Theme Templates

There are three ways (that I know of) to create a theme for JBoss Nukes. 1) Create a module using a Java class that must have ThemeSupport.java in its hierarchy, or 2) Create an HTML file with relatively few key taglines to be recognized by ThemeTemplate.java. 3) Create a hybrid of Java and HTML using the Nukes templating system. This is rare and no examples exist as of yet. Of these three, the HTML file is the easiest for non-Java programmers, but both can be done.

Setting up the Structure

In any case, a common structure must be followed for the templates to work with Nukes, as shown in the following figure. The first step is creating a new folder named after the theme to be created. Inside the folder, another folder must be created called "src". There will also be some build files in this directory, but that will be covered later.

Create four more folders in the src directory: bin, etc, main, and resources. The simplest folder to fill is the etc directory, which contains the manifest files. The main folder will be the key difference for the Java version, such that it will actually contain the source file, where the HTML file will be kept in the bin folder. The bin folder is where the images will be kept and, in the case of the HTML version only, the cascading style sheet and HTML file template. The resources folder will be the hardest to fill as it contains the deployment descriptors and properties files.

Inside etc, there will be two files named nukes-<name of theme>-ear.mf and nukes-<name of theme>-sar.mf, where <name of theme> should be replaced by the actual name of the theme to be created (i.e. nukes-unisysTheme-ear.mf) (<name of theme> will appear several times in this document and will have the same meaning). The content of the files are the same as all of the manifest files in the entirety of Nukes. There is a printed version provided in the examples, or the copy and paste option works just as well. To find another manifest, enter any module (note: build, thirdparty, and tools are not modules) and inside the source folder there will be an etc folder.

The main folder will have the directory tree of "org/jboss/nukes/core/themes/<name of theme>/", which will furthermore be known as the **Tree**. For the HTML version, this is all the main directory will contain, as there is no Java source code to maintain.

The bin folder will have a directory named nukes-<name of theme>-lib-jar. This directory will contain the Tree, as defined above. At the bottom of the Tree, there will be at least one more folder called images, where the images to be used in the theme will be stored. If using the HTML version, a style folder containing a style.css file should be created as well as the theme.html file, which is the template itself. More will be explained on those two later in this document.

The resources folder will have three directories inside it: nukes-<name of theme>-ear, nukes-<name of theme>-lib-jar, and nukes-<name of theme>-sar. The "ear" and "sar" folders will each have a folder called META-INF where their respective xml descriptors will be held. The "jar" folder will contain the Tree along with two new files and the same style folder with style.css file as in the bin folder (a simple copy and paste will take care of this).

For the "ear" folder's META-INF directory, there must be two xml files: application.xml and jboss-app.xml. Examples have been provided along with this document. The application.xml file will define the theme library file for the Enterprise Java Bean and the jboss-app.xml file defines the source archive for the JBoss service. These terms may be confusing, but the example does work. If the example is copied, replacing the "unisysTheme" with the name of the theme being created, then the new files should work as well.

The “sar” folder’s META-INF directory will have the jboss-service.xml file. There are two examples provided, one for each version. There are five basic elements to consider when constructing the new service file for the theme to be created (refer to the numbered labels in the examples).

1. Where to find the code.
    a. If using the HTML version, this line will always be **code=“org.jboss.nukes.theme.ThemeTemplate”**.
    b. For the Java version, it will represent the path to reach the Java source code or the “package”.
2. What to name the theme.
    a. This is for the JBoss Jmx-console page. It needs to know what the service (or theme) is to be called.
    b. The line should always be **name=“nukes.themes:name=<name of theme>”** or errors may occur.
3. What the service depends on.
    a. This list of depends tags define which services should be already started before this service can start.
    b. By default, the **<depends>nukes.modules:name=core</depends>** should always be there.
    c. If using the HTML version, then it is a good idea to depend on the html module: **<depends>nukes.modules:name=html</depends>**
4. How to create the service.
    a. Every service (including this theme) will be or rely on a Java class. Java classes have constructors, which creates an instance of itself when called.
    b. The arg tag defines which constructor to call.
        i. By default, there will always be the “empty” constructor with no arguments.
        ii. Nukes uses EJB’s to persist data in the database. To create an instance that will manage this, use the **<arg type=“boolean” value=“true”/>** constructor. The HTML version should always use this.
5. Attributes the service has.
    a. Attributes define what the service is.
    b. For both versions, the security attribute should be defined.
        i. Security is defined by permissions. The permission will define the group name, permission pattern, and the level of permission granted.
        ii. The pattern “**.*:.*:.***” removes all restrictions for the service to every part of the portal for the level specified.
            1. I’m not too familiar with how to change these pattern to be useful. Every change I’ve made caused an error so far.
            2. The Nukes on JBoss forum is an excellent source of information.
    c. For the HTML version, two extra attributes must be defined.
        i. “Id”
            1. This tells the class loader where to find the HTML template file.
            2. It should always be **<attribute name=“Id”>jar:/theme.html</attribute>**
        ii. “Root”
            1. This tells the resource loader where to find the resource files, such as the properties and style sheets.
            2. It should be defined as the Tree, as described above.

Two lines were skipped in the description of the example because they do not change and I do not know exactly what they mean.

The "jar" folder will contain the Tree along with the style folder, a languages folder, and a nukes-web_1_0.dtd file. The Resource.properties file will be kept inside the languages directory at the bottom of the tree with the style folder. The example provided shows that it contains global definitions for background colors and text colors. To use this handy feature, when defined a background color for an HTML element, such as a table column, use the property name qualified by the service name surrounded by ${} tags (i.e. `<td bgcolor="${testTheme.BGCOLOR1}">` Now to change certain colors without having to sort through HTML code, just change the color value in the properties file and redeploy the theme.

The nukes-web_1_0.dtd file is only four lines each defining a web element, of which I am unfamiliar with. This file should be copied **exactly** into the core directory within the Tree.

Build files

Build files in Nukes have a common structure. The first line is the xml document version and encoding definition. Following that is the Doctype definition with project entity paths to other xml documents describing common features. These two statements should appear in every build file:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE project [
   <!ENTITY buildmagic SYSTEM "../tools/etc/buildfragments/buildmagic.ent">
   <!ENTITY tools      SYSTEM "../tools/etc/buildfragments/tools.ent">
   <!ENTITY libraries  SYSTEM "../tools/etc/buildfragments/libraries.ent">
   <!ENTITY modules    SYSTEM "../tools/etc/buildfragments/modules.ent">
   <!ENTITY defaults   SYSTEM "../tools/etc/buildfragments/defaults.ent">
   <!ENTITY targets    SYSTEM "../tools/etc/buildfragments/targets.ent">
]>
```

The rest of the build file will define the project to be built, or rather this small piece of the project. It will start with the <project> tag and a couple of attributes to set. The first is the default target to call when ant is invoked with this build file (type "build" in the command prompt in the theme or modules root directory). Typically, this is set to "main", which should perform the necessary class compilation and packaging. The second attribute is the name of the project, but this can be optional.

```
<project default="main" name="testTheme Nukes on JBoss Theme">
```

Next, the entities declared in the Doctype line must be included using the '&' symbol.

```
&buildmagic;
&libraries;
&modules;
&defaults;
&tools;
&targets;
```

Now build targets must be defined. The targets are defined using the <target> tag. Attributes that must be defined in the target tag are the name and a "depends" clause. The "depends" clause is set to the name of another target or targets that must be executed prior to this target. The root target is defined in the buildmagic enitity; therefore, every target defined in this project will have a "depends" clause.

There is another optional attribute called "unless", which disables the target from executing. This is useful for building a small part of the project while skipping a time consuming process. Typically, the attribute is set to disable qualified with the current target name. The "description" attribute is also optional and is set with a short string description of what the task does.

Example:

```
<target name="init" unless="init.disable" description="Initializes project."
depends="_buildmagic:init">
</target>
```

Targets are useless without properties defined to be executed, unless the target set in the "depends" clause executes what is needed. There are several special tags that can be used to aid in building the project.

`<property>` `- defines an ant property or variable value.`

```
      name - name of attribute
      value - value of attribute
```

Properties are typically used for configurations and will most likely be found in a target named "configure".

Example: `<property name="jndi-root" value="nukes/testTheme"/>`

`<call>` `- invokes a specified build target.`

```
      target - target to invoke
```

Calls are normally found in configuration targets to invoke targets from the included entities.

Example:
```
<!-- Configure thirdparty libraries -->
 <call target="configure-libraries"/>
```

`<path>` `- defines a classpath to be included in the project build.`

```
      id - reference name of classpath
      refid - reference name of already defined path
```

Paths are usually defined during configuration as well. The id attribute is the name of the path to include, while the actual path is defined by a group of <fileset> tags (see below). A refid is a means of localizing an already defined path. The referenced path is defined in another file, such as the libraries entity, and redefined to be included locally.

`<fileset>` `- defines a set of classes or jars to include in the path.`

```
            dir - directory that contains the classes or jars
            includes - filter mapping of the file types to include (e.g. "*.jar" will
            include all jar type files)
```

`<include>` `- used to specify an individual file to include in a fileset.`

```
            When the include tag is used, the fileset attribute "includes" is not
            used.
            name - name of file to include
```

`<filterset>` `- defines a set of tokens to be replaced by the specified values.`

```
            id - name of the filter set
```

<filter> - defines an individual token and value to replace it with.

token - symbol to replace

value - value to replace token with

Filter sets are not normally used. The only build file that makes use of filter sets is the nukes core project build. The filter set is defined within the configure target.

Example:

```
<filterset id="module.filterset">
   <filter token="nukes.datasource"
           value="${nukes.datasource}"/>
   <filter token="nukes.database.user"
           value="${nukes.database.user}"/>
   <filter token="nukes.datbase.pass"
           value="${nukes.database.pass}"/>
</filterset>
```

<mkdir> - creates a directory within the specified path.

dir - path and directory name to create

<ejbdoclet> - defines how Enterprise Java Beans should be generated.

destdir - directory where the generated classes will be copied

excludedtags - javadoc tags to exclude from class generation, typical values are: "@version,@author"

ejbspec - Enterprise Java Bean specification version, should be set to "2.0"

After the ejbdoclet tag attributes have been set, a file set (see above and provided example) must be defined providing the source code for the Enterprise Java Beans.

<localinterface> - by adding this tag, the Enterprise Java Bean local interface will be generated

havingClassTag - will only create the interface with the given class tag

<localhomeinterface> - by adding this tag, the Enterprise Java Bean local home interface will be generated

havingClassTag - will only create the interface with the given class tag

`<deploymentdescriptor>` - `sets attributes for the generated Java Bean deployment descriptor defining the Java Bean methods.`

```
                          xmlencoding - encoding type for the descriptor document,
                          should be "UTF-8"
                          destdir - directory to copy the generated deployment
                          descriptor, should be a META-INF directory
                          description - short string describing the Beans being
                          deployed
                          displayname - display name for the Bean classes
```

`<jboss>` - `sets attributes for the generated Java Bean deployment descriptor defining the server interactions with the Java Beans.`

```
          xmlencoding - encoding type for the descriptor document, should be "UTF-8"
          version - version of the JBoss server the Beans will be deployed to,
          currently should be set to "3.2"
          destdir - directory to copy the generated deployment descriptor, should be a
          META-INF directory
          mergedir="${source.resources}/${nukes.mergedir}"
          datasource - datasource used to create a connection to the database used by
          the Java Beans
          typemapping - SQL to Java type mapping descriptor
```

`<jar>` - `creates an archive with the specified classes.  There must be a manifest defined for ejb, sar, and ear archives.`

```
        jarfile - directory path and name of the archive file
        manifest - for ejb, sar, and ear archives only, specifies the path and name of
        the manifest file
```

`<require>` - `checks for the file specified in the path, creates one if not there.`

```
            file - path and name of the necessary file
```

The require tag is used primarily within the “deploy” target to be sure the archives will have a directory to be deployed to.

`<copy>` - `copies the specified file to the specified directory.`

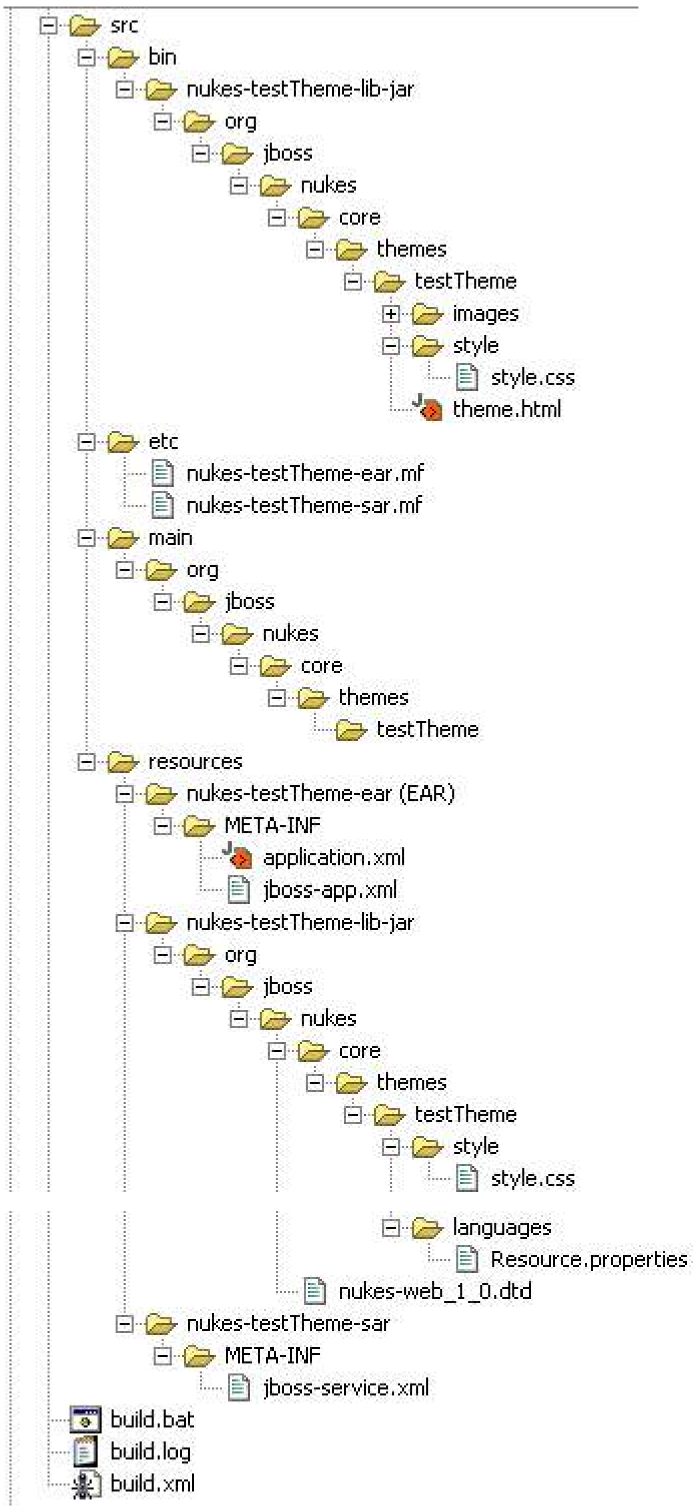
```
         file - path and name of file to copy
         todir - directory to copy file to
```

The copy tag is used primarily within the “deploy” target to deploy the archives.

