component will also provide the ReviewApplicationFilterBuilder class with factory methods to create these search filters:

1. Filter by auction ID
2. Filter by user ID
3. Filter by application status ID

The component will support composite AND, OR and NOT filters of any of the filters defined above. These filters are already provided by the Search Builder component so the design just needs to make sure they are supported.

1.2.4 Thread-Safety

The component is required to be thread-safe.

1.2.5 Performance

Performance is an essential requirement for this component since this component will serve the most often visited TopCoder website pages and the responses need to be real-time. It will also be used by a standalone background process that will constantly monitor all open auctions and assign applicants to review positions.

DB operations have the most dramatic effect on performance so the number of DB requests need to be minimized and the queries need to be written efficiently. Here are some considerations that designers need to implement:

1. All new DB tables except review_auction and review_application are considered static, meaning their content will not change during the component lifetime. Therefore only a single query to these tables needs to be performed and the content should be cached afterwards. For example, ReviewAuctionManager#searchOpenAuctions() methods should not construct new ReviewAuctionType, ReviewApplicationRole etc. objects by querying the DB for each new request. Instead, they should reuse the cached content or cached ReviewAuctionType, ReviewApplicationRole etc. objects. If the constructed objects are cached in memory, a good design must also take care not to return shared mutable objects that can be modified by the caller code (which will make the result of the future calls to searchOpenAuctions methods incorrect).
2. ReviewAuctionManager#searchOpenAuctions() methods return a list of ReviewAuction objects. To initialize the ReviewAuction#openPositions list an additional DB call is required (see section 1.3.1 for the details). It is important to make a single DB call that would include results for all auctions and not perform separate DB calls for each individual auction.

1.2.6 Logging and Error handling

The component will log its actions and errors at an appropriate level of details.

The component will have a strategy for handling all errors and exceptions (e.g. trying to delete a non-existent record etc.).

1.3 Required Algorithms

Designers must describe all algorithms used for the component.

1.3.1 Computing open review positions

The ReviewAuction and ReviewApplicationRole classes define the general structure of the review positions, including review roles and the number of positions for each review role. However for an actual project some review positions may have already been taken (for example, when a PM manually adds a reviewer in the Online Review). A project can also be configured in the Online Review to require less than 3 reviewers (which is the default number of reviewers). In such cases not all review positions will be open.

Constructing the ReviewAuction#openPositions list requires a special attention so the high-level algorithm is explained here:

1. Get resource roles for the project (see the performance note in section 1.2.5). You can use this SQL query for that:

   ```
   select distinct project_id, resource_role_id from resource where project_id in (LIST OF
   PROJECT IDs)
   ```
2. Determine how many reviewers are still needed for the project, which is equal to max(reviewers_required,0). Let's denote this value N.
3. Iterate through the ReviewAuctionType#applicationRoles list (remember it should be already ordered by the review_application_role_lu.order_index column). For each element in this list:
   a. Check if this application role is open. Iterate through ReviewApplicationRole#resourceRoles list for that application role and check if ReviewApplicationResourceRole#uniqueRole is true and if there's already a resource with that resourceRoleId in the project (see the list from step 1). If both conditions hold true for at least one element in the ReviewApplicationRole#resourceRoles list then this application role is considered closed.
   b. If the application role is closed put value 0 to the ReviewAuction#openPositions list, meaning all positions for that application role are closed.
   c. If the application role is open put value min(N, ReviewApplicationRole#positions) to the ReviewAuction#openPositions list. Decrease N by min(N, ReviewApplicationRole#positions).

1.4 Example of the Software Usage

The component will be used by the TopCoder website for the review signup pages for listing all open review auctions and managing review applications from the review board members.

The component will also be used by a standalone background process that will constantly monitor all open auctions and assign applicants to review positions.

1.5 Future Component Direction

Any enhancement must be approved either in forum or in email with managers to eliminate over-complicating the component with useless functionality. All performance optimizations are encouraged and do not require an explicit approval.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None, only API interface will be provided.

2.1.2 External Interfaces

The component must meet the interface shown on the class diagram (see interface.gif)

Any change to this interface needs to be approved.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5, Java 1.6

2.1.4 Package Structure

com.topcoder.management.review.application

com.topcoder.management.review.application.search

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

Configuration for the dependency components: Search Builder, DB Connection Factory

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 2.0.0
- Configuration API 1.1.0
- Configuration Persistence 1.0.2
- Search Builder 1.4.1
- Database Abstraction 2.0.0
- DB Connection Factory 1.1.0
- Logging Wrapper 1.2.0

**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:

Any third party library needs to be approved.

3.2.4 QA Environment:

- Java 1.5
- RedHat Linux 4
- Windows 2000
- Windows 2003

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
- Configuration Files for the Search Builder component

3.4.2 Help / User Documentation

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