Review Management Persistence 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 implementation of ReviewManager interface defined in Review Management component. This implementation uses JDBC to access review data in Informix database.

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.
- "Update review" method was updated to remove associated uploads when required.
- Some trivial source code changes are performed to make the component meet TopCoder standards.
- Some performance optimizations were performed.

1.1 Design Patterns

Strategy pattern – InformixReviewPersistence can be plugged as a strategy into DefaultReviewManager; InformixReviewPersistence uses IDGenerator and DBConnectionFactory implementation instances as strategies.

DAO/DTO pattern – InformixReviewPersistence is DAO for Review, Comment, CommentType and some other DTOs.

1.2 Industry Standards

JDBC, SQL

1.3 Required Algorithms

1.3.1 Transaction of InformixReviewPersistence

All the methods of InformixReviewPersistence that involve in the database operations, should put the database queries in one transaction. Take the createReview(Review,String) method for instance, in order to create Review, this method will also be responsible for creating the associated Item and Comment. All these database operations should be put in one transaction. The connection is retrieved from DBConnectionFactory, and it should be closed after operation to release to connection pool.

Note that searchReview method will fall into two transactions. One transaction is used to retrieve an array of review ids, and the other to initialize the review array.

1.3.2 Create the Review in the persistence

This algorithm is also responsible for creating the associated Comment, and Item, but not for CommentType. Please note that the comment_type_lu table is simply a look-up table.

This method is responsible for setting the creation user and modification user for the review instance. The creation user and modification user is identical to the passed in operation.

Before creating the review instance in the database, all the associated review properties should be validated.

Please see the Review Data Structure component about which properties should be validated, and the corresponding valid values. Review class in the Review Data Structure component might add validate() method to perform the data validation. If that, the validate() method will be called before inserting the instance into the database.

```
Impl note:

1. Get the review id and check that if it exists in the database. If exist, throw
DuplicateReviewEntityException.

2. get id from reviewIDGenerator, set it on review, and insert the review
properties into the review table

3. get all comments from review, generate id for each comment(using
reviewCommentIDGenerator), and insert the comment into the table.

4. get all the items from review, generate id for each item(using
reviewItemIDGenerator), and insert the item into the table.

5. for each item in the step 4, get all the item comments, and generate the id for
them(using reviewItemCommentIDGenerator), and insert them into the item_comment
table.

6. if any exception occurred, reset the review properties to  their initial values,
and roll back.

Note that: the source_id field represents the author id
```

1.3.3 Update Review in the persistence

This algorithm is also responsible for creating, deleting, updating the associated Items, Comments, and Item Comments. The algorithm is also responsible for set operator on Review#modificationUser.

Before updating the review instance in the database, all the associated review properties should be validated.

Please see the Review Data Structure component about which properties should be validated, and the corresponding valid values. Review class in the Review Data Structure component might add validate() method to perform the data validation. If that, the validate() method will be called before inserting the instance into the database.

```
Impl note:

1. get the review id, and check that it exists in the database. otherwise, throw
   ReviewEntityNotFoundException.

2. get the Review bean properties, and update them on "review" table.

3. select all the comment id from review_comment table which belong to the given
   review. Assume that idSet keeps the comment ids retrieved in this step.

4. get the review comments, for each comment, get the comment id,

if the id exists in idSet, update the comment, and remove the id from idSet.

if not exists, get the id(using reviewCommentIDGenerator), set the id on comment,
   and create the comment.

5. If idSet is not empty, iterate through the set, and remove the comment from
   review_comment table for each comment id

7. select all the review item ids from review_item table which belong to the given
   review.. Assume that itemIdSet keeps the items ids retrieved in this step.

8. get the review items, for each item id:

if the id exists in the itemIdSet, update the item, and remove the id from
   itemIdSet. Note that update the item involve creating, updating and deleting the
   item comments. t

If the id not exists, get the id from reviewItemIDGenerator, and create the item.
```

Note that creating item will involve creating all the associated item comments.

9. if itemIdSet is not empty, iterate through the set, for each id, remove it from review_item table. Additionally uploads associated with review items must be removed.

