超详细的Java容器、面板及四大布局管理器应用讲解！

发布于2020-09-23 11:17:56 阅读 851

> 本文主要讲解Swing程序设计中容器、面板及四大布局管理器的详细使用、包括实例程序讲解、使用注意及使用技巧分享、敬请阅读！

Hello！你好哇，我是灰小猿！

之前在进行Java的窗体开发时经常会把容器、面板与布局管理混淆，从而不能正确的使用这三种属性对窗体应用进行布局。所以今天在这里记录一下Java窗体中面板及常见的四大布局管理器的用法。

什么是容器?

在Java的GUI界面设计中，关于容器的理解，从字面意思我们就可以认为它是存放控件的地方，而这个地方依托在窗体之上，常用的容器是container。

而关于container容器我们应该有这样的认识：Swing组件中的窗体通常是与容器相关联的，所以在一般情况下，建立完JFrame窗体后，我们会调用getContent方法将窗体转换为容器，之后再在该容器中添加控件或布局管理器。关于控件在container容器中添加和删除用以下两种方法：

```
Container.add(); //为容器添加控件
Container.remove(); //为容器添加控件
```

实例程序如下：

```
public class ConJFrame extends JFrame{

    public ConJFrame() {
        setTitle("Container容器");
        setSize(400,400);
        Container container = getContentPane();      //将窗体变为容器
        /****************在容器中添加一个标签******************/
        JLabel jLabel = new JLabel("这是一个容器");
        jLabel.setBounds(100, 100, 200, 50);
        container.add(jLabel);  //将控件加入到容器

        /****************在容器中添加一个按钮后移除************/
        JButton jButton = new JButton("这是一个按钮");
        jButton.setBounds(100, 200, 200, 50);
        container.add(jButton);     //将按钮加入到容器
        container.remove(jButton);      //将按钮从容器中移除

        setLayout(null);    //清空布局管理器
        setVisible(true);   //设置窗体可见
        setLocationRelativeTo(null);    //设置窗体居中
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);    //设置窗体关闭方式
    }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new ConJFrame();
    }

}
```

效果如下：

什么是面板?

关于面板的解释，你可以认为它也是一个容器，但是这个容器必须添加在其他的容器中，在Swing中常用的面板有两种，分别是JPanel面板和JScrollPane面板，别对这两种面板的用途进行介绍：

JPanel面板

在JPanel面板中我们可以添加一些组件来对该面板中的内容进行布局，之所以它具备这样的功能，是因为JPanel面板同样也继承了java.awt.Container类，因此具有Container容器的功能，

但是与Container容器不同的就是：Container容器不需要添加在其他容器中，而JPanel面板必须添加在其他容器中。

因此我们可以理解为JPanel面板其实就是对一个大的容器的划分，将Container容器根据一定的规则（布局管理）划分成了一个个小的面板。因此JPanel面板的作是与布局管理器相结合的，

JScrollPane面板

先来看一种在界面设计时常见的问题：在一个较小的界面中显示一个较大的内容的情况，对于这种情况，我们常用的方法就是将较小的容器设置为JScrollPane面因是因为JScrollPane面板是自带滚动条的，并且同时它也是一种容器，**这也是在做相关开发时我们设置滚动条常用的一种方法。**

使用JScrollPane面板时需要注意以下两个问题：

1. JScrollPane面板中只能布置一个控件，

2. JScrollPane面板不能使用布局管理器

因此如果想要在JScrollPane面板中显示多个控件，就需要首先将控件布局在JPanel面板中，之后将JPanel面板作为一个整体组件添加到JScrollPane面板中，

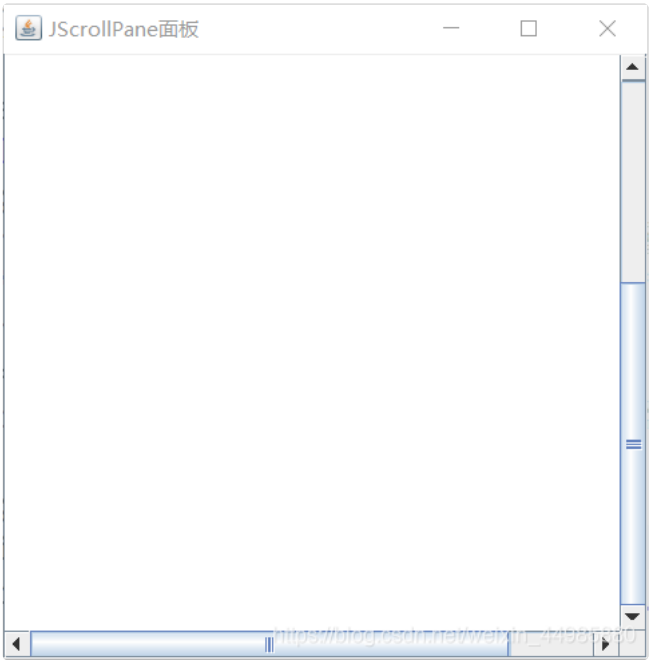
通过下面程序对JScrollPane面板进行实践：

以下程序是在JScrollPane面板中加入一个文本框，实现一个带有滚动条的文本框。

```
public class JScrollPaneClass extends JFrame{

    public JScrollPaneClass() {
        setTitle("JScrollPane面板");
        setSize(400,400);
        Container container = getContentPane();      //将窗体变为容器
        /****************在JScrollPane面板中添加一个文本框******************/
        JTextArea jTextArea = new JTextArea();
        JScrollPane jsp = new JScrollPane(jTextArea);
        container.add(jsp);      //将JScrollPane面板加入容器

        setVisible(true);   //设置窗体可见
        setLocationRelativeTo(null);    //设置窗体居中
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);    //设置窗体关闭方式
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new JScrollPaneClass();
    }
```

```
20
21  }
```

效果如下：



什么是布局管理器?

关于布局管理器，在Swing中，每个组件都有一个具体的位置和大小，而在容器中真正去摆放这些组件时其实是很难判断其具体位置和大小的，布局管理器就提对swing组件的排版方式，因此使用布局管理器就可以很有效的处理整个窗体中组件的布局方式，Swing提供的常用的布局管理器有四种：绝对布局管理器、流布器、边界布局管理器、网格布局管理器，接下来会对这四种布局管理器进行探讨。

绝对布局管理器

除了网格布局管理器、流布局管理器、边界布局管理器这三种布局方式以外，还有一种较为不同的布局方式就是绝对布局，所谓绝对布局，就是按照一定的坐标组件的坐标和大小硬性的设置在窗体上。

使用绝对布局时首先有一点需要注意：就是要先取消默认布局管理器，方法为：

> setLayout(null); //清空布局管理器，即取消原来的边界布局管理器

至于为什么要这样做，详细的解释可以看我的这篇文章：“盘点Java窗体中关于默认布局管理器容易踩的坑”

之后再使用以下方法对组件进行绝对定位：

> setBounds(x,y,width,height);
> //其中x表示组件基于容器左上角的横坐标、y表示纵坐标，width表示组件的宽，height表示组件的高

请看下面实例：在容器中采用绝对布局添加三个控件，并赋于横纵坐标和按钮的长宽：

```
public class AbsolutelyLayoutClass extends JFrame{

    public AbsolutelyLayoutClass() {
        setTitle("绝对布局管理器");
        setSize(600,300);
        Container container = getContentPane();      //将窗体变为容器
        /****************使用绝对布局管理器布局组件******************/
        JButton jb1 = new JButton("这是绝对布局1");
        jb1.setBounds(200, 50, 200, 30);    //为组件设置绝对坐标
        container.add(jb1);     //将组件添加到容器

        JButton jb2 = new JButton("这是绝对布局2");
        jb2.setBounds(200, 100, 200, 30);   //为组件设置绝对坐标
        container.add(jb2); //将组件添加到容器

        JButton jb3 = new JButton("这是绝对布局3");
        jb3.setBounds(200, 150, 200, 30);   //为组件设置绝对坐标
        container.add(jb3); //将组件添加到容器

        setLayout(null);    //清空布局管理器，即取消原来的边界布局管理器
```

```
        setVisible(true);    //设置窗体可见
        setLocationRelativeTo(null);    //设置窗体居中
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);    //设置窗体关闭方式
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new AbsolutelyLayoutClass();
    }

}
```

