Util Java

Java Code

```
/*
 * Copyright: (c) 2004-2009 Mayo Foundation for Medical Education and
 * Research (MFMER). All rights reserved. MAYO, MAYO CLINIC, and the
 * triple-shield Mayo logo are trademarks and service marks of MFMER.
 *
 * Except as contained in the copyright notice above, or as used to identify
 * MFMER as the author of this software, the trade names, trademarks, service
 * marks, or product names of the copyright holder shall not be used in
 * advertising, promotion or otherwise in connection with this software without
 * prior written authorization of the copyright holder.
 *
 * Licensed under the Eclipse Public License, Version 1.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *                 http://www.eclipse.org/legal/epl-v10.html
 *
 */
package org.LexGrid.LexBIG.example;

import java.io.PrintWriter;

import org.LexGrid.LexBIG.DataModel.Core.CodingSchemeSummary;
import org.LexGrid.LexBIG.Impl.dataAccess.ResourceManager;
import org.LexGrid.LexBIG.Impl.logging.Logger;
import org.apache.commons.cli.HelpFormatter;
import org.apache.commons.cli.Options;

/**
 * Utility functions to support the examples.
 *
 * @author <A HREF="mailto:johnson.thomas@mayo.edu">Thomas Johnson</A>
 */
class Util {
    static final private String _lineReturn = System.getProperty("line.separator");
    static final private Logger _logger = ResourceManager.instance().getLogger();
    static final private PrintWriter _printWriter = new PrintWriter(System.out);

    /**
     * Outputs messages to the error log and console, with additional tagging to
     * assist servicability.
     *
     * @param message
     *            The message to display.
     * @param cause
     *            Error associated with the message.
     */
    static void displayAndLogError(String message, Throwable cause) {
        displayTaggedMessage(message);
        _logger.error(message, cause);
    }

    /**
     * Outputs messages to the error log and console, with additional tagging to
     * assist servicability.
     *
     * @param cause
     *            Error associated with the message.
     */
    static void displayAndLogError(Throwable cause) {
        displayAndLogError(cause.getMessage(), cause);
    }

    /**
```

```
     * Outputs a standard message to console indicating supported command line
     * options.
     *
     * @param syntax
     *            Named syntax.
     * @param options
     *            Provided options.
     * @param example
     *            Example usage, if applicable.
     * @param parseErr
     *            Error that occurred parsing the command line, if applicable.
     */
    static void displayCommandOptions(String syntax, Options options, String example, Throwable parseErr) {
        displayMessage("");
        if (parseErr \!= null) {
            displayMessage("Unable to parse command options>> " + parseErr.getMessage());
            displayMessage("");
        }
        try {
            new HelpFormatter().printHelp(_printWriter, 80, syntax, "", options, 0, 0, "", true);
        } finally {
            _printWriter.flush();
        }
        if (example \!= null) {
            displayMessage("");
            displayMessage("Example: " + example);
        }
    }

    /**
     * Displays a message to the console.
     *
     * @param message
     *            The message to display.
     */
    static void displayMessage(String message) {
        try {
            _printWriter.println(message);
        } finally {
            _printWriter.flush();
        }
    }

    /**
     * Displays a message to the console, with additional tagging to assist
     * servicability.
     *
     * @param message
     *            The message to display.
     */
    static void displayTaggedMessage(String message) {
        displayTaggedMessage(message, null, null);
    }

    /**
     * Displays a message to the console, with additional tagging to assist
     * servicability.
     *
     * @param message
     *            The message to display.
     * @param cause
     *            Optional error associated with the message.
     * @param logID
     *            Optional identifier as registered in the LexBIG logs.
     */
    static void displayTaggedMessage(String message, Throwable cause, String logID) {
        StringBuffer sb = new StringBuffer("[VKC:LB] ").append(message);
        if (cause \!= null) {
            String causeMsg = cause.getMessage();
            if (causeMsg \!= null &amp;amp;&amp;amp; \!causeMsg.equals(message)) {
                sb.append(_lineReturn).append("\t*** Cause: ").append(causeMsg);
```

```
            }
        }
        if (logID \!= null) {
            sb.append(_lineReturn).append("\t*** Refer to message with ID = ").append(logID)
                    .append(" in the log file.");
        }
        displayMessage(sb.toString());
    }

    /**
     * Display a list of available code systems and
     *
     * @return The coding scheme summary for the selected code system; null if
     *         no valid selection was made.
     */
    static CodingSchemeSummary promptForCodeSystem() {
        return new CodingSchemeSelectionMenu().displayAndGetSelection();
    }

    /**
     * Returns common text to append to displayed help for commands that allow
     * the user to prompt for coding scheme information instead of providing urn
     * and version information as parameters.
     *
     * @return String
     */
    static String promptForSchemeHelp() {
        return "\n" + "\nNote: If the URN and version values are unspecified, a list of"
                + "\navailable coding schemes will be presented for user selection.";
    }

}
```