Terms Of Use 1.0 Component Specification

1. Design

The component defines data model and functionality for managing Terms Of Use and their relations to users and contest roles. The existing Terms Of Use functionality is implemented in EJBs, which brings unnecessary complications to the applications using them. Additionally several enhancements are to be done to the existing functionality.

This component rewrites the existing Terms Of Use functionality with removal of EJBs and adding several new methods. The grouping of project/role to terms of use is done additionally.

1.1 Design Patterns

1.1.1 Strategy

TermsOfUseDaoImpl, UserTermsOfUseDaoImpl, ProjectTermsOfUseDaoImpl are Strategies of TermsOfUseDao, UserTermsOfUseDao, ProjectTermsOfUseDao respectively in some external context.

1.1.2 DAO/DTO

TermsOfUseDao, UserTermsOfUseDao, ProjectTermsOfUseDao are DAOs for the DTO TermsOfUse and TermsOfUseType.

1.2 Industry Standards

SQL
JDBC

1.3 Required Algorithms

Please refer to TCUML method doc for details not listed below.

1.3.1 Logging

Logging will happen in all public methods, but not in setters/getters.
- Method entrance and exit will be logged with DEBUG level.
  Entrance format: [Entering method {className.methodName}]
  Exit format: [Exiting method {className.methodName}]. Only do this if there are no exceptio ns.
- Method request and response parameters will be logged with DEBUG level
  Format for request parameters: [Input parameters[{request_parameter_name_1}: { request_parameter_value_1}, {request_parameter_name_2}: { request_parameter_value_2}, etc.)]]

  Format for the response: [Output parameter {response_value}]. Only do this if there are no e xceptions and the return value is not void.
- All exceptions will be logged at ERROR level, and automatically log inner exceptions as well.
  Format: Simply log the text of exception: [Error in method {className.methodName}: Details {error details}]

Constructors TermsOfUse and TermsOfUseType classes do not require logging.
The access to the logger for dao implementations should be done by calling protected getter #getLog().

1.3.2 Database schema update

There are two new fields to be added to the tables:

- "group_ind" field should be added to the project_role_terms_of_use_xref table. This is an integer

field that is used to group links by groups. The terms of use entities with the same group index form the terms of use group. The user can take the specific role for the project only if he has all terms of use from at least one group.

- "member_agreeable" field should be added to the terms_of_use table. This is a field of the same type as the "electronically_signable" field and indicates whether a user can agree to the terms of use (either electronically or by sending a hard copy).

The updated 01_common_oltp_main_schema.sql schema is provided in this design.

1.4 Component Class Overview

TermsOfUse:
This class defines simple entity of terms of use. It is associated with the database table terms_of_use. It provides model fields and getters/setters for them.

TermsOfUseType:
This class defines simple entity of terms of use type. It provides model fields and getters/setters for them.

TermsOfUseDao:
This interface defines the dao for manipulating the TermsOfUse entity. It simply provides CRUD operations on this entity and retrieval operation by the terms of use type. It also provides retrieval and update operation for terms of use type.

UserTermsOfUseDao:
This interface defines the dao for manipulating the TermsOfUse to User links. It simply provides CRUD operations on the links and several helper operations whether user has specific user terms or ban for them.

ProjectTermsOfUseDao:
This interface defines the dao for manipulating the TermsOfUse to Project links. It simply provides create/delete and search operations on the links.

BaseTermsOfUseDao:
This class is the base class defined for the daos of this component. It simply provides common logger, database connection factory and id generator. The subclasses access these fields by protected getters.

TermsOfUseDaoImpl:
This class is the default implementation of TermsOfUseDao. It utilizes the DB Connection Factory to get access to the database. The configuration is done by the Configuration API.

UserTermsOfUseDaoImpl:
This class is the default implementation of UserTermsOfUseDao. It utilizes the DB Connection Factory to get access to the database. The configuration is done by the Configuration API.

