GWT - FORMPANEL WIDGET

Introduction

The **FormPanel** widget represents a panel that wraps its contents in an HTML <FORM> element.

Class declaration

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

```
public class FormPanel
   extends SimplePanel
      implements FiresFormEvents,
         com.google.gwt.user.client.ui.impl.FormPanelImplHost
```

Class constructors

S.N.	Constructor & Description
1	**FormPanel** Creates a new FormPanel.
2	**protected FormPanel***Elementelement* This constructor may be used by subclasses to explicitly use an existing element.
3	**protected FormPanel***Elementelement, booleancreateIFrame* This constructor may be used by subclasses to explicitly use an existing element.
4	**FormPanel***NamedFrameframeTarget* Creates a FormPanel that targets a NamedFrame.
5	**FormPanel***java. lang. Stringtarget* Creates a new FormPanel.

Class methods

S.N.	Function name & Description
1	**void add Form Handler** *FormHandlerhandler* Deprecated. Use add Submit Complete Handler *com. google. gwt. user. client. ui. FormPanel. SubmitCompleteHandler* and add Submit Handler *com. google. gwt. user. client. ui. FormPanel. SubmitHandler* instead

2

Handler Registration addSubmit Complete Handler
FormPanel. SubmitCompleteHandlerhandler

Adds a FormPanel.Submit Complete Event handler.

3

HandlerRegistration addSubmitHandler*FormPanel. SubmitHandlerhandler*

Adds a FormPanel.SubmitEvent handler.

4

java.lang.String getAction

Gets the 'action' associated with this form.

5

java.lang.String getEncoding

Gets the encoding used for submitting this form.

6

java.lang.String getMethod

Gets the HTTP method used for submitting this form.

7

java.lang.String getTarget

Gets the form's 'target'.

8

protected void onAttach

This method is called when a widget is attached to the browser's document.

9

protected void onDetach

This method is called when a widget is detached from the browser's document.

10

boolean onFormSubmit

Fired when a form is submitted.

11

void onFrameLoad

12

void removeFormHandler*FormHandlerhandler*

Deprecated. Use the HandlerRegistration.removeHandler method on the object returned by and add*Handler method instead

13

void reset

Resets the form, clearing all fields.

14

void setAction*java. lang. Stringurl*

Sets the 'action' associated with this form.

15

void setEncoding*java. lang. StringencodingType*

Sets the encoding used for submitting this form.

16

void setMethod*java. lang. Stringmethod*

Sets the HTTP method used for submitting this form.

17

void submit

Submits the form.

18

static FormPanel wrap*Elementelement*

Creates a FormPanel that wraps an existing <form> element.

19

static FormPanel wrap*Elementelement, booleancreateIFrame*

Creates a FormPanel that wraps an existing <form> element.

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

FormPanel Widget Example

This example will take you through simple steps to show usage of a FormPanel 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;
}
```

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>FormPanel 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 FormPanel 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.FileUpload;
import com.google.gwt.user.client.ui.FormPanel;
import com.google.gwt.user.client.ui.FormPanel.SubmitCompleteEvent;
```

```
import com.google.gwt.user.client.ui.FormPanel.SubmitEvent;
import com.google.gwt.user.client.ui.ListBox;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.TextBox;
import com.google.gwt.user.client.ui.VerticalPanel;

public class HelloWorld implements EntryPoint {

   public void onModuleLoad() {
      // Create a FormPanel and point it at a service.
      final FormPanel form = new FormPanel();
      form.setAction("/myFormHandler");

      // Because we're going to add a FileUpload widget,
      // we'll need to set the form to use the POST method,
      // and multipart MIME encoding.
      form.setEncoding(FormPanel.ENCODING_MULTIPART);
      form.setMethod(FormPanel.METHOD_POST);

      // Create a panel to hold all of the form widgets.
      VerticalPanel panel = new VerticalPanel();
      panel.setSpacing(10);
      form.setWidget(panel);

      // Create a TextBox, giving it a name so that it will be submitted.
      final TextBox tb = new TextBox();
      tb.setWidth("220");

      tb.setName("textBoxFormElement");
      panel.add(tb);

      // Create a ListBox, giving it a name and
      // some values to be associated with its options.
      ListBox lb = new ListBox();
      lb.setName("listBoxFormElement");
      lb.addItem("item1", "item1");
      lb.addItem("item2", "item2");
      lb.addItem("item3", "item3");
      lb.setWidth("220");
      panel.add(lb);

      // Create a FileUpload widget.
      FileUpload upload = new FileUpload();
      upload.setName("uploadFormElement");
      panel.add(upload);

      // Add a 'submit' button.
      panel.add(new Button("Submit", new ClickHandler() {
         @Override
         public void onClick(ClickEvent event) {
            form.submit();
         }
      }));

      // Add an event handler to the form.
      form.addSubmitHandler(new FormPanel.SubmitHandler() {
         @Override
         public void onSubmit(SubmitEvent event) {
            // This event is fired just before the form is submitted.
            // We can take this opportunity to perform validation.
            if (tb.getText().length() == 0) {
               Window.alert("The text box must not be empty");
               event.cancel();
            }
         }
      });

      form.addSubmitCompleteHandler(new FormPanel.SubmitCompleteHandler() {
         @Override
```

```
         public void onSubmitComplete(SubmitCompleteEvent event) {
            // When the form submission is successfully completed,
            // this event is fired. Assuming the service returned
            // a response of type text/html, we can get the result
            // here.
            Window.alert(event.getResults());
         }
      });

      DecoratorPanel decoratorPanel = new DecoratorPanel();
      decoratorPanel.add(form);
      // 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: