ExportingCSVFiles

NOTE: This is outdated information that applies only to Tapestry 4.

Exporting CSV Files

This component allows to export, i.e. download, tabular data in the comma separated values (CSV) format. The CSV format is described in RFC 4180 (http://www.rfc-editor.org/rfc/rfc4180.txt). It is useful e.g. to export data to MS Excel or OpenOffice Calc. The component is a special page with content type "text/csv". It should work with JDK 1.4/5.0 and Tapestry 4.0. The source code of this component is also available on Tassel in the "Other" category.

CSVExportPage.java

The page class of the component. This page has no template file. We overwrite finishLoad to hinder Tapestry from searching for a template file. Maybe there is a nicer way to do this.

```
package contrib.export;

import java.util.Iterator;
import java.util.List;

import org.apache.tapestry.AbstractPage;
import org.apache.tapestry.IMarkupWriter;
import org.apache.tapestry.IRequestCycle;
import org.apache.tapestry.engine.IPageLoader;
import org.apache.tapestry.spec.IComponentSpecification;
import org.apache.tapestry.util.ContentType;
import org.apache.tapestry.web.WebResponse;

public abstract class CSVExportPage extends AbstractPage {

        public static final String MIME_TYPE = "text/csv";

        public abstract char getSeparator();
        public abstract void setSeparator(char separator);

        public abstract String getFileName();
        public abstract void setFileName(String fileName);

        public abstract String getCharset();
        public abstract void setCharset(String charset);

        public abstract List getHeaderLine();
        public abstract void setHeaderLine(List headerLine);

        public abstract List getDataRows();
        public abstract void setDataRows(List dataRows);

        public abstract WebResponse getWebResponse();

        /**
         * Invoked from a page that wants to export the given data to a CSV file.
         *
         * @param fileName The file name that is given to the CSV file. It should end
         *   with the extension '.csv'.
         * @param charset The character set that is used for the data in the file.
         *   The charset is encoded in the MIME type of the document. Can be <code>null</code>
         *   if no charset should be defined.
         * @param headerLine The list of column names (list of String elements). Can be
         *   <code>null</code> if no header line should be contained in the CSV file.
         * @param dataRows A list of data rows. A data row is a List object containing
         *   String objects for the data elements.
         */
        public void doExport(String fileName, String charset, List headerLine, List dataRows, IRequestCycle
cycle)
        {
                setFileName(fileName);
```

```
                setCharset(charset);
                setHeaderLine(headerLine);
                setDataRows(dataRows);
                cycle.activate(this);
        }

        public ContentType getResponseContentType()
        {
                String contentType = MIME_TYPE;
                if (getCharset() != null) {
                        contentType += "; charset=" + getCharset();
                }
                if (getHeaderLine() != null) {
                        contentType += "; header=present";
                }
                else {
                        contentType += "; header=absent";
                }
                return new ContentType(contentType);
        }

        /** finishLoad is overridden here so that no template file is loaded. */
        public void finishLoad(IRequestCycle cycle, IPageLoader loader, IComponentSpecification specification)
        {
        }

        protected void renderComponent(IMarkupWriter writer, IRequestCycle cycle) {
                // Set the file name of the exported file.
                getWebResponse().setHeader("Content-Disposition",
                        "inline; attachment; filename=" + getFileName());

                // Set special headers to work around a bug in IE with https.
                getWebResponse().setHeader("Cache-Control", "max-age=0");
                getWebResponse().setHeader("Pragma", "public");

                if (getHeaderLine() != null) {
                        renderRow(getHeaderLine(), writer);
                }

                Iterator rowIter = getDataRows().iterator();
                while (rowIter.hasNext()) {
                        List row = (List) rowIter.next();
                        renderRow(row, writer);
                }
        }

        private void renderRow(List row, IMarkupWriter writer) {
                Iterator colIter = row.iterator();
                while (colIter.hasNext()) {
                        String rowElement = (String) colIter.next();
                        writer.printRaw(rowElement);
                        if (colIter.hasNext()) {
                                writer.print(getSeparator());
                        }
                }
                writer.print('\n');
        }
}
```

CSVExportPage.page

The page specification for the export component. We don't use the standard field separator COMMA (',') as the initial value of the "separator" property because we had problems with MS Excel recognizing it as a separator.

```
<?xml version="1.0"?>
<!DOCTYPE page-specification PUBLIC
  "-//Apache Software Foundation//Tapestry Specification 4.0//EN"
  "http://jakarta.apache.org/tapestry/dtd/Tapestry_4_0.dtd">

<page-specification class="contrib.export.CSVExportPage">

        <property name="separator" initial-value="';'"/>
        <property name="fileName" initial-value="'Export.csv'"/>
        <property name="charset"/>
        <property name="headerLine"/>
        <property name="dataRows"/>

        <inject property="webResponse" object="infrastructure:response"/>

</page-specification>
```

How to use it

To export data with the component you have to prepare the data as Lists of Strings. The example listener method below shows how the data has to be prepared and how the CSVExportPage is used. All the properties of the page, except the "separator" property, can be set with the method doExport. See the JavaDoc comment of the method doExport in the CSVExportPage class above for a description of its parameters.

```
        public void exportPersons(IRequestCycle cycle) {
                List headerData = Arrays.asList("Name", "Age", "City");

                List exportData = new ArrayList();
                List row = Arrays.asList("J. Young", "39", "New York");
                exportData.add(row);
                row = Arrays.asList("P. Meier", "64", "Berlin");
                exportData.add(row);
                row = Arrays.asList("H. Smith", "56", "London");
                exportData.add(row);

                CSVExportPage page = (CSVExportPage) cycle.getPage("CSVExportPage");
                page.doExport("Export.csv", null, headerData, exportData, cycle);
        }
```