Zaval GUI Designer Package
Version 1.0

Tutorial

Zaval Creative Engineering Group
http://www.zaval.org

Contents

Introduction to the Zaval GUI Designer Package

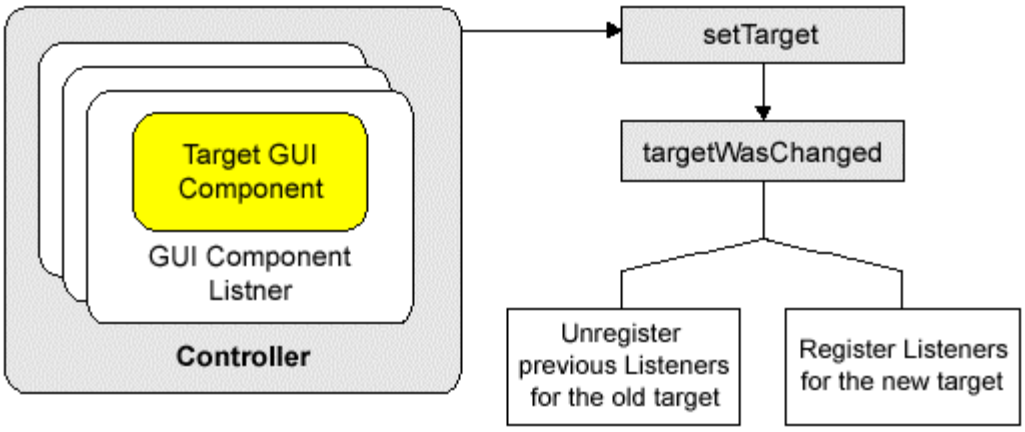
Zaval GUI Designer Package is a reference implementation of visual component builder components. These components allow to control size, location, cursor type for the Java AWT/Swing controls, including various types of standard AWT, Swing controls and user defined non-standard controls. These components can be used to provide wide range of drag-n-drop operations, property editors of external builders and so on.

When to use Zaval GUI Designer Package

Zaval GUI Designer Package can be used for various IDE development (WYSIWYG forms builder) or/and in any GUI development based on AWT and Swing libraries. This is a reference implementation of visual component builder.

Controller-Component

Controller is a class that manages a Java GUI Component (that is base on Java AWT library). The library provides special abstract class – *org.zaval.awt.gdp.Controller* that is base class for different controllers. This is very simple class. It binds the specified GUI component (it is called "target") with the controller class implementation. The image bellow illustrates the controller meaning:



Any controller has a GUI Component as a target. To control the target behavior as rule necessary to register one or more listeners. The controller class has *setTarget* method that is used to set the specified target component. The method performs *targetWasChanged* method that has an old target reference and the new target reference as the input arguments. The method is abstract and it is supposed to be used to register appropriate listeners for the new target and unregister listeners for the old target component.

Controller implementations

This chapter describes shortly set of controllers that is available within the package.

Controller	Description
Drag controller	This is the simplest controller implementation. Its main purpose is to convert appropriate mouse events to the special drag events, the events usage is more useful and handy.
Cursor controller	This implementation controls cursor type for the target component depending on the mouse pointer location (relatively the target

	component).
Shape controller	This controller allows to control size and location for the specified component.
Designer controller	This controller provides more universal way to control size and location for the specified component.

Drag Controller

This controller is represented with *org.zaval.awt.gdp.DragController* class. The main purpose of the class is to transform mouse events that are performed by the target component into *org.zaval.awt.gdp.event.MouseDragEvent* events and to provide listeners support for this event type. The new *org.zaval.awt.gdp.event.MouseDragEvent* type can be more useful in a case if it is necessary to organize drag and drop, for example. Using the controller an application can be notified when the drag process has been started, continued or stopped. Using *getDestination* method of the drag event it is possible to get destination component. The sample bellow illustrates this controller usage:

```
…
Button button = new Button(“Ok”);
DragController controller = new DragController(button);
controller.addMouseDragListener (new MouseDragListener()
{
    public void dragStarted(MouseDragEvent e) {}
    public void dragMoved (MouseDragEvent e) {}

    public void dragStopped(MouseDragEvent e) {
        System.out.println (“The destination is:”+e.getDestination().getDestComponent());
    }
} );
…
```

