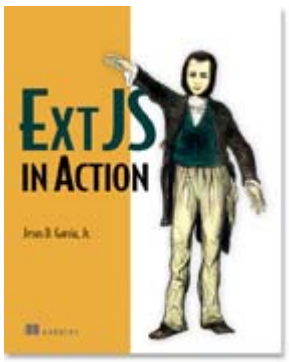
MANNING PUBLICATIONS

The Table Layout

Excerpted from



Ext JS in Action

EARLY ACCESS EDITION

Jesus Garcia

MEAP Release: February 2009
Softbound print: January 2010 (est.) | 425 pages
ISBN: 9781935182115

This article is taken from the book *Ext JS in Action*. As part of a chapter on organizing your components in a wide variety of layouts, this segment demonstrates how to use the table layout, in which you can position child components like a traditional HTML table.

The table layout gives you complete control over how you want to visually organize your components. Many of us are used to building HTML tables the traditional way, where we actually write the HTML code. Building a table of Ext Components, however, is different as we specify the content of the table cells in a single dimension array, which can get a little confusing. Once you've done these exercises, you'll be an expert in this layout. Let's create a 3 x 3 table layout:

Listing 1 A vanilla table layout

```
var myWin = new Ext.Window({
    height       : 300,
    width        : 300,
    border       : false,
    autoScroll   : true,
    title        : 'A Window with a Table layout',
    layout       :'table',                                    (1)
    layoutConfig : {
       columns : 3                                            (2)
    },
    defaults : {                                              (3)
       height : 50,
       width  : 50
    },
    items : [
       {
          html : '1'
       },
       {
          html : '2'
       },
       {
```

```
            html : '3'
        },
        {
            html : '4'
        },
        {
            html : '5'
        },
        {
            html : '6'
        },
        {
            html : '7'
        },
        {
            html : '8'
        },
        {
            html : '9'
        }
    ]
});

myWin.show();
```

{1} Specifying the layout as table
{2} Set the number of columns for the table to 3
{3} Set default size for each box to 50x50

The code in Listing 1 creates a Window Container that has nine boxes stacked in a 3x3 formation like in Figure 1. I want to make sure we highlight a few items, the most obvious of which should be the layout parameter **{1}** being set to 'table'. Next, we set a layoutConfig **{2}** object, which sets the number of columns. Always remember to set this property when using this layout. Lastly, we're setting the defaults **{3}** for all of the child items to 50 pixels wide by 50 pixels high.

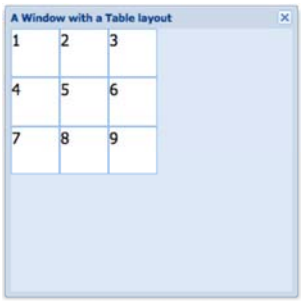


Figure 1 The results of our first simple table layout.

Often we need sections of the table to span multiple rows or multiple columns. To accomplish this, we specify either the rowspan or colspan parameters explicitly on the child items. Let's modify our table so the child items can span multiple rows or columns

Listing 2 Exploring rowspan and colspan

```
items : [
    {
        html    : '1',
        colspan : 3,                                    (1)
        width   : 150
    },
    {
        html    : '2',
        rowspan : 2,                                    (2)
        height  : 100
    },
```

```
    {
        html : '3'
    },
    {
        html    : '4',
        rowspan : 2,                                          (3)
        height  : 100
    },
    {
        html : '5'
    },
    {
        html : '6'
    },
    {
        html : '7'
    },
    {
        html : '8'
    },
    {
        html    : '9',
        colspan : 3,                                          (4)
        width   : 150
    }
]
```

{1} Set colspan to 3 and width to the total width to 150 pixels
{2} Set rowspan to 2 and height to 100 pixels
{3} Set rowspan to 2 and height to 100 pixels
{4} Set colspan to 3 and width to 150 pixels

In Listing 2, we reuse the existing Container code from Listing 1 and replace the child items array. We set the colspan attribute for the first panel **{1}** to 3, and manually set its width to fit the total known width of the table, which is 150 pixels. Remember that we have 3 columns of default 50x50 child containers. Next, we set the rowspan of the second child **{2}** item to 2 and its height to the total of two rows, which is 100 pixels. We do the exact same thing for Panel 4 **{3}**. The last change involves panel 9, which has the exact same attributes as panel 1 **{4}**. The rendered change should look just like Figure 2.

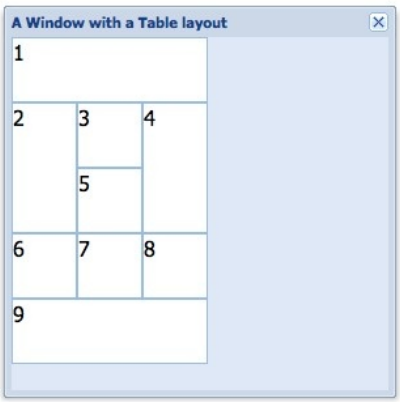


Figure 2 When using the table layout, you could specify rowspan and colspan for a particular Component, which will make it occupy more than one cell in the table.

When using the Table layout, you should remember a few things. First, determine the total number of columns that will be used and specify it in the layoutConfig parameter. Also, if you're going to have Components span rows and/or columns, be sure to set their dimensions accordingly, otherwise the components laid out in the table will not seem to be aligned correctly.

The Table Layout is extremely versatile and can be used to create any type of box-based layout that your imagination conjures up, with the main limitation being that there is no parent-child size management.