GWT - POPUPPANEL WIDGET

Introduction

The **PopupPanel** widget represents a panel that can **pop up** over other widgets. It overlays the browser's client area *andanypreviously − createdpopups*.

Class declaration

Following is the declaration for **com.google.gwt.user.client.ui.PopupPanel** class:

```
public class PopupPanel
   extends SimplePanel
      implements SourcesPopupEvents, EventPreview,
         HasAnimation, HasCloseHandlers<PopupPanel>
```

Class constructors

S.N.	Constructor & Description
1	**PopupPanel** Creates an empty popup panel.
2	**PopupPanel***booleanautoHide* Creates an empty popup panel, specifying its **auto-hide** property.
3	**PopupPanel***booleanautoHide, booleanmodal* Creates an empty popup panel, specifying its **auto-hide** and **modal** properties.

Class methods

S.N.	Function name & Description
1	**void addAutoHidePartner***Elementpartner* Mouse events that occur within an autoHide partner will not hide a panel set to autoHide.
2	**HandlerRegistration addCloseHandler***CloseHandler < PopupPanel > handler* Adds a CloseEvent handler.
3	**void addPopupListener***PopupListenerlistener* Deprecated. Use addCloseHandler*com. google. gwt. event. logical. shared. CloseHandler* instead

4

void center

Centers the popup in the browser window and shows it.

5

protected Element getContainerElement

Override this method to specify that an element other than the root element be the container for the panel's child widget.

6

protected Element getGlassElement

Get the glass element used by this PopupPanel.

7

java.lang.String getGlassStyleName

Gets the style name to be used on the glass element.

8

int getOffsetHeight

Gets the panel's offset height in pixels.

9

int getOffsetWidth

Gets the panel's offset width in pixels.

10

int getPopupLeft

Gets the popup's left position relative to the browser's client area.

11

int getPopupTop

Gets the popup's top position relative to the browser's client area.

12

protected Element getStyleElement

Template method that returns the element to which style names will be applied.

13

java.lang.String getTitle

Gets the title associated with this object.

14

void hide

Hides the popup and detaches it from the page.

15

void hide*booleanautoClosed*

Hides the popup and detaches it from the page.

16

boolean isAnimationEnabled

Returns true if animations are enabled, false if not.

17

boolean isAutoHideEnabled

Returns true if the popup should be automatically hidden when the user clicks outside of it.

18

boolean isAutoHideOnHistoryEventsEnabled

Returns true if the popup should be automatically hidden when the history token changes, such as when the user presses the browser's back button.

19

boolean isGlassEnabled

Returns true if a glass element will be displayed under the PopupPanel.

20

boolean isModal

Returns true if keyboard or mouse events that do not target the PopupPanel or its children should be ignored.

21

boolean isPreviewingAllNativeEvents

Returns true if the popup should preview all native events, even if the event has already been consumed by another popup.

22

boolean isShowing

Determines whether or not this popup is showing.

23

boolean isVisible

Determines whether or not this popup is visible.

24

boolean onEventPreview*Eventevent*

Deprecated. Use onPreviewNativeEvent*com. google. gwt. user. client. Event. NativePreviewEvent* instead

25

boolean onKeyDownPreview*charkey, intmodifiers*

Deprecated. Use onPreviewNativeEvent*com. google. gwt. user. client. Event. NativePreviewEvent* instead

26

boolean onKeyPressPreview*charkey, intmodifiers*

Deprecated. Use onPreviewNativeEvent*com. google. gwt. user. client. Event. NativePreviewEvent* instead

27

boolean onKeyUpPreview*charkey, intmodifiers*

Deprecated. Use onPreviewNativeEvent*com. google. gwt. user. client. Event. NativePreviewEvent* instead

28

protected void onPreviewNativeEvent*Event. NativePreviewEventevent*

29

protected void onUnload

This method is called immediately before a widget will be detached from the browser's document.

30

void removeAutoHidePartner*Elementpartner*

Remove an autoHide partner.

31

void removePopupListener*PopupListenerlistener*

Deprecated. Use the HandlerRegistration.removeHandler method on the object returned by addCloseHandler*com. google. gwt. event. logical. shared. CloseHandler* instead

32

void setAnimationEnabled*booleanenable*

Enable or disable animations.

33

void setAutoHideEnabled*booleanautoHide*

Enable or disable the autoHide feature.

34

void setAutoHideOnHistoryEventsEnabled*booleanenabled*

Enable or disable autoHide on history change events.

35

void setGlassEnabled*booleanenabled*

When enabled, the background will be blocked with a semi-transparent pane the next time it is shown.

36

void setGlassStyleName*java. lang. StringglassStyleName*

Sets the style name to be used on the glass element.

37

void setHeight*java. lang. Stringheight*

Sets the height of the panel's child widget.

38

void setModal*booleanmodal*

When the popup is modal, keyboard or mouse events that do not target the PopupPanel or its children will be ignored.

39

void setPopupPosition*intleft, inttop*

Sets the popup's position relative to the browser's client area.

40

void setPopupPositionAndShow*PopupPanel. PositionCallbackcallback*

Sets the popup's position using a PopupPanel.PositionCallback, and shows the popup.

41

void setPreviewingAllNativeEvents*booleanpreviewAllNativeEvents*

When enabled, the popup will preview all native events, even if another popup was opened after this one.

42

void setTitle*java. lang. Stringtitle*

Sets the title associated with this object.

43

void setVisible*booleanvisible*

Sets whether this object is visible.

44

void setWidget*Widgetw*

Sets this panel's widget.