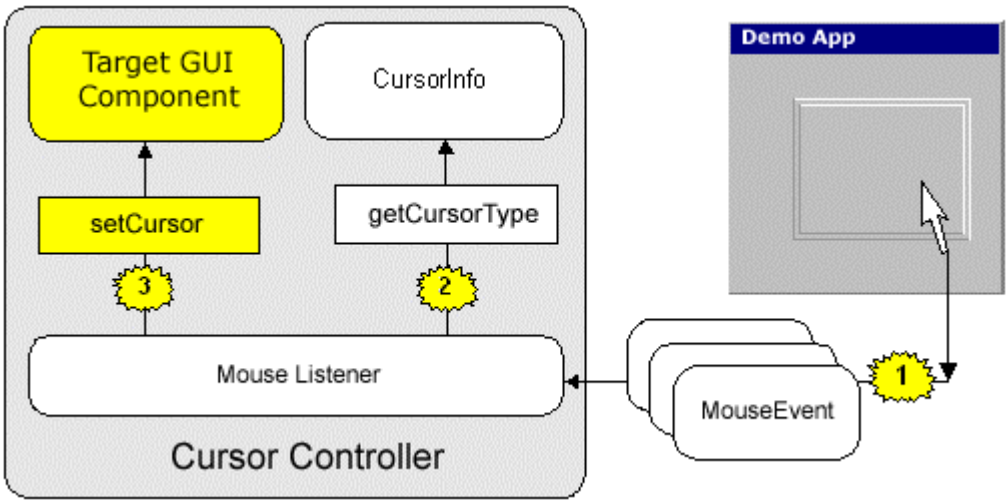
The following listeners are registered within the controller for the target component

- java.awt.event.MouseListener
- java.awt.event.MouseMotionListener

Note: The controller can work incorrectly for some types of the Java GUI components. The problem is in specific mouse events performing (in other words, the Java performs mouse events wrongly for some components). It is impossible to fix the problem within the controller only. One of such components is *java.awt.Label* component. In this case performing of the mouse dragged event type is stopped whenever the mouse exit event type has been performed.

Cursor Controller

This controller is represented with *org.zaval.awt.gdp.CursorController* class. Its main purpose is to control mouse cursor type for the specified target component. The cursor type is defined by a special *org.zaval.awt.gdp.CursorInfo* interface depending on the location relatively the specified component. The controller uses an implementation of the interface, use *setCursorInfo* method to specify the new cursor info instance. The image bellow illustrates the controller functionality:



Controller gets mouse events by registered mouse listener from the target component (on the image above the target component is the border panel). Than, *getCursorType* method is called with the listener to define a cursor type for the specified target component and the current mouse pointer location. At last, the returned value with the cursor info class is used to set the cursor for the target component. It is possible to listen when the target component's cursor has been changed: for this purpose use *addActionListener* and *removeActionListener* methods of the controller.

The sample bellow illustrates this controller usage:

```
…
Button button = new Button("Ok");
CursorController controller = new CursorController(button);
controller.addActionListener (new ActionListener()
{
   public void actionPerformed(ActionEvent e) {
     System.out.println ("The cursor type has been changed");
}
} );
…
```

The following listeners are registered with the controller for the target component

- java.awt.event.MouseListener
- java.awt.event.MouseMotionListener

Shape Controller

This controller is represented by *org.zaval.awt.gdp.ShapeController* class. Its main purpose is to control size and location of the specified target component. It is possible to resize and relocate the target component by using mouse. This controller uses two other controllers:

- **Cursor controller.** It is used to bind an appropriate cursor type with the specified area of the target component. The cursor type that is returned with the controller is necessary for the shape controller to define what part of the target component should be resized or if the target has to be moved. The following cursor types (see *java.awt.Cursor* class) exist:

Cursor Type	Description
NE_RESIZE_CURSOR	Resize north-east part of the target component.
NW_RESIZE_CURSOR	Resize north-west part of the target component.
S_RESIZE_CURSOR	Resize south part of the target component.
SE_RESIZE_CURSOR	Resize south-east part of the target component.

SW_RESIZE_CURSOR	Resize south-west part of the target component.
N_RESIZE_CURSOR	Resize north part of the target component.
W_RESIZE_CURSOR	Resize west part of the target component.
E_RESIZE_CURSOR	Resize east part of the target component.
MOVE_CURSOR	Move the target component.

- **Drag controller.** It is necessary for the shape controller to listen drag events for the target component. The events signal the controller to start resizing or moving the target component.

Pay attention how the controller moves target component. There is a feature – the target component can be moved to the new parent component if the container component contains the top–left corner of the target. But, the package provides special empty interface – *DgnComponent*. Implement the interface for a container that cannot be a new parent for the target. For example, you want to create something like a simple designer application. On the one hand the application should contain special area where the new forms are designed and on the other hand the application can contain toolbar, inspector and so on. The components cannot be a parent for the designed components, so it is necessary to implement *DgnComponent* interface for its or its parent components.

The sample bellow illustrates this controller usage (it's very simple):

```
…
Button button = new Button("Ok");
ShapeController controller = new ShapeController(button);
…
```

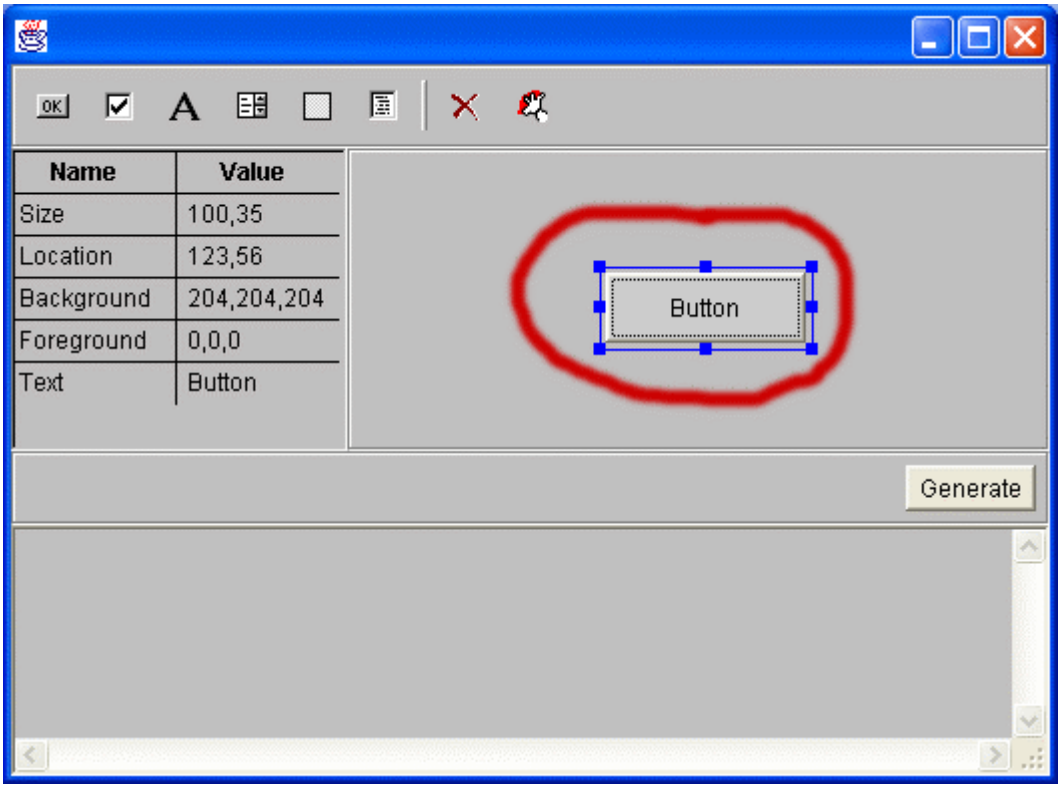
The following listeners are registered with the controller for the target component

- java.awt.event.MouseListener
- java.awt.event.MouseMotionListener

Note: The controller can work incorrectly for some types of the Java GUI components. It is component specific. Use Designer controller in this case (see the next chapter).

Designer Controller

This controller is represented with *org.zaval.awt.gdp.DgnController* class. It is a variation of the shape controller that has been described in previous chapter. The controller provides more complex and universal way to control size and location of the specified target component. The image below is a fragment of an application that uses the controller:



On the image above we set a shape controller for the button component (marked with red). The shape controller is active at the image. In this case special designer container is used for the target component. The container paints special border (this is blue border around the button) that allows to resize and move the target component. The designer controller uses shape controller for the designer container.

The shape controller can have non-active state. In this case the designer container is not used and the target component is shown as is. To activate the designer controller for the given target component you have to click on it.

It is possible to listen when the designer controller has been activated or deactivated by using *addDgnControllerListener* and *removeDgnControllerListener* methods. The controller performs *DgnControllerEvent* events when it is activated or deactivated.
The sample bellow illustrates this controller usage:

```
…
Button button = new Button(“Ok”);
DgnController controller = new DgnController(button);
…
//*************************************************************************
// if it is necessary to listen whenever the controller has been activated or deactivated
// use the code bellow
//*************************************************************************

controller. addDgnControllerListener (new DgnControllerListener ()
{
   public void dgnStarted(DgnControllerEvent e){
      …
   }

   public void dgnFinished (DgnControllerEvent e){
     …
   }
} );
…
```

The following listeners are registered with the controller for the target component

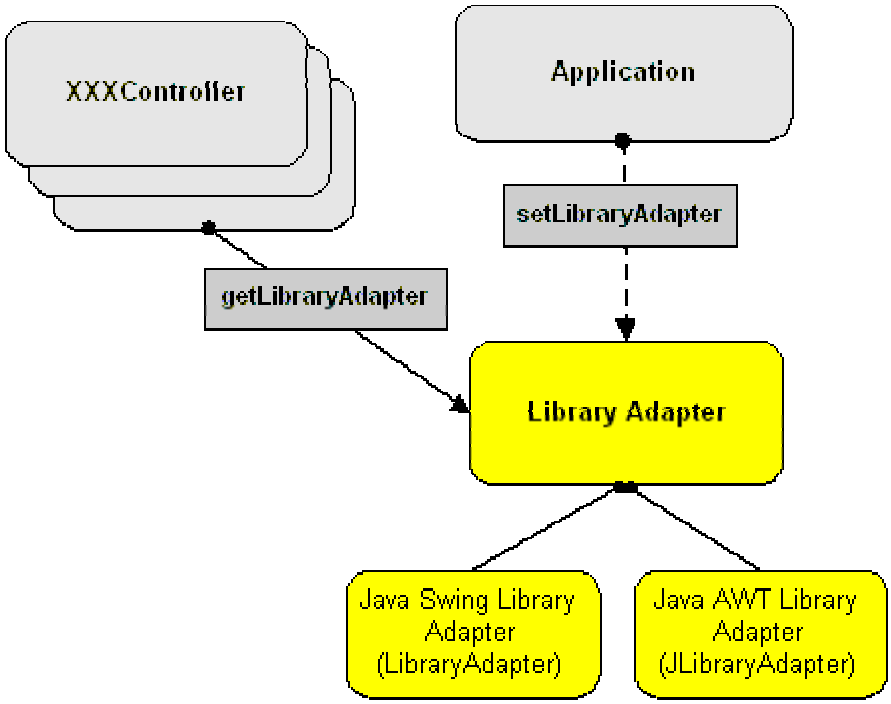
- java.awt.event.MouseListener

Swing/AWT Support

The package can be used with Java AWT library or Java Swing library. The controllers use special class *LibraryAdapter.* The class contains set of methods that is library specific. Use *getLibraryAdapter* static method of *LibraryAdapter* class to get current library adapter. The package has two implementations of the adapters:

- **LibraryAdapter.** This adapter is used to work with the Java AWT library components and the adapter is set as default.
- **JLibraryAdapter.** This adapter is used to work with the Java Swing library components.

Use *setLibraryAdapter* static method of *LibraryAdapter* class to set appropriate adapter to be used with the controllers classes. The image below explains the adapter meaning:



The sample bellow shows what you should do in a case if the Java Swing Library is used:

```
…
// ****************************************************************************
//
//  Set appropriate library adapter before any controllers will be used
//
// ****************************************************************************

LibraryAdapter.setLibraryAdapter(new JLibraryAdapter());
…
```

Further product plans

In the future the following features are going to be added:

- The inspector component (allows to control properties for the target object).
- Code generator (allows java code generation basing on the GUI form content).
- Zaval LwVCL support.

Support available

All support for library usage and problems should be sent directly to support@zaval.org with “Re: Zaval GUI Designer Package Support” in subject line and plain text in the message body, describing your request and/or your problem. Since this software is distributed under the General Public License and is maintained by its authors on non-commercial basis, your request will be answered as soon as possible, but no later than 5 business days.

The Zaval Creative Engineering Group carries out its software customization/new software development on the regular basis. For more info contact us at info@zaval.org.

Stay informed!

Now you can receive information on latest products’ updates and hotfixes via email. This is a low-traffic list (1-2 messages per month). To subscribe, send blank mail to news-subscribe@zaval.org.