ProjectTermsOfUseDaoImpl:
This class is the default implementation of ProjectTermsOfUseDao. It utilizes the DB Connection Factory to get access to the database. The configuration is done by the Configuration API.

1.5 Component Exception Definitions

TermsOfUseDaoException:
This exception is thrown, when any exception occurred while executing the dao operations.

TermsOfUsePersistenceException:
This exception is thrown, when any generic exception occurred with persistence.

EntityNotFoundException:
This exception is thrown, when the entity in the database is not found.
UserBannedException:
This exception is thrown, when user was banned to perform specific operation.
TermsOfUseDaoConfigurationException:
This exception is thrown, when there is any problem with the configuration of the component. It is thrown in the constructors of the dao implementations.

1.6 Thread Safety

The component is thread - safe. The configuration of DAO implementations is done in constructor in thread – safe manner. The initialized parameters are never changed. DAO implementations are immutable and thread – safe, the usage of database connection factory and id generator is thread – safe. The entities in this component are not thread – safe, but only the entity instances are shared only to one thread. Under these conditions the components is thread – safe.

2. Environment Requirements

2.1 Environment

Development language: Java 1.5

Compile target: Java 1.5

2.2 TopCoder Software Components

- Base Exception 2.0 – used to define exceptions in the component
- Configuration API 1.0 – used for configuration in the component.
- DB Connection Factory 1.1.0 – used for obtaining connection to the database.
- ID Generator 3.0.0 – is used for id generation.
- Logging Wrapper 1.2.0 – used for logging

2.3 Third Party Components

None.

3. Installation and Configuration

3.1 Configuration Parameters

Configuration for UserTermsOfUseDao, ProjectTermsOfUseDao:

Name	Description	Value
dbConnectionFactoryConfig	The configuration for DBConnectionFactoryImpl, used to obtain connection to the database.	com.topcoder.configuration.ConfigurationObject. It cannot be null. Required.
loggerName	The logger name used to perform logging.	java.lang.String. It cannot be null. Required

Configuration for TermsOfUseDao:

Name	Description	Value
dbConnectionFactoryConfig	The configuration for DBConnectionFactoryImpl, used to obtain connection to the database.	com.topcoder.configuration.ConfigurationObject. It cannot be null. Required.
loggerName	The logger name used to perform logging.	java.lang.String. It cannot be null. Required
idGeneratorName	The id name to generate ids for the entities.	java.lang.String. It cannot be null. Required.

3.2 Dependencies Configuration

None.

3.3 Package Structure

com.cronos.termsofuse.model
com.cronos.termsofuse.dao
com.cronos.termsofuse.dao.impl

4. Usage Notes

4.1 Required steps to test the component

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

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

  b. Update test_files/config.xml and

  test_files/com/topcoder/db/connectionfactory/DBConnectionFactoryImpl.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

The sample configuration file for database connection and logging is provided below

```
<CMConfig>
  <Config name="termsOfUseDao">
    <Property name="dbConnectionFactoryConfig">
      <Property name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
        <Property name="connections">
          <Property name="default">
            <Value>InformixJDBCConnection</Value>
          </Property>
          <Property name="InformixJDBCConnection">
            <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/common_oltp:informixserver=ol_topcoder
                </Value>
              </Property>
              <Property name="user">
                <Value>informix</Value>
              </Property>
              <Property name="password">
                <Value>123456</Value>
              </Property>
            </Property>
          </Property>
        </Property>
```

```
    </Property>
  </Property>
  <Property name="loggerName">
    <Value>loggerName</Value>
  </Property>
  <Property name="idGeneratorName">
    <Value>idGenerator</Value>
  </Property>
</Config>
<Config name="userTermsOfUseDao">
  <Property name="dbConnectionFactoryConfig">
    <Property name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
      <Property name="connections">
        <Property name="default">
          <Value>InformixJDBCConnection</Value>
        </Property>
        <Property name="InformixJDBCConnection">
          <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/common_oltp:informixserver=ol_topcoder
              </Value>
            </Property>
            <Property name="user">
              <Value>informix</Value>
            </Property>
            <Property name="password">
              <Value>123456</Value>
            </Property>
          </Property>
        </Property>
      </Property>
    </Property>
  </Property>
  <Property name="loggerName">
    <Value>loggerName</Value>
  </Property>
</Config>
<Config name="projectTermsOfUseDao">
  <Property name="dbConnectionFactoryConfig">
    <Property name="com.topcoder.db.connectionfactory.DBConnectionFactoryImpl">
      <Property name="connections">
        <Property name="default">
          <Value>InformixJDBCConnection</Value>
        </Property>
        <Property name="InformixJDBCConnection">
          <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/common_oltp:informixserver=ol_topcoder
              </Value>
            </Property>
            <Property name="user">
              <Value>informix</Value>
            </Property>
            <Property name="password">
              <Value>123456</Value>
            </Property>
```

```
            </Property>
          </Property>
        </Property>
      </Property>
    </Property>
    <Property name="loggerName">
      <Value>loggerName</Value>
    </Property>
  </Config>
</CMConfig>
```

