[TopCoder]

Review Management 1.2 Component Specification

1. Design

All changes made in the version 1.2 are marked with **blue**.

All new items in the version 1.2 are marked with **red**.

Reviews are produced based on scorecards. A review holds a collection of items which address each of the questions on the scorecard. It also consists of the author that produced the review, the submission it addresses and the scorecard template it is based on. Various types of comments can be attached to the review or to each review item. A committed review must address all questions on the corresponding scorecard, and will have its overall score available.

The component provides the management functionalities to create, update, delete or search reviews. The review persistence logic is pluggable and is not provided in this component.

Changes in the version 1.2:

- Development language is changed to Java 1.5. Thus generic parameters are explicitly specified for all generic types in the source code.
- Added a method for deleting an existing review.
- Some trivial source code changes are performed to make the component meet TopCoder standards.
- The component is made thread safe.

1.1 Design Patterns

Strategy pattern – DefaultReviewManager uses pluggable ReviewPersistence implementation instance as a strategy; ReviewManager together with its implementation can be used in some external strategy context.

Delegate pattern – all methods of DefaultReviewManager simply delegate execution to the pluggable ReviewPersistence implementation instance.

DAO/DTO pattern – ReviewManager and ReviewPersistence are DAOs for Review, Comment, CommentType and some other DTOs.

1.2 Industry Standards

None

1.3 Required Algorithms

None

1.4 Component Class Overview

ChainFilter

This ChainFilter class provides the convenient methods to construct composite filters with "chain" operation. It will be used for complex search conditions.

It supports and, or and not operations

DefaultReviewManager

This DefaultReviewManager class is the default implementation of ReviewManager. It actually delegates the work to the pluggable persistence to create, update, delete and search review entities. The review entity ids will be generated by the inner persistence class. Additionally, It can also be used to add comment for review and item, and get all the comment types from the manager.

Changes in 1.2:

- Added removeReview() method.

Helper

Package private helper class to simplify the argument checking and string parsing.

Changes in 1.2:

- Specified generic parameter for generic class.

- Changed visibility of all helper methods to package private.

ReviewManager [interface]

This ReviewManager interface defines the contract of managing the review entities. It provides the management functionalities to create, update, delete and search review entities. It can also be used to add comment for review and item, and get all the comment types from the manager.

Changes in 1.2:

- Added removeReview() method.

- Implementations of this interface are now required to be thread safe.

ReviewPersistence [interface]

This interface defines the contract of managing the review entities in the persistence. It provides the persistence functionalities to create, update, delete and search reviews. Additionally, application users can also add comment for review and item, and get all the comment types from the persistence. The constructor of the implementation must take a string argument as namespace.

Changes in 1.2:

- Added removeReview() method.

- Implementations of this interface are now required to be thread safe.

1.5 Component Exception Definitions

ConfigurationException

This exception is thrown from the constructor taking a namespace argument, if failed to load configuration values from the ConfigManager or the configuration values are invalid.

Changes in 1.2:

- Renamed "msg" parameters to "message".

- Added constructors that accept ExceptionData.

DuplicateReviewEntityException

This exception will be thrown by DefaultReviewManager and implementations of ReviewPersistence if application tries to create a duplicated review entity.

Changes in 1.2:

- Renamed "msg" parameters to "message".

- Added constructors that accept Throwable and ExceptionData.

ReviewEntityNotFoundException

This exception will be thrown by DefaultReviewManager and implementations of ReviewPersistence if application tries to get or update a non-existing review entity

Changes in 1.2:

- Renamed "msg" parameters to "message".

- Added constructors that accept Throwable and ExceptionData.

ReviewManagementException

This exception will be thrown by ReviewManager, ReviewPersistence and their implementations if any error occurs during the management of review entities.

Changes in 1.2:

- Extends BaseCriticalException instead of BaseException.

- Renamed "msg" parameters to "message".

- Added constructors that accept ExceptionData.

ReviewPersistenceException

This exception will be thrown by DefaultReviewManager and implementations of ReviewPersistence if any error occurred during persisting the review entity.

Changes in 1.2:

- Renamed "msg" parameters to "message".

- Added constructors that accept ExceptionData.

1.6 Thread Safety

This component is thread safe starting from the version 1.2.

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

DefaultReviewManager is immutable, and thread safe when entities passed to it are used by the caller in thread safe manner. It uses thread safe ReviewPersistence implementation instance.

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

2.2 TopCoder Software Components

Configuration Manager 2.2.1 – is used for configuring this component.

Base Exception 2.0 – defines the base class for custom exceptions defined in this component.

Review Data Structure 1.0 – defines Review, CommentType and Comment entities used in this component.

Search Builder 1.3.2 – defines Filter interface 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

None

3. Installation and Configuration

3.1 Package Name

com.topcoder.management.review

3.2 Configuration Parameters

3.2.1 Configuration of DefaultReviewManager

The following table describes the structure of properties from Configuration Manager namespace used for configuring DefaultReviewManager class.

Parameter	Description	Values

persistence. persistence_class	The full class name of ReviewPersistence implementation to be used by the manager.	String. Not empty. Required.
persistence. persistence_namespace	The namespace used for configuring ReviewPersistence implementation instance (if provided, it is passed as the only constructor argument; otherwise a default constructor is used).	String. Not empty. Optional.

3.3 Dependencies Configuration

Please see docs of dependency components to configure them properly.

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 Sample configuration

```
<CMConfig>
  <Config name="com.topcoder.management.review.DefaultReviewManager">
    <Property name="persistence">
      <Property name="persistence_namespace">
        <Value>com.topcoder.management.review.MockReviewPersistence</Value>
      </Property>
      <Property name="persistence_class">
        <Value>com.topcoder.management.review.MockReviewPersistence</Value>
      </Property>
    </Property>
  </Config>
</CMConfig>
```

4.3.2 API usage

```
 // create a manager instance from default namespace.
manager = new DefaultReviewManager();

// create a manager instance from specified namespace.
manager = new DefaultReviewManager(NAMESPACE);

// create a manager instance from given persistence.
manager = new DefaultReviewManager(persistence);

// create a review into the manager.
manager.createReview(review, "createReviewer");

// create another review into the manager.
manager.createReview(new Review(2), "createReviewer");

// update the review with id = 2 in the manager.
Review updatedReview = new Review(2);
updatedReview.addItem(new Item(1));
manager.updateReview(updatedReview, "updateReviewer");

// update this review again in the manager.
updatedReview.addItem(new Item(2));
manager.updateReview(updatedReview, "updateReviewer");
```

```
// get the review from manager with its id.
Review getReview = manager.getReview(1);

// get all comment types from manager.
CommentType[] commentTypes = manager.getAllCommentTypes();

// add comment for review in manager.
Comment reviewComment = new Comment(1, 10001, "good");
manager.addReviewComment(review.getId(), reviewComment, "someReviewer");

// add comment for item in manager.
Comment itemComment = new Comment(1, 10001, "ok");
manager.addReviewComment(1, itemComment, "someReviewer");

// search the review from manager with simple filter.

// search for the reviews which have been committed.
Review[] reviews = manager.searchReviews(equalFilter, true);

// search the review from manager with 'chain' filter.

// search for the reviews which have been committed and
// project is greater than 10001, or reviewer id is less than 10000.
ChainFilter cf = new ChainFilter(equalFilter);
cf = cf.and(greaterThanFilter).or(lessThanFilter);

reviews = manager.searchReviews(cf.getFilter(), true);

// remove review with ID equal to 1
manager.removeReview(1, "manager");
```

5. Future Enhancements

None