Starting a Web Application

The following document illustrates how to get started writing a web application using common code that we have been using or developing over the last few years. We also describe how to deploy the application to a web server.

Web Application Skeleton

To download the skeleton of the web application do a **cvs export** of the module **org-glast-web-base-application** (CVSROOT is *:ext:USERNAME@glast-java.slac.stanford.edu:/cvs/java* where *USERNAME* is you). If you are using Tortoise CVS you have to select "CVS checkout .." and unser "Options" select "Export".

You should now be able to open this maven1 project in Netbeans and deploy it to the bundled Tomcat Server.

GLAST Commons

The GLAST Commons is a collections of common code that can and should be shared across different GLAST web application providing a uniform look and feel. The following items are provided by the GLAST Commons and are used in the skeleton provided above:

- **style sheet** If included in your jsp page, it will give your page the style shared by all other applications. This style sheet is loaded when needed via the web, so any change to the GLAST Commons style sheet will propagate to all the applications using it. By default the style sheet is included in the Sitemesh templates described below, so you should not have to include it.
- **GLAST Logo** This servlet creates the GLAST logo for a provided title. This is also included by default in the Sitemesh templates.
- **Date and Date/Time Picker** These are javascript functions that make it easier to select either Dates and/or Times in your application. Examples are provided in the jsp file: **src/webapp/pickers.jsp**

Filters

The GLAST Commons provide three filters that are by default included in the above skeleton application:

- **Login** A filter that monitor the login status of users with our central single-sign-on CAS server. This filter makes sure that once a user logged in, its logged-in status is preserved across applications.
- **Cookies** Most of our applications require cookies. This filter checks that the user's browser allows cookies to be set and warns the user if they are disabled.
- **Multipart** This filter allows form data, including uploaded files, to be submitted using enctype="multipart/form-data".

To disable any of these filters you'll have to remove their specification in **src/webapp/WEB-INF/web.xml**.

Sitemesh

Sitemesh is used to decorate the pages, making sure they all get a similar look and feel. Please refer to the sitemesh documentation for more information.

The main sitemesh related files are the following:

- **src/webapp/WEB-INF/sitemesh.xml** Sitemes configuration file in which appropriate modules are loaded to handle frames.
- **src/webapp/WEB-INF/decorators.xml** This is the mapping between your jsp pages and the templates to be used to decorate them. In this example we map all the pages to a common decorator, but for three pages belonging to a *frame* which need to have special decorators. The following are the decorators used:
  - **src/webapp/decorators/basicDecorator.jsp** common decorator for all pages. It adds a common style sheed from the GLAST commons, a style sheet specifit to this application (src/webapp/css/style.css), it adds the logo on the top right of the page, the login button and the footer with the last modified date.
  - **src/webapp/decorators/justFooterDecorator.jsp** decorator that just adds the footer, used for the main window of frames. It adds the last modified date at the bottom of the page.
  - **src/webapp/decorators/justHeaderDecorator.jsp** decorator that just decorates the header with the logo and the login button. This is used for the header of a frame.
  - **src/webapp/decorators/justStyleDecorator.jsp** decorator that just adds style sheets. Used for the menu/tree window of a frame.

The above decorators share common code stored in the form of tag files:

- **src/webapp/WEB-INF/tags/decorators/footerDecorator.tag** Prints the last modified date in the footer.
- **src/webapp/WEB-INF/tags/decorators/headerDecorator.tag** Adds the logo and the login button to the header.

Finally for sitemesh to work a filter needs to be added to the application. It's the filter that is responsible for mangling your jsp code with the decorating templates. This is done in **src/webapp/WEB-INF/web.xml** with the following code:

```
<filter>
    <filter-name>sitemesh</filter-name>
    <filter-class>com.opensymphony.module.sitemesh.filter.PageFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>sitemesh</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

If you are not interested in sitemesh you can remove the filter definition in web.xml and all the above files.

Display Tag For Tables

We use the Display tag library to create sortable tables. We provide a simple jsp example (**src/webapp/table.jsp**) that displays a sample list of items created using the java classes in **src/main/java/org/glast/web/base/table**.

To use the Display tag library all you have to do is to include the tag library definition at the top of your jsp page:

```
<%@taglib prefix="display" uri="http://displaytag.sf.net" %>
```

and use the prefix **display** to access the table tags.

Additionally, if you specify the table's class to be **datatable**, as done in the example below, you will get the default styles provided by the GLAST Commons and used by most of the other GLAST web applications.

```
<display:table  class="datatable" name="${table}"  defaultorder="ascending"  sort="list" id="tableId" >
```

Relevant to Display Tag is the properties file **src/webapp/WEB-INF/classes/displaytag.properties** which allows some configuration; please refer to the Display tag configuration documentation for a list of all the available properties.

AIDATLD For Plots

We create our plots using the AIDATLD jsp tags. The jsp file **src/webapp/aidaPlot.jsp** shows a sample plot extracted from the AIDATLD examples.

FreeHEP WebUtil

The FreeHEP WebUtil library provides some additional common code to be used in web application:

- **tabs** An example is shown in **src/webapp/tabs.jsp**. These tags have the advantage that the body of the tag is executed only the the tab is selected.
- **Non Available Filter** Allows to disable the web application and redirect incoming request to a different page (in this case to **src/webapp/nonAvailable.jsp**). To try it out go to *http://your deployment host/WebBaseApplication/admin/available/*
- **tree** Allows the creation of navigation trees as shown in the jsp file **src/webapp/tree.jsp**.

Related to the FreeHEP WebUtil code is the properties file **src/webapp/WEB-INF/classes/freehepWebapp.properties** that allows to configure the above items. Please refer to the FreeHEP WebUtil documentation for a list of the available parameters.

Logging

By default logging is turned on, using *java.util.logging*. The logging parameters are controlled via the properties file **src/webapp/WEB-INF/classes/logging.properties**. For additional information on logging in Tomcat, please refer to the following documentation.