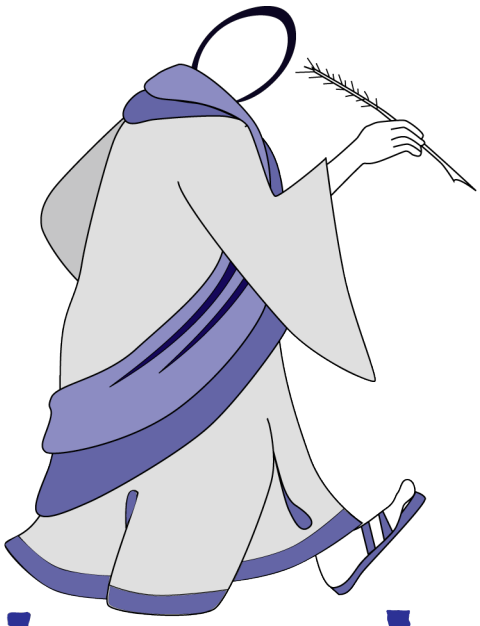

calendar
bedework
www.bedework.org

Bedework Calendar Design Guide

Bedework version 3.3.1

Last modified: April 25, 2007

Bedework Design Guide

This guide is a resource for web designers who would like to customize the look and layout of the Bedework calendar web clients. No programming experience is assumed, though a basic understanding of how web applications work is helpful.

This guide will focus primarily on the public web interface, but the concepts apply to all client development.

Contents

Overview

Skins

MainCampus skin

A generic skin is provided with each web application in the quickstart release. The public client's MainCampus calendar suite comes with three simple color schemes (red, green, and blue) to get you started. Fonts, colors, and most layout specifics are controlled by css.

Within the quickstart directory, the source skin files for the web applications can be found in the following three directories:

```
bedework/deployment/webpublic/webapp/resources/
bedework/deployment/webuser/webapp/resources/
bedework/deployment/webadmin/webapp/resources/
```

Each application has a default skin named *default.xsl* and one or more associated css stylesheets in the same directory. Public calendar suite skin sets are found in the webpublic directory. Here you will also find skins for producing rss feeds, javascript feeds, and output appropriate for use on PDAs and cell phones, as well as television screens.

Examples of how members of the Bedework community have implemented skins can be viewed by selecting from the "production examples" pull-down list in the demo public client or from our listing on the Bedework website, "Who's using Bedework?":

http://www.bedework.org/bedework/update.do?artcenterkey=35

Rensselaer Polytechnic

Queens University


Dalhousie University

The "quickstart" distribution comes with skins in place and ready to use. If you rebuild the application (you will want to do this to create a production release), the skins are copied into:

```
jakarta-tomcat-5.0.28/webapps/ROOT/calrsrc.MainCampus and
[ jakarta-tomcat-5.0.28/webapps/ROOT/calrsrc.OtherCalSuite and ]
jakarta-tomcat-5.0.28/webapps/ROOT/ucalrsrc and
jakarta-tomcat-5.0.28/webapps/ROOT/caladminrsrc
```

Once deployed, the skins can be manipulated directly within the Tomcat webapps directory for quickest development (or copied in from the source folder -- our recommended approach); but, if you want to keep your changes, make certain to keep or copy edited files in the source folder, or your work will be overwritten when the application is rebuilt (by issuing the "ant clean.deploy" or "ant clean.deploy.debug" commands).

The locations of the skins and their deployment directories are defined in the file:

```
bedework/config/configs/{clone}.properties
```

It is strongly recommended that you begin with the "demoskins" build (default) and that you make a backup copy of the source folder before you begin.

Prerequisites

Changing headers, footers, colors, and fonts can be accomplished with an understanding of XHTML and CSS. Changing global layout or presentation behavior requires an understanding of XML and XSL (xslt and xpath, in particular). It is highly recommended that you read through the specifications for these technologies at the World Wide Web Consortium.

Because the calendar front-end uses XSLT to transform XML, you must use valid XML markup for all templates. Skins that produce HTML must be written using XHTML (regardless of the final output). Skins without valid markup will fail to transform.