45

void setWidth*java. lang. Stringwidth*

Sets the width of the panel's child widget.

46

void show

Shows the popup and attach it to the page.

47

void showRelativeTo*UIObjecttarget*

Normally, the popup is positioned directly below the relative target, with its left edge aligned with the left edge of the target.

Methods inherited

This class inherits methods from the following classes:

- com.google.gwt.user.client.ui.UIObject

- com.google.gwt.user.client.ui.Widget
- com.google.gwt.user.client.ui.Panel
- com.google.gwt.user.client.ui.SimplePanel
- java.lang.Object

PopupPanel Widget Example

This example will take you through simple steps to show usage of a PopupPanel Widget in GWT. Follow the following steps to update the GWT application we created in *GWT - Create Application* chapter:

Step	Description
1	Create a project with a name *HelloWorld* under a package *com.tutorialspoint* as explained in the *GWT - Create Application* chapter.
2	Modify *HelloWorld.gwt.xml*, *HelloWorld.css*, *HelloWorld.html* and *HelloWorld.java* as explained below. Keep rest of the files unchanged.
3	Compile and run the application to verify the result of the implemented logic.

Following is the content of the modified module descriptor **src/com.tutorialspoint/HelloWorld.gwt.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<module rename-to='helloworld'>
  <!-- Inherit the core Web Toolkit stuff.                        -->
  <inherits name='com.google.gwt.user.User'/>

  <!-- Inherit the default GWT style sheet.                       -->
  <inherits name='com.google.gwt.user.theme.clean.Clean'/>

  <!-- Specify the app entry point class.                         -->
  <entry-point class='com.tutorialspoint.client.HelloWorld'/>

  <!-- Specify the paths for translatable code                    -->
  <source path='client'/>
  <source path='shared'/>

</module>
```

Following is the content of the modified Style Sheet file **war/HelloWorld.css**.

```
body{
   text-align: center;
   font-family: verdana, sans-serif;
}
h1{
   font-size: 2em;
   font-weight: bold;
   color: #777777;
   margin: 40px 0px 70px;
   text-align: center;
}
.gwt-PopupPanel {
   border: 3px solid #000000;
   padding: 3px;
   background: white;
}

.gwt-PopupPanelGlass {
```

```
   background-color: #000;
   opacity: 0.3;
   filter: alpha(opacity=30);
}

.gwt-PopupPanel .popupContent {
   border: none;
   padding: 3px;
   background: gray;
}
```

Following is the content of the modified HTML host file **war/HelloWorld.html**.

```
<html>
<head>
<title>Hello World</title>
   <link rel="stylesheet" href="HelloWorld.css"/>
   <script language="javascript" src="helloworld/helloworld.nocache.js">
   </script>
</head>
<body>

<h1>PopupPanel Widget Demonstration</h1>
<div ></div>

</body>
</html>
```

Let us have following content of Java file **src/com.tutorialspoint/HelloWorld.java** which will demonstrate use of PopupPanel widget.

```
package com.tutorialspoint.client;

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.event.dom.client.ClickEvent;
import com.google.gwt.event.dom.client.ClickHandler;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.DecoratorPanel;
import com.google.gwt.user.client.ui.HasHorizontalAlignment;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.PopupPanel;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.VerticalPanel;

public class HelloWorld implements EntryPoint {

   private static class MyPopup extends PopupPanel {

      public MyPopup() {
         // PopupPanel's constructor takes 'auto-hide' as its boolean
         // parameter. If this is set, the panel closes itself
         // automatically when the user clicks outside of it.
         super(true);

         // PopupPanel is a SimplePanel, so you have to set it's widget
         // property to whatever you want its contents to be.
         setWidget(new Label("Click outside of this popup to close it"));
      }
   }

   public void onModuleLoad() {
      Button b1 = new Button("Click me to show popup");
      b1.addClickHandler(new ClickHandler() {
         public void onClick(ClickEvent event) {
            // Instantiate the popup and show it.
             new MyPopup().show();
         }
```

```
      });

      Button b2 = new Button("Click me to show popup partway"
      +" across the screen");
      b2.addClickHandler(new ClickHandler() {
         public void onClick(ClickEvent event) {
            // Create the new popup.
            final MyPopup popup = new MyPopup();
            // Position the popup 1/3rd of the way down and across
            // the screen, and show the popup. Since the position
            // calculation is based on the offsetWidth and offsetHeight
            // of the popup, you have to use the
            // setPopupPositionAndShow(callback)  method. The alternative
            // would be to call show(), calculate  the left and
            // top positions, and call setPopupPosition(left, top).
            // This would have the ugly side effect of the popup jumping
            // from its original position to its new position.
            popup.setPopupPositionAndShow(new PopupPanel.PositionCallback(){
               public void setPosition(int offsetWidth, int offsetHeight) {
                  int left = (Window.getClientWidth() - offsetWidth) / 3;
                  int top = (Window.getClientHeight() - offsetHeight) / 3;
                  popup.setPopupPosition(left, top);
               }
            });
         }
      });
      VerticalPanel panel = new VerticalPanel();
      panel.setHorizontalAlignment(HasHorizontalAlignment.ALIGN_CENTER);
      panel.setSpacing(10);
      panel.add(b1);
      panel.add(b2);

      DecoratorPanel decoratorPanel = new DecoratorPanel();
      decoratorPanel.add(panel);
      // Add the widgets to the root panel.
      RootPanel.get().add(decoratorPanel);
   }
}
```

Once you are ready with all the changes done, let us compile and run the application in development mode as we did in GWT - Create Application chapter. If everything is fine with your application, this will produce following result: