Database Abstraction 2.0 Requirements Specification

1. Scope

1.1 Overview

The Database Abstraction component is intended to provide a common set of Java classes that abstract the mapping of database data types to the corresponding java objects. Mappings are provided for Oracle, Informix and MS SQLServer.

This enhancement of the component adds a type conversion mechanism, so that data retrieval will be independent of JDBC driver implementations.

Version 2.0 will upgrade the component to use the best practice of JDK 1.5 (e.g. Generic), latest TC Component Standard and JDBC 3.0.
Currently TCUML contains many leftovers from ZUML conversion, e.g. <p> tags in documentation, <<realize>> stereotypes, wrong colors of exceptions, etc. Such issues in the documentation must be cleaned up in this update.
You should synchronize documentation with the existing source code.

1.2 Logic Requirements

1.2.1 Compatibility

N/A

1.2.2 Type Conversion

CustomResultSet should be able to automatically convert data types with the getters. The conversion mechanism should mimic the result set implemented in JDBC drivers.

Currently the component only supports whatever data type is used from the JDBC result set that is used to produce the CustomResultSet. For example, if the result set uses Timestamp for a column it should also be able to return the data as Date or Time. In another scenario an underlying byte array should be able to support a binary stream.

1.2.3 JDBC 3 New Features

http://download.oracle.com/javase/1.5.0/docs/guide/jdbc/getstart/appendixB.html#1006300

- Ability to have multiple open ResultSet objects
- Holdable cursor support
- BOOLEAN data type
- Making internal updates to the data in Blob and Clob objects
- Retrieving and updating the object referenced by a Ref object
- DATALINK/URL data type
- Rowsets (A Feature Introduced in the JDBC 2.0 Optional Package)

Other features are out of scope.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

The current TopCoder Statistics portion of the website uses a generic result set container that runs predefined queries. The data is captured by the result set container, once the necessary data type mapping takes place, and then passes the collection of data to the viewing controller. At this point JSPs take the data and render it for clients to view.

1.5 Future Component Direction

None.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

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

2.1.4 Package Structure

com.topcoder.util.sql

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

None.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

JDBC 3.0

3.2.2 TopCoder Software Component Dependencies:

None.

**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:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- 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.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the ‘Documentation’ tab of TCUML.