CREACION DE INTERFACES GRAFICAS DE USUARIO (GUI) PARA LA WEB

Universidad Surcolombiana
Tecnología en Desarrollo de Software
Factoría de Software
Autor: Ing. Juan Antonio Castro Silva
Versión: Beta 01 19NOV2012-0632

Introducción:

En este modulo vamos a trabajar con Sencha ExtJs una librería de

Objetivos:

Al finalizar este modulo el estudiante estará en capacidad de:

- Crear una interface gráfica de usuario aplicando el paradigma de orientación a objetos(clases, atributos, métodos).
- Crear una grilla
- Crear barras de herramientas
- Manejar los eventos de botones
- Crear un formulario
- Crear Ventanas
- Crear Paneles
- Crear almacenes de datos
- Validar formularios
- Crear diferentes tipos de campos
- Crear una aplicación aplicando el patrón de diseño MVC (Modelo, Vista, Controlador)

INTRODUCCION

Ext JS es una framework (marco de trabajo – librería) RIA (Rich Internet Application - Aplicaciones Ricas de Internet) independiente del browser (cross-browser), fácil de usar con componentes de interface de usuario (UI) ricos, utilizado por mas de un millón de desarrolladores alrededor del mundo.

POR QUE USAR EXTJS ???

LICENCIAMIENTO

SOPORTE

HERRAMIENTAS

COMUNIDAD

USUARIOS

DESARROLLO EN EL TIEMPO - HISTORIAL

JUSTIFICAR

COMPARAR CON OTROS PRODUCTOS

EJEMPLOS DE USO

DOCUMENTACION

EJEMPLOS

0. ANTES DE INICIAR

Antes de iniciar con este modulo es necesario contar con un editor de texto y un navegador con algunas herramientas para desarrolladores.

Recomendamos seguir usando el entorno de desarrollo integrado (IDE) eclipse y Firefox (con FireBug) o Google Chrome (con Developer Tools).

Todos los ejemplos requieren el SDK (Software Development Kit) de Ext JS 4, el cual esta disponible como una descarga libre en el sitio web de Sencha http://www.sencha.com/products/extjs/.

Additionally, some make use of the Sencha SDK Tools, which can be downloaded from http://www.sencha.com/products/sdk-tools/.

Si bien cada problema es un ejemplo independiente, necesitamos incluir el SDK y adicionar el método Ext.onReady a nuestro archivo HTML, el cual ejecutará la función pasada cuando todo sea plenamente cargado. Prepare un archivo HTML con lo siguiente, el cual puede ser usado como un punto de inicio para la mayoría de los ejemplos.

```
<html>
<head>
    <link rel="stylesheet" type="text/css" href="extjs/resources/css/ext-all.css">
    <script type="text/javascript" src="extjs/ext-all-debug.js">
    </script>
    <script type="text/javascript">
        Ext.onReady(function () {
        //El código fuente del ejemplo va aquí
       });
    </script>
</head>
<body>
</body>
</html>
```

El código fuente de los ejemplos suministrado con este modulo puede ser ejecutado como un proyecto independiente o importando cada carpeta de capitulo dentro de la carpeta de ejemplos.

1. INSTALACION

PANTALLA DESCARGA

EXPLICACION ARCHIVOS Y CARPETAS

HOLA MUNDO – MI PRIMER PROGRAMA – EJEMPLO

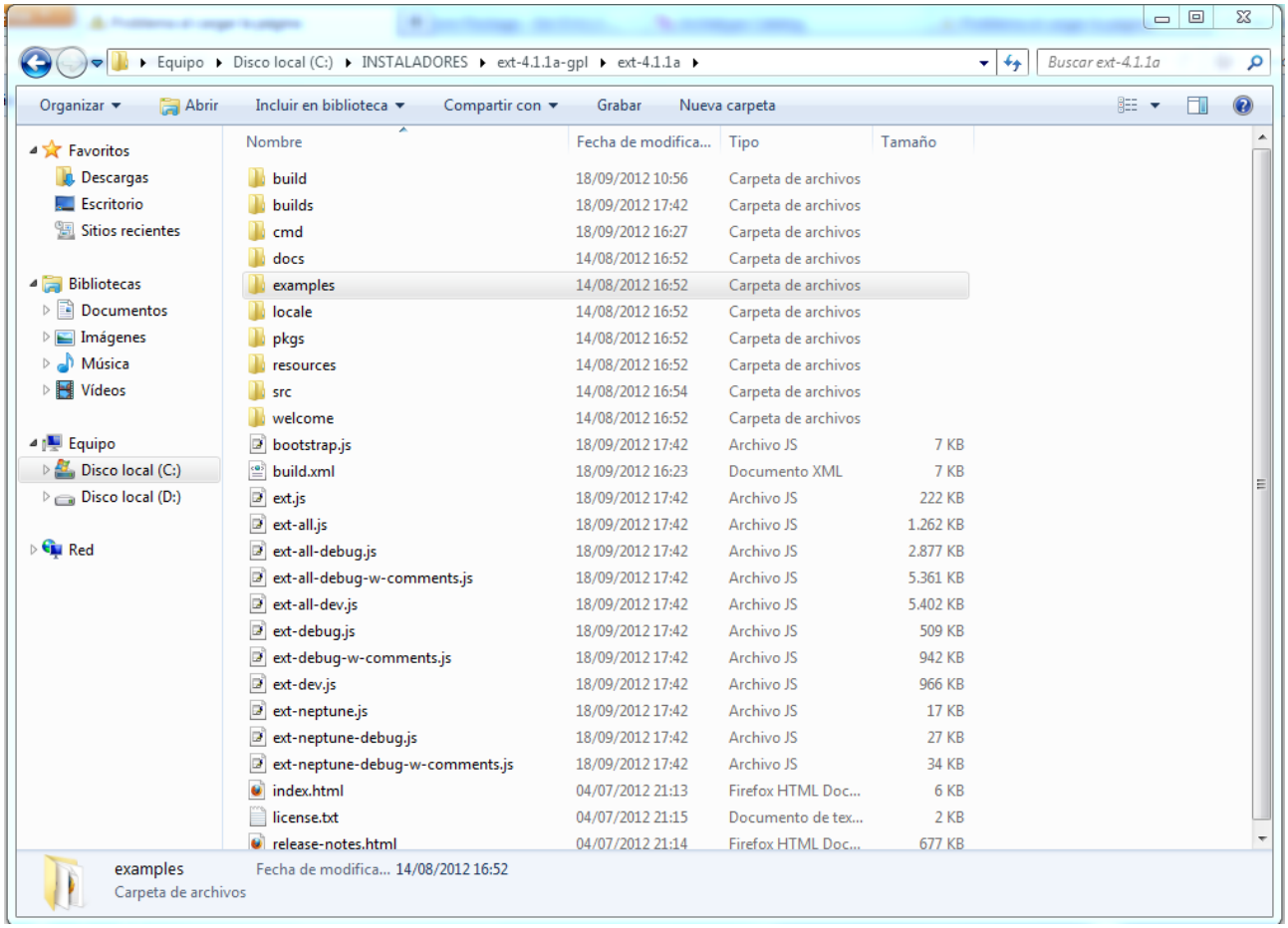
Crear una aplicación web en jboss

1. Descargue el SDK de Ext JS 4 desde el sitio web de Sencha http://www.sencha.com/products/extjs/.

2. Descomprima el archivo descargado (ext-4xxx.zip).

3. Renombre la carpeta descomprimida y renombrela como extjs y cópiela dentro de la aplicación de jboss gui.war.

La carpeta con el SDK de Ext JS contiene lo siguiente:



El archivo ext-all.js y ext.js contienen el framework de Ext JS 4,

El archivo ext.css contiene la hoja de estilos del SDK

Mi primer programa (Hola Mundo)

Para probar el SDK Ext JS 4 vamos a crear un programa de ejemplo:

Cree una carpeta llamada gui01 dentro del directorio de publicación y adicione un archivo HTML con el siguiente contenido:

[gui01/index.html]

```
01 <html>
02 <head>
03     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
04     <link rel="stylesheet" type="text/css" href="../extjs/resources/css/ext-all.css">
05     <script type="text/javascript" src="../extjs/ext-all-debug.js">
06     </script>
07     <script type="text/javascript">
08         Ext.onReady(function () {
09               //El código fuente del ejemplo va aquí
10               Ext.Msg.alert('Aplicación', "Hola Mundo!");
11         });
12     </script>
13 </head>
14 <body>
15 </body>
16 </html>
```

Para hacer uso del SDK debe incluir la hoja de estilos y los archivos java script necesarios mediante las siguientes lineas:

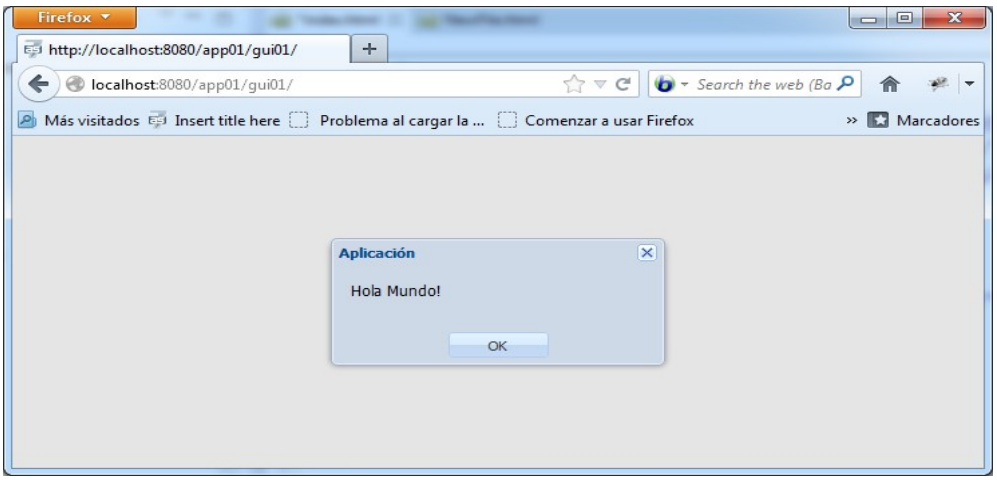
<link rel="stylesheet" type="text/css" href="../extjs/resources/css/ext-all.css">

<script type="text/javascript" src="../extjs/ext-all-debug.js">

El método **Ext.onReady(function () {…});** se ejecuta después que se han cargado los archivos *.css y *.js, como el nombre indica cuando la pagina web este lista –preparada (**onReady**).

La linea `Ext.Msg.alert('Aplicación', "Hola Mundo!");` muestra un cajón de mensaje con el titulo "Aplicación" y el mensaje "Hola Mundo!".

Para probar la pagina web abra un browser y apunte a la dirección donde esta publicada, para nuestro caso http://localhost:8080/app01/gui01/.



1. ORIENTACION A OBJETOS

1.1 Crear una Clase

Si bien JavaScript no es un lenguaje basado en clases, es posible simular clases usando su estructura prototipica. Ext JS 4 introduce una nueva forma de definir clases.

En esta sección explicaremos como definir clases usando el nuevo sistema y se darán detalles de acerca de las características que este tiene para ofrecer. Lo haremos creando una clase personalizada para modelar una persona, con un método que muestra algunos detalles acerca de esta.

Como hacerlo...

El método Ext.define se utiliza para definir nuevas clases. Usa una definición basada en texto, dejando al framework el cuidado por el nombre del espacio (namespacing) y concretar la definición de la clase:

1. Llame al método Ext.define con nuestro nombre de clase y objeto de configuración.

   ```
   //Define una nueva clase "Persona" bajo el nombre del espacio de nombre "Uscosoft"
   Ext.define('Uscosoft.Persona', {
       //la configuración de la clase va aquí
   });
   ```

2. Adicione atributos y métodos al objeto configuración:

   ```
   Ext.define('Uscosoft.Persona', {
       apellido: 'Einstein',
       nombre: 'Albert',
       getDetalles: function(){
          alert('Mi nombre es ' + this.nombre + ' ' + this.apellido);
        }
   });
   ```

3. Ahora adicionamos el tercer parámetro opcional del método Ext.define, el cual es una función que es ejecutada después de que la clase ha sido definida, entre el ámbito (scope) de la nueva clase creada:

   ```
   Ext.define('Uscosoft.Persona', {
       apellido: 'Einstein',
       nombre: 'Albert',
       getDetalles: function(){
          alert('Mi nombre es ' + this.nombre + ' ' + this.apellido);
        }
   }, function(){
       console.log('La clase Uscosoft.Persona esta definida!');
   });
   ```

4. Finalmente, creamos una instancia (objeto) de la nueva clase, mostramos el

valor del atributo apellido y llamamos su método getDetalles:

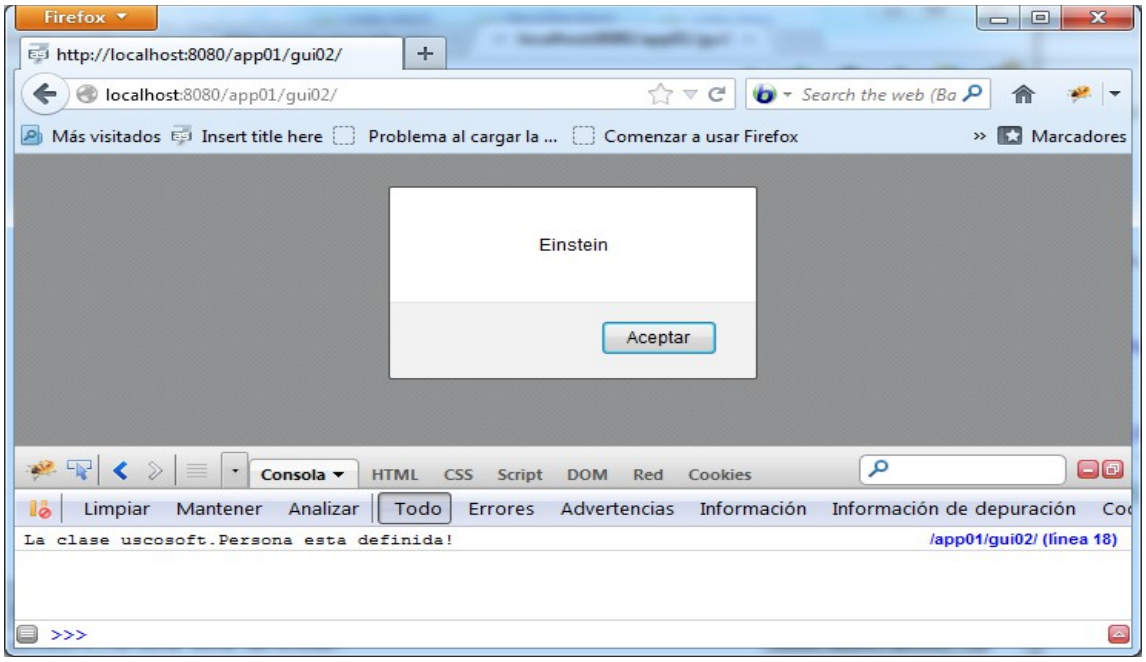
```
var persona1 = Ext.create('Uscosoft.Persona');
alert(persona1.apellido); // alerts 'Einstein'
persona1.getDetalles(); // alerts 'Mi nombre es Albert Einstein'
```

Cree un archivo HTML con el siguiente contenido en una carpeta llamada gui02:

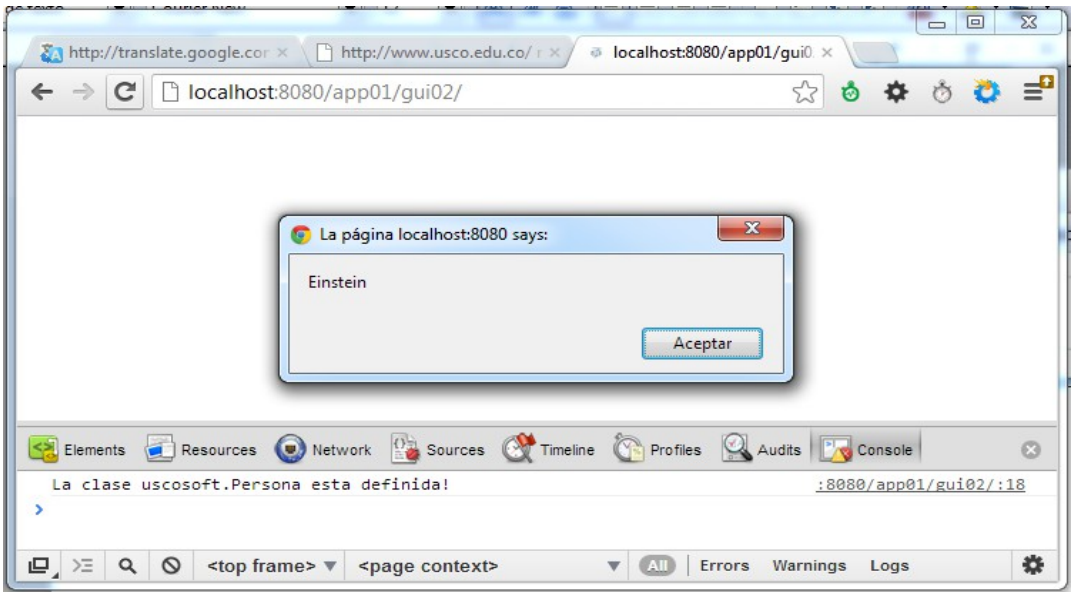
[gui02/index.html]

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" type="text/css" href="../extjs/resources/css/ext-all.css">
    <script type="text/javascript" src="../extjs/ext-all-debug.js">
    </script>
    <script type="text/javascript">
        Ext.onReady(function () {
            //El código fuente del ejemplo va aquí
            Ext.define('Uscosoft.Persona', {
                apellido: 'Einstein',
                nombre: 'Albert',
                getDetalles: function(){
                    alert('Mi nombre es ' + this.nombre + ' ' + this.apellido);
                }
            },
            function(){
                console.log('La clase Uscosoft.Persona esta definida!');
            });
            var persona1 = Ext.create('Uscosoft.Persona');
            alert(persona1.apellido); // muestra 'Einstein'
            persona1.getDetalles(); // muestra 'Mi nombre es Albert Einstein'
        });
    </script>
</head>
<body>
</body>
</html>
```

El resultado en el browser Firefox con Firebug:



El resultado en el navegador Google Chrome con Herramientas para Desarrolladores:



La instrucción `console.log('La clase Uscosoft.Persona esta definida!');` utiliza método log() del objeto console para imprimir un mensaje de texto en el Firebug o en las Herramientas para Desarrolladores de los navegadores Firefox o Google Chrome respectivamente.

Es importante resaltar que para que se pueda tener acceso al objeto console se debe usar el SDK de Ext JS 4 en modo depuración, incluyendo el archivo ***ext-all-debug.js***:

```
<script type="text/javascript" src="../extjs/ext-all-debug.js">
```

3. DISTRIBUCION DE LOS COMPONENTES (layout)

En esta sección vamos a explorar el sistema de layout (distribución) en Ext JS 4 y demostraremos como usar estos layouts para ubicar sus componentes de la interface de usuario. Los layouts con los que trabajaremos son FitLayout, BorderLayout, HBox layout, VBox layout, ColumnLayout, TableLayout, AccoridionLayout, CardLayout, AnchorLayout, y AbsoluteLayout. El ejemplo final combinará un numero de estos layouts para crear un framework para una aplicación rica estilo escritorio.

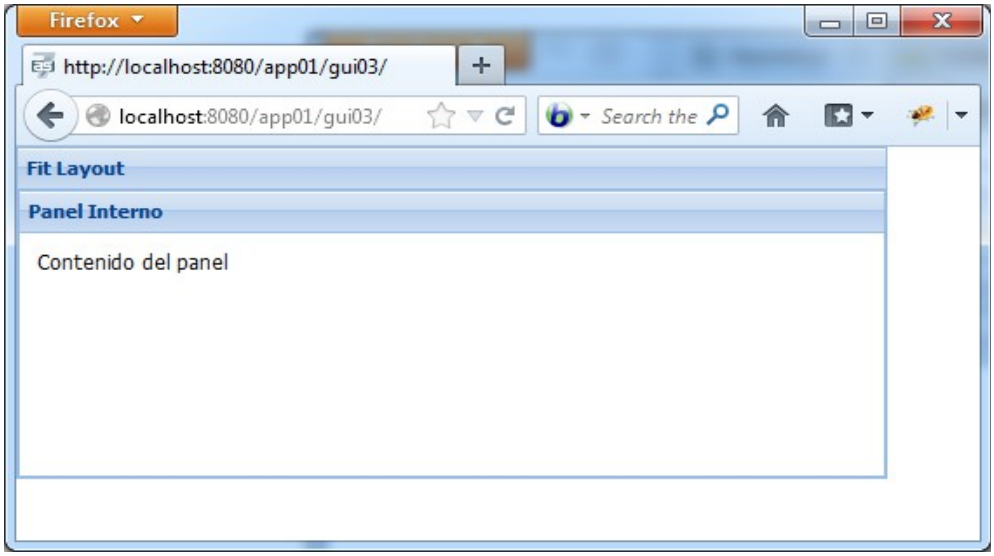
3.1 FitLayout

El FitLayout hace posible para usted expandir un componente para llenar su contenedor padre. El FitLayout es fácil de usar y no requiere configuración. El pantallazo (screenshot) abajo muestra como un FitLayout se puede usar para expandir automáticamente un panel para tomar toda el área de su contenedor (padre):

Para crear una distribución de componentes tipo fit (ajustado) establecemos la propiedad layout como 'fit' (linea 06).

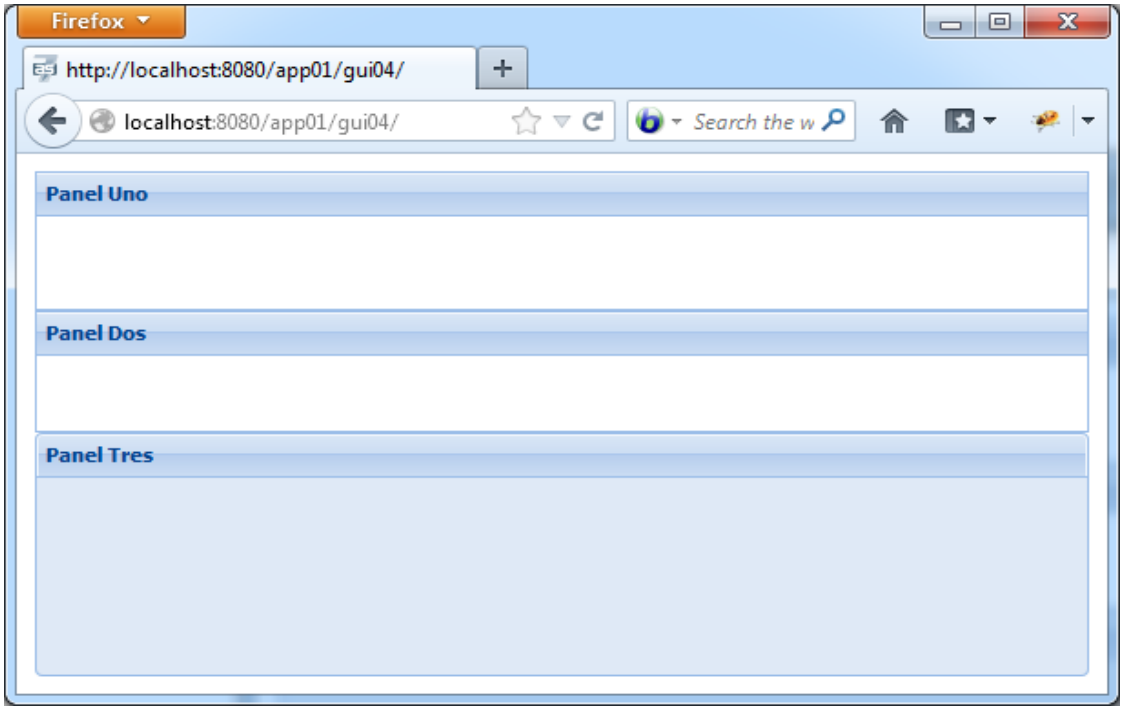
```
01 Ext.onReady(function () {
02     Ext.create('Ext.panel.Panel', {
03        title: 'Fit Layout',
04        width: 500,
05        height: 200,
06        layout: 'fit',
07        items: {
08            title: 'Panel Interno',
09            html: 'Contenido del panel',
10            bodyPadding: 10,
11               border: true
12            },
13            renderTo: Ext.getBody()
14        });
15 });
```

Observe que el panel interno ocupa todo el espacio del panel que lo contiene:



3.2 Crear un layout vertical flexible con VBoxes

El VBoxLayout le permite alinear componentes verticalmente hacia abajo en un contenedor. El pantallazo de abajo muestra tres paneles alineados verticalmente, dividiendo el espacio disponible entre ellos:



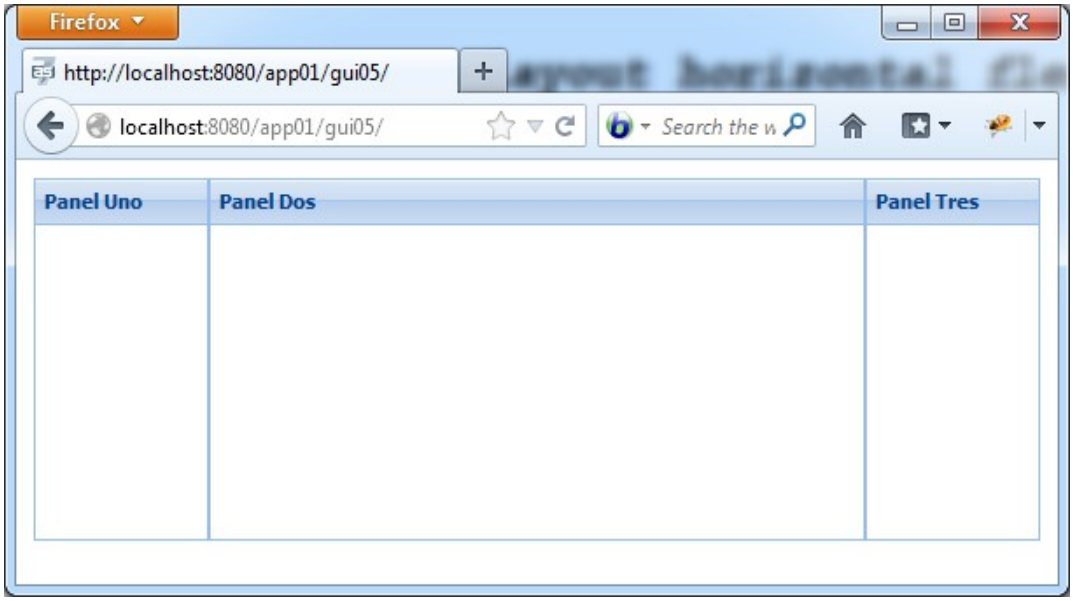
propiedad layout como 'fit' (linea 06).

```
01 Ext.onReady(function () {
02     Ext.create('Ext.container.Viewport', {
03        layout: {
04            type: 'vbox',
05            align: 'stretch',
06            padding: 10
07        },
08        items: [{
09            xtype: 'panel',
10            title: 'Panel Uno',
11            height: 80
12        }, {
13            xtype: 'panel',
14            title: 'Panel Dos',
15            flex: 1
16        }, {
17            xtype: 'panel',
18            title: 'Panel Tres',
19            frame: true,
20            flex: 2
21        }]
22     });
23 });
```

Observe que el panel interno ocupa todo el espacio del panel que lo contiene:

3.3 Crear un layout horizontal flexible con HBoxes

En

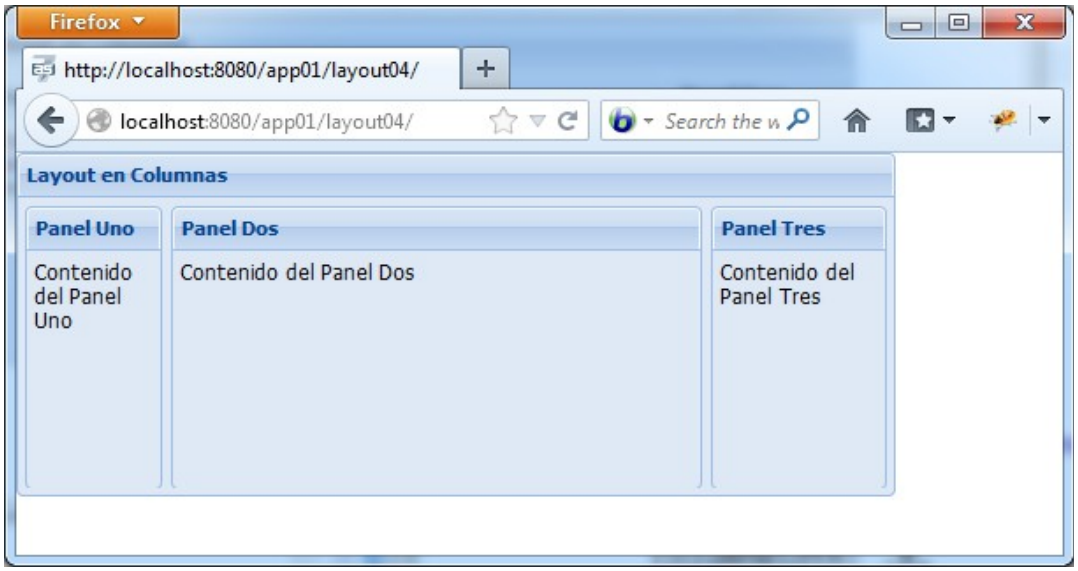


Código

