CodingSchemeSelectionMenu 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.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Formatter;
import java.util.List;

import org.LexGrid.LexBIG.DataModel.Collections.CodingSchemeRenderingList;
import org.LexGrid.LexBIG.DataModel.Core.CodingSchemeSummary;
import org.LexGrid.LexBIG.DataModel.InterfaceElements.CodingSchemeRendering;
import org.LexGrid.LexBIG.Exceptions.LBException;
import org.LexGrid.LexBIG.Impl.LexBIGServiceImpl;
import org.LexGrid.LexBIG.LexBIGService.LexBIGService;

/**
 * Helper class used to display a list of available code systems and return the
 * user selection.
 *
 */
public class CodingSchemeSelectionMenu {
    static final String Dash5 = "-----";
    static final String Dash15 = "---------------";
    static final String Dash25 = "-------------------------";
    static final String Dash30 = "------------------------------";

    public CodingSchemeSelectionMenu() {
        super();
    }

    /**
     * Displays a list of available coding schemes.
     *
     * @return The ordered list of coding scheme summaries as displayed.
     * @throws LBException
     *             If an error occurs displaying the list.
     */
    protected List<CodingSchemeSummary> displayCodingSchemes() throws LBException {
        List<CodingSchemeSummary> choices = new ArrayList<CodingSchemeSummary>();

        LexBIGService lbs = LexBIGServiceImpl.defaultInstance();
        CodingSchemeRenderingList schemes = lbs.getSupportedCodingSchemes();

        if (schemes.getCodingSchemeRenderingCount() == 0)
            Util.displayMessage("No coding schemes found.");
```

```
    else {

        Formatter f = new Formatter();

        String format = "%-5.5s|%-30.30s|%-25.25s|%-15.15s\n";
        Object[] hSep = new Object[] { Dash5, Dash30, Dash25, Dash15 };
        f.format(format, hSep);
        f.format(format, new Object[] { "#", "Local Name", "Version", "Tag" });
        f.format(format, hSep);
        CodingSchemeRendering[] csr = schemes.getCodingSchemeRendering();
        for (int i = 1; i <= csr.length; i++) {
            String nu = String.valueOf(i);

            CodingSchemeSummary css = csr[i - 1].getCodingSchemeSummary();
            choices.add(css);

            // Evaluate local name
            String localName = css.getLocalName();
            if (localName != null && localName.length() > 30)
                localName = localName.substring(0, 28) + ">>";

            // Evaluate version
            String version = css.getRepresentsVersion();
            if (version != null && version.length() > 25)
                version = version.substring(0, 23) + ">>";

            // Evaluate tag(s)
            String[] tags = csr[i - 1].getRenderingDetail().getVersionTags().getTag();
            String tag = tags.length > 0 ? tags[0] : "";
            if (tag != null && tag.length() > 15)
                tag = tag.substring(0, 13) + ">>";

            // Output the first line
            f.format(format, new Object[] { nu, localName, version, tag });

            // Output additional tags
            for (int j = 1; j < tags.length; j++) {
                tag = tags[j];
                if (tag != null && tag.length() > 10)
                    tag = tag.substring(0, 8) + ">>";
                f.format(format, "", "", "", tag);
            }

            // Output separator
            f.format(format, hSep);
        }
        Util.displayMessage(f.out().toString());
        Util.displayMessage("");
        Util.displayMessage("NOTE: >> indicates column value exceeds the available width.");
    }
    return choices;

}

/**
 * Display the list of available coding schemes and process the user
 * selection.
 *
 * @return A coding scheme summary correponding to the user selection; null
 *         if no selection is made or an error occurs.
 */
public CodingSchemeSummary displayAndGetSelection() {
    CodingSchemeSummary css = null;
    try {
        // Display choices ...
        List<CodingSchemeSummary> choices = displayCodingSchemes();

        if (choices.size() > 0) {
            // Process the user input ...
            try {
                Util.displayMessage("Enter the number of the Coding scheme to use, then <Enter> :");
```

```
                    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
                    int i = new Integer(br.readLine()).intValue();
                    if (i > 0 && i <= choices.size())
                        css = choices.get(i - 1);
                    else
                        Util.displayMessage("Entered value is out of range.");
                } catch (NumberFormatException nfe) {
                    Util.displayMessage("Entered value must be numeric.");
                } catch (IOException ioe) {
                    Util.displayAndLogError("An error occurred while processing the entered value", ioe);
                }
            } else {
                Util.displayMessage("No code systems are available.");
            }
        } catch (Exception e) {
            Util.displayAndLogError("An error occurred processing the available code systems", e);
        }
        return css;
    }
}
```