GUI Framework

GUIFramework (GFW) is the basic module for developing Swing GUIs at SLAC.

Current Version

0-0-3
Changes: added access to split panes.

Requirements **TO-DO**

Developer's Guide

Abstract

This guide describes in detail how to use GFW in a Swing application. In general, you create your own Swing components (either programmatically, or with a GUI builder) and add them to the appropriate GFW frame.

Overview

We recommend the following steps:

1. #Check out GFW from CVS into Eclipse.
2. #Read and run the example code.
3. #Create custom components for the desired GFW frame
4. #Customize GFW.
    - #Add event listeners to the appropriate widgets.
    - #Set web help URL.
5. #Test the application.
6. Optional features.
    - #Info Text Pane
    - #Progress Bar
    - #Split Panes
        - #Make split-pane divider disappear
7. #Tips.

Check out GFW from CVS into Eclipse

GFW is located in $CVSROOT/physics/GUIFramework

Read and run the example code

An example can be found in the class

```
edu.stanford.slac.gfw.example.GfwExample
```

When running the example, please observe

- the different GFW frames
- the various areas in which the custom components end up
- how the GUI behaves when it is resized

Create custom components for the desired GFW frame

For a quick development (e.g. GUI mock ups), we recommend the Netbeans GUI Builder.
Here's a nice overview of all Swing components: http://java.sun.com/docs/books/tutorial/ui/features/components.html

Customize GFW

1. Add the GFW to your Eclipse project buildpath. You can do this either by including the GUIFramework project from your workspace, or by including the GUIFramework.jar from its production location (/usr/local/lcls/physics/GUIFramework/jar/GUIFramework.jar).
2. Instantiate the custom components. For instance, if you have not yet created the main class, now you would do so. Note, that this class is likely to be in a different package to the User Interface package if you are separating UI components from application code.
3. Create a BasicFrame or a ModelFrame instance of the GFW, e.g

```
final BasicFrame myFrame = new BasicFrame("My Frame"); //'final', because it's used later in an
anonymous class
```

or

```
final ModelFrame myFrame = new ModelFrame("My Model Frame");
```

4. Add custom components to various areas of the GFW frame, such as
   - Title Bar
     ```
     myFrame.addToTitleBar(myComponent);
     ```
   - Tab 1
     ```
     myFrame.setTopTab1(myComponent);
     ```
   - Tab 2
     ```
     myFrame.setTopTab2(myComponent);
     ```
   - New Tab
     ```
     myFrame.addTopTab(myComponent, "My Label");
     ```
   - Options Panel
     ```
     myFrame.addToOptions(myComponent);
     ```
   - Bottom ScrollPane
     ```
     myFrame.setBottomScrollPane(myComponent);
     ```
5. Set application version
   ```
   BasicFrame myFrame = ...; // see above
   JLabel appVersionLabel = myFrame.getBasicPanel().getStatusPanel().getAppVersionLabel();
   appVersionLabel.setText("my version"); //run in the GUI thread, of before displaying the frame
   ```
6. #Add event listeners to the appropriate widgets
7. Display the GFW frame
   ```
   SwingUtilities.invokeLater(new Runnable() {
           public void run() {
                   myFrame.setVisible(true);
           }
   });
   ```

Add event listeners to the appropriate widgets

Below we describe the recommended way, but nothing prevents you from using any other technique.

- Extend *BasicFrameListener* by overriding the default behavior and/or adding your own methods, e.g.

```
public class MyFrameListener extends BasicFrameListener {

	@Override
	public void topTabChanged(ChangeEvent e) {
		System.out.println("Tab changed");
	}

	public void myComponentActivated(EventObject e){
		System.out.println("My component activated");
	}
}
```

- Override *registerAsDelegateForSwingListeners* method of the *BasicFrameController* class (find out more about Swing listeners), e.g.

```
public class MyFrameController extends BasicFrameController {

	protected void registerAsDelegateForSwingListeners(
			final MyFrameListener myFrameListener) {
		super.registerAsDelegateForSwingListeners(myFrameListener); //don't forget this!
		Component myComponent = ...; //get your component, e.g. by using SwingUtil, or through
methods of your GUI class

		myComponent.addComponentListener(new ComponentListener(){
			public void componentActivated(EventObject e){
				myFrameListener.myComponentActivated(e);
			}
		});
	}

	public MyFrameController(BasicFrame myFrame) {
		super(myFrame);
	}

}
```

- Add your BasicFrameListener to your BasicFrameController

```
BasicFrameListener myFrameListener = new MyFrameListener();
BasicFrameController myFrameController = new MyFrameController(myFrame);
myFrameController.addBasicFrameListener(myFrameListener);
```

Set web help URL

If you want to provide help via the default GFW way, set the URL of your help website.

```
BasicFrameListener myFrameListener = ...;
myFrameListener.setHelpUrl("http://www.google.com");
```

Test the application

Optional features

Info Text Pane

Located in the status panel, the info text pane can be used to display some info beyond messaging.

```
JTextPane infoTextPane = bf.getBasicPanel().getStatusPanel().getInfoTextPane();
infoTextPane.setText("my info"); //run in the GUI thread, of before displaying the frame
```

Progress Bar

Located in the status panel, the progress bar could be used to display the progress of your program.

- Determinate

```
JProgressBar progressBar = bf.getBasicPanel().getStatusPanel().getProgressBar();
progressBar.setMinimum(0);
progressBar.setMaximum(10);
progressBar.setValue(6);
```

- Indeterminate

```
JProgressBar progressBar = bf.getBasicPanel().getStatusPanel().getProgressBar();
progressBar.setIndeterminate(true);
```

  To stop:

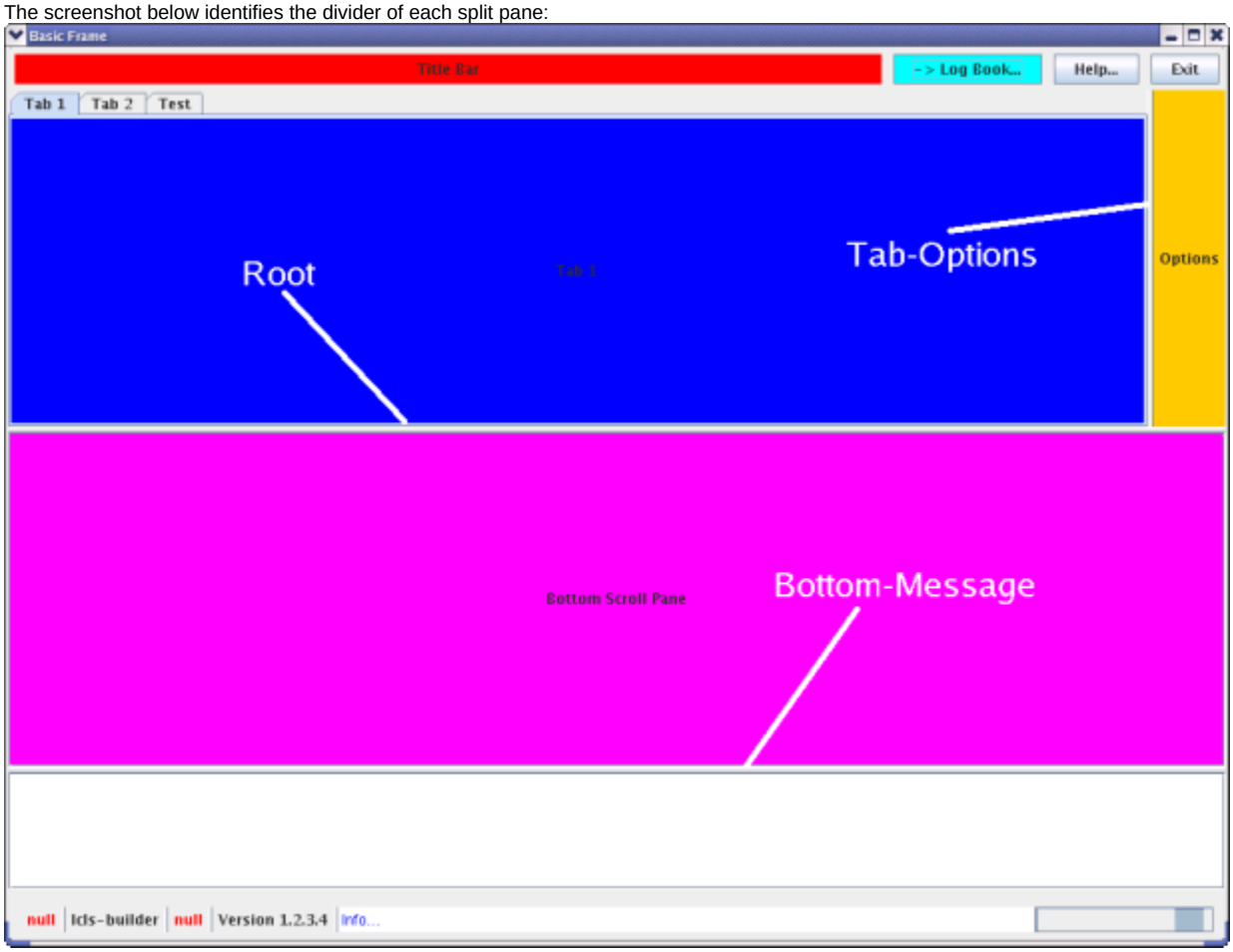
```
progressBar.setValue(progressBar.getMinimum());
progressBar.setIndeterminate(false);
```

Split Panes

The BasicPanel class has methods to access SplitPanes:

```
myBasicFrame.getBasicPanel().getRootSplitPane();
myBasicFrame.getBasicPanel().getBottomMessageSplitPane();
myBasicFrame.getBasicPanel().getTabOptionsSplitPane();
```

The screenshot below identifies the divider of each split pane:



The following methods of the JSplitPane class are particularly useful:

```
get/setDividerLocation();
get/setDividerSize();
get/setLastDividerLocation();
get/setResizeWeight();
```

See also http://java.sun.com/docs/books/tutorial/uiswing/components/splitpane.html.

Make split-pane divider disappear

It seems to be an undocumented feature, but here is how you can make the split-pane divider disappear:

```
splitPane.setDividerSize(-1);
```

SwingUtil

Tips

- When adding to the GUI some long text that doesn't fit, update the frame, e.g.

```
BasicFrame myFrame = ...;
myFrame.validate(); //run in the GUI thread
```