10. If any error occurred, roll back, and reset the review properties to their initial statuses..

1.3.4 Get Review from the persistence

Get the Review instance from the persistence with given id

Impl note:

1. if id is not positive, throw IllegalArgumentException

2. check that id exists in the database, if not exists, throw ReviewEntityNotFoundException

3. get all comment types from comment_type_lu table.

4. create a Review instance, select all the Review properties and Comments from the review, review_comment and set them on the instance.

Notice that the source_id represents Review#author.

The step 4 must be done in one SQL query

5. select all the review item and item comments from review_item and review_item_comment tables where review_id is equal to given id,

and create the Item instance for each row,

Note that the Item#getDocument() is identical review_item.upload_id

The step 5 must also be done in one SQL query.

1.3.5 Search for the review in the persistence

Search the persistence with given filter. If complete is false, the associated items and comments of the matching review will not be retrieved.

In the version 1.0, the filter supports at most five fields:

scorecardType --- the score card type, must be java.Long type

submission --- the review submission id, must be java.Long type

reviewer --- the author of the review, must be java.Long type

project --- the project id of the review, must be java.Long type

committed --- indicate if the review has been committed, must be java.lang.Integer type.

Either new Integer(1), representing committed, or new Integer(0), represent not committed

Impl note:

1. get an array of review ids by calling this.searchBundle.search(filter). If the returned CustomResultSet is empty, simply return empty array.

2. if complete is false, select all the review properties from review table:

select * from review where review_id IN (,,,,,) // the (,,,) is an array of review ids retrieved in step 1

Initialize an array of review with data retrieved in the above sql and return it.

Review#getAuthor() is identical to review.resource_id

```
2. if complete is true get all comment types  from comment_type_lu table

2.1 retrieve review properties and review comments from review , review_comment
tables

where review.review_id=review_comment.review_id and

review_id IN (....) order by review_id, review_comments.review_comment_id

Initialize the array of review with the data retrieved in the above sql and the
comment types retrieved in step 2

2.2 retrieve review items and item comments from review_item:

select review_id, item properties(document, question, etc), item comment properties
from review_item, review_item_comment where

review_item.review_id IN (.....) and

review_item.review_item_id=review_item_comment.review_item_id

order by review_id, review_item_id, review_item_comment_id

Initialize the reviews with items retrieved in the above sql and the comment types
retrieved in step 2.
```

1.3.6 Delete review from the persistence

This algorithm corresponds to InformixReviewPersistence#removeReview() method. It should remove respective records from review, review_comment, review_item, review_item_comment and upload tables. Please see implementation notes in TCUML for details.

1.4 Component Class Overview

Helper

Helper class for this component.

Changes in 1.2:

- Specified generic parameters for all generic types in the code.

- Added deleteEntities() overload for optimized record deletion.

Helper.DataType [abstract]

This class is a wrapper for type safe retrieval of values from a ResultSet. This class has been introduced to consist the behaviors of different databases and JDBC drivers so that always the expected type is returned (getObject(int) does not sufficiently do this job as the type of the value is highly database-dependent (e.g. for a BLOB column the MySQL driver returns a byte[] and the Oracle driver returns a Blob)).<p/>This class contains a private constructor to make sure all implementations of this class are declared inside Helper

Instances are provided to users via constants declared in Helper - so this class defines some kind of 'pseudo-enum' which cannot be instantiated externally

Helper.DateType

This class is a wrapper for type safe retrieval of values from a ResultSet. The values retrieved by the getValue(java.sql.ResultSet, int) implementation of this DataType are assured to be of type Date or to be null in case the ResultSet value was null.

Changes in 1.2:

- Added empty private constructor.

Helper.FloatType

This class is a wrapper for type safe retrieval of values from a ResultSet. The values retrieved by the getValue(java.sql.ResultSet, int) implementation of this DataType are assured to be of type Float or to be null in case the ResultSet value was null.

Changes in 1.2:

- Added empty private constructor.

Helper.LongType

This class is a wrapper for type safe retrieval of values from a ResultSet. The values retrieved by the getValue(java.sql.ResultSet, int) implementation of this DataType are assured to be of type Long or to be null in case the ResultSet value was null.

Changes in 1.2:

- Added empty private constructor.

Helper.StringType

This class is a wrapper for type safe retrieval of values from a ResultSet. The values retrieved by the getValue(java.sql.ResultSet, int) implementation of this DataType are assured to be of type String or to be null in case the ResultSet value was null.

Changes in 1.2:

- Added empty private constructor.

InformixReviewPersistence

This class is responsible for creating, updating, searching and deleting the review entities stored in the Informix database. Additionally, application users can also add comment for review and item, and get all the comment types from the persistence.

Changes in 1.2:

- Added removeReview() method.

- Updated deleteReviewItems() to remove associated uploads.

- Specified generic parameters for all generic types in the code.

- Made class thread safe by synchronizing getAllCommentTypes() methods.

LogMessage

Encapsulates the entry log data and generates consistent log messages.

Changes in 1.2:

- Made immutable fields final.

1.5 Component Exception Definitions

None

1.6 Thread Safety

This component is thread safe starting from the version 1.2.

InformixReviewPersistence is mutable, since it has mutable cachedCommentTypes. But thus class is thread safe when entities passed to it are used by the caller in thread safe manner, since cachedCommentTypes is used in the synchronized getAllCommentTypes() method only.

LogMessage has mutable logMessage field, and thus it is not thread safe. But InformixReviewPersistence always uses it in thread safe manner - LogMessage instances are never accessed from multiple threads.

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

Review Management 1.2 – defines ReviewPersistence interface implemented in this component.

Configuration Manager 2.2.1 – is used for configuring this component.

Logging Wrapper 1.2 – is used for logging errors and debug information.

Search Builder 1.3.3 – defines Filter, SearchBundle and SearchBundleManager used in this component for searching reviews in persistence.

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

ID Generator 3.0.2 – is used for generating IDs of reviews, review items and comments.

Data Validation 1.1.1 – defines ObjectValidator, NotValidator and NullValidator used in this component for validating reviews.

Database Abstraction 2.0 – defines CustomResultSet used in this component when extracting review data obtained from persistence.

DB Connection Factory 1.1 – is used for establishing database connections.

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.persistence
com.topcoder.management.review.persistence.logging

3.2 Configuration Parameters

3.2.1 Configuration of InformixReviewPersistence

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

Parameter	**Description**	**Values**
connection.name	The connection name used when retrieving connection from DB Connection Factory.	String. Not empty. Required.
connection. factory_namespace	The namespace used in DB Connection Factory component.	String. Not empty. Required.
connection.factory_class	The full class name of DBConnectionFactory implementation to be used.	String. Not empty. Required.
search_bundle_manager _namespace	The namespace of SearchBundleManager.	String. Not empty. Required.
search_bundle_name	The name of the SearchBundle to be used when searching for reviews.	String. Not empty. Required.

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

1. Drop all tables that you previously use for testing this component.
2. Use **test_files/sql/ddl.sql** to create all tables that are needed for this component.
3. Configure database connection details in **test_files/dbfactory.xml**
4. Run tests

   Drop tables with **test_files/sql/drop.sql.**

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.persistence.InformixReviewPersistence">
    <Property name="connection">
      <!-- the connection name to retrieve connection in DB Connection Factory, required -->
      <Property name="name">
        <Value>failureTest</Value>
      </Property>
      <!-- The DBConnectionFactory class name to be used, required -->
      <Property name="factory_namespace">
          <Value>com.topcoder.db.connectionfactory.DBConnectionFactoryImpl</Value>
      </Property>
      <Property name="factory_class">
          <Value>com.topcoder.db.connectionfactory.DBConnectionFactoryImpl</Value>
      </Property>
    </Property>
    <Property name="search_bundle_manager_namespace">
      <Value>com.topcoder.search.builder.SearchBundleManager</Value>
    </Property>
    <Property name="search_bundle_name">
      <Value>ReviewSearchBundle</Value>
    </Property>
  </Config>
</CMConfig>
```

4.3.2 API usage

API of this component is not expected to be used directly by the user. Instead InformixReviewPersistence is expected to be plugged into DefaultReviewManager defined in Review Management component.

Please see CS of Review Management component for a sample API usage of DefaultReviewManager.

5. Future Enhancements

None