Time Tracker Reject Reason V3.2 Component Specification

Note: The red sections are the differences from V3.1

1. Design

The Time Tracker Reject Reason custom component is part of the Time Tracker application. It provides an abstraction of reject reasons for many of the other components in the system. This component handles the persistence and other business logic required by the application.

This component consists of two layers, a DAO layer with two DAOs which provides the Create, Retrieve, Update, Delete, Enumerate and Search methods for their respective entity, and an Access layer which leverages the EJB's transaction management ability to manage transaction across all components of Time Tracker Application by using two stateless session beans with Container Managed Transaction. A POJO is used as Business Delegate and Service Locator at this layer to access the DAO's business methods.

Interfaces RejectReasonDAO and RejectEmailDAO are designed for pluggable persistence for reject reason entry and reject email entry. Implementation of Informix database is provided and any other data store can be easily plugged-in.

The actual value object entities, RejectReason and RejectEmail, are modeled as data beans that follow the JavaBean naming convention. Other classes used by the session beans are also Serializable for the EJB's specification and easy persistence and network transfer.

1.1 Design Patterns

Business Delegate pattern is used by both RejectReasonDAOAccess and RejectEmailDAOAccess. They delegate all business methods to corresponding session beans.

Service Locator pattern is also used by both RejectReasonDAOAccess and RejectEmailDAOAccess. They locate the service providers – the local interfaces of the session beans.

Data Access Object pattern is used by both RejectReasonDAO and RejectEmailDAO

The Strategy Pattern RejectReasonDAO and RejectEmailDAO are all strategies for persisting and retrieving information from a data store

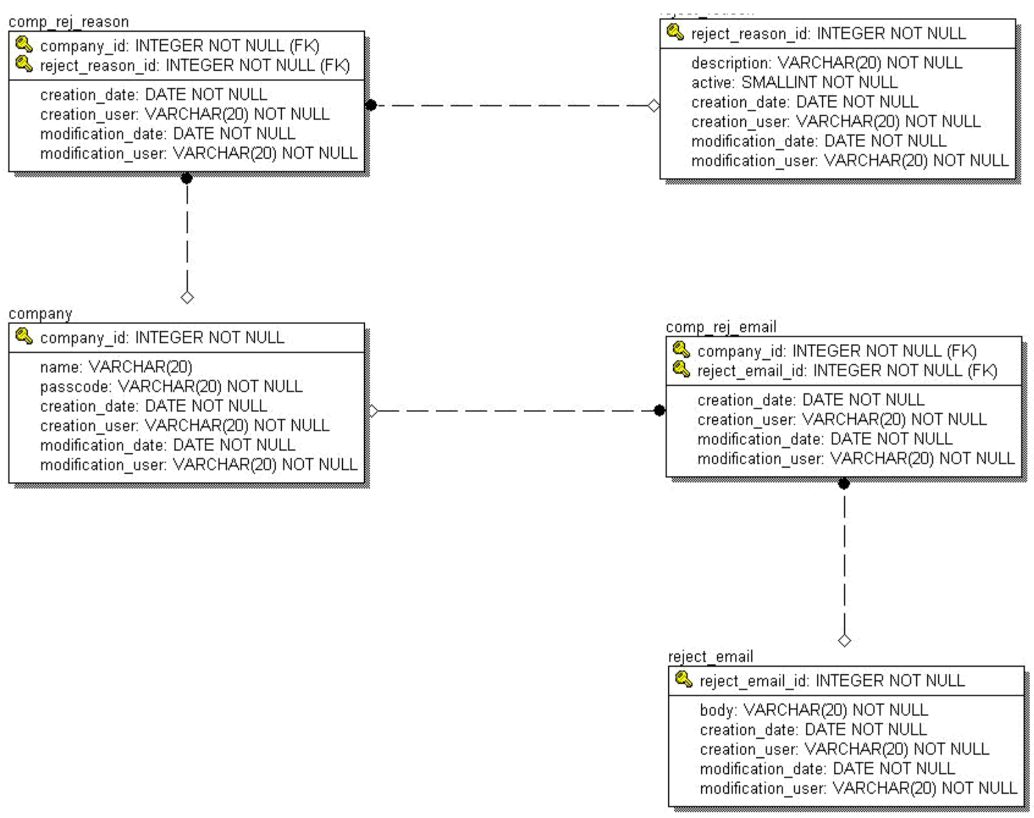
1.2 Industry Standards

JDBC

1.3 Required Algorithms

1.*3.1 Database Schema:*

The DbDAOs that we will be modeling will be accessing these particular tables:


