组装控件(草稿) (SEUG)

组装控件是指利用一到多个现成的控件组合而成的新控件，使用区别于普通的新控件开发方法的、十分快捷的控件封装技巧。组装控件根据其定义方法又可以分为配置型和对象型两种，其中配置型组装控件是比较常用的。

配置型组装控件

由于组装控件是利用现成的控件组合而成的，因此大部分情况下我们只需要通过XML把控件的组合方式定义出来，预设一些属性、再根据需要绑定上一些事件代码就完成了。即使偶尔需要一些更复杂点的逻辑，也可以利用控件的监听器（listener）中的Java代码来进行强化。

Hello World

先来看一个最简单的组装控件的例子。

```
<Button id="ToggleButton" toggleable="true"/>
```

此组装控件只由一个现成的Button控件组成，示例中我们预设了该Button的toggleable属性的值为true。这样，这个按钮就变成了一种开关式的按钮。

组装控件的声明必须有id，我们推荐直接使用新控件的名称作为这段声明的id。比如我们将给新控件起名为ToggleButton，因此我们在上例中定义Button的id为ToggleButton。

> 需要注意的是我们在这里定义的id，并不是最终用户使用该组装控件时，该控件在运行时具有的id。事实上这两个id之间完全没有关系，运行时的控件id完全是由未来的控件使用者决定的。

有了上面的定义，接下来我们就需要将这段配置真正的注册成一个组装控件了。假设上面的定义我们是放在classpath:com/bstek/dorado/sample/AssembledComponents.view.xml中的，那么我们应该在home:components-context.xml中添加下面的配置：

```
<bean parent="dorado.assembledComponentTypeRegister">
 <property name="name" value="ToggleButton" />
 <property name="src" value="com.bstek.dorado/sample/AssembledComponents#ToggleButton" />
</bean>
```

这样，只要在IDE中更新项目的配置规则，我们立刻就可以在IDE控件栏中看到一个名为ToggleButton的新控件了。

> 注意components-context.xml文件的声明中需要引入spring-dorado-7.1.xsd，否则无法完全支持下面介绍的各项功能。例如：
>
> ```
> <?xml version="1.0" encoding="UTF-8"?>
> <beans xmlns="http://www.springframework.org/schema/beans"
> xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
> xmlns:d="http://www.bstek.com/dorado/schema"
>  xsi:schemaLocation="http://www.springframework.org/schema/beans
> http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
> http://www.bstek.com/dorado/schema
> http://www.bstek.com/dorado/schema/spring-dorado-7.1.xsd">
>  ... ... ...
> </beans>
> ```

动态id

上面的例子中，我们介绍了一个由单一控件封装而成的组装控件，不过在其实应用的场景中，组装控件往往不会这么简单，由多个控件封装成的组装控件才能更加凸显出此功能的强大。为了让多个控件形成一个整体，我们应该使用容器型的控件或者具体容器特征的控件作为组装控件的根，例如：Container、Panel等或TabControl、SplitPanel等。可是一旦一个组装控件中包含了多个子控件，我们很容易就会碰到一个新问题——这些子控件之间如何进行相互的访问?

比如组装控件中有一个DataSet和DataGrid，我们希望这里的DataGrid能够绑定到这个DataSet；或者组装控件中有一个Button和一个TextEditor，我们希望点击Button时能够为TextEditor赋一个值。在传统的页面开发模式下，这些都不是什么难事，只要给控件定义一个id，就可以通过这个id进行绑定操作或者api访问了。然而在组装控件中我们却不能这样做，因为id对于一个View而言必须是唯一的，如果我们给上述这些子控件定义一个固定的id，我们难保将来这个组装控件不会再某个View中被使用两次，如果这种情况出现必然会导致Dorado抛出id重复的异常。要解决这个问题，我需要给这些子控件定义一个“动态id”。

定义动态id的方法是利用EL表示式，例如我们设置子DataSet的id为${acomp.id("dataSet1")}
，这里的acomp是AssembledComponent的缩写。之后就可以设置DataGrid的dataSet属性为${acomp.id("dataSet1")}，表示绑定到前面的DataSet。在Javascript代码中同样也是通过这个表达式来访问这个DataSet，例如：

```
view.get("#${acomp.id('dataSet1')}").flush();
或
view.id("${acomp.id('dataSet1')}").flush();
```

对于组装控件的根控件，前面我们提到过，建议直接使用新控件的名称作为其id。看起来这不像是一个动态id，然后实际上他的id也是动态的。因为这个控件的id是由将来的组装控件使用者决定的，我们目前定义的id仅仅用在注册组装控件时。因此，如果我们要在组装控件的内部Javascript中访问根控件同样要利用动态id的方法。假设根控件的id是SimpleCRUD，那么获得他的实际id的方法就是 ${acomp.id("SimpleCRUD")}。

虚拟属性

有时，我们会希望为组装控件添加一些新的属性，以便于实现一些特定的功能。比如我们在组装控件中放置了一个子Button，现在我们希望得IDE的属性列表中能够出现一个叫buttonCaption的属性，以便于用户通过IDE为这个子Button设置标题。这种功能我们称之为虚拟属性，定义虚拟属性的方法如下：

```
<bean parent="dorado.assembledComponentTypeRegister">
 <property name="name" value="ButtonPanel" />
 <property name="src" value="com.bstek.dorado/sample/AssembledComponents#ButtonPanel" />
 <d:virtual-property name="buttonCaption" />
</bean>
```

上面是定义虚拟属性的方法，接下来我们还要设法利用虚拟属性的值。比如，上面的buttonCaption属性，现在我们只要在那个子Button的caption属性中这样定义：${acomp.prop("buttonCaption")}。

（待完善）

虚拟事件

有时，我们会希望把某个子控件的事件暴露给用户，或者将组装控件中的某些操作以事件的方式暴露给用户。比如我们在组装控件中放置了一个子Button，我们希望当用户点击这个按钮能够触发一个事件，而这个事件可以交由组装控件的使用者任意的定义。在这时候，我就需要使用到虚拟事件。

定义虚拟事件的方法如下：

```
<bean parent="dorado.assembledComponentTypeRegister">
 <property name="name" value="ButtonPanel" />
 <property name="src" value="com.bstek.dorado/sample/AssembledComponents#ButtonPanel" />
 <d:virtual-event name="onSubButtonClick" />
</bean>
```

接下来，我们在定义组装控件时为子Button添加一个onClick事件的处理，在其中触发这个虚拟事件：

```
var buttonPanel = view.get("#${acomp.id('ButtonPanel')}"); // 获得组装控件的根控件
buttonPanel.fireEvent("onSubButtonClick", buttonPanel, null); //
触发虚拟事件，此方法的第二个参数相当于传入虚拟事件的self参数，第二个参数相当于传入虚拟事件的arg参数
```