```
<delete> - deletes the specfied file.
           file - path and name of file to delete
```

The delete tag is used primarily within the “undeploy” target to be sure the archives to be undeployed will be removed from the server.

The best policy for constructing a build file for a new theme or module is to copy an existing one from another similar project. If the project does not use Enterprise Java Beans (such as most themes), the ejbdoclet tag may be omitted. See the example provided with this document for a fully constructed build file.



```
src
    bin
        nukes-testTheme-lib-jar
            org
                jboss
                    nukes
                        core
                            themes
                                testTheme
                                    images
                                    style
                                        style.css
                                    theme.html
    etc
        nukes-testTheme-ear.mf
        nukes-testTheme-sar.mf
    main
        org
            jboss
                nukes
                    core
                        themes
                            testTheme
    resources
        nukes-testTheme-ear (EAR)
            META-INF
                application.xml
                jboss-app.xml
        nukes-testTheme-lib-jar
            org
                jboss
                    nukes
                        core
                            themes
                                testTheme
                                    style
                                        style.css

                                    languages
                                        Resource.properties
                            nukes-web_1_0.dtd
        nukes-testTheme-sar
            META-INF
                jboss-service.xml
build.bat
build.log
build.xml
```

Figure 1: example structure NOTE: testTheme is the name of the theme created for this example.

HTML Version

HTML templating requires the template file to be named theme.html. The file is like any other HTML source file except that comments play a bigger role in defining the formatting of certain sections of a page. There are six sections to be defined: header, footer, left, middle, right, and message. Two other sections allow for customization of table formats: openTable and closeTable. Within the <head> tags of the file, there should be two lines. The first line is to define the base for all path reference tags and is customarily <base href="../../">. This makes all path references (href, src, etc…) relative to the core directory within the Tree. The second line imports the style sheet to be used for this template: <style type="text/css">@import url ("themes/testTheme/style/style.css");</style>. These two lines may not be required and not the only formatting tools allowed. Care must be taken as these two lines are the only two that have been tested.

Inside the <body> tags, the sections must be defined. There are some restrictions to the order of the left, right, and middle section definitions, but the others may be defined in any order. The header section definition starts with the <!-- BEGIN header --> comment. This tells the ThemeTemplate class to begin parsing the header source. Within the header section, the left and middle block formats must be defined, each starting with a respective <!-- BEGIN <block section name> --> tag and ending with a <!-- END <block section name> --> tag. The value for <block section name> can be “left”, “middle”, or “right” (See Figure 2). The left block format must be defined before the middle block format and the header section must end (use <!-- END header -->) after the middle block format definition. The footer can now be defined using the same tag syntax except replace “header” with “footer”. Within the footer section, the right block format must be defined first, and then any footer formatting may be defined before ending the footer section. Usually, the footer formatting is left out.

Typically, the formatting is structured using the <table> tags and lays out the page in blocks.
The example below shows how the sections are related:

```
<!-- BEGIN header -- >
<table>
<row>
<table>
Header bar
            </table>
</row>
<row>
            <table>
<row>
<column>
<!-- BEGIN left -- >
<table>
Left formatting
                              </table>
<!-- END left -- >
                        </column>
                        <column>
<!-- BEGIN middle -- >
                              <table>
                                          Middle formatting
                              </table>
<!-- END middle -- >
<!-- END header -- >
```

Header		
Left	Middle	Right
Footer		

```
<!-- BEGIN footer -- >
                        </column>
                  <column>
<!-- BEGIN right -- >
                              <table>
                                    Right formatting
<!-- END right -- >
                              </table>
</column>
</row>
</table>
</row>
<row>
            <table>
                  Footer content           Not shown in example
            </table>
</row>
</table>
<!-- END footer -- >
```

Figure 2: Abstract Html template structure.

In the example, the entire template is set within a table from header to footer, which will be referred to as the Template Table. The header bar content is defined as the first row and within the row is a set of table tags to format the header bar. The next row of the Template Table will define the Left, Middle, and Right block formats. Within this row is a set of table tags defining one row with three columns, each column for a block format and each format defined within their own table tags. The last row of the Template Table is for the footer content, but this is usually left out.

There are three more sections to define, openTable, closeTable, and message. The open and clost table sections are simply the series of HTML tags with options that should be used to define how to open a table within a block for content formatting, and how to close that same table. The message section is only used if there are special messages that should be displayed to the user (or if the adminmessages module has been deployed). Typically, it is a combination of the openTable definition and the closeTable definition with the line <div style="text-align: center" class="pn_title">{MESSAGE}</div> in the middle. Note that the style and class options may change depending on the style definition created for the template (See the Style section).

The format definitions only describe what the layout around the content. To place the title and content of a block in a specific place within the layout, special template tags have been defined. The general syntax is {<block section name>.(TITLE|CONTENT)}. The {MESSAGE} tag (above) is another special template tag defined to insert the message content in that specific place in the layout.

An example of a right block definition using the special tag syntax:

```
<table width="100%" border="0" cellspacing="1" cellpadding="3">
     <tr>
          <td bgcolor="#ffffff" class="pn-title">{right.TITLE}</td>
     </tr>
     <tr>
          <td bgcolor="#ffffff" class="pn-normal">{right.CONTENT}</td>
     </tr>
</table>
```

For a more detailed example, see the source provided with this document.

Java Version

The Java templating version uses a simple Java class to format the layout of the page. The class must have `ThemeSupport.java` in its hierarchy. It must have a no-argument constructor that invokes super(). It should also have a boolean persistent flag constructor that invokes super(persistent), but this may not be necessary. Seven methods should be defined: block, blocks, header, footer, openTable, closeTable, and process.

The methods that deal with the html formatting of the sections are header, footer, block, openTable, and closeTable. The blocks method iterates through the list of blocks to render them on the page and process defines the main structure of the Html document, including the DOCTYPE, style import, html, head, and body tags. The process method is the main driver for the theme.

Method signatures:

```
public void block(Writer writer, int side, int index,
                  PageFragment fragment)
                              throws IOException {}

public void footer(PageResult result, Writer writer)
            throws IOException {}

public void header(PageResult result, Writer writer)
            throws IOException {}

public void process(PageResult result, Writer writer)
            throws IOException {}

protected void closeTable(Writer writer)
            throws IOException {}

protected void openTable(Writer writer)
            throws IOException {}

private void blocks(Writer writer, int side, List blocks)
            throws IOException {}
```

The Writer class is an abstract class for writing to character streams. The writer parameter has been defined to write to the page document that will be rendered on the screen. The only method that should be used is `writer.write(String)`, where String is an Html source string that defines a formatting feature.

Example:

```
writer.write("<table width=\"150\" border=\"0\" cellspacing=\"0\"
cellpadding=\"5\">\n" + "<tr>\n" + "<td class=\"pn-title-lblock\">");
```

The side parameter defines which side the block is to be rendered.
The integer codes: 0 = left, 1 = middle, and 2 = right.

The index parameter represents the index number (starting at zero) of the block currently being rendered. This is typically not used except for debug purposes.

The PageFragment class holds fragments of the page to be rendered, such as the body and title of a single block or section. The fragment parameter is the handle to use to reference the page fragments. These are used in conjunction with the writer.write and process methods to render the title and content, respectively, on the page.

Example:
```
writer.write(fragment.getTitle());
writer.write("</td>\n" + "</tr>\n" + "<tr>\n <td valign=\"top\">");
process(fragment.getBody(), writer);
```

The List class is an implementation of a generic ordered collection. The blocks parameter is a list of the blocks to be rendered to the page. This method is a simple for iteration loop over the blocks list.

Example:
```
int index = 0;
for (Iterator i = blocks.iterator(); i.hasNext();) {
PageFragment block = (PageFragment) i.next();
      block(writer, side, index++, block);
}
```

The PageResult class holds the list of fragments that will be rendered on the screen. The getFragments method retrieves the list of blocks to be rendered based on an integer value passed. The values are in the Constants class and are `Constants.SIDE_LEFT`, `Constants.SIDE_CENTRE`, `Constants.SIDE_RIGHT`. The two other available sections are the head (Html source between the <head> tags), using getHead() method, and the main body, using getMain(). getBody() method. The head of the document is written to the page using the `writer.write(result.getHead().toString());` command, while the rest of the body is sent to the ThemeSupport class process method using the `process(result.getMain().getBody(), writer);` command.

Example process method:

```
public void process(PageResult result, Writer writer)
            throws IOException {
      /* set up the style sheet import statement and write it to the
       * result print stream
       */
result.getHead().getPrintWriter().print("<style
      type=\"text/css\">@import url(\"themes/" + getName() +
      "/style/style.css\");</style>");

      // write the DOCTYPE statement to the page document
// include the html and head tags
writer.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01" +
      "Transitional//EN\">\n<html>\n<head>\n");

      // write the import statement defined above to the page document
writer.write(result.getHead().toString());

      // close the head tag and open the body tag
writer.write("</head><body>\n");

      // render the header, left, and middle blocks on the page
header(result, writer);

      // process and render the main content on the page
process(result.getMain().getBody(), writer);

      // render the right block and footer on the page
footer(result, writer);

      // close the body and html tags
writer.write("</body>\n</html>\n");
}
```

The open and close table methods usually have only one `writer.write` command with an Html source string defining how a table should be opened and closed, respectively.

Example openTable method:

```
protected void openTable(Writer writer)
            throws IOException {
      writer.write("<br><table width=\"100%\" border=\"1\" " +
            "cellspacing=\"1\" cellpadding=\"0\"><tr><td>\n <table" +
            "width=\"100%\"" + " border=\"0\" cellspacing=\"1\"" +
            " cellpadding=\"5\"><tr><td>\n");
}
```

Example closeTable method:

```
protected void closeTable(Writer writer)
            throws IOException {
      writer.write("</td></tr></table></td></tr></table>\n");
}
```

The header method defines the layout and content of the header bar for the page, as well as the layout of the section to contain the left and middle blocks. The footer method defines the layout of the section to contain the right blocks as well as the layout and content of the footer bar.

Example header method:

```
writer.write(…layout and content html source strings for header bar…);
writer.write(…layout html source strings for left section…);
blocks(writer, Constants.SIDE_LEFT,
result.getFragments(Constants.SIDE_LEFT));
writer.write(…layout html source strings for middle section…);
blocks(writer, Constants.SIDE_CENTRE,
result.getFragments(Constants.SIDE_CENTRE));
```

Example footer method:

```
writer.write(…layout html source strings for right section…);
blocks(writer, Constants.SIDE_RIGHT,
result.getFragments(Constants.SIDE_RIGHT));
writer.write(…layout and content html source strings for footer bar…);
```

The block method defines the layout for each block according to the side the block will be rendered. This is typically done with a **`switch`** `(side)` statement with the **`case`** statements checking for the `Constants.SIDE_`<block side> attributes. The title of the block is rendered using a `writer.write` method, while the block content is rendered by calling the ThemeSupport process method with the PageFragment fragment body.

Example block method:

```
switch (side) {
case Constants.SIDE_LEFT :
            writer.write(…layout html source strings…);
            writer.write(fragment.getTitle());
            writer.write(…layout html source strings…);
            process(fragment.getBody(), writer);
            writer.write(…layout html source strings…);
            break;

      case Constants.SIDE_CENTRE :
            writer.write(…layout html source strings…);
            writer.write(fragment.getTitle());
            writer.write(…layout html source strings…);
            process(fragment.getBody(), writer);
            writer.write(…layout html source strings…);
            break;

      case Constants.SIDE_RIGHT :
            writer.write(…layout html source strings…);
            writer.write(fragment.getTitle());
            writer.write(…layout html source strings…);
            process(fragment.getBody(), writer);
            writer.write(…layout html source strings…);
            break;
}
```

For a more detailed example, see the source provided with this document.

Hybrid

This section describes a way to use both Java and Html templates. This is based on the Templating concept used throughout Nukes. For the hybrid version, the structure will be the same as the Html version, except there will be Java source code to maintain in the main folder (see the Structure section). The Java source will still have to define the same methods as described above; however, the content and signature of the methods may change. The Html source will still have to define the sections as described in the Html section, but with Java support, the source may add context elements to enhance the theme. Context elements are defined in the Context section, App B.

The Java side must define two more methods: start and stop. The start method must invoke super.start() and create a template repository to load the Html version templates to be rendered. The stop method must invoke super.stop(). It must also declare five Template type variables, typically with the same names as the templates to render:

Examples:

```
TemplateLoader repository;
Template header, footer, openTable, closeTable, message;

public void start() throws Exception
{
 super.start();
 // create the repository of Document type - load the html source
 Document doc = repository.loadTemplate("/theme.html");

 // add the template sections
 repository.addTemplate("header",
                                      /*
                                       * the /node/loop… string is an
                                       * XPath for the dom4j Node element
                                       * more info here:
                                       * http://www.w3.org/TR/xpath
                                       */
  (Element)doc.selectSingleNode("/node/loop[@name='header']/node"));

 repository.addTemplate("footer",
  (Element)doc.selectSingleNode("/node/loop[@name='footer']/node"));

 repository.addTemplate("openTable",
  (Element)doc.selectSingleNode("/node/loop[@name='openTable']/node"));

 repository.addTemplate("closeTable",
  (Element)doc.selectSingleNode("/node/loop[@name=' " +
   "closeTable ']/node"));

 repository.addTemplate("message",
  (Element)doc.selectSingleNode("/node/loop[@name='message']/node"));

 // create the template and assign them to the Template handles
 header = repository.createTemplate("header");
 footer = repository.createTemplate("footer");
 openTable = repository.createTemplate("openTable");
 closeTable = repository.createTemplate("closeTable");
 message = repository.createTemplate("message");
}

public void stop()
{
 super.stop();
}
```

The only method signature that should change is for block:

```
public void block(DelegateContext ctx, int side, int index,
                             PageFragment fragment)
throws IOException {}
```

The reason that the writer has been changed to a DelegateContext is because the block method will no longer need to write any Html source. The block source is already in the theme.html file, the only thing that the block method can do is define contexts for the block sides (using the switch statement is best). Since the theme.html source is now being loaded by this Java class, it will no longer have the {<block section name>**.**(TITLE|CONTENT)} automatically defined. This means for every case in the switch that deals with a block side, the appropriate context must be defined and appended to the context using the ctx.next("<context name>") method. The method returns a new instance of DelegateContext, which should be assigned to a new handle. Then use the <handle name>.put("<property name>", "<property value>") method to add the TITLE and CONTENT properties to the context.

Example:

```
DelegateContext block;
switch (side) {
  case Constants.SIDE_LEFT :
      block = ctx.next("left");
      block.put("TITLE", fragment.getTitle());
      block.put("CONTENT", fragment.getBody());
      .
```

```
        .       /* other such stuff */
        .
    break;

    case Constants.SIDE_CENTRE :
        block = ctx.next("middle");
        block.put("TITLE", fragment.getTitle());
        block.put("CONTENT", fragment.getBody());
        .
        .       /* other such stuff */
        .
    break;

    case Constants.SIDE_RIGHT :
        block = ctx.next("right");
        block.put("TITLE", fragment.getTitle());
        block.put("CONTENT", fragment.getBody());
        .
        .       /* other such stuff */
        .
    break;
  }
```

The method content for header, footer, message, openTable and closeTable will be changed to define contexts for those templates. The very least (and most sufficient for the table methods) is to use the Context.NULL_CONTEXT attribute and pass the empty context into the render method. If more is desired, see the Context section, App B.

Example:

```
protected void openTable(Writer writer)
      throws IOException {
  openTable.render(Context.NULL_CONTEXT, writer);
}
```

The render method is from the Template class and is used to render the loaded template on the page. This is why it is customary to name the template variables after the templates to be rendered. It avoids confusion as to which template instance's render method should be called.

Style

The style sheet is a simple mechanism for adding style, such as background colors, fonts, etc…, to web documents. Style sheets in CSS are made up of *rules*. A *rule* is a statement about one stylistic aspect of one or more elements. A *style sheet* is a set of one or more rules that apply to an HTML document. Each rule has three parts:

1. the *selector*, which tells the browser which part of the document is affected by the rule;
2. the *property*, which specifies what aspect of the layout is being set;
3. and the *value*, which gives the value for the style property.

H1 {color: blue}

The rule above sets the color of all first-level headings (h1). H1 is the selector (it selects all <h1> tags), color is the property to set, and blue is the value. In effect, this will treat all h1 tags as <h1 color='blue'>. Note html elements, such as H1, BODY, TABLE, etc…, are considered *type selectors*; there are other kinds of selectors as shown in the examples provided with this document.

To use the same formatting for multiple selectors, simply use a comma separated list when defining the rule. In addition, multiple properties can be set within one declaration.

Example: H1, H2, H3 {color:green
font-weight: bold}.

To "glue" the styles defined into the html source, the most flexible way is by using the @import statement. This is done in the <head> tags of the source and uses the <style> tags to tell the browser what style should be applied to the document. The style needs to define the type, which in this case should be type="text/css". This means a cascading style sheet written with plain text is the document type. The @import statement must define the url of the document. This is done with a url(<path to style.css>); statement.

Example:

```
<style type="text/css">@import url("themes/testTheme/style/style.css");</style>
```

More on Selectors

The type selectors have already been defined above. Other selectors include the universal selector (*), class selector (.<class name>), id selectors, and type selectors with pseudo-classes or pseudo elements (<type selector>:<pseudo-class or element name>).

The universal selector defines document wide style, such as a common font type. Defining a property values for an element, such as H1, will override the value set in the universal selector block, since H1 is more specific than *.

```
* {font-size: 20px} /* entire document has font size of 20 pixels */
H1 {font-size: 3px} /* except for h1 elements, font size is 3 pixels */
```

Class selectors allow for multiple style formats for single elements or for any element to reference. For example, if the links at the top of a web page are to have a different font size than the links at the bottom of the page, and all content in the middle will have a different font, then a possible style definition could be:

```
a.top-links {font-size: 12px}
a.bottom-links {font-size: 16px}
.middle-content {font-size: 30px}
```

To reference these styles for the desired <a> tags, the class attribute must be set.

Example of a link declaration at the top of the page with middle content underneath it:

```
<a class="top-links" href="www.example.com">
<span class="middle-content">This is relatively large text.</span>
```

NOTE: When defining the class name, the period must be used, but when referencing the name, no period is used.

Id selectors are similar to class selectors. The 'ID' attribute is guaranteed to have a unique value over the document. It can therefore be of special importance as a style sheet selector, and can be addressed with a preceding '#':

```
#z98y { letter-spacing: 0.3em }   /* general id selector rule */
H1#z98y { letter-spacing: 0.5em } /* <h1> specific id selector rule */

<P ID=z98y>Wide text</P>  /* to reference an id selector,
                             set the ID attribute */
```

In the above example, the first selector matches the 'P' element due to the 'ID' attribute value. The second selector specifies both an element type ('H1') and an ID value, and will therefore not match the 'P' element.

Pseudo-classes, used with the anchor tag, define states of an element, such as a mouse hovering on a link. Pseudo-elements, used commonly with the paragraph tag, define special pieces of an element's content. The anchor tag classes are:

- `a:link - normal link attributes`
- `a:active - properties to set when the mouse button is pressed on a link`
- `a:hover - properties to set when the mouse is hovering over a link`
- `a:visited - properties to set when a link has been visited`

The paragraph tag elements are:

- `p:first-line - properties to set for the first line of a paragraph`
- `p:first-letter - properties to set for the first letter of a paragraph`

The pseudo type definitions may be used in combination with the class selector definitions; however, only one class selector may be defined per pseudo type.

Example:

```
A.my-class:hover {color: orange}
```

More on properties

This is a list of common properties that are defined within style blocks.

Property Name	Property Value
Font properties	
Font-family	Comma separated list of font family names and generic font family names. BODY { font-family: gill, Helvetica, sans-serif } **family-name** The name of a font family of choice. In the last example, "gill" and "Helvetica" are font families. **generic-family** In the example above, the last value is a generic family name. The following generic families are defined: •serif' (e.g. Times) •sans-serif' (e.g. Helvetica) •cursive' (e.g. Zapf-Chancery) •fantasy' (e.g. Western) •monospace' (e.g. Courier)
Font-style	Normal, italic, or oblique. Default is normal. H1, H2, H3 { font-style: italic }
Font-variant	Normal or small-caps. Default is normal. H3 { font-variant: small-caps }
Font-size	An absolute size, relative size, length, or percentage. Default is medium. P { font-size: 12pt; } BLOCKQUOTE { font-size: larger } EM { font-size: 150% } EM { font-size: 1.5em } absolute-size An absolute-size keyword is an index to a table of font sizes computed and kept by the UA. Possible values are: [xx-small | x-small | small | medium | large | x-large | xx-large]. On a computer screen a scaling factor of 1.5 is suggested between adjacent indexes; if the 'medium' font is 10pt, the 'large' font could be 15pt. Different media may need different scaling factors. Also, the UA should take the quality and availability of fonts into account when computing the table. The table may be different from one font family to another. relative-size A relative-size keyword is interpreted relative to the table of font sizes and the font size of the parent element. Possible values are: [larger | smaller]. For example, if the parent element has a font size of 'medium', a value of 'larger' will make the font size of the current element be 'large'. If the parent element's size is not close to a table entry, the UA is free to interpolate between table entries or round off to the closest one. The UA may have to extrapolate table values if the numerical value goes beyond the keywords. Length and percentage values should not take the font size table into account when calculating the font size of the element. Negative values are not allowed.
Font-weight	Normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900 Default is normal. Normal weight is equal to 400. Bold weight is equal to 700. H1 { font-weight: 700 } /* bold */

Font	Any combination of property values for (note: the “/” character is literal): • font-style, font-variant, and font-weight. • font-style, font-variant, font-weight, font-size. • font-style, font-variant, font-weight, font-size / line-height. • font-size. • font-size / line-height. • font-size / line-height and font-family. • font-family. • font-style, font-variant, font-weight, and font-family. • font-style, font-variant, font-weight, font-size / line-height and font-family. P { font: 12pt/14pt sans-serif } P { font: 80% sans-serif } P { font: x-large/110% "new century schoolbook", serif } P { font: bold italic large Palatino, serif } P { font: normal small-caps 120%/120% fantasy }
Color properties	
Color	Natural color name or RGB(0-255, 0-255, 0-255) where 0-255 is an integer within the range of 0 to 255. The first integer determines red, the second determines green, and the third determines blue. RGB (0,0,0) is black, and RGB(255,255,255) is white. EM { color: red } /* natural language */ EM { color: rgb(255,0,0) } /* RGB range 0-255 */ Natural color names include hex values as well (e.g. #ffffff is white).
Background-color	Natural color name or transparent. Default is transparent. `H1 { background-color: #F00 }`
Background-image	URL of an image or none. Default is none. BODY { background-image: url(marble.gif) background-color: #F00 } P { background-image: none } When setting a background image, one should also set a background color that will be used when the image is unavailable. When the image is available, it is overlaid on top of the background color.

Background-repeat	Repeat, repeat-x, repeat-y, or no-repeat. Default is repeat.
	BODY {
	  background: red url(pendant.gif);
	  background-repeat: repeat-y;
	}
	
	A value of 'repeat' means that the image is repeated both horizontally and vertically. The 'repeat-x' ('repeat-y') value makes the image repeat horizontally (vertically), to create a single band of images from one side to the other. With a value of 'no-repeat', the image is not repeated.

Background-attachment	Scroll or fixed. Default is scroll.
	BODY {
	  background: red url(pendant.gif);
	  background-repeat: repeat-y;
	  background-attachment: fixed;
	}

Background-position	A pair of percentages or lengths, or any single key word or pair of keywords from the following list: • top • center • bottom • left • right Default is 0% 0%. BODY { background: url(banner.jpeg) 100% 0% } /* right top */ BODY { background: url(banner.jpeg) top center } /* 50% 0% */ BODY { background: url(banner.jpeg) 10cm 10cm } /* 10 cm to the right, 10 cm from the top */ BODY { background: url(banner.jpeg) bottom } /* 50% 100% */ With a value pair of '0% 0%', the upper left corner of the image is placed in the upper left corner of the box that surrounds the content of the element (i.e., not the box that surrounds the padding, border or margin). A value pair of '100% 100%' places the lower right corner of the image in the lower right corner of the element. With a value pair of '14% 84%', the point 14% across and 84% down the image is to be placed at the point 14% across and 84% down the element. With a value pair of '2cm 2cm', the upper left corner of the image is placed 2cm to the right and 2cm below the upper left corner of the element. If only one percentage or length value is given, it sets the horizontal position only; the vertical position will be 50%. If two values are given, the horizontal position comes first. Combinations of length and percentage values are allowed, e.g. '50% 2cm'. Negative positions are allowed. One can also use keyword values to indicate the position of the background image. Keywords cannot be combined with percentage values, or length values. The possible combinations of keywords and their interpretations are as follows: • 'top left' and 'left top' both mean the same as '0% 0%'. • 'top', 'top center' and 'center top' mean the same as '50% 0%'. • 'right top' and 'top right' mean the same as '100% 0%'. • 'left', 'left center' and 'center left' mean the same as '0% 50%'. • 'center' and 'center center' mean the same as '50% 50%'. • 'right', 'right center' and 'center right' mean the same as '100% 50%'. • 'bottom left' and 'left bottom' mean the same as '0% 100%'. • 'bottom', 'bottom center' and 'center bottom' mean the same as '50% 100%'. • 'bottom right' and 'right bottom' mean the same as '100% 100%'. If the background image is fixed with regard to the canvas (see the 'background-attachment' property above), the image is placed relative to the canvas instead of the element.
Background	Any combination of property values for: • background-color • background-image • background-repeat • background-attachment • background-position BODY { background: red } P { background: url(chess.png) gray 50% repeat fixed } The 'background' property always sets all the individual background properties. In the first rule of the above example, only a value for 'background-color' has been given and the other individual properties are set to their initial value. In the second rule, all individual properties have been specified.
Text properties	
Word-spacing	Normal or a length. Default is normal. `H1 { word-spacing: 1mm }` This rule will increase the spacing 1mm above normal.
Letter-spacing	Normal or a length. Default is normal. BLOCKQUOTE { letter-spacing: 0.1mm } Here, the letter-spacing between each character in 'BLOCKQUOTE' elements would be increased by '0.1mm'.
Text-decoration	None or any combination of the keywords: underline, overline, line-through, and blink. Default is none. A:link, A:visited, A:active { text-decoration: underline }

Vertical-align	A percentage or any one of the keywords: baseline, sub, super, top, text-top, middle, bottom, text-bottom. Default is baseline. EM { vertical-align: middle } Note: ‘parent’ refers to any html elements enclosing the element using the rule (e.g. <H1>The headline <EM>is</EM> important!</H1> the h1 tag is the parent of the em tag). The root parent is typically <body> since most tags lie within. 'baseline' align the baseline of the element (or the bottom, if the element doesn't have a baseline) with the baseline of the parent 'middle' align the vertical midpoint of the element (typically an image) with the baseline plus half the x-height of the parent 'sub' subscript the element 'super' superscript the element 'text-top' align the top of the element with the top of the parent element's font 'text-bottom' align the bottom of the element with the bottom of the parent element's font Another set of properties are relative to the formatted line that the element is a part of: 'top' align the top of the element with the tallest element on the line 'bottom' align the bottom of the element with the lowest element on the line Using the 'top' and 'bottom' alignment, unsolvable situations can occur where element dependencies form a loop. Percentage values refer to the value of the 'line-height' property of the element itself. They raise the baseline of the element (or the bottom, if it has no baseline) the specified amount above the baseline of the parent. Negative values are possible. E.g., a value of '-100%' will lower the element so that the baseline of the element ends up where the baseline of the next line should have been. This allows precise control over the vertical position of elements (such as images that are used in place of letters) that do not have a baseline.
Text-transform	None or any one of the keywords: capitalize, uppercase, or lowercase. Default is none. H1 { text-transform: uppercase} 'capitalize' uppercases the first character of each word 'uppercase' uppercases all letters of the element 'lowercase' lowercases all letters of the element 'none' neutralizes value of parent (see note in vertical-align).
Text-align	Left, right, center or justify. Default is browser dependent. DIV.center { text-align: center }
Text-indent	A length or percentage. Default is 0. P { text-indent: 3cm }
Line-height	Normal, a number, a length, or a percentage. Default is normal. DIV { line-height: 1.2; font-size: 10pt } /* number */ DIV { line-height: 1.2em; font-size: 10pt } /* length */ DIV { line-height: 120%; font-size: 10pt } /* percentage */ The three rules in the example above have the same resultant line height.

For more information go to http://www.w3.org/TR/REC-CSS1 and http://www.w3.org/TR/REC-CSS2/.

Appendix A: Special words

This section will describe some of the more important special words that Nukes uses for its templating, and the syntax required to utilize them.

TemplateAnalyzer.jj

The template analyzer is a parsing definition file that helps the WriterParser class recognize what text to replace with mapped values. In other words, it defines the syntax of the special tags that allow template writers to "plug in" variable data. This is a snippet of the analyzer file:

```
<BEGIN: "<!-- BEGIN " (~[" "])+ " -->" >

<END: "<!-- END " (~[" "])+ " -->" >

<PROPERTY: "${" (~["}"])* "}" >

<REF: "{" (<LITTERAL> ".")* <LITTERAL> "}" >

<#LITTERAL: (["A"-"Z","a"-"z","0"-"9","-","_"])* >

<TEXT: ~["\n"] >

<NEWLINE: "\n" >
```

Some pieces should look familiar, such as the "<!-- BEGIN". What the first line means is that the BEGIN symbol is defined by the string "<!-- BEGIN " with any non space characters and ends with the string " -->". This will recognize the special template comments such as <!-- BEGIN header -->. The second line does the same thing but for the END tag.

A property symbol starts with a "${" and contains any characters to make a property name and ends with the "}" (by this definition, this means spaces can be part of the property name). It is not recommended to have spaces as part of the property name due to the possibility of code (like in the WriterParser class) that may not support this. Instead, use the Java naming conventions or separate words with a '_'.

Properties are non-context specific, but are still restricted by visibility. By context specific, a context can be delegated to areas within a template programmatically (using the Java version). Properties can be used without the specifying a context, but must be qualified by the module name.

Example:

```
output.text("${quotes._CONFIRM_DELETE}");
output.formSubmit("${core._YES}");
```

In the above example, the output variable is an instance of the Html class, which constructs html tags around the passed in values. The text(String) and formSubmit(String) methods each take a string, builds the appropriate html source and appends it to the string buffer to be printed to the page later. The properties defines are for the quotes module and core. The quotes module property values are defined in a Resource_en_US.properties file, which defines the _CONFIRM_DELETE property in US English.

Example:

```
_CONFIRM_DELETE=Delete This Quote?
```

This should only be visible to (or used in) modules and templates within the "quotes" package. For a template created with the structure defined in the first section, ${<template name>.<property name>} will be the syntax to use.

Core is the Nukes core and has a global visibility. Every module and template has access to the core properties, which will be listed later.

A REF symbol, or reference, is similar to the PROPERTY symbol definition, except it is not defined with a '$'. The reference properties are context specific (see Context section, App B) and the reference names are defined by the LITERAL symbol. The LITERAL symbol describes literals as any set of capital and lowercase letters, digits, hyphens, and underscores (no spaces).

The TEXT symbol is anything that has not already been defined, except for the newline character. A newline, which typically has its own symbol definition, is a special Java (C++, etc…) symbol for the carriage return and line feed (analogous to pressing the enter key in a word processor) (may not be the same for other programming languages).

Key Handlers

The CoreModule class creates two key handlers: config and encodeurl. Key handlers are another means of creating reserved words using a special syntax. For each key handler created, an implementation of the handler must be provided.

The config key handler implementation grants access to the Nukes configuration variable mappings. One such mapping is for the site name, which can be manipulated within the JBoss jmx-console. To access the site name, the properties syntax must be used with the config qualifier and "siteName" property name. Note: Unlike the property syntax, to qualify a key handler the '**:**' character must be used instead of the '**.**'.

Example:

```
${config:siteName}          /* This will retrieve the value entered for the web site name */
```

Config Properties:

This is a listing of the config properties:

siteName	name of the web site.
defaultTheme	default theme for the web site.
slogan	web site slogan.
metakeywords	metakey words used for html properties and descriptors.
startPage	default web page (e.g. index.html)
defaultUserPage	default web page for a user.
favicon	path to web site icon.
minAge	minimum age to enter site.
minPass	minimum number of characters for a password.
defaultGroup	default group id number.
fileUploadSizeMax	maximum size of a file to upload.
banners	flag to indicate if bannars will be used.
themeChange	flag to indicate if themes can change.
charset	character set used for encoding.

The encodeurl key handler implementation uses the URLEncoder class, which provides a function to encrypt URL segments, keeping data that must be passed from page to page private. To encode a URL segment, the properties syntax must be used, qualified with the encodeurl keyword using the '**:**' character followed by the value that must be encoded.

Example:
```
/* encodes the name of an index to be created */
<a href='index.html?module=index&op=action&id=2&name=${encodeurl:" +
name.getCanonicalName() + "}'>Create</a>
```

Core Properties:

This is a listing of the available properties and values in the core. These properties have global visibility and can be accessed using the ${core.<property name>} syntax.

```
_ADMINMENU=Administration Menu
_ERRTRYHOME=Try the Home Page
_ERRMAILED=The details of this error have automatically been mailed to the webmaster.
_ERRSORRY=We're sorry. The page you requested
_ERRMAIL404=A 404 error was encountered by
_ERRUPPERCASE=Using UPPER CASE CHARACTERS
_ERRSTARTHERE=Or you can simply try to start from the
_ERRURLEND=Url ends with
_ERRALLLOWER=all names are in lower case only
_ERRMAILON=on
_ERRPAGENF=Page Not Found on
_ERR404REP=404 Error Report
_ERR404=404 Error Message\:
_ERRDOESNTEXIST=doesn't exist on
_ERRMAILREF=The referring page was\:
_ERRMAILURI=The URI which generated the error is\:
_ERRHP=home page
_ERRALLPAGES=all pages on
_ERRCOMMONM=Common Mistakes
_ERRENDWITH=end with
_ERRCOMMONH=Here are the most common mistakes in accessing
_TEXTLINK=Text link
_UPDATEFAILED=Attempted update of block failed
_DESCRIPTION255=Description\: (255 characters max)
_MPROBLEM=A Problem occurred\!
_DESCRIPTION=Description
_REMEMBERME=Remember me
_ACTIVE=Active
_SMEMBERS=Members
_PARENT=Parent
_ADDED=Added\:
_BOTLINKNAME3=Developers
_BOTLINKNAME2=Tasks
_USER=User
_STAFF=Staff
_BOTLINKNAME1=Report Bugs
_BACKTOTOP=Back to top
_LOGOUT=Logout
_GUESTS=guests
_EDITORREVIEW=Editor review
_LOGIN=Login
_DAYS=days
_BADAUTHKEY=No authorization to carry out operation
_BUTTONLINK=Button link
_LINKTITLE=Link Title
_NICKNAME=Username
_QUESTION=Secret question
_ANSWER=Secret answer
_SHOW=Show
_THENUMBER=The number
_REQUIRED=(required)
_OPTIONS=Options
```

```
_LAST30DAYS=Last 30 days
_UREALNAME=Real name
_WEBLINKS=Web Links
_REGISTEREDUSERS=Registered users
_EDIT=Edit
_COOKIE=COOKIE
_NEWUSER=New user
_DATESTRING=%A, %B %d @ %H\:%M\:%S
_DAY=Day
_SAVECHANGES=Save Changes
_BY=by
_ALREADYEXIST=already exists\!
_MAIN=Main
_ROOT=Root
_URL=URL
_NAME=Name
_TITLE=Title
_NICK2LONG=Username is too long. It must be less than 25 characters
_MODARGSERROR=Bad arguments for API function
_REQUEST=REQUEST
_UNREGISTEREDUSERS=Unregistered users
_TITLEAZ=Title (A to Z)
_BTN_CHANGEINFO=Change Info
_USERANDIP=Username and IP are recorded, so please don't abuse the system.
_TITLEZA=Title (Z to A)
_GET=GET
_CURRENTLY=We have
_BLOCKSNOAUTH=Not authorized to carry out operation on blocks
_DETAILS=Details
_ERRORINVNICK=Error in Nick Name.  Could be caused by a space in the name
_RESULTS=Results
_DBSELECTERROR=Database select error
_GOBACK=[ <a href\='javascript\:history.go(-1)'>Back</a> ]
_WELCOMETO=Welcome to
_NAMERESERVED=ERROR\: This username is reserved
_LOOKSRIGHT=Does this look right?
_DELETE=Delete
_COMMIT=Commit changes
_TOTALOF=Total of
_AUTHOR=Author
_IGNORE=Ignore
_MOREABOUT=More about
_CONTENT=Content
_LDESCRIPTION=Description\: (255 characters max)
_DATE=Date
_TEMPLATENAME=Example item name
_FUNCTIONS=Functions
_POSTEDBY=Posted by
_ONLINE=online
_TIMEZONES=IDLW NT HST YST PST MST CST EST AST GMT-3\:30 GMT-3 AT WAT GMT CET EET BT GMT+3\:30
GMT+4 GMT+4\:30 GMT+5 GMT+5\:30 GMT+6 WAST CCT JST ACS GST GMT+11 NZST
_THANKSBROKEN=Thank you for helping us maintain this directory's integrity.
_PREVIOUS=Previous
_NOTE=Note\:
_PLAINTEXT=Plain Old Text
_30DAYS=30 days
_DAY_OF_WEEK_SHORT=Sun Mon Tue Wed Thu Fri Sat
_ENV=ENV
_WARNING=Warning
_PRINTER=Printer friendly page
_MADE=\ made.
_HELLO=Hello
_YEAR=Year
```

```
_SERVER=SERVER
_VISIT=Visit
_OPTION=Option
_AND=and
_UP=Up
_FILES=FILES
_INSTRUCTIONS=Instructions
_ADDITIONALDET=Additional details
_TO=To
_NICKTAKEN=ERROR\: Username already taken
_NEW=New
_INDB=in our database
_GUEST=guest
_ALL=All
_TIMEZONEOFFSET=Time zone offset
_ACCESS_OVERVIEW=Overview
_ACCESS_EDIT=Edit
_REFRESH=Refresh
_YOUARENOTREGGED=You are not a registered user or you have not logged in.
_SHOWTOP=Show top
_UMONTH=Month
_RETURNTO=Return to
_MEMBER=member
_STATUS=Status
_DATELONG=%A, %B %d, %Y
_THREAD=Thread
_2WEEKS=2 weeks
_ALLOWEDHTML=Allowed HTML\:
_ADDURL=Add this URL
_ACTIVATE=Activate
_CONFIGURE=Configure
_DATETIMEBRIEF=%b %d, %Y - %I\:%M %p
_OWNER=Owner
_YOUAREREGGED=You are a registered user and are logged in.
_NOTRIGHT=Something is not right with passing a variable to this function. This message is just to keep things from messing up down the road
_START=Start
_LASTWEEK=Last week
_ACCEPT=Accept
_UNLIMITED=Unlimited
_DATETIMELONG=%A, %B %d, %Y - %I\:%M %p
_MODIFY=Modify
_INACTIVE=Inactive
_CREATEFAILED=Creation attempt failed
_ACCESS_ADMIN=Admin
_HTMLFORMATED=HTML formatted
_FLAT=Flat
_OPTIONAL=(optional)
_NONE=None
_LAST=Last
_DATEWRITTEN=Date written
_SITENAME=Site name
_SAVE=Save
_TEMPLATE=Template (example)
_APILOADFAILED=Unable to load API.
_ACCESS_DELETE=Delete
_FAILED=Failed\!
_MONTH_LONG=January February March April May June July August September October November December
_DOWN=Down
_TZOFFSETS=0 1 2 3 4 5 6 7 8 8.5 9 10 11 12 13 14 15 15.5 16 16.5 17 17.5 18 19 20 21 21.5 22 23 24
_LOADFAILED=Load of module failed
_ACCESS_READ=Read
_ON=on
```

```
_REPORTBROKEN=Report broken link
_OK=OK
_OF=of
_THEREARE=There are
_LANGUAGE_ENG=English
_YES=Yes
_DEACTIVATE=Deactivate
_SELECTPAGE=Select page
_POST=POST
_NO=No
_ERRORURLEXIST=ERROR\: This URL is already listed in the database\!
_CHARSET=ISO-8859-1
_NEXT=Next
_PREVIEW=Preview
_SIGNATURE=Signature
_USERNAME=Username
_PASSWORD=Password
_PASSWORDAGAIN=Confirm your password
_BEPATIENT=(please be patient)
_TEMPLATENUMBER=Example item number
_PAGE=Page
_NICKNOSPACES=ERROR\: There cannot be spaces in the username
_TIMEBRIEF=%I\:%M %p
_SUBJECT=Subject
_SEARCH_MEMBERS=Search Members
_EMAIL=E-mail
_LANGUAGE=Language
_LOGINCREATE=Login/Create an account
_NESTED=Nested
_DAY_OF_WEEK_LONG=Sunday Monday Tuesday Wednesday Thursday Friday Saturday
_LINKSDATESTRING=%d-%b-%Y
_SEARCH=Search
_READS=Reads
_EXTRANS=Extrans (html tags to text)
_SESSION=SESSION
_NOSUBJECT=No subject
_AUTOLINKSNOSUCHLINK=No such link present
_ADD=Add
_EXTENDEDTEXT=Extended text
_MODERATE=Moderate
_ORIGINAL=Original
_NEWTHISWEEK=New this week
_SEARCH_NO_MEMBERS=Members\: No members matched your search.
_ACCESS_MODERATE=Moderate
_TEMPLATENOAUTH=Not authorized to access Template module
_OFALL=of all
_THANKSFORINFO=Thank you for the information.
_SUBMIT=Submit
_ONN=on...
_MONTH_SHORT=Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
_GLOBALS=GLOBALS
_DATEBRIEF=%b %d, %Y
_1WEEK=1 week
_ACCESS_ADD=Add
_ADDEDON=Added on
_HITS=hits
_USERINFO=User info
_YOURNAME=Your name
_CONFIRM=Confirm
_EXTRAINFO=Extra Information
_PERMISSIONSNOAUTH=You are not authorized to carry out that operation
_ID=ID
_ONLINEMANUAL=Online manual
```

```
_MEMBERS=members
_SUBMITTER=Submitter
_UNKNOWN=Unknown
_TEMPLATENOSUCHITEM=No such item
_PROPOSED=Proposed
_ACCESS_NONE=None
_DELETEFAILED=Deletion attempt failed
_CLICK=Please click
_GO=GO
_LOGOUTEXIT=Logout/Exit
_THISISYOURPAGE=This is your personal page
_INFOCHANGED=Account information changed. Returning to your account page.
```

Appendix B: Contexts

Contexts are defined programmatically, or within the Java code using the DelegateContext class. A subtle example would be the left, middle, and right block format definitions. Each is actually a context defined in the ThemeTemplate class using the DelegatContext class. To define a section of a template to be within a context, use the <!-- BEGIN <context name> --> and <!-- END <context name> --> tags. This is illustrated in the examples in the Html Version. The reference property is used to display the title and content of the block using the {<context name>.(TITLE | CONTENT)} syntax, or **context syntax** . The example {right**.**TITLE} shows that "right" is the context defined for the right block section and the property defined is TITLE. The value of the property is set within the code using a put("<property name>", "<property value>") command.

NOTE: The header, footer, openTable, closeTable, and message sections within a theme are *templates*. They are contexts in their own right and are not defined using the DelegateContext class (as far as I know). They are root contexts and can still have context properties defined, but they do not have to be qualified by the context name (e.g. {HEADERPROPERTY}, if defined for the header, could be used, as is, within the header template definition).

Contexts are generally not used for themes, but in order to use them a Java/Html version hybrid should be used (see Hybrid section).

To use contexts, a template document must first be loaded into the repository. This is done in the start method of the Java class that defines the contexts for the template. First a Document must be declared and a template file must be loaded into it. If there are template files defined outside of the Tree structure described in the Structure section, then a class loader must be defined (e.g. `CLLoader loader = new CLLoader("org/jboss/nukes/addons/modules/template/templateFolder");`)

for that path and passed into the constructor for repository (e. g. `repository = new TemplateLoader(loader);`); otherwise, the ComponentSupport class takes care of repository.

```
Document doc = repository.loadTemplate("/template.tpl");
```

Template files can be **.**html or **.**tpl (for template). The tpl files are like the html files only they define just the sections to be rendered, like a snippet of a full html source file. The <!-- BEGIN <template name> --> and <!-- END <template name> --> tags are used to define the template sections to render.

Example of template.tpl:

```
<!-- BEGIN main -->
<div align="center">{WELCOME}
<table align="center">
<form action="index.html" method="GET">
<input type="hidden" name="module" value="template"/>
<input type="hidden" name="op" value="action"/>
<tr><td>{YOURNAME} : </td><td><input type="text" name="name" value=""/></td></tr>
<tr><td colspan="2"><input type="submit"/></td></tr>
</form>
</table>
</div>
<!-- END main -->
<!-- BEGIN action -->
<div align="center">{WELCOME}
</div>
<!-- END action -->
```

Next the templates from the template file must be added to the repository. Note: repository is defined as a class attribute with data type `TemplateLoader`.

```
repository.addTemplate("main", (Element)doc.selectSingleNode("/node/loop
[@name='main']/node"));
repository.addTemplate("action", (Element)doc.selectSingleNode("/node/loop
[@name='action']/node"));
```

The string parameter passed into the doc.selectSingleNode method is in XPath format. It is a special format used in xml parsing by the dom4j**.**Node class. The important part is the @name='<template name>'. The template name is the name used in the <!-- BEGIN <template name> --> and <!-- END <template name> --> tags to define the template sections (not the context sections, which use the same format). Once the repository has all of the templates for the file, create the templates and assign them to Template instances. Note: templateMain and templateAction are defined as class attributes with data type`Template`.

```
      templateMain = repository.createTemplate("main");
      templateAction = repository.createTemplate("action");
```

The next step is to implement methods to define the contexts for the templates and render them. It is customary to name the methods after the templates.

Example:

```
public void main(Page page)
{
   // Create a DelegateContext for replacing variables in the
   // templates.
   DelegateContext ctx = new DelegateContext();

   /* Add mapping for context with a localized string
    * directory src/resources/<module or theme name>-lib-jar
    * /org/jboss/<"modules" or "themes">
    * /<name of module or theme>/languages/
    * Resource_<language code>_<country code>.properties
    *
    * In this example the directory path is:
    * src/resources/template-lib-jar/org/jboss/modules/template/
    * Resource_en_US.properties
    */ see Appendix A
        // property name,    property value
      ctx.put("WELCOME", "${template.WELCOME}");

      // Display page with context specified.
      templateMain.render(ctx, page.getBodyWriter());
   }

public void action(Page page)
{
   // Get the parameter
   String name = page.getParameter("name");

   // If no name has been provided, just render the main page again
   if (name == null || name.length() == 0)
   {
      main(page);
      return;
   }
   // Create a DelegateContext for replacing variables in the
   // templates.
   DelegateContext ctx = new DelegateContext();

   /* Add mapping for context with a localized string
    * directory src/resources/<module or theme name>-lib-jar
    * /org/jboss/<"modules" or "themes">
    * /<name of module or theme>/languages/
    * Resource_<language code>_<country code>.properties
    *
    * In this example the directory path is:
    * src/resources/template-lib-jar/org/jboss/modules/template/
    * Resource_en_US.properties
    */ see Appendix A
     // property name,          property value
   ctx.put("WELCOME", "${template.WELCOME} " + name);

   // Display page with context specified.
   templateAction.render(ctx, page.getBodyWriter());
}
```

Using the ctx.put method to define a property name WELCOME, the template can now use the context syntax to plug the variably changing value into the document without having to edit the source each time.

```
<!-- BEGIN main -->
<div align="center">{WELCOME} // context property defined
<table align="center">
       .
       .
       .
```

Earlier in the document, the context syntax was defined as {<context name>.<property name>} (previously, (TITLE | CONTENT) was used in place of <property name>). In the above example, the WELCOME property was not qualified by any context name. This is because there is no context name defined since the property is used in the root context, or template context. The template context is the section of the template between the <!-- BEGIN <template name> --> and <!-- END <template name> --> tags. Any context properties used within the template context are automatically qualified as root properties.

To define sub-contexts, the ctx.next("<context name>") method must be used. The method returns a DelegateContext instance that should be assigned to an appropriately named handle. Then the "put" method can be used with the new handle to assign properties to the sub-context.

Example:

```
public void main(Page page)
{
   // Create a DelegateContext for replacing variables in the
   // templates.
   DelegateContext ctx = new DelegateContext();

   /* Add mapping for context with a localized string
    * directory src/resources/<module or theme name>-lib-jar
    * /org/jboss/<"modules" or "themes">
    * /<name of module or theme>/languages/
    * Resource_<language code>_<country code>.properties
    *
    * In this example the directory path is:
    * src/resources/template-lib-jar/org/jboss/modules/template/
    * Resource_en_US.properties
    */ see Appendix A
        // property name,    property value
      ctx.put("WELCOME", "${template.WELCOME}");

      // create a sub-context with handle name sub
      DelegateContext sub = ctx.next("sub-context");
      // add property to sub-context
      sub.put("HELLO", "Hello World!");

      // Display page with context specified.
      templateMain.render(ctx, page.getBodyWriter());
   }
```

For the context to be used, it must be qualified within the template file. The template file **must** define the context section or errors will occur. The section definition uses the <!-- BEGIN <context name> --> and <!-- END <context name> --> tags and definitions may be nested within the template definition (see HTML Version section examples with the left, middle, and right block definitions).

Example:

```
<!-- BEGIN main -->
<div align="center">{WELCOME}
<table align="center">
      <!-- BEGIN sub-context -->
      <tr><td>{sub-context.HELLO}</td></tr>
      <!-- END sub-context -->
</table>
</div>
<!-- END main -->
```

Nested contexts can be created in a similar fashion. Once the sub-context has been created, using the "next" method to create another sub-context inside the previous creating a **context tree**. Consequently, any properties defined for the nested context must be qualified by the names of the context tree from the root to the nested context name, separated by '**.**' characters.

Example:

```
public void main(Page page)
{
   // Create a DelegateContext for replacing variables in the
   // templates.
   DelegateContext ctx = new DelegateContext();

   /* Add mapping for context with a localized string
    * directory src/resources/<module or theme name>-lib-jar
    * /org/jboss/<"modules" or "themes">
    * /<name of module or theme>/languages/
    * Resource_<language code>_<country code>.properties
    *
    * In this example the directory path is:
    * src/resources/template-lib-jar/org/jboss/modules/template/
    * Resource_en_US.properties
    */ see Appendix A
        // property name,    property value
      ctx.put("WELCOME", "${template.WELCOME}");

      // create a sub-context with handle name sub
      DelegateContext sub = ctx.next("sub-context");
      // add property to sub-context
      sub.put("HELLO", "Hello World!");

      // create a nested context
      DelegateContext nested = sub.next("nested");
      // add a property
      nested.put("GOODBYE", "Goodbye World!");
      // add as many as needed
      nested.put("NEVERMIND", "Nevermind, I'll stay.");

      // Display page with context specified.
      templateMain.render(ctx, page.getBodyWriter());
   }
```

```
<!-- BEGIN main -->
<div align="center">{WELCOME}
<table align="center">
      <!-- BEGIN sub-context -->
      <tr><td>{sub-context.HELLO}</td></tr>
            <!-- BEGIN nested -->
            <tr><td>{sub-context.nested.GOODBYE}</td></tr>
            <tr><td>{sub-context.nested.NEVERMIND}</td></tr>
            <!-- END nested -->
      <!-- END sub-context -->
</table>
</div>
<!-- END main -->
```

Content control using contexts

Context definitions within a template will only be rendered if the context name is added to the context tree. This means that if there is content within a web page that should only be displayed given a certain condition is true, then define that content within a context section. Inside the template method that will render the web page template, check for the condition. If the condition is true, use the "next" method to add the context and effectively the content. Otherwise, the content will not appear.

NOTE: Contexts **cannot** be added to the context tree if there is no definition for it in the template source. Errors will occur.

Example:

```
public void main(Page page)
{
   // Create a DelegateContext for replacing variables in the
   // templates.
   DelegateContext ctx = new DelegateContext();

   /* Add mapping for context with a localized string
    * directory src/resources/<module or theme name>-lib-jar
    * /org/jboss/<"modules" or "themes">
    * /<name of module or theme>/languages/
    * Resource_<language code>_<country code>.properties
    *
    * In this example the directory path is:
    * src/resources/template-lib-jar/org/jboss/modules/template/
    * Resource_en_US.properties
    */ see Appendix A
      // property name,    property value
      ctx.put("WELCOME", "${template.WELCOME}");
```

```
        // create a sub-context with handle name sub
        DelegateContext sub = ctx.next("sub-context");
        // add property to sub-context
        sub.put("HELLO", "Hello World!");

        if(condition == true) {
              // create a nested context
              DelegateContext nested = sub.next("nested");
              // add a property
              nested.put("CONTENT", "Woo Hoo!");
        } else if (otherCondition == true) {
              // if only special content is needed, properties do
              // not have to be defined (creates an empty leaf on the
              // context tree)
              sub.next("just-content");
        } else {
          ; // no content to display
        }

        // Display page with context specified.
        templateMain.render(ctx, page.getBodyWriter());
    }

<!-- BEGIN main -->
<div align="center">{WELCOME}
<table align="center">
      <!-- BEGIN sub-context -->
      <tr><td>{sub-context.HELLO}</td></tr>
            <!-- BEGIN nested -->
            <tr>
            <td>The first condition is true if your see this!</td>
            <td>{sub-context.nested.CONTENT}</td>
            </tr>
            <!-- END nested -->
            <!-- BEGIN just-content -->
            <tr><td>The second condition is true</td></tr>
            <!-- END just-content -->
      <!-- END sub-context -->
</table>
</div>

<!-- END main -->
```

Appendix C: Examples

Manifest file content:

```
Manifest-Version: 1.0
Created-By: @java.vm.version@ (@java.vm.vendor@)
Specification-Title: @specification.title@
Specification-Version: @specification.version@
Specification-Vendor: @specification.vendor@
Implementation-Title: @implementation.title@
Implementation-URL: @implementation.url@
Implementation-Version: @implementation.version@
Implementation-Vendor: @implementation.vendor@
Implementation-Vendor-Id: @implementation.vendor.id@
```

Application.xml:

```
<?xml version="1.0"?>
<!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.2//EN"
"http://java.sun.com/j2ee/dtds/application_1_2.dtd">

<application>
      <display-name>Nukes Unisys Theme</display-name>
      <description></description>

      <module>
            <ejb>nukes-unisysTheme-lib.jar</ejb>
      </module>

<!-- for hybrid themes that use database persistence, add the ejb jar as well
   <module>
      <ejb>nukes-unisysTheme-ejb.jar</ejb>
   </module>
-->

</application>
```

Jboss-app.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

<jboss-app>
   <module>
      <service>nukes-unisysTheme.sar</service>
   </module>
</jboss-app>
```

Jboss-service.xml (for Html version):

NOTE: the numbers (1., 2., 3., etc…) are not to be included, they are related to the explanation in the Setting Up the Structure section.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE server>

<server>
   <mbean
1.    code="org.jboss.nukes.theme.ThemeTemplate"
```

```
2.     name="nukes.themes:name=extralite"

       xmbean-dd=""
       xmbean-code="org.jboss.nukes.component.NukesMBean">

3.     <depends>nukes.modules:name=core</depends>
       <depends>nukes.modules:name=html</depends>

4.     <constructor>
          <arg type="boolean" value="true"/>
       </constructor>

5.     <xmbean>
        <attribute name="Security">
           <security>
              <permission group="Admins" pattern=".*:.*:.*"
                   level="ADMIN"/>
           </security>
        </attribute>
        <attribute name="Id">jar:/theme.html</attribute>
        <attribute name="Root">
             org/jboss/nukes/core/themes/extralite</attribute>
       </xmbean>

   </mbean>

</server>
```

Jboss-service.xml (for Java version):

NOTE: the numbers (1., 2., 3., etc…) are not to be included, they are related to the explanation in the Setting Up the Structure section.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE server>

<server>
   <mbean
1.     code="org.jboss.nukes.core.themes.unisysTheme.UnisysTheme"

2.     name="nukes.themes:name=unisysTheme"

       xmbean-dd=""
       xmbean-code="org.jboss.nukes.component.NukesMBean">

3.     <depends>nukes.modules:name=core</depends>

4.     <constructor>
          <arg type="boolean" value="true"/>
       </constructor>

5.     <xmbean>
          <attribute name="Security">
             <security>
                <permission group="Users" pattern=".*:.*:.*"
                   level="READ"/>
                <permission group="Admins" pattern=".*:.*:.*"
                   level="ADMIN"/>
             </security>
          </attribute>
       </xmbean>
   </mbean>
</server>
```

Resource.properties:

```
BGCOLOR1=#ffffff
BGCOLOR2=#cccccc
BGCOLOR3=#ffffff
BGCOLOR4=#eeeeee
BGCOLOR5=#000000
BGCOLOR6=#7fff00
TEXTCOLOR1=#ffffff
TEXTCOLOR3=#000000
```

Nukes-web_1_0.dtd

```
<!ELEMENT nukes-web (nukes-mapping)>
<!ELEMENT nukes-mapping (url-pattern|servlet-name)*>
<!ELEMENT url-pattern (#PCDATA)>
<!ELEMENT servlet-name (#PCDATA)>
```

Build.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE project [
   <!ENTITY buildmagic
      SYSTEM "../tools/etc/buildfragments/buildmagic.ent">
   <!ENTITY tools
      SYSTEM "../tools/etc/buildfragments/tools.ent">
   <!ENTITY libraries
      SYSTEM "../tools/etc/buildfragments/libraries.ent">
   <!ENTITY modules
      SYSTEM "../tools/etc/buildfragments/modules.ent">
   <!ENTITY defaults
      SYSTEM "../tools/etc/buildfragments/defaults.ent">
   <!ENTITY targets
      SYSTEM "../tools/etc/buildfragments/targets.ent">
]>

<!-- $Id: build.xml,v 1.19 2004/03/14 11:59:06 sgwood Exp $ -->

<!--+==========================================================+-->
<!--| Nukes (The OpenSource CMS) Build File                    |-->
<!--|                                                          |-->
<!--| Distributable under GPL license.                         |-->
<!--| See terms of license at http://www.gnu.org.              |-->
<!--|                                                          |-->
<!--| This file has been designed to work with the 'tools'     |-->
<!--| module and Buildmagic extentions.                        |-->
<!--+==========================================================+-->

<project default="main" name="testTheme Nukes on JBoss Theme">

   <!--+==========================================================+-->
   <!--| Setup                                                    |-->
```

```
<!--|                                                          |-->
<!--| Include the common build elements.                       |-->
<!--|                                                          |-->
<!--| This defines many different targets, properties & paths.  |-->
<!--| It also sets up the basic extension tasks among other     |-->
<!--| things.                                                   |-->
<!--+==========================================================+-->

&buildmagic;
&libraries;
&modules;
&defaults;
&tools;
&targets;

<!-- ========================================================== -->
<!-- Initialization                                             -->
<!-- ========================================================== -->

<!--
   | Initialize the build system.  Must depend on _buildmagic:init'.
   | Other targets should depend on 'init' or things will
   | mysteriously fail.
 -->

<target name="init" unless="init.disable"
        depends="_buildmagic:init">
</target>

<!--+==========================================================+-->
<!--| Configuration                                            |-->
<!--|                                                          |-->
<!--| This target is invoked by the Buildmagic initialization   |-->
<!--| logic and should contain module specific configuration    |-->
<!--| elements.                                                 |-->
<!--+==========================================================+-->


<target name="configure" unless="configure.disable">
   <property name="jndi-root" value="nukes/testTheme"/>

   <!-- Configure thirdparty libraries -->
   <call target="configure-libraries"/>
   <path id="library.classpath">
      <fileset dir="${jboss.home}/client" includes="*.jar"/>
      <fileset dir="${jboss.home}/lib" includes="*.jar"/>
      <fileset dir="${jboss.home}/server/all/lib" includes="*.jar"/>
   </path>

   <!-- Configure modules -->
   <call target="configure-modules"/>
   <path id="dependentmodule.classpath">
      <path refid="jboss.nukes.classpath"/>
   </path>

   <filterset id="module.filterset">
     <!--
     <filter token=""
             value=""/>
     -->
   </filterset>
```

```
      <!--+========================================+-->
      <!--| Override any default properties here. |-->
      <!--+========================================+-->

      <!-- Configure defaults & build tools -->
      <call target="configure-defaults"/>
      <call target="configure-tools"/>

      <!--+========================================+-->
      <!--| Define module specific elements here. |-->
      <!--+========================================+-->
      <property name="javadoc.private" value="true"/>
      <property name="javadoc.protected" value="false"/>

   </target>


   <!--+==========================================================+-->
   <!--| Compile                                                  |-->
   <!--|                                                          |-->
   <!--| This target should depend on other compile-* targets for  |-->
   <!--| each different type of compile that needs to be performed,|-->
   <!--| short of documentation compiles.                         |-->
   <!--+==========================================================+-->

   <target name="compile"
      description="Compile all source files."
      depends="generate-classes,
               _default:compile-classes,
               _default:compile-etc,
               _default:compile-resources,
               _default:compile-ddl">
      <!-- Add module specific elements here. -->
   </target>

   <!-- Generate all class files -->
   <target name="generate-classes" depends="init">

      <mkdir dir="${build.gen.classes}"/>

<!-- example for generating Enterprise Java Beans for Hybrid themes using database
persistence
      <ejbdoclet
         destdir="${build.gen.classes}"
         mergedir="${source.resources}/ejb-jar/"
         excludedtags="@version,@author"
         ejbspec="2.0">

         <fileset dir="${source.java}">
            <include name="**/*EJB.java"/>
         </fileset>

         <localinterface havingClassTag="nukes.testTheme"/>
         <localhomeinterface havingClassTag="nukes.testTheme"/>

         <deploymentdescriptor
            xmlencoding="UTF-8"
          destdir="${build.resources}/nukes-testTheme-ejb-jar/META-INF"
            description=""
            displayname=""/>
         <jboss xmlencoding="UTF-8"
            version="3.2"
          destdir="${build.resources}/nukes-testTheme-ejb-jar/META-INF"
            mergedir="${source.resources}/${nukes.mergedir}"
            datasource="java:${nukes.datasource}"
            typemapping="${nukes.type-mapping}"/>
      </ejbdoclet>

-->

   </target>

   <!--+===========================================================+-->
   <!--| Generate Output                                           |-->
   <!--|                                                           |-->
   <!--| Generates the target output for this module. Target output|-->
   <!--| is the output which is ment to be released or used by     |-->
   <!--| external modules.                                         |-->
   <!--+===========================================================+-->

   <target name="output"
      description="Generate all target output."
      depends="compile">

      <mkdir dir="${build.lib}"/>

      <!-- nukes-testTheme-lib.jar -->
      <jar jarfile="${build.lib}/nukes-testTheme-lib.jar">
         <fileset dir="${source.bin}/nukes-testTheme-lib-jar"
            includes="**/*"/>
         <fileset dir="${build.resources}/nukes-testTheme-lib-jar"
            includes="**/*"/>
         <fileset dir="${build.classes}" includes="**/*"/>
      </jar>

      <!-- nukes-testTheme.sar -->
      <jar jarfile="${build.lib}/nukes-testTheme.sar"
            manifest="${build.etc}/nukes-testTheme-sar.mf">
         <fileset dir="${build.resources}/nukes-testTheme-sar"
            includes="**/*"/>
      </jar>

      <!-- nukes-testTheme.ear -->
      <jar jarfile="${build.lib}/nukes-testTheme.ear"
            manifest="${build.etc}/nukes-testTheme-ear.mf">
         <fileset dir="${build.lib}">
            <include name="nukes-testTheme.sar"/>
            <include name="nukes-testTheme-lib.jar"/>
         </fileset>
         <fileset dir="${build.resources}/nukes-testTheme-ear"
            includes="**/*"/>
      </jar>

   </target>


   <!-- ========================================================== -->
   <!-- Cleaning                                                   -->
   <!-- ========================================================== -->
   <!-- Clean up all build output -->
   <target name="clean" depends="_default:clean">
      <!-- Add module specific elements here. -->
   </target>

   <!--+=========================================================+-->
   <!--| Documents                                               |-->
   <!--|                                                         |-->
   <!--| Generate all documentation for this module.             |-->
   <!--+=========================================================+-->

   <target name="docs" depends="_default:docs">
      <!-- Add module specific elements here. -->
   </target>

   <!-- ========================================================== -->
   <!-- Misc.                                                      -->
   <!-- ========================================================== -->

   <target name="main" depends="most"/>
   <target name="all" depends="_default:all"/>
   <target name="most" depends="_default:most"/>
   <target name="help" depends="_default:help"/>

   <!-- ========================================================== -->
   <!-- Deployment                                                 -->
   <!-- ========================================================== -->

   <!--
      | Deploy the application
    -->
   <target name="deploy"
      description="Deploy."
      depends="output">
      <require file="${jboss.home}/server/${nukes.deploy.dir}/nukes"/>
      <copy file="${build.lib}/nukes-testTheme.ear"
            todir="${jboss.home}/server/${nukes.deploy.dir}/nukes"/>
   </target>

    <!--
      | deploy the app
    -->
    <target name="deploy-all"
            description="Deploy entire component"
            depends="deploy"/>

   <!--
      | Undeploy the application
    -->
   <target name="undeploy"
      description="Undeploy."
      depends="init">
      <delete
            file="${jboss.home}/server/${nukes.deploy.dir}/nukes/nukes-
                  testTheme.ear"/>
   </target>

</project>
```

Style.css:

```
* {
FONT: 12px Tahoma, Verdana, sans-serif;
}

BODY  {
BACKGROUND: #FFFFFF;
COLOR: #333333;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
MARGIN-TOP: 2;
MARGIN-LEFT: 2;
MARGIN-RIGHT: 2;
MARGIN-BOTTOM: 2;
}

TD {
FONT-FAMILY: Verdana, Helvetica, sans-serif;
FONT-SIZE: 8pt;
}

A {
BACKGROUND: none;
COLOR: #4C5EA8;
TEXT-DECORATION: underline
}

A:active {
BACKGROUND: none;
COLOR: #CC6600;
TEXT-DECORATION: underline
}

A:visited {
BACKGROUND: none;
COLOR: #4C5EA8;
TEXT-DECORATION: underline
}

A:hover {
BACKGROUND: none;
COLOR: #CC6600;
TEXT-DECORATION: underline
}

.pn-title {
FONT: bold 9pt Verdana, Helvetica, sans-serif;
BACKGROUND: none;
COLOR: #CC6600;
TEXT-DECORATION: none
}

A.pn-title {
FONT: bold 9pt Verdana,Helvetica,sans-serif;
BACKGROUND: none;
COLOR: #CC6600;
TEXT-DECORATION: underline
}

A.pn-title:active {
BACKGROUND: none;
COLOR: #CC6600;
FONT: bold 9pt Verdana, Helvetica, sans-serif;
```

```
TEXT-DECORATION: underline
}

A.pn-title:visited {
BACKGROUND: none;
COLOR: #CC6600;
FONT: bold 9pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: underline
}

A.pn-title:hover {
BACKGROUND: none;
COLOR: #4C5EA8;
FONT: bold 9pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

A.pn-normal {
BACKGROUND: none;
COLOR: #000000;
TEXT-DECORATION: underline
}

A.pn-normal:active {
BACKGROUND: none;
COLOR: #F3F3F3;
FONT: 8pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

A.pn-normal:visited {
BACKGROUND: none;
COLOR: #4C5EA8;
FONT: 8pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

A.pn-normal:hover {
BACKGROUND: none;
COLOR: #CC6600;
FONT: 8pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

A.pn-hometext {
BACKGROUND: none;
COLOR: #000000;
FONT: 8pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

A.pn-hometext:active {
BACKGROUND: none;
COLOR: #000000;
FONT: 8pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

A.pn-hometext:visited {
BACKGROUND: none;
COLOR: #4C5EA8;
```

```
FONT: 8pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

A.pn-hometext:hover {
BACKGROUND: none;
COLOR: #CC6600;
FONT: 8pt Verdana,Helvetica,sans-serif;
TEXT-DECORATION: underline
}

:TD.pn-hometext {
BACKGROUND: none;
COLOR: #000000;
FONT: 8pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none: line-height: 1
}