```
01 Ext.onReady(function () {
02     Ext.create('Ext.container.Viewport', {
03        layout: {
04            type: 'hbox',
05            align: 'stretchmax',
06            padding: 10
07        },
08        items: [{
09            xtype: 'panel',
10            title: 'Panel Uno',
11            height: 200,
12            width: 100
13        }, {
14            xtype: 'panel',
15            title: 'Panel Dos',
16            flex: 1
17        }, {
18            xtype: 'panel',
19            title: 'Panel Tres',
20            width: 100
21        }]
22     });
23 });
```

3.4 Mostrar contenido en columnas

El eqfwr

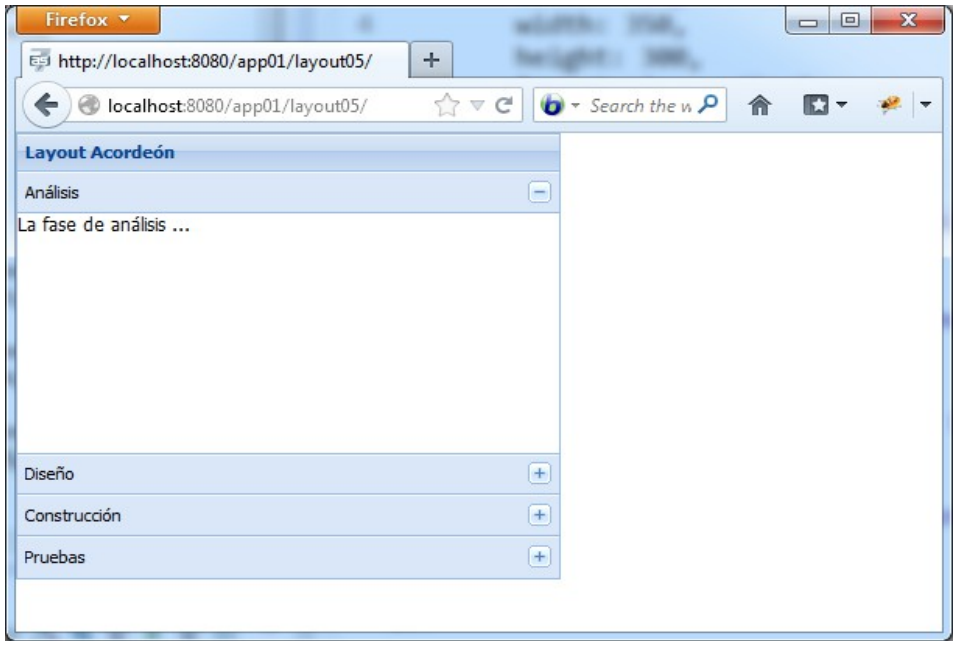


Código

```
01 Ext.onReady(function () {
02     Ext.create('Ext.Panel', {
03         title: 'Layout en Columnas',
04         width: 500,
05         height: 200,
06         layout: 'column',
07         frame: true,
08         defaults: {
09             height: 165,
10             frame: true
11         },
12         items: [{
13             title: 'Panel Uno',
14             columnWidth: .2,
15             html: 'Contenido del Panel Uno'
16         }, {
17             title: 'Panel Dos',
18             columnWidth: .8,
19             margin: '0 5 0 5',
20             html: 'Contenido del Panel Dos'
21         }, {
22             title: 'Panel Tres',
23             width: 100,
24             html: 'Contenido del Panel Tres'
25         }],
26         renderTo: Ext.getBody()
27     });
28 });
```

3.5 Layouts colapsables con acordeones

El layout tipo acordeón

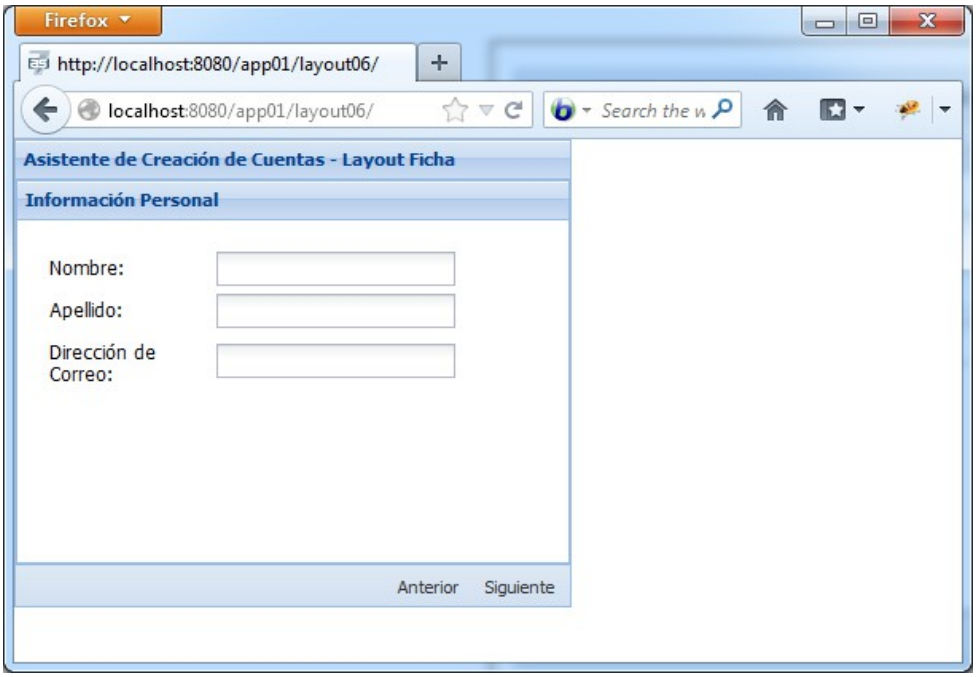


Código

