BrowserInputModule

sample browser input module to get a client browser name. sample created by RomanHrivik

cocoon.xconf

```
        <component-instance class="com.yourpackage.BrowserModule" logger="core.modules.input" name="browser">
                                <default-browser name="unknown"/>
                                <browser name="explorer" useragent="MSIE"/>
                                <browser name="netscape" useragent="Mozilla"/>
        </component-instance>
```

sitemap.xmap

```
<map:match pattern="header">
        <map:generate src="header{browser:clientbrowser}.xml"/>
        <map:transform src="header.xslt"/>
        <map:serialize type="xml"/>
</map:match>
```

Source code

```
package com.yourpackage;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;

import org.apache.avalon.framework.configuration.Configuration;
import org.apache.avalon.framework.configuration.ConfigurationException;
import org.apache.avalon.framework.thread.ThreadSafe;
import org.apache.cocoon.components.modules.input.AbstractInputModule;
import org.apache.cocoon.environment.ObjectModelHelper;

/**
 *
 * Browser input module<br>
 * in cocoon.xconf set how you want identify browsers<br>
 * <pre>
 *    &lt;component-instance class="com.yourpackage.BrowserModule" logger="core.modules.input" name="browser"
/&gt;<br>
 *               &lt;default-browser name="explorer"/&gt;<br>
 *               &lt;browser name="explorer" useragent="MSIE"/&gt;<br>
 *               &lt;browser name="pocketexplorer" useragent="MSPIE"/&gt;<br>
 *               &lt;browser name="handweb" useragent="HandHTTP"/&gt;<br>
 *               &lt;browser name="avantgo" useragent="AvantGo"/&gt;<br>
 *               &lt;browser name="imode" useragent="DoCoMo"/&gt;<br>
 *               &lt;browser name="opera" useragent="Opera"/&gt;<br>
 *               &lt;browser name="lynx" useragent="Lynx"/&gt;<br>
 *               &lt;browser name="java" useragent="Java"/&gt;<br>
 *               &lt;browser name="wap" useragent="Nokia"/&gt;<br>
 *               &lt;browser name="wap" useragent="UP"/&gt;<br>
 *               &lt;browser name="wap" useragent="Wapalizer"/&gt;<br>
 *               &lt;browser name="mozilla5" useragent="Mozilla/5"/&gt;<br>
 *               &lt;browser name="mozilla5" useragent="Netscape6/"/&gt;<br>
 *               &lt;browser name="netscape" useragent="Mozilla"/&gt;<br>
 *   &lt;/component-instance&gt;<br>
 *</pre>
 *<br>
 *in sitemap use {browser:clientbrowser}<br>
 *<br>
 *
 * @author Roman Hrivik
```

```
 *
 */

public class BrowserModule extends AbstractInputModule implements ThreadSafe  {

    /**
     * element name for browser
     */
    private static final String ELEMENT_BROWSER = "browser";
    /**
     * attribute name for browserName for browser element
     */
    private static final String ATTR_BROWSERNAME_ELEMENT_BROWSER = "name";
    /**
     * attribute name for useragent for browser element
     */
    private static final String ATTR_USERAGENT_ELEMENT_BROWSER = "useragent";
    /**
     * element name for default-browser
     */
    private static final String ELEMENT_DEFAULT_BROWSER = "default-browser";
    /**
     * attribute name for browserName for defaultBrowser
     */
    private static final String ATTR_BROWSERNAME_ELEMENT_DEFAULT_BROWSER = "name";

    /**
     * input parameter name to get type of client browser in sitemap use {browser:clientbrowser}
     */
    private static final String INPUT_PARAMETER_CLIENT_BROWSER = "clientbrowser";

    /**
     * keeps browserConfiguration (browser,agent)
     */
    private List browserList;
    /**
     * default browser to be returned when none from configuration will be founded
     */
    private String defaultBrowser = "";

        /* (non-Javadoc)
         * @see org.apache.avalon.framework.configuration.Configurable#configure(org.apache.avalon.framework.
configuration.Configuration)
         */
        public void configure(Configuration conf) throws ConfigurationException {
        super.configure(conf);

        browserList = new ArrayList();

        Configuration[] parameters = conf.getChildren();

        for (int i = 0; i < parameters.length; i++) {
            String elementName = parameters[i].getName();
            String browserName = parameters[i].getAttribute(ATTR_BROWSERNAME_ELEMENT_BROWSER, "");
            String userAgent = parameters[i].getAttribute(ATTR_USERAGENT_ELEMENT_BROWSER, "");