Consider the following data is presented in the database(the create_date and modify_date are omitted for simplicity):

terms_of_use:

terms_of_use_id	terms_text	terms_of_use_type_id	title	electronically_agreeable	url	member_agreeable
1	text1	1	t1	0	url1	1
2	text2	1	t2	1	url2	0
3	text3	1	t3	1	url3	0
4	text4	2	t4	0	url4	1

user_terms_of_use_xref:

user_id	terms_of_use_id
1	1
1	2
2	1
3	3

user_terms_of_use_ban_xref:

user_id	terms_of_use_id
1	3
2	4
3	4

project_role_terms_of_use_xref:

project_id	resource_role_id	terms_of_use_id	sort_order	group_ind
1	1	1	1	0
1	1	2	2	0
1	1	3	3	0
1	2	1	1	1
2	1	2	1	2

4.3.2 Sample Component Usage

This demo shows the usage of daos and sample results.

```
// Create the configuration object
ConfigurationObject configurationObject = TestsHelper.getConfig(TestsHelper.CONFIG_TERMS);
// Instantiate the dao implementation from configuration defined above
TermsOfUseDao termsOfUseDao = new TermsOfUseDaoImpl(configurationObject);

// Create simple TermsOfUse to persist
TermsOfUse terms = new TermsOfUse();
```

```
terms.setTermsOfUseTypeId(3);
terms.setTitle("t5");
terms.setElectronicallySignable(true);
terms.setUrl("url5");
terms.setMemberAgreeable(false);

// Persist the TermsOfUse
terms = termsOfUseDao.createTermsOfUse(terms, "");
```

The terms_of_use contents should be following:

terms_of_use_id	terms_text	terms_of_use_type_id	title	electronically_agreeable	url	member_agreeable
1	text1	1	t1	0	url1	1
2	text2	1	t2	1	url2	0
3	text3	1	t3	1	url3	0
4	text4	2	t4	0	url4	1
5		3	t5	1	url5	0

```
// Set terms of use text
termsOfUseDao.setTermsOfUseText(terms.getTermsOfUseId(), "text5");
```

The terms_of_use contents should be following:

terms_of_use_id	terms_text	terms_of_use_type_id	title	electronically_agreeable	url	member_agreeable
1	text1	1	t1	0	url1	1
2	text2	1	t2	1	url2	0
3	text3	1	t3	1	url3	0
4	text4	2	t4	0	url4	1
5	text5	3	t5	1	url5	0

```
// Get terms of use text. This will return "text5".
String termsOfUseText = termsOfUseDao.getTermsOfUseText(terms.getTermsOfUseId());

// Update some information for TermsOfUse
terms.setMemberAgreeable(true);

// And update the TermsOfUse
terms = termsOfUseDao.updateTermsOfUse(terms);
```

