皮肤部件

上一节讲解了皮肤分离机制的原理，将一个普通组件拆分成了两个组件。本节说明拆分后逻辑组件是如何访问皮肤里的子项的。这里引入一个皮肤部件(SkinPart)的概念。

首先看一个自定义按钮的例子：

Button.ts

```
    class Button extends eui.Component{
        public constructor(){
            super();
        }
        public labelDisplay:eui.Label;
        private _label:string = "";
        public get label():string{
            return this._label;
        }
        public set label(value:string){
            this._label = value;
            if(this.labelDisplay){
                this.labelDisplay.text = value;
            }
        }
        protected childrenCreated():void{
            super.childrenCreated();
            if(this.labelDisplay){
                this.labelDisplay.text = this._label;
            }
        }
    }
```

上述代码的关键点：所有的可定制皮肤组件都必须继承自eui.Component或它的子类，Component的每个子类都封装了一定的功能，写自定义组件时根据需求选择不同的组件继承即可。这里是一个自定义的按钮，没有额外功能，所以直接继承Component类。然后为自定义按钮声明了一个label属性，作用是修改按钮上显示的文本。每个组件都有一个childrenCreated()方法，它会在组件初始化完成后回调，子类通常覆盖这个方法来访问一些延迟实例化的子项。

下面是这个自定义按钮对应的皮肤：

ButtonSkin.exml

```
<e:Skin class="ButtonSkin" xmlns:e="http://ns.egret.com/eui"> <e:Image source="imag
e/button.png" width="100%" height="100%"/> <e:Label id="labelDisplay" textAlign="cen
ter" verticalAlign="middle" left="20" right="20" top="10" bottom="10"/> </e:Skin>
```

通过如下代码可以将皮肤附加到组件：

```
var button = new Button();
button.skinName="resource/ButtonSkin.exml" //假设Button.exml在resource文件夹下。
this.addChild(button);
```

其中，Button上定义了一个labelDisplay的公开变量，ButtonSkin里也有一个id为labelDisplay的Label节点，在皮肤附加到逻辑组件上时，会自动匹配双方的同名变量和id，这些同名变量就叫做“皮肤部件”(SkinPart)，例如负责显示文本标签的labelDisplay对象，匹配结果就是把ButtonSkin上的Label节点引用赋值给了 `button.labelDisplay` 属性。正是因为这个皮肤部件的自动匹配功能，在按钮初始化完成后，才可以直接访问 `this.labelDisplay` 属性。逻辑组件就是通过定义皮肤部件对皮肤里的子节点进行操作的。

关于皮肤部件，这里还要注意：所有可定制皮肤的组件都定义了各自不同的皮肤部件。当为某个组件定制皮肤时，实际上分为创建对应的皮肤部件和显示图片素材两部分。如果只添加了图片素材，而没有声明并实例化对应变量名的皮肤部件，逻辑组件将无法正常工作。具体皮肤部件名请参考每个组件定义的public公开变量列表。