```
01 Ext.onReady(function () {
02     Ext.create('Ext.panel.Panel', {
03        title: 'Layout Acordeón',
04        width: 350,
05        height: 300,
06        layout: 'accordion',
07        renderTo: Ext.getBody(),
08        items: [{
09            title: 'Análisis',
10            html: 'La fase de análisis ...'
11        }, {
12            title: 'Diseño',
13            html: 'El diseño comprende ...'
14        }, {
15            title: 'Construcción',
16            html: 'Los lenguajes de programación ...'
17        }, {
18            title: 'Pruebas',
19            html: 'Las pruebas de software ...'
20        }]
21     });
22 });
```

3.5 Mostrar componentes apilados con CardLayouts

El CardLayout nos permite cdfvbnm,

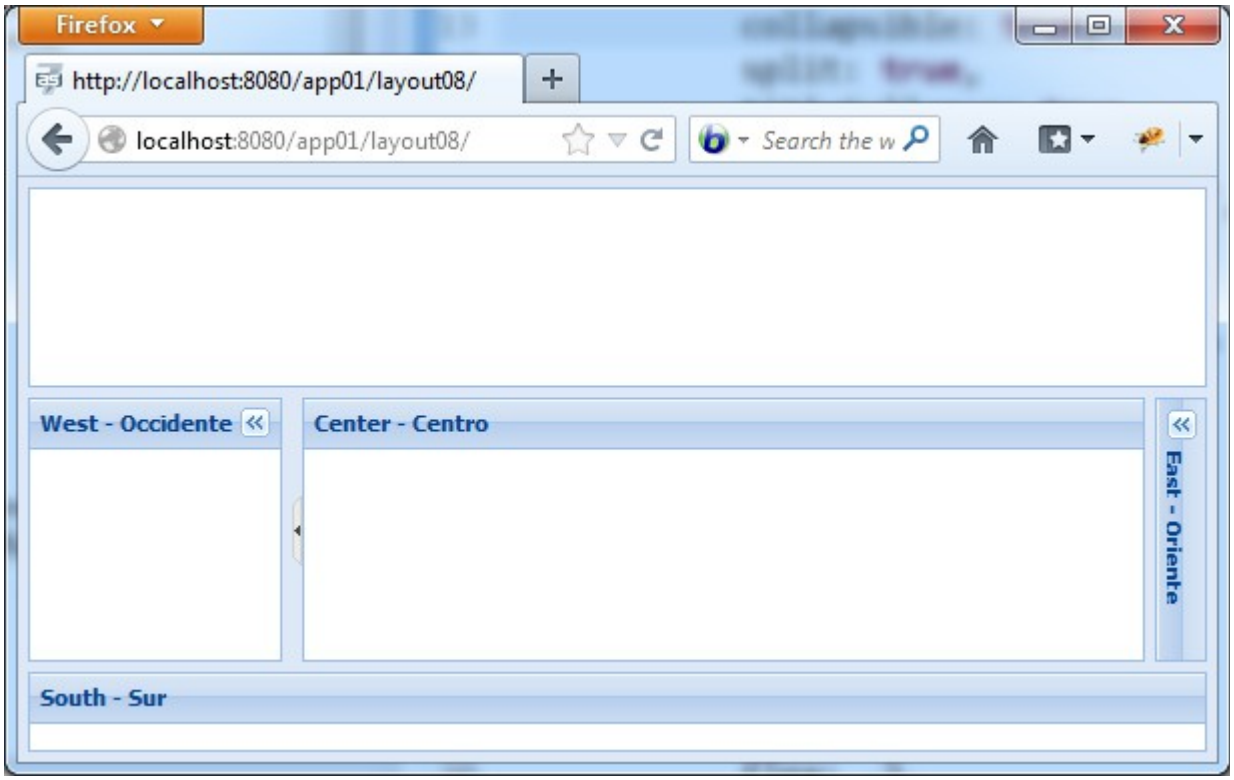


Código

```
Ext.onReady(function () {
    var card1 = new Ext.panel.Panel({
        bodyStyle: 'padding: 20px',
        title: 'Información Personal',
        items: [{
            xtype: 'textfield',
            fieldLabel: 'Nombre'
        }, {
            xtype: 'textfield',
            fieldLabel: 'Apellido'
        }, {
            xtype: 'textfield',
            fieldLabel: 'Dirección de Correo',
            vtype: 'email'
        }]
    });
    var card2 = new Ext.panel.Panel({
        bodyStyle: 'padding: 20px',
        title: 'Información de la Cuenta',
        items: [{
            xtype: 'textfield',
            fieldLabel: 'Usuario'
        }, {
            xtype: 'textfield',
            fieldLabel: 'Clave',
            inputType: 'password'
       }]
    });
    var card3 = new Ext.panel.Panel({
        bodyStyle: 'padding: 20px',
        title: 'Creación de la Cuenta Exitosa!',
        html: 'Exito!'
    });
    var panel = Ext.create('Ext.panel.Panel', {
        title: 'Asistente de Creación de Cuentas - Layout Ficha',
        width: 350,
        height: 300,
        layout: 'card',
        renderTo: Ext.getBody(),
        items: [card1, card2, card3],
        bbar: ['->',
            {
            xtype: 'button',
            text: 'Anterior',
            handler: function(btn){
                var layout = panel.getLayout();
                if (layout.getPrev()) {
                    layout.prev();
                }
            }
            }, {
                xtype: 'button',
                text: 'Siguiente',
                handler: function(btn){
                var layout = panel.getLayout();
                    if (layout.getNext()) {
                        layout.next();
                    }
                }
            }]
    });
});
```

3.7 Crear aplicaciones de pantalla completa con el BorderLayout

El BorderLayout permite crear un distribución de los componentes por regiones (Norte, Sur, Occidente, Centro, Oriente):