Structure of the Template Directories

The skin directory for a client deployment of the calendar is called the application root or "appRoot". This is the directory in which the designer will work. The appRoot has the following structure:

appRoot = *http://a-web-server-path/to/your/template/directory/*
which for the default quickstart build is
appRoot = *http://localhost:8080/calrsrc/* and
appRoot = *http://localhost:8080/ucalrsrc/* and
appRoot = *http://localhost:8080/caladminrsrc/*

The appRoot is then further modified at run time for portals and calendar suites. If we running under a portal the approot has "." + portalPlatform appended Continuing with the above example, if the portal platform is "uPortal2" the root becomes
appRoot = *http://localhost:8080/ucalrsrc*.uPortal2

In addition, we append the calendar suite name for public clients.
So if the calendar suite is MainCampus we have
appRoot = *http://localhost:8080/ucalrsrc.MainCampus*
or for the portlet
appRoot = *http://localhost:8080/ucalrsrc.uPortal2.MainCampus*

```
appRoot/
    |-- default/
    |     | -- default/
    |             |
    |             |-- default.xsl
    |              -- default.css
     -- images/
```

The top-level directories of the appRoot define **locales**. The default locale is "*default*". You may add directories here using a locale name such as "en_US" or "fr_CA". Browsers provide Bedework with a locale; if a directory is found with a name that matches the locale provided by the browser, Bedework will use the skins found there. Otherwise, the default locale directory will be used. This level is also a convenient location for template *images*.

The next level down defines the **browser type**. Bedework looks first for a folder whose name is associated with the user-agent of the visiting browser. If found, Bedework will use that folder to deliver the skin. If not found, Bedework will use the default directory seen here. Currently, the acceptable folder names are:

default, Netscape4, MSIE, Opera, Mozilla, and PDA.

Bedework can be forced to use a specifc browser type and skin and can have multiple skins per browser type. See Actions & Parameters for more information about this. (Note: as of version 3.3.1 PDA browser sniffing is very much out-of-date; to use the PDA directory found in the quickstart release, add the request parameter *&browserTypeSticky=PDA. &browserTypeSticky=default* will reset this to the default path.)

For production deployment, we encourage placing the appRoot directories on an external web server where they can be modified without the need for redeploying the application.

Making Basic Stylistic Changes

Each xslt skin is (mostly) self-contained in a file and is made up of a series of xsl *templates*. If you have a good grasp of XHTML and CSS, you can modify the graphics, colors, and general feel of a skin by editing one of the examples (such as default.xsl/default.css). Skins are cached in memory, so if you choose to edit a live skin (e.g. in the Tomcat webapps directory) you must refresh the stylesheet by appending your query string with *refreshXslt=somestring*. For example:

```
  http://localhost:8080/cal/setup.do?refreshXslt=yes or
  http://localhost:8080/cal/event/eventView.do?eventId=2&refreshXslt=
yes
```

NOTE: Be careful editing a live skin; though it is quick and convenient, if you do not copy your changes into the source skin directory, your edits will be overwritten during the next build/deploy. Again, we strongly encourage you to make a backup copy of the source folder before you begin.

Actions & Parameters

What are Actions & Parameters?

Typical of web applications, Bedework receives HTTP requests and returns responses based on the page or action requested and the parameters sent in the query string. Look at the following URL:

```
http://localhost:8080/cal/main/setViewPeriod.do?b=de&viewType=weekVie
w&date=20061121
```

Bedework is built in the Apache Struts MVC framework, and as such does not reference "web pages" through URLs directly. ***setViewPeriod.do*** in the URL above calls a Java "action" that returns a response based on the query string "date=20061121". This URL tells the personal calendar to set the current view to November 21, 2006.

Parameters can be strung together on a query string like so:

```
http://hostname:port/context/action?param1=value&param2=value
```

Normal Actions & Render Actions

Request processing in Bedework is divided into two parts: a normal action that may change the state of the application, and a render action that returns the resulting state for display. This is required, among other things, for Bedework to run as a portlet.

For each normal action that is called, Bedework will automatically redirect to the appropriate render action. Normal actions take a ".do" extension. Render actions have an ".rdo" extension.

As the developer of a skin, you will be primarily concerned with normal actions, and it is these that will be presented in the XSL stylesheets for users to click. All actions and parameters described below are normal actions.

Bedework Actions & Parameters in Detail

All actions used by the Bedework guest and personal web clients are described in

```
bedework/projects/webapps/webclient/war/WEB-INF/struts-config.xml.
```

All actions used by the Bedework admin web client are described in

```
bedework/projects/webapps/webadmin/war/WEB-INF/struts-config.xml.
```

For detailed information about the actions and parameters used in Bedework, please see the API reference found from the Bedework Wiki.

Parameters used primarily to effect skins are described in the next section: Bedework Skin Parameters.

Bedework Skin Parameters

These parameters provide control over stylesheets. They can be added to any Bedework URL and combined with any other parameter.

1. **setappvar**=*key(value)*
    - applicaton variable: used to pass a key/value pair into the XML output
    - This feature is the equivalent of passing a parameter between pages in other frameworks.
    - You can pass as many appvars as you need.
    - appvars, once set, persist through a user session
    - To change the value of an appvar, send the same key with a different value.

2. **skinName**=*name*
3. **skinNameSticky**=*name*
    - These parameters explicitly select an xslt skin. skinName will switch the skin for one request/response; the sticky version will switch the skin for the remainder of the user session or until a different skin is called. name is the file name of an XSLT document without the .xsl extension. For example:

```
appRoot/
    |-- default/
    |     | -- default/
    |             |
    |             |-- default.xsl
    |             |-- default.css
    |             |-- shiny.xsl
    |             |-- shiny.css
    |             |-- lavared.xsl
    |              -- lavared.css
     -- images/
```

- If unspecified, Bedework uses the default skin. The URL to select "shiny.xsl" might look like this: http://hostname/cal/setup.do?skinNameSticky=shiny

4. **browserType**=*default, MSIE, Netscape, Netscape4, Mozilla, PDA, other*
5. **browserTypeSticky**=*default, MSIE, Netscape, Netscape4, Mozilla, PDA, other*
    - These parameters explicitly select a browserType folder. browserType will switch the folder for one request/response; the sticky version will switch the folder for the remainder of the user session or until a different browserType is called. For example:

```
appRoot/
    |-- default/
    |     | -- default/
    |     |       |
    |     |       |-- default.xsl
    |     |        -- default.css
    |     | -- Mozilla/
    |             |
    |             |-- default.xsl
    |              -- default.css
     -- images/
```

- If unspecified, Bedework uses the folder that most closely matches the user-agent of the requesting browser. If not found, the "default" folder will be used. The example above would use the Mozilla folder for an incoming Mozilla

browser. You may also create your own folder and call it explicitly, though Bedework cannot automatically associate it with a user-agent. (Note: the PDA user-agent list is not currently up-to-date; to use the PDA browser path, call it explicitly.)

6. **refreshXslt**=*string*
   - XSL skins are cached once per user session to improve performance. If you make a change to your skin, you need to pass this parameter to reload it. string can be any value; we typically set it to "yes". Example: update.do?catcenterkey=12&skinName=lavared&refreshXslt=yes

7. **noxslt**=*string*
   - This parameter turns off the XSLT filter and returns raw xml in the response. You can look at the xml by selecting "view source" from your browser. This feature is very important when designing skins because it allows you to reference the exact XML you are trying to transform. string can be any value; we typically set it to "yes". Example: update.do?catcenterkey=12&noxslt=yes

XML

Bedework XML Structure

In Bedework, you can effectively look at the XML in two ways:

1. In a user client (guest or personal) append noxslt=yes to the query string and view the page source. For example:

```
http://localhost:8080/cal/setup.do?noxslt=yes
```

2. Look at the JSP pages which produce the XML output. These can be found in:

```
bedework/projects/webapps/webclient/war/docs
and
bedework/projects/webapps/webadmin/war/docs
```

However, each "page" of an XML response takes the following form:

```
<bedework> (or <bedeworkadmin>)

  [header data/variables and urlPrefixes for building links]

  <page>pageName</page>

  [page specific XML]

  [footer data/variables (if any)]

</bedework>
```

In most cases, the XSL stylesheets look first at the *pageName* found in the incoming XML and branch to an appropriate XSL template based on that. Look to the default template (the first in the stylesheets, matching on "/") to see a (mostly) complete listing of all incoming "pages" and their associated XSL templates.

Bedework XSLT

XSLT References

- XSLT & XPath Quick Reference (PDF):
  http://www.mulberrytech.com/quickref/XSLT_1quickref-v2.pdf

- XPath Specification
  http://www.w3c.org/TR/xpath

- XSLT Specification
  http://www.w3c.org/Style/XSL/