Search Builder 1.3 Requirements Specification

1. Scope

1.1 Overview

The Search Builder component provides an API for both simple and complex searches against a persistence data store (LDAP, database). The component allows a user to programmatically create searches and will translate them into the appropriate query string for the data store, execute the query against the configured data store connection and return the result set to the user for further processing.

This version aims to fix two issues with the previous implementation, with no functional additions.

1.2 Logic Requirements

1.2.1 Compatibility

Compatibility of the SearchBundle, SearchBundleManager and the filter package should be guaranteed. The API of the implementation oriented classes, including ConnectionStrategy, ConnectionInformation, SearchStringBuilder can be altered if necessary.

1.2.2 Connections

The component should not leave any connections open. All connections must be closed after the search is executed.

The invocation of the search should be stateless. Multiple threads should be able to execute searches concurrently over separate connections.

1.2.3 Prepared Statement

The methods that take in a Filter must use prepared statement to execute the searches. In general there should be a way to bind the parameters after the search string is built. The changes made to satisfy this requirement should be transparent to the end user.

1.2.4 Is Null Filter

An IS NULL type of filter should be added. The filter will not take any values, and will search for any record which contains null for the specified parameter name.

1.3 Required Algorithms

No specific algorithms are required.

1.4 Example of the Software Usage

A web-based search page will be developed for a client and will utilize the search criteria component to build and execute searches against either a database or LDAP and display the results.

1.5 Future Component Direction

Subcomponents for different types of persistence stores may be developed.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

None.

2.1.3 *Environment Requirements*

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

2.1.4 *Package Structure*

com.topcoder.search.builder

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?*

None.

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 Poseidon.