效果如下：

在使用绝对布局管理器时值得注意的就是：在使用绝对布局之前要调用setLayout(null)方法来告知编辑器，这里将不再使用默认的布局管理器。

流布局管理器

流布局管理器（FlowLayout）具有的特点就是：在其中的组件可以像“流”一样按照从左到右的顺序依次的排列，直到该行排列完毕后再从下一行继续排列。在默认情况下，组将在流布局管理器中都是以居中排列的，当然排列的顺序是可以改变的，

FlowLayout类中的构造函数中有如下三种：

Public FlowLayout();

Public FlowLayout(int alignment);

Public FlowLayout(int alignment, int horizGap, int vertGap);

以上构造函数中的alignment参数表示组件在采用流布局管理器后在每一行的具体排放位置。可以赋的值为：

FlowLayout.LEFT = 0;

FlowLayout.CENTER = 1;

FlowLayout.RIGHT = 2;

以上三个值被赋予以后，表示组件在流布局管理器中每一行的摆放位置和摆放顺序，如当alignment=0时，流布局管理器中的组件按照从左到右的顺序排列，当alignment=1时，流布局管理器中的组件按照从中间向两端的顺序排列。

在Public FlowLayout(int alignment, int horizGap, int vertGap);构造方法中，后面的参数horizGap和vertGap分别表示以像素为单位指定组件之间的水平间隔和垂直间隔。

关于流布局管理器的具体使用可以参考如下实例：

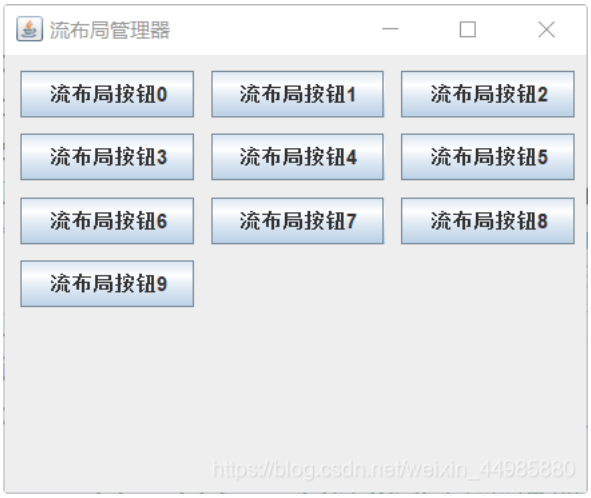
在该窗体中按照流布局管理器添加10个按钮

```
public class FlowLayoutClass extends JFrame{

    public FlowLayoutClass() {
        setTitle("流布局管理器");
        setSize(600,300);
        Container container = getContentPane();      //将窗体变为容器
        /****************使用流布局管理器布局组件******************/
        //将容器设置为从左向右排列、组件水平间隔和垂直间隔分别为10的流布局管理器
        container.setLayout(new FlowLayout(0,10,10));
        for (int i = 0; i < 10; i++) {
            container.add(new JButton("流布局按钮" + i));
        }
        //setLayout(null);  //清空布局管理器，即取消原来的边界布局管理器
        setVisible(true);    //设置窗体可见
        setLocationRelativeTo(null);    //设置窗体居中
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);    //设置窗体关闭方式
```

```
    }
    public static void main(String[] args) {
        new FlowLayoutClass();

    }
}
```

效果如下：

在我们拉动窗体大小变化时，其中的组件也会按照流布局的特点发生改变，这就验证了流布局管理器中的组件按照从左到右的顺序依次摆放，直到该行占满后再行开始摆放。



边界布局管理

在默认不指定窗体布局的情况下，Swing组件的布局模式就是边界布局（BorderLayout），该布局管理器的特征就是组件会按照一定要求布满整个容器的边界，

边界布局管理器可以将容器划分成东、南、西、北、中五个区域，在容器中添加组件时，我们可以设置组将放入到哪一个区域中，关于区域的控制可以使用BorderLayout类中的成员方法来确定，关于这些成员变量的具体含义可以参考下表：