comp_rej_reason
company_id: INTEGER NOT NULL (FK)
reject_reason_id: INTEGER NOT NULL (FK)
creation_date: DATE NOT NULL
creation_user: VARCHAR(20) NOT NULL
modification_date: DATE NOT NULL
modification_user: VARCHAR(20) NOT NULL
reject_reason_id: INTEGER NOT NULL
description: VARCHAR(20) NOT NULL
active: SMALLINT NOT NULL
creation_date: DATE NOT NULL
creation_user: VARCHAR(20) NOT NULL
modification_date: DATE NOT NULL
modification_user: VARCHAR(20) NOT NULL
company
company_id: INTEGER NOT NULL
name: VARCHAR(20)
passcode: VARCHAR(20) NOT NULL
creation_date: DATE NOT NULL
creation_user: VARCHAR(20) NOT NULL
modification_date: DATE NOT NULL
modification_user: VARCHAR(20) NOT NULL
comp_rej_email
company_id: INTEGER NOT NULL (FK)
reject_email_id: INTEGER NOT NULL (FK)
creation_date: DATE NOT NULL
creation_user: VARCHAR(20) NOT NULL
modification_date: DATE NOT NULL
modification_user: VARCHAR(20) NOT NULL
reject_email
reject_email_id: INTEGER NOT NULL
body: VARCHAR(20) NOT NULL
creation_date: DATE NOT NULL
creation_user: VARCHAR(20) NOT NULL
modification_date: DATE NOT NULL
modification_user: VARCHAR(20) NOT NULL

1.3.2 SQL Statements

The implementation notes for the DAO methods contain details on the SQL statements that need to be executed. And the developer is expected to have enough basic knowledge of JDBC to be able to implement the needed functionality.

1.3.2 Searching

Searching is executed in this component by using the TopCoder Software Component Search Builder which supports both simple and complex searches against a persistent data store. The Search Builder component allows a user to programmatically create searches. It will translate them into the appropriate query string, execute the query against the configured data store connection, and return the result set to use for further processing (See more at the documents of Search Builder 1.3.1 http://software.topcoder.com/catalog/c_component.jsp?comp=11884906).

By leveraging Search Builder, two convenience classes RejectReasonSearchBuilder and RejectEmailSearchbuilder are used respectively to build specific filters for search reject reason and reject email recorders in database. These built filters will be passed to corresponding search method in DbRejectReasonDAO or in DbRejectEmailDAO to perform the search job. And then the search method of DbRejectReasonDAO or DbRejectEmailDAO will delegate the search job to the search method of SearchBundle class in the Search Builder, which is responsible for creating needed SQL query string and executing the query against the database connection.

1.3.3 Auditing

Audit flag is used in every method may change the data in persistence. Instances of com.topcoder.timetracker.audit.AuditManager are hold by DAOs to perform the audit tasks. In the implementation notes of methods of DAOs the creation of AuditHead which is required to insert an audit record is depicted, and the following clarifies how to create AuditDetail objects to add to the AuditHead. When creating a new entry (audit header action type is CREATE), then the oldValue for each detail is null. When deleting an entry (audit header action type is DELETE), then the newValue for each detail is null. When updating an entry, the old value is retrieved from the data store to populate the AuditDetail's old value.

And both AuditManager and this component leverage the EJB Container to manage transaction, the operation on this component will join transaction existed or create a new transaction when there is none of it, that is to say the operations on this component and the corresponding auditing operation will be in the same transaction. And when operation fails, the EJB Container will roll back the transaction to ensure data integrity.

1.3.4 Configuration of TXDatasource

DataSource should be configured in DBConnectionFactory by retrieving the connection that it uses from the configured TXDatasource in JBoss.. A sample configuration file for DBConnectionFactory will looks like:

```
<Config name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
    <Property name="connections">
        <Property name="default">
            <Value>time_tracker_application</Value>
        </Property>
        <Property name="time_tracker_application">
            <Property name="producer">
                <Value>
                  com.topcoder.db.connectionfactory.producers.JNDIConnectionProducer
               </Value>
            </Property>
            <Property name="parameters">
                <Property name="jndi_name">
                    <Value>java:jdbc/TXDatasource</Value>
                </Property>
            </Property>
        </Property>
    </Property>
</Config>
```

1.4 Component Class Overview

Package com.topcoder.timetracker.rejectreason

RejectReason

This bean represents a possible reason why a time or expense entry may be rejected by the Project Manager.

RejectEmail

This bean represents an email template that is sent to the contractor when a time or expense entry has been rejected by the Project Manager.

RejectReasonDAO [interface]

This interface defines the necessary methods that a RejectReason DAO should support. Create, Retrieve, Update, Delete and Enumerate (CRUDE) methods are provided.

RejectEmailDAO [interface]

This interface defines the necessary methods that a RejectEmail DAO should support. Create, Retrieve, Update, Delete and Enumerate (CRUDE) methods are provided.

RejectReasonSearchBuilder

This is a convenience class that may be used to build filters for performing searches in the RejectReasonDAO. It provide the methods to build fundamental search filter as well as methods to compose filters with AND, OR and NOT logic

RejectEmailSearchBuilder

This is a convenience class that may be used to build filters for performing searches in the RejectEmailDAO. It provide the methods to build fundamental search filter as well as methods to compose filters with AND, OR and NOT logic

Package com.topcoder.timetracker.rejectreason.informix

DbRejectReasonDAO

This class is an Informix database implementation of the RejectReasonDAO interface. It is capable of persisting and retrieving RejectReason information from the database. Create, Retrieve, Update, Delete and Enumerate (CRUDE) methods are provided.

DbRejectEmailDAO

This class is an Informix database implementation of the RejectEmailDAO interface. It is capable of persisting and retrieving RejectEmail information from the database. Create, Retrieve, Update, Delete and Enumerate (CRUDE) methods are provided.

Package com.topcoder.timetracker.rejectreason.ejb

RejectReasonDAOLocalHome [interface]

LocalHome interface for the RejectReasonDAO. It contains only a single no-param create method that produces an instance of the local interface. It is used to obtain a handle to the Stateless SessionBean.

RejectEmailDAOLocalHome [interface]

LocalHome interface for the RejectEmailDAO. It contains only a single no-param create method that produces an instance of the local interface. It is used to obtain a handle to the Stateless SessionBean.

RejectReasonDAOLocal [interface]

This is a Local interface for RejectReasonDAO. It contains exactly the same methods as RejectReasonDAO interface.

RejectEmailDAOLocal [interface]

This is a Local interface for RejectEmailDAO. It contains exactly the same methods as RejectEmailDAO interface.

RejectReasonDAOAccess

This is a Business Delegate/Service Locator that may be used within a J2EE application. It is responsible for looking up the local interface of the SessionBean for RejectReasonDAO, and delegating any calls to the session bean.

RejectEmailDAOAccess

This is a Business Delegate/Service Locator that may be used within a J2EE application. It is responsible for looking up the local interface of the SessionBean for RejectEmailDAO, and delegating any calls to the session bean.

RejectReasonSessionBean [session bean]

This is a Stateless SessionBean that is used to provide business services to manage RejectReason within the Time Tracker Application. It contains the same methods as RejectReasonDAO, and delegates to an instance of RejectReasonDAO

RejectEmailSessionBean [session bean]

This is a Stateless SessionBean that is used to provide business services to manage RejectEmail within the Time Tracker Application. It contains the same methods as RejectEmailDAO, and delegates to an instance of RejectEmailDAO

1.5 Component Exception Definitions

1. Standard Exceptions

IllegalArgumentException
This exception is thrown when the parameters passed to a function invalid. (e.g. empty string, null argument etc.) Please refer to the method exception documentation for more details. Empty string means the length of string is zero after it is trimmed.

2. Custom Exceptions

RejectReasonDAOException

This exception extends the BaseException and is thrown from RejectReasonDAO interface. It is also parent exception of RejectReasonNotFoundException and RejectReasonDAOConfigurationException.

RejectReasonNotFoundException

This exception extends RejectReasonDAOException and is thrown if the involved RejectEmail was not found in the data store.

RejectReasonDAOConfigurationException

This exception extends RejectReasonDAOException and is thrown from constructor of RejectReasonSessionBean when any of needed environment variables to create an instance of RejecReasonDAO can't be found or with an illegal value.

RejectEmailDAOException

This exception extends the BaseException and is thrown from RejectEmailDAO interface. It is also parent exception of RejectEmailNotFoundException and DbRejectEmailDAOConfigurationException.

RejectEmailNotFoundException

This exception extends RejectEmailDAOException and is thrown if the involved RejectEmail was not found in the data store.

RejectEmailDAOConfigurationException

This exception extends RejectEmailDAOException and is thrown from constructor of RejectEmailSessionBean when any of needed environment variables to create an instance of RejecEmailDAO can't be found or with an illegal value.

1.6 Thread Safety

In this component that both RejectReasonDAO and RejectEmailDAO are immutable and thread safe. And the two helpers, RejectReasonSearchBuilder and RejectEmailSearchBuilder are stateless and also thread safe. But the Value Objects, RejectReason and RejectEmail are both mutable and not thread safe, it is advised to be used concurrently only for read-only access. Otherwise, each thread is expected to work with its own instance. The developer should be aware that the thread-safe issue extends to the resources used by this component.
In this version Container Managed Transaction is leveraged to ensure the thread safe of access database. The database connection is configured as TXDATASOURCE in the deploy descriptor of the session bean, and the session bean is configured with REQUIRED trans-attribute in the deploy description (see the ejb-jar.xml for more details).

2. Environment Requirements

2.1 Environment

Development language: Java1.4
Compile target: Java1.4

2.2 TopCoder Software Components

Configuration Manager 2.1.5 – Configuration Manager is used to configure data in this component.
Object Factory 2.0.1 – It is used to create AuditManager objects in this design.
DB Connection Factory 1.0 – It is used to load configurable database connection
ID Generator 3.0 – It is used to generate ids for entities need in this component.
Search Builder 1.3.1 – It is used to build SQL statements with different simple condition and combined conditions and to execute SQL query on database with these statements.
BaseException 1.0 – It is extended by all the custom exceptions of this component.
Time Tracker Common 3.1 – is used to provide the TimeTrackerBean base class.

JNDI Context Utility 1.0 - is used by the business delegates to look up the home interface of the session bean.

Time Tracker Audit 3.2 – is used to perform the optional audit.

2.3 Third Party Components

None

3. Installation and Configuration

3.1 Package Name

com.topcoder.timetracker.rejectreason

com.topcoder.timetracker.rejectreason.informix

com.topcoder.timetracker.rejectreason.ejb

3.2 Configuration Parameters

(excluding those in the deploy descriptor, which can be seen at ejb-jar.xml)

Parameter	Description	Values
context_name	The context name used by JNDIUtils. Option	Must be non-empty string.

