[TopCoder Software]

Requirements Specification

1. Scope

1.1 Overview

The Object Formatter component provides a generic API for formatting java objects. This component can be used to centralize data formatting within an application instead of having each "tier" of the application define its own formatting methods.

1.2 Logic Requirements

1.2.1 Standard Formatting

- Provide the ability to simplify the process of formatting strings, numbers and dates.
- The underlying formatting objects for these standard objects can wrap the existing formatting capabilities provided through Java.
- The component must initially support formatting of the following object types and their associated primitives:
    - Integer
    - Short
    - Long
    - Byte
    - Float
    - Double
    - Date
- Default formats should be as follows:
    - Integer: ##,###
    - Short: ###
    - Long: ##,###
    - Byte: #
    - Float: ##,###.00
    - Double: ##,###.00
    - Date: mm/dd/yyyy

1.2.2 Miscellaneous

- Provide the interface to write additional object formatters for any object that does not have a formatter associated with it (i.e. a custom Address object).
- Objects may have a number of predefined formats or support a custom format string (such as “GyMwWdDFEaHkKhmsSzZ” as is provided through java.text.SimpleDateFormat). Either solution provides users with the ability to select the formatting of their choice. This may be straightforward for dates and numbers, as the format strings and corresponding parsers are already available through Java, but is more complex with custom objects (i.e. the Address object).
- Formats may apply to multiple object types. For example, a given format object may be valid for both an Integer and Short object.
- It is up to the designer to determine how to define and apply multiple formats for a given object.
- An object must be configurable with a default format.

1.3 Example of the Software Usage

The Object Formatter will be used to centralize and simplify the process of formatting data within an application. For example, an address may need to be displayed in different formats for a jsp

application, an applet and within the database.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

- Ability to pass in an object and have it formatted according to the default formatting for that object type.
- Ability to pass in an object and the required format.
- All methods should return a string representation of the formatted object.

2.1.3 Environment Requirements

- Development language: Java1.4
- Compile target: Java1.2, Java1.3, Java1.4

2.1.4 Package Structure

Com.topcoder.util.format

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 Third Party Component, Library, or Product Dependencies:

None.

3.2.3 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000

3.3 Design Constraints

3.3.1 Development Standards:

- It is required that all code be clearly commented.
- All code must adhere to javadoc standards, and, at minimum, include the @author, @param, @return, @throws and @version tags.
    - When populating the author tag, use your TopCoder member handle. In addition, please do not put in your email addresses.
    - Copyright tag: Copyright © 2002, TopCoder, Inc. All rights reserved
- For standardization purposes, code must use a 4-space (not tab) indentation.
- Do not use the ConfigurationManager inside of EJB's. The usage of the java.io.* package to read/write configuration files can potentially conflict with a restrictive security scheme inside the EJB container.
- All code must adhere to the Code Conventions outlined in the following:

http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html

3.3.2 Database Standards:

- Table and column names should be as follows: table_name, not tableName.
- Every table must have a Primary Key.
- Always use Long (or database equivalent) for the Primary Key.
  For maximum portability, database objects (table names, indexes, etc) should be kept to 18 characters or less. Informix and DB2 have an 18-character limit.

3.4 Required Documentation

3.4.1 Design Documentation

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

3.4.2 Help / User Documentation

- Javadocs must provide sufficient information regarding component design and usage.