The terms_of_use contents should be following:

terms_of_use_id	terms_text	terms_of_use_type_id	title	electronically_agreeable	url	member_agreeable
1	text1	1	t1	0	url1	1
2	text2	1	t2	1	url2	0
3	text3	1	t3	1	url3	0
4	text4	2	t4	0	url4	1
5	text5	3	t5	1	url5	1

```
// Retrieve some terms of use. The third row will be returned
terms = termsOfUseDao.getTermsOfUse(3);

// Delete terms of use
termsOfUseDao.deleteTermsOfUse(5);
```

The terms_of_use contents should be following:

terms	terms_text	terms_of	title	electronically	url	membe

_of_use_id		_use_type_id		_agreeable		r_agreeable
1	text1	1	t1	0	url1	1
2	text2	1	t2	1	url2	0
3	text3	1	t3	1	url3	0
4	text4	2	t4	0	url4	1

```
// Retrieve all terms of use. All rows will be returned
List<TermsOfUse> allTerms = termsOfUseDao.getAllTermsOfUse();

// Create the configuration object
configurationObject = TestsHelper.getConfig(TestsHelper.CONFIG_USER_TERMS);
// Instantiate the dao implementation from configuration defined above
UserTermsOfUseDao userTermsOfUseDao = new UserTermsOfUseDaoImpl(configurationObject);

// Create user terms of use to user link
userTermsOfUseDao.createUserTermsOfUse(3, 2);
```

The user_terms_of_use_xref contents should be following:

user_id	terms_of_use_id
1	1
1	2
2	1
3	3
3	2

```
// Remove user terms of use to user link
userTermsOfUseDao.removeUserTermsOfUse(3, 3);
```

The user_terms_of_use_xref contents should be following:

user_id	terms_of_use_id
1	1
1	2
2	1
3	2

```
// Retrieve terms of use. This will return user terms with ids 1 and 2.
List<TermsOfUse> termsList = userTermsOfUseDao.getTermsOfUseByUserId(1);

// Retrieve users by terms of use. This will return ids 1 and 3.
List<Long> userIdsList = userTermsOfUseDao.getUsersByTermsOfUseId(2);

// Check whether user has terms of use. Will return false
boolean hasTerms = userTermsOfUseDao.hasTermsOfUse(3, 3);

// Check whether user has terms of use ban. Will return true
boolean hasTermsBan = userTermsOfUseDao.hasTermsOfUseBan(1, 3);

// Create the configuration object
configurationObject = TestsHelper.getConfig(TestsHelper.CONFIG_PROJECT_TERMS);
// Instantiate the dao implementation from configuration defined above
ProjectTermsOfUseDao projectTermsOfUseDao = new ProjectTermsOfUseDaoImpl(configurationObject);

// Create user terms of use to project link
projectTermsOfUseDao.createProjectRoleTermsOfUse(2, 2, 3, 2, 0);
```

The project_role_terms_of_use_xref contents should be following:

project_id	resource_role_id	terms_of_use_id	sort_order	group_ind

1	1	1	1	0
1	1	2	2	0
1	1	3	3	0
1	2	1	1	1
2	1	2	1	2
2	2	3	2	0

```
// Remove user terms of use to project link
projectTermsOfUseDao.removeProjectRoleTermsOfUse(2, 2, 3, 0);
```

The project_role_terms_of_use_xref contents should be following:

project_id	resource_role_id	terms_of_use_id	sort_order	group_ind
1	1	1	1	0
1	1	2	2	0
1	1	3	3	0
1	2	1	1	1
2	1	2	1	2

```
// Get terms of use with non-member-agreeable terms
// Will return two lists:
// 1st with ids: 1,2,3
// 2nd with ids: 1
Map<Integer, List<TermsOfUse>> termsGroupMap = projectTermsOfUseDao.getTermsOfUse(1, new int[] {1, 2}, true);

// Get terms of use without non-member-agreeable terms
// Will return one list:
// 1st with ids: 1
termsGroupMap = projectTermsOfUseDao.getTermsOfUse(1, new int[] {1, 2}, false);
```

5. Future Enhancements

None at the moment