.pn-logo {
BACKGROUND: none;
COLOR: #191919;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
LETTER-SPACING: 7pt;
TEXT-DECORATION: none
}

.pn-sub {
BACKGROUND: none;
COLOR: #666666;
FONT: 7pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none
}

A.pn-sub {
BACKGROUND: none;
COLOR: #191919;
FONT: 8pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: underline
}

A.pn-sub:active    {
BACKGROUND: none;
COLOR: #F3F3F3;
FONT: 8pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: underline
}

A.pn-sub:visited {
BACKGROUND: none;
COLOR: #4C5EA8;
FONT: 8pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: underline
}

A.pn-sub:hover {
BACKGROUND: none;
COLOR: #CC6600;
FONT: 8pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: underline
}
```

```
.pn-logo {
BACKGROUND: none;
COLOR: #FFFFFF;
FONT-WEIGHT: bold;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
LETTER-SPACING: 3px;
TEXT-DECORATION: none
}

A.pn-logo {
BACKGROUND: none;
COLOR: #FFFFFF;
FONT-WEIGHT: bold;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
LETTER-SPACING: 3px;
TEXT-DECORATION: none
}

A.pn-logo:active {
BACKGROUND: none;
COLOR: #FFFFFF;
FONT-WEIGHT: bold;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
LETTER-SPACING: 3px;
TEXT-DECORATION: none
}

A.pn-logo: visited {
BACKGROUND: none;
COLOR: #FFFFFF;
FONT-WEIGHT: bold;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
LETTER-SPACING: 3px;
TEXT-DECORATION: none
}

A.pn-logo:hover   {
BACKGROUND: none;
COLOR: #CC6600;
FONT-WEIGHT: bold;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
LETTER-SPACING: 3px;
TEXT-DECORATION: none
}

.pn-logo-7pt {
BACKGROUND: none;
COLOR: #FFFFFF;
FONT-SIZE: 7pt;
FONT-WEIGHT: bold;
FONT-FAMILY: Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none
}

.pn-pagetitle {
BACKGROUND: none;
COLOR: #4C5EA8;
FONT: bold 10pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none
}
```

```
.pn-title-rblock {
BACKGROUND: none;
COLOR: #4C5EA8;
FONT: bold 10pt Verdana, Helvetica, sans-serif;
LETTER-SPACING: 0px;
TEXT-DECORATION: none
}

.pn-title-lblock {
BACKGROUND: none;
COLOR: #4C5EA8;
FONT: bold 10pt Verdana, Helvetica, sans-serif;
LETTER-SPACING: 1.5px;
TEXT-DECORATION: none
}

.pn-rblock {
BACKGROUND: none;
COLOR: #000000;
FONT: 8pt Verdana, Helvetica, sans-serif;
LETTER-SPACING: 0px;
TEXT-DECORATION: none
}

.pn-bartitle {
BACKGROUND: none;
COLOR: #003399;
FONT: bold 8pt Verdana, Helvetica, sans-serif;
LETTER-SPACING: 2px;
TEXT-DECORATION: none
}

.pn-normal {
BACKGROUND: none;
ALIGN: left;
COLOR: #191919;
FONT: 8pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none
}


.pn-footer {
BACKGROUND: none;
COLOR: #FFFFFF;
FONT: 6pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none
}

.pn-art {
BACKGROUND: none;
COLOR: #191919;
FONT: 10pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none;
line-height: 1.5
}

TD.pn-normal {
BACKGROUND: none;
COLOR: #666666;
FONT: 10pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION:
```

```
none: line-height: 1.5
}

TD.pn-title {
BACKGROUND: none;
COLOR: #4C5EA8;
FONT: bold 14pt Verdana, Helvetica, sans-serif;
TEXT-DECORATION: none
}

INPUT {
background : #EBF2FD;
color: #000000;
font-size: 10px;
}

INPUT.pn-text {
BACKGROUND : #F3F3F3;
COLOR: #000000;
BORDER: solid 1px #000000;
FONT-SIZE: 10pt
}

INPUT.r-button {
BACKGROUND : none;
COLOR: #000000;
FONT-SIZE: 10pt
}

INPUT.pn-button {
BACKGROUND : #F3F3F3;
COLOR: #000000;
BORDER: 1px solid #000000;
FONT-SIZE: 10pt;
border-collapse: collapse
}

TEXTAREA.pn-text {
BACKGROUND : #F3F3F3;
COLOR: #000000;
BORDER: solid 1px #000000;
FONT-SIZE: 10pt;
border-bottom: 1px dashed
}

SELECT.pn-text {
BACKGROUND : #F3F3F3;
COLOR: #000000;
BORDER: solid 1px #000000;
FONT-SIZE: 10pt
}

/* Centre blocks and admin messages */

.message-centre {
FONT: normal 10pt Verdana, Helvetica, sans-serif;
border-style: none;
}

.message-centre .border2 {
border-style: none;
```

```
border-width: 0;
padding: 0;
}

TD.message-centre {
FONT: normal 10pt Verdana, Helvetica, sans-serif;border-style: none;
border: 1px;
padding: 3px;
}

.message-centre .pn-title {
FONT: bold 10pt Verdana, Helvetica, sans-serif;
color: #4C5EA8;
}

.pn-code {
font-family: Courier, 'Courier New', sans-serif; font-size: 11px; color: #006600;
background-color: #FAFAFA; border: #D1D7DC; border-style: solid;
border-left-width: 1px; border-top-width: 1px; border-right-width: 1px; border-bottom-
width: 1px;
}
.pn-quote {
font-family: Verdana, Arial, Helvetica, sans-serif; font-size: 11px; color: #444444;
line-height: 125%;
background-color: #FAFAFA; border: #D1D7DC; border-style: solid;
border-left-width: 1px; border-top-width: 1px; border-right-width: 1px; border-bottom-
width: 1px;
}
```

Theme templates:

Html versions:

Extralite:

```
<html>
<head>
  <base href="../../">
  <style type="text/css">
          @import url("themes/extralite/style/style.css");</style>
</head>
<body>

<!-- BEGIN header -->
<table border="0" cellpadding="4" cellspacing="0" width="100%"
      align="center">
  <tr>
    <td bgcolor="#ffffff">
      <table border="0" cellspacing="0" cellpadding="3" width="100%"
            bgcolor="#ffffff">
        <tr>
          <td>
            <a href="index.html">
              <img src="modules/core/images/nukes_logo_orange.jpg"
                   Alt="${core._WELCOMETO} ${config:siteName}"
                   border="0">
            </a>
          </td>
        </tr>
      </table>
    </td>
```

```
</tr>
<tr>
  <td valign="top" width="100%" bgcolor="#ffffff">
    <table border="0" cellspacing="0" cellpadding="2" width="100%">
      <tr>
        <td valign="top" width="150" bgcolor="#ffffff">
          <!-- BEGIN left -->
          <table border="0" cellspacing="0" cellpadding="0"
                width="100%" bgcolor="#000000">
            <tr>
              <td>
                <table width="100%" border="0" cellspacing="1"
                      cellpadding="3">
                  <tr>
                    <td bgcolor="#ffffff"
                      class="pn-title">{left.TITLE}</td>
                  </tr>
                  <tr>
                    <td bgcolor="#ffffff"
                      class="pn-normal">{left.CONTENT}</td>
                  </tr>
                </table>
              </td>
            </tr>
          </table>
          <br/>
          <!-- END left -->
          <img src="themes/extralite/images/pix.gif" border="0"
                width="100%" height="1" alt="">
        </td>
        <td>&nbsp;&nbsp;</td>
        <td valign="top">
          <!-- BEGIN middle -->
          <table border="0" cellspacing="0" cellpadding="0"
                width="100%" bgcolor="#000000">
            <tr>
              <td>
                <table width="100%" border="0" cellspacing="1"
                      cellpadding="3">
                  <tr>
                    <td bgcolor="#ffffff"
                      class="pn-title">{middle.TITLE}</td>
                  </tr>
                  <tr>
                    <td bgcolor="#ffffff"
                      class="pn-normal">{middle.CONTENT}</td>
                  </tr>
                </table>
              </td>
            </tr>
          </table>
          <br/>
          <!-- END middle -->
          <!-- END header -->
          <!-- BEGIN footer -->
        </td>
        <td>&nbsp;&nbsp;</td>
        <td valign="top" width="150" bgcolor="#ffffff">
          <!-- BEGIN right -->
          <table border="0" cellspacing="0" cellpadding="0"
                width="100%" bgcolor="#000000">
              <tr>
                <td>
                  <table width="100%" border="0" cellspacing="1"
                        cellpadding="3">
                    <tr>
                      <td bgcolor="#ffffff"
                        class="pn-title">{right.TITLE}</td>
                    </tr>
                    <tr>
                      <td bgcolor="#ffffff"
                        class="pn-normal">{right.CONTENT}</td>
                    </tr>
                  </table>
                </td>
              </tr>
            </table>
            <br/>
            <!-- END right -->
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<!-- END footer -->

<!-- BEGIN openTable -->
<table width="100%" border="0" cellspacing="1" cellpadding="0"
      bgcolor="#cccccc">
  <tr>
    <td>
      <table width="100%" border="0" cellspacing="1" cellpadding="8"
            bgcolor="#ffffff">
        <tr>
          <td>
<!-- END openTable -->

<!-- BEGIN closeTable -->
          </td>
        </tr>
      </table>
    </td>
  </tr>
</table>
<br/>
<!-- END closeTable -->

<!-- BEGIN message -->
<table width="100%" border="0" cellspacing="1" cellpadding="0"
      bgcolor="#cccccc">
  <tr>
    <td>
      <table width="100%" border="0" cellspacing="1" cellpadding="8"
            bgcolor="#ffffff">
        <tr>
          <td><div style="text-align: center"
                  class="pn_title">{MESSAGE}</div></td>
        </tr>
      </table>
    </td>
  </tr>
```

```
</table>
<br/>
<!-- END message -->

</body>
</html>
```

Imagic:

```
<html>
<head>
  <base href="../../">
  <style type="text/css">
          @import url("themes/imagic/style/style.css");</style>
</head>
<body>

<!-- BEGIN header -->
<div align="center">
   <table width="100%" border="0" cellpadding="0" cellspacing="0">
      <tr>
         <td colspan="2" valign="top">
            <table width="100%" border="0" cellpadding="0"
                  cellspacing="0" style="margim-bottom:10px">
               <tr>
                  <td colspan="2" valign="top">
                     <table width="100%" border="0" cellpadding="0"
                              cellspacing="0" bgcolor="#2288FF">
                        <tr>
                           <td colspan="2" valign="top">
                              <div id="siteName">
                                    &nbsp;${config:siteName}</div>
                           </td>
                        </tr>
                        <tr bgcolor="#446699">
                           <td id="slogan">
                              <div style="color: #FFFFFF">
                                    &nbsp;${config:slogan}</div>
                           </td>
                           <td width="2%" align="right" nowrap>
                              <div align="right" class="slogan"
                                    style="font-weight: bold;">
                                 <!-- usertime -->&nbsp;&nbsp;
                              </div>
                           </td>
                        </tr>
                     </table>
                  </td>
               </tr>
               <tr>
                  <td colspan="2">
                     <img src="themes/imagic/images/ombra.gif"
                        width="100%" height="10" border="0">
                  </td>
               </tr>
            </table>
         </td>
      </tr>
      <tr>
         <td valign="top" colspan="2">
```

```
<table width="100%" border="0" cellpadding="0"
      cellspacing="0">
   <tr>
      <td width="5" valign="top" bgcolor="#FFFFFF">
         <img src="themes/imagic/images/spacer.gif"
             width="5" height="1">
      </td>
      <td width="150" valign="top" bgcolor="#FFFFFF">
      <!-- BEGIN left -->
         <table border="0" cellpadding="0" cellspacing="0"
             width="150">
            <tr>
               <td align="right" style="font-weight: bold;
                        border-bottom:1px solid #2288ff">
                   {left.TITLE}</td>
            </tr>
         </table>
         <table border="0" cellpadding="0" cellspacing="0"
             width="150" bgcolor="#F9F9F9">
            <tr valign="top">
               <td bgcolor="#F9F9F9">{left.CONTENT}</td>
            </tr>
         </table>
         <br/>
      <!-- END left -->
      </td>
      <td width="5" valign="top" bgcolor="#FFFFFF">
         <img src="themes/imagic/images/spacer.gif"
             width="9" height="1">
      </td>
      <td valign="top" bgcolor="#FFFFFF">
         <br/>
         <div style="clear:both;line-height:5px;"></div>
                   <!-- END header --><!-- BEGIN footer -->
                   <div style="clear:both;
                         line-height:5px;"></div>
      </td>
      <td width="5" valign="top" bgcolor="#FFFFFF">
         <img src="themes/imagic/images/spacer.gif"
             width="9" height="1">
      </td>
      <td width="150" valign="top" bgcolor="#FFFFFF">
      <!-- BEGIN right -->
         <table border="0" cellpadding="0" cellspacing="0"
             width="160">
            <tr>
               <td style="font-weight: bold;
                        border-bottom:1px solid #2288ff">
                {right.TITLE}</td>
            </tr>
         </table>
         <table border="0" cellpadding="0" cellspacing="0"
             width="160" bgcolor="#F9F9F9">
            <tr valign="top">
               <td bgcolor="#F9F9F9">{right.CONTENT}</td>
            </tr>
         </table>
         <br/>
      <!-- END right -->
      </td>
      <td width="5" valign="top" bgcolor="#FFFFFF">
```

```
                      <img src="themes/imagic/images/spacer.gif"
                         width="5" height="1">
                  </td>
               </tr>
            </table>
         </td>
      </tr>
   </table>
   <table width="100%" cellpadding="1" cellspacing="0" border="0"
      bgcolor="#FFFFFF">
      <tr align="center">
         <td width="100%" colspan="3">
         <!-- banner display -->
         <br/>
         <!-- footmsg -->
          </td>
      </tr>
   </table>
</div>
<!-- END footer -->