            if(ELEMENT_BROWSER.equals(elementName)) {

                browserList.add(new BrowserDataConfig(browserName, userAgent));

                if(getLogger().isDebugEnabled()) {
                    getLogger().debug("BrowserModule: conf added: browserName=" + browserName + " userAgent=" +
userAgent);
                }

            }
            else if(ELEMENT_DEFAULT_BROWSER.equals(elementName)) {
                defaultBrowser = parameters[i].getAttribute(ATTR_BROWSERNAME_ELEMENT_DEFAULT_BROWSER, "");
            }

                }
```

```
        }

        /* (non-Javadoc)
         * @see org.apache.cocoon.components.modules.input.InputModule#getAttribute(java.lang.String, org.
apache.avalon.framework.configuration.Configuration, java.util.Map)
         */
        public Object getAttribute(
                String name,
                Configuration modeConf,
                Map objectModel)
                throws ConfigurationException {

        if (INPUT_PARAMETER_CLIENT_BROWSER.equals(name)) {
            /*
             * get the current browser
             */
            if(browserList != null) {
                String userAgent = ObjectModelHelper.getRequest(objectModel).getHeader("User-Agent");
                if(userAgent != null) {
                    // loop over config
                    for (Iterator iter = browserList.iterator();
                                                iter.hasNext();
                                                ) {
                                                BrowserDataConfig config = (BrowserDataConfig) iter.next();

                        if(getLogger().isDebugEnabled()) {
                            getLogger().debug("BrowserModule: checking: userAgent=" + userAgent + " that
contains confAgent=" + config.getUserAgent());
                        }

                        if(userAgent.indexOf(config.getUserAgent()) != -1) {
                            return config.getBrowserName();
                        }

                                        }
                }
            }

            /*
             * otherwise return defaultBrowser
             */
            return defaultBrowser;
                } else {
            // unknown attribute
            return null;
                }
        }




        /* (non-Javadoc)
         * @see org.apache.cocoon.components.modules.input.InputModule#getAttributeNames(org.apache.avalon.
framework.configuration.Configuration, java.util.Map)
         */
        public Iterator getAttributeNames(Configuration modeConf, Map objectModel)
                throws ConfigurationException {

                    return browserList.iterator();
        }


        /* (non-Javadoc)
         * @see org.apache.cocoon.components.modules.input.InputModule#getAttributeValues(java.lang.String, org.
apache.avalon.framework.configuration.Configuration, java.util.Map)
         */
        public Object[] getAttributeValues(
                String name,
                Configuration modeConf,
                Map objectModel)
```

```
                throws ConfigurationException {

                    return browserList.toArray();
        }


    /**
     * BrowserDataConfig helper class
     *
     * @author hrivik
     *
     */
    private class BrowserDataConfig {
        private String browserName;
        private String userAgent;

        BrowserDataConfig() {}

        BrowserDataConfig(String browserName, String userAgent) {
            setBrowserName(browserName);
            setUserAgent(userAgent);
        }

                /**
                 * @return browserName
                 */
                public String getBrowserName() {
                        return browserName;
                }

                /**
                 * @param browserName
                 */
                public void setBrowserName(String browserName) {
                        this.browserName = browserName;
                }

                /**
                 * @return userAgent
                 */
                public String getUserAgent() {
                        return userAgent;
                }

                /**
                 * @param userAgent
                 */
                public void setUserAgent(String userAgent) {
                        this.userAgent = userAgent;
                }

    }

}
```