3.3 Dependencies Configuration

None

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

None

4.3 Demo

4.3.1 Manage RejectReason

```
// The user
```

```
String username = "topcoder";

// create RejectReasonDAOAccess
RejectReasonDAOAccess rejectReasonDAOAccess = new
RjectReasonDAOAccess("test");

// create RejectReason with audit
long companyId = 1;
String description = "something";
boolean active = true;
RejectReason rejectReason_1 = new RejectReason();
rejectReason_1.setCompanyId(companyId);
rejectReason_1.setDescription(description);
rejectReason_1.setActive(active);
boolean isAudit = true;
long rejectReason_1_Id =
rejectReasonDAOAccess.createRejectReason(rejectReason_1,username,
isAudit).getId();

// create another RejectReason without audit
RejectReason rejectReason_2 = new RjectReason();
rejectReason_2.setCompanyId(2);
rejectReason_2.setDescription("another_thing");
rejectReason_2.setActive(true);
isAudit = false;
long rejectReason_2_Id =
rejectReasonDAOAccess.createRejectReason(rejectReason_2,username,
isAudit).getId();
isAudit = true;

// Retrieve RejectReason
RejectReason rr =
rejectReasonDAOAccess.retrieveRejectReason(rejectReason_1_id);

// Update RejectReason
rr.setDescription("new_thing");
rejectReasonDAOAccess.updateRejectReason(rr, username, isAudit);

// Delete RejectReason
rejectReasonDAOAccess.deleteRejectReason(rr, isAudit);

// List RejectReason
rejectReasonDAOAccess.listRejectReasons();

// Search RejectReason
// create RejectReasonSearchBuilder
RejectReasonSearchBuilder rejectReasonSearchBuilder =
new RejectReasonSearchBuilder();

// simple search
Filter filter =
rejectReasonSearchBuilder.createdByUserFilter(username);
RejectReason[] rejectReasonResults =
rejectReasonDAOAccess.searchRejectReason(filter);

// complex search;
Filter filter_1 =
rejectReasonSearchBuilder.createdByUserFilter(username);
Filter filter_2 = rejectReasonSearchBuilder.not(filter1);
filter = rejectReasonSearchBuilder.and(filter1, filter2);
// note: this search should return an empty result set
rejectReasonResults =
rejectReasonDAOAccess.searchRejectReason(filter);
```

4.3.2 *Manage RejectEmail*

```
// The user
String username = "topcoder";

// create RejectEmailDAOAccess
RejectEmailDAOAccess rejectEmailDAOAccess = new
RjectEmailDAOAccess("test");

// create RejectEmail with audit
long companyId = 1;
String body= "something";
RejectEmail rejectEmail_1 = new RejectEmail();
rejectEmail_1.setCompanyId(companyId);
rejectEmail_1.setBody(description);
isAudit = true;
long rejectEmail_1_Id =
rejectEmailDAOAccess.createRejectEmail(rejectEmail_1, username,
isAudit).getId();

// create another RejectEmail without audit
RejectEmail rejectEmail_2 = new RjectReason();
rejectEmail_2.setCompanyId(2);
rejectEmail_2.setBody("another_thing");
isAudit = false;
long rejectEmail_2_Id =
rejectEmailDAOAccess.createRejectEmail(rejectEmail_2, username,
isAudit).getId();
isAudit = true;

// Retrieve RejectEmail
RejectEmail rr =
rejectEmailDAOAccess.retrieveRejectEmail(rejectEmail_1_id);

// Update RejectEmail
rr.setBody("new_thing");
rejectEmailDAOAccess.updateRejectEmail(rr, username, isAudit);

// Delete RejectEmail
rejectEmailDAOAccess.deleteRejectEmail(rr, isAudit);

// List RejectEmail
rejectEmailDAOAccess.listRejectEmails();

// Search RejectEmail
// create RejectEmailSearchBuilder
RejectEmailSearchBuilder rejectEmailSearchBuilder =
new RejectEmailSearchBuilder();

// simple search
Filter filter =
rejectEmailSearchBuilder.createdByUserFilter(username);
RejectEmail[] rejectEmailResults =
rejectEmailDAOAccess.searchRejectEmail(filter);

// complex search;
Filter filter_1 =
rejectEmailSearchBuilder.createdByUserFilter(username);
Filter filter_2 = rejectEmailSearchBuilder.not(filter1);
filter = rejectEmailSearchBuilder.and(filter1, filter2);
// note: this search should return an empty result set
rejectEmailResults =
rejectEmailDAOAccess.searchRejectEmail(filter);
```

5. Future Enhancements

Other database systems maybe plugged in for some client environments. Multiple user stores may be used for the same client environment.