<!-- BEGIN message -->
<table width="100%" border="0" cellspacing="1" cellpadding="0"
      bgcolor="#2288FF">
   <tr>
      <td>
         <table width="100%" border="0" cellspacing="1" cellpadding="8"
            bgcolor="#F9F9F9">
            <tr>
               <td><div class="pn-title" style="text-align: center">
                        {MESSAGE}</div></td>
            </tr>
         </table>
      </td>
   </tr>
</table>
<br/>
<!-- END message -->

<!-- BEGIN openTable -->
<table width="100%" border="0" cellspacing="1" cellpadding="0"
      bgcolor="#2288FF">
   <tr>
      <td>
         <table width="100%" border="0" cellspacing="1" cellpadding="8"
            bgcolor="#F9F9F9">
            <tr>
               <td>
<!-- END openTable -->

<!-- BEGIN closeTable -->
               </td>
            </tr>
         </table>
      </td>
   </tr>
</table>
<br/>
<!-- END closeTable -->

</body>
```

```
</html>
```

Java Version:

UnisysTheme.java

```
/**
 * Nukes: The OpenSource CMS
 * Distributable under GPL license.    See
 * terms of license at gnu.org.
 */
package org.jboss.nukes.core.themes.unisysTheme;

import org.jboss.nukes.Constants;
import org.jboss.nukes.handler.PageFragment;
import org.jboss.nukes.handler.PageResult;
import org.jboss.nukes.theme.ThemeSupport;

import java.io.IOException;
import java.io.Writer;

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


/**
 * @author <a href="mailto:michael.mcnamara1@unisys.com">Michael L. P. McNamara</a>
 * @version $Revision: 1.19 $
 */
public class UnisysTheme
      extends ThemeSupport {
    /**
     * Creates a new UnisysTheme object.
     */
    public UnisysTheme() {
          super();
    }

    /**
     * Creates a new UnisysTheme object.
     *
     * @param persistent true to persist in database
     */
    public UnisysTheme(boolean persistent) {
          super(persistent);
    }

    /**
     * render blocks using theme
     *
     * @param writer Abstract class for writing to character streams.
     * @param side side of page
     * @param index index of block
     * @param fragment fragment of page to render
     *
     * @throws IOException unknown exception
     */
    public void block(Writer writer, int side, int index,
                      PageFragment fragment)
          throws IOException {
          switch (side) {
                  case Constants.SIDE_LEFT :
                     writer.write("<table width=\"150\" border=\"0\" " +
                                  "cellspacing=\"0\" cellpadding=\"5\">\n" +
                                  "<tr>\n" + "<td class=\"pn-title-lblock\">");
                     writer.write(fragment.getTitle());
                     writer.write("</td>\n" + "</tr>\n" + "<tr>\n" +
                                  "<td valign=\"top\">");
                     process(fragment.getBody(), writer);
                     writer.write("\n" + "</td>\n" + "</tr>\n" + "</table>\n" +
                                  "<table width=\"150\" border=\"0\" " +
                                  "cellspacing=\"0\" cellpadding=\"1\">\n" +
                                  "<tr>\n" + "<td>\n" + "<hr size=\"1\">\n" +
                                  "</td>\n" + "</tr>\n" + "</table>\n");

                     break;

                  case Constants.SIDE_CENTRE :
                     writer.write("<table width=\"100%\" border=\"0\" " +
                                  "cellspacing=\"0\" cellpadding=\"5\">\n" +
                                  "<tr>\n" + "<td class=\"pn-title\">");
                     writer.write(fragment.getTitle());
                     writer.write("\n" + "</td>\n" + "</tr>\n" + "</table>\n");

                     writer.write("<font class=\"pn-normal\">");
                     process(fragment.getBody(), writer);
                     writer.write("</font>");

                     break;

                  case Constants.SIDE_RIGHT :
                     writer.write("<br><table width=\"140\" border=\"0\" " +
                                  "background=\"themes/unisysTheme/images/" +
                                  "right_bkg.gif\" cellpadding=\"0\" " +
                                  "cellspacing=\"0\">\n <tr>\n" +
                                  "<td valign=\"top\" height=\"13\" colspan=\"2\">\n"
+
                                  "<div align=\"left\">" +
                                  "<img src=\"themes/unisysTheme/images/" +
                                  "corner_top_right.gif\" width=\"15\" " +
                                  "height=\"13\"></div>\n </td>\n" +
                                  "<td rowspan=\"3\" " +
                                  "background=\"themes/unisysTheme/images/" +
                                  "right_bkg.gif\" width=\"5\">" +
                                  "<img src=\"themes/unisysTheme/images/blank.gif\" "
+
                                  "width=\"2\" height=\"100%\" alt=\"\" " +
                                  "border=\"0\"></td>\n" +
                                  "</tr>\n <tr>\n " +
                                  "<td class=\"pn-title-rblock\" colspan=\"2\">");
                     writer.write(fragment.getTitle());
                     writer.write("\n" + "</td>\n" + "</tr>\n" + "<tr>\n" +
                                  "<td width=\"8%\">" +
                                  "<img src=\"themes/unisysTheme/images/blank.gif\" "
+
                                  "width=\"10\" height=\"1\" alt=\"\" " +
                                  "border=\"0\"></td>\n" +
                                  "<td width=\"91%\">");
                     process(fragment.getBody(), writer);
                     writer.write("</td>\n" + "</tr>\n" + "<tr>\n" +
                                  "<td height=\"11\" colspan=\"2\" " +
                                  "valign=\"bottom\">\n" +
```

```java
                              "<div align=\"left\">" +
                              "<img src=\"themes/unisysTheme/images/" +
                              "corner_bottom_right.gif\" width=\"11\" " +
                              "height=\"11\"></div>\n" +
                              "</td>\n" + "</tr>\n" + "</table>");

                break;
        }
    }

    /**
     * render footer on page
     *
     * @param result content to render
     * @param writer Abstract class for writing to character streams.
     *
     * @throws IOException unknown error
     */
    public void footer(PageResult result, Writer writer)
        throws IOException {
        writer.write("</td>" +
                     "<td width=\"140\" valign=\"top\" bgcolor=\"#FFFFFF\">");
        blocks(writer, Constants.SIDE_RIGHT,
                result.getFragments(Constants.SIDE_RIGHT));
        writer.write("<br></td>\n" + "</tr>\n" + "</table>\n" +
                     "<table width=\"100%\" border=\"0\" cellspacing=\"0\" " +
                     "cellpadding=\"0\" bgcolor=\"#ffffff\">\n" +
                     "<tr bgcolor=\"#ffffff\">\n" +
                     "<td height=\"10\"><img src=\"themes/unisysTheme/images/" +
                     "rightbar.gif\" width=\"1\" height=\"1\" " +
                     "border=0 alt=\"\"></td>\n" +
                     "</tr>\n" + "<tr bgcolor=\"#ffffff\">\n" + "<td>\n" +
                     "<table width=\"90%\" border=\"0\" cellspacing=\"0\" " +
                     "cellpadding=\"5\" align=\"center\">\n" +
                     "<tr>\n" + "<td>\n" + "<div align=\"center\">\n" +
                     "<font class=\"pn-sub\">\n");
        writer.write("<a href=\"http://www.unisys.com/about__unisys/" +
                     "copyright/index.htm\">Copyright © Unisys 2004</a> / " +
                     "<a href=\"privacy.html\">Privacy Policy</a> / " +
                     "<a href=\"http://iwww4.unisys.com/policies/" +
                     "LEG/LEG6_1.asp\">For Internal Use Only</a> / " +
                     "<a href=\"contact.html\">Contact</a>");
        writer.write("</font>\n" + "</div>\n" + "</td>\n" + "</tr>\n" +
                     "</table>\n" + "</td>\n" + "</tr>\n" + "</table>\n" +
                     "</table>" + "</body>");
    }

    /**
     * render the header on the page
     *
     * @param result content to render
     * @param writer Abstract class for writing to character streams.
     *
     * @throws IOException unknown error
     */
    public void header(PageResult result, Writer writer)
        throws IOException {
        writer.write("<body text=\"#333333\" link=\"#000000\" alink=\"#FF9900\"" +
                     " vlink=\"#CC6600\" topmargin=\"0\" marginheight=\"0\" " +
                     "marginwidth=\"0\" leftmargin=\"0\" rightmargin=\"0\" " +
                     "bgcolor=\"#D0D0D0\">" +
                     "<table cellpadding=\"0\" cellspacing=\"0\" border=\"0\" " +
                     "width=\"100%\" bgcolor=\"#D0D0D0\">\n" +
                     "<tr>\n <td colspan=\"6\" valign=\"top\" height=\"91\" " +
                     "background=\"themes/unisysTheme/images/" +
                     "corner_bottom_right.gif\">\n" +
                     "<a href=\"index.html\">" +
                     "<img src=\"themes/unisysTheme/images/" +
                     "unisys_logo_and_tagline.gif\" border=\"0\" " +
                     "align=\"top\"></a>\n </td>\n" + "</tr>\n" + "<tr>\n");
        writer.write("<td background=\"themes/unisysTheme/images/" +
                     "topgrad_blue.jpg\" valign=\"left\" align=\"center\" " +
                     "width=\"16%\" height=\"21\">&nbsp;<a class=\"pn-sub\" " +
                     "href=\"http://iwww4.unisys.com/Directory/CdPs.asp\">" +
                     "Corporate Directory</a>\n");
        writer.write("<td background=\"themes/unisysTheme/images/" +
                     "topgrad_blue.jpg\" valign=\"left\" align=\"center\" " +
                     "width=\"16%\" height=\"21\">&nbsp;<a class=\"pn-sub\" " +
                     "href=\"http://webtime.ea.unisys.com:8010/images/" +
                     "uniwt001\">WebTime</a>\n");
        writer.write("<td background=\"themes/unisysTheme/images/" +
                     "topgrad_blue.jpg\" valign=\"left\" align=\"center\" " +
                     "width=\"16%\" height=\"21\">&nbsp;<a class=\"pn-sub\" " +
                     "href=\"http://iwww.unisys.com/employee/default.asp\">" +
                     "Employee Network</a>\n");
        writer.write("<td background=\"themes/unisysTheme/images/" +
                     "topgrad_blue.jpg\" valign=\"left\" align=\"center\" " +
                     "width=\"16%\" height=\"21\">&nbsp;<a class=\"pn-sub\" " +
                     "href=\"http://iwww.ess.unisys.com/servlets/psportal/" +
                     "peoplesoft8_PSFP/?cmd=login\">Employee Self Service</a>\n");

        if (!getApi().userLoggedIn()) {
            writer.write("<td background=\"themes/unisysTheme/images/" +
                        "topgrad_blue.jpg\" valign=\"right\" align=\"center\" "
+
                        "width=\"10%\" height=\"21\">&nbsp;<a class=\"pn-sub\"
" +
                        "href=\"index.html?module=user&op=loginscreen\">" +
                        "Sign On</a>\n");
        } else {
            writer.write("<td background=\"themes/unisysTheme/images/" +
                        "topgrad_blue.jpg\" valign=\"right\" align=\"center\" "
+
                        "width=\"10%\" height=\"21\">&nbsp;<a class=\"pn-sub\"
" +
                        "href=\"index.html?module=user&op=logout\">" +
                        "Sign Off</a>\n");
        }

        writer.write("</td>\n" + "</tr>\n" + "</table>\n" +
                     "<table width=\"100%\" border=\"0\" cellspacing=\"0\" " +
                     "cellpadding=\"0\">\n <tr>\n" +
                     "<td width=\"150\" valign=\"top\" bgcolor=\"#ffffff\">\n" +
                     "<table width=\"100%\" border=\"0\" cellspacing=\"0\" " +
                     "cellpadding=\"0\">\n <tr bgcolor=\"#ffffff\">\n" +
                     "<td><img src=\"themes/unisysTheme/images/blank.gif\" " +
                     "width=\"1\" height=\"17\" alt=\"\" border=\"0\">\n" +
                     "</td>\n" + "</tr>\n" + "<tr>\n <td valign=\"top\">\n");
        blocks(writer, Constants.SIDE_LEFT,
                result.getFragments(Constants.SIDE_LEFT));
        writer.write("</td>\n" + "</tr>\n" + "</table>\n" + "</td>\n" +
                     "<td width=\"100%\" valign=\"top\" align=\"left\" " +
                            "bgcolor=\"#F0F8FF\">\n");

            blocks(writer, Constants.SIDE_CENTRE,
                     result.getFragments(Constants.SIDE_CENTRE));
      }

      public void process(PageResult result, Writer writer)
            throws IOException {
            result.getHead().getPrintWriter().print("<style type=\"text/css\">@import
" +
                                                     "url(\"themes/" + getName() +
                                                     "/style/style.css\");</style>");
            writer.write("<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 4.01
Transitional//" +
                         "EN\">\n<html>\n<head>\n");
            writer.write(result.getHead().toString());
            writer.write("</head><body>\n");
            header(result, writer);
            process(result.getMain().getBody(), writer);
            footer(result, writer);
            writer.write("</body>\n</html>\n");
      }

      /**
       * Html tags to close an open table
       *
       * @param writer Abstract class for writing to character streams.
       *
       * @throws IOException unknown error
       */
      protected void closeTable(Writer writer)
            throws IOException {
            writer.write("</td></tr></table></td></tr></table>\n");
      }

      /**
       * Html tags to open a table - must have a corresponding closeTable method
       *
       * @param writer Abstract class for writing to character streams.
       *
       * @throws IOException unknown error
       */
      protected void openTable(Writer writer)
            throws IOException {
            writer.write("<br><table width=\"100%\" border=\"1\" cellspacing=\"1\" " +
                         "cellpadding=\"0\"><tr><td>\n" +
                         "<table width=\"100%\" border=\"0\" cellspacing=\"1\" " +
                         "cellpadding=\"5\"><tr><td>\n");
      }

      /**
       * Render blocks on the page
       *
       * @param writer Abstract class for writing to character streams.
       * @param side side to render block - 0 == left, 1 == center, 2 == right
       * @param blocks list of blocks to render
       *
       * @throws IOException unknown error
       */
      private void blocks(Writer writer, int side, List blocks)
            throws IOException {
```

```
          int index = 0;

          for (Iterator i = blocks.iterator(); i.hasNext();) {
                PageFragment block = (PageFragment) i.next();
                block(writer, side, index++, block);
          }
    }
}
```