成员变量	含义
BorderLayout.NORTH	在容器中添加组件时，组件置于顶端
BorderLayout.SOUTH	在容器中添加组件时，组件置于底端
BorderLayout.EAST	在容器中添加组件时，组件置于右端
BorderLayout.WEST	在容器中添加组件时，组件置于左端
BorderLayout.CENTER	在容器中添加组件时，组件置于中间开始填充，直到与其他组件边界连接

关于边界布局管理器的具体使用可以参考如下实例，将容器划分为东、南、西、北、中五个区域，并在各区域中添加组件。

```
public class BorderLayoutClass extends JFrame{
```

```
    public BorderLayoutClass() {
        setTitle("边界布局管理器");
        setSize(600,300);
        Container container = getContentPane();      //将窗体变为容器
        /*****************使用边界布局管理器布局组件******************/
        //将容器设置为边界布局管理器
        container.setLayout(new BorderLayout());
        container.add(new JButton("我是北部区域"),BorderLayout.NORTH);     //将按钮加入到北部区域
        container.add(new JButton("我是西部区域"),BorderLayout.WEST);      //将按钮加入到西部区域
        container.add(new JButton("我是中部区域"),BorderLayout.CENTER);    //将按钮加入到中部区域
        container.add(new JButton("我是东部区域"),BorderLayout.EAST);      //将按钮加入到东部区域
        container.add(new JButton("我是南部区域"),BorderLayout.SOUTH);     //将按钮加入到南部区域
        //setLayout(null);  //清空布局管理器，即取消原来的边界布局管理器
        setVisible(true);   //设置窗体可见
        setLocationRelativeTo(null);    //设置窗体居中
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);    //设置窗体关闭方式
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new BorderLayoutClass();
    }

}
```

效果如下：

在该方法中直接在add()方法中新建组件，并且在后方加入表示组件位置的参数。

网格布局管理器

网格布局管理器（GridLayout）从字面意思就可以理解，就是将容器按照行列划分成特定的网格，在网格布局管理器中每一个网格的大小都是一样的，并且网格的个数是由划分的行和列决定的，如一个两行两列的网格布局，将会划分成四个大小相同的网格。

在网格布局中的组将会按照从左到右、从上到下的顺序加入到网格中，而且加入到网格中的组件都会将网格填满，同时改变窗体的大小，网格的大小也会随之改

网格布局中常用的布局管理器有如下两种：

Public GridLayout(int rows, int columns);

Public GridLayout(int rows, int columns, int horizGap, int vertGap);

其中的rows和columns分别表示网格布局的行和列，**这两个参数中只有一个可以为0，**表示为一行或一列可以摆放多个组件，horizGap和vertGap两个参数和流布局管理器中的一样，只不过在流布局管理器中表示的是组件之间的水平和垂直间距，而在网格布局管理器中表示网格之间的水平和垂直间距，

关于网格布局管理器的具体使用参考如下实例，

将容器设置为4行5列的网格，网格之间的水平和垂直间距为10像素。并在其中加入20个按钮

```
public class GridLayoutClass extends JFrame{

    public GridLayoutClass() {
        setTitle("网格布局管理器");
        setSize(600,300);
        Container container = getContentPane();      //将窗体变为容器
        /*****************使用网格布局管理器布局组件******************/
        //将容器设置为4行5列网格布局管理器，网格之间的水平和垂直间距都为10像素
        container.setLayout(new GridLayout(4,5,10,10));
        for (int i = 0; i < 20; i++) {
            container.add(new JButton("网格" + i));
```

```
        }
        //setLayout(null);  //清空布局管理器，即取消原来的边界布局管理器
        setVisible(true);   //设置窗体可见
        setLocationRelativeTo(null);    //设置窗体居中
        setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);    //设置窗体关闭方式
    }
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        new GridLayoutClass();
    }

}
```

效果如下

在拉动窗体改变大小时，网格的大小也会随之改变。

容器、面板、布局管理器之间的关系

关于Swing窗体开发中的容器、面板、布局管理器之间有如下的关系：

面板应该设置在容器之中，

布局管理器可以设置在容器或面板之中，

布局管理器中还可以嵌套面板，在该面板中还可以再添加布局管理器，

只有深刻的理解了这三者之间的关系，才能在窗体开发中很好的结合使用。

觉得有用记得点赞关注哟！

容器 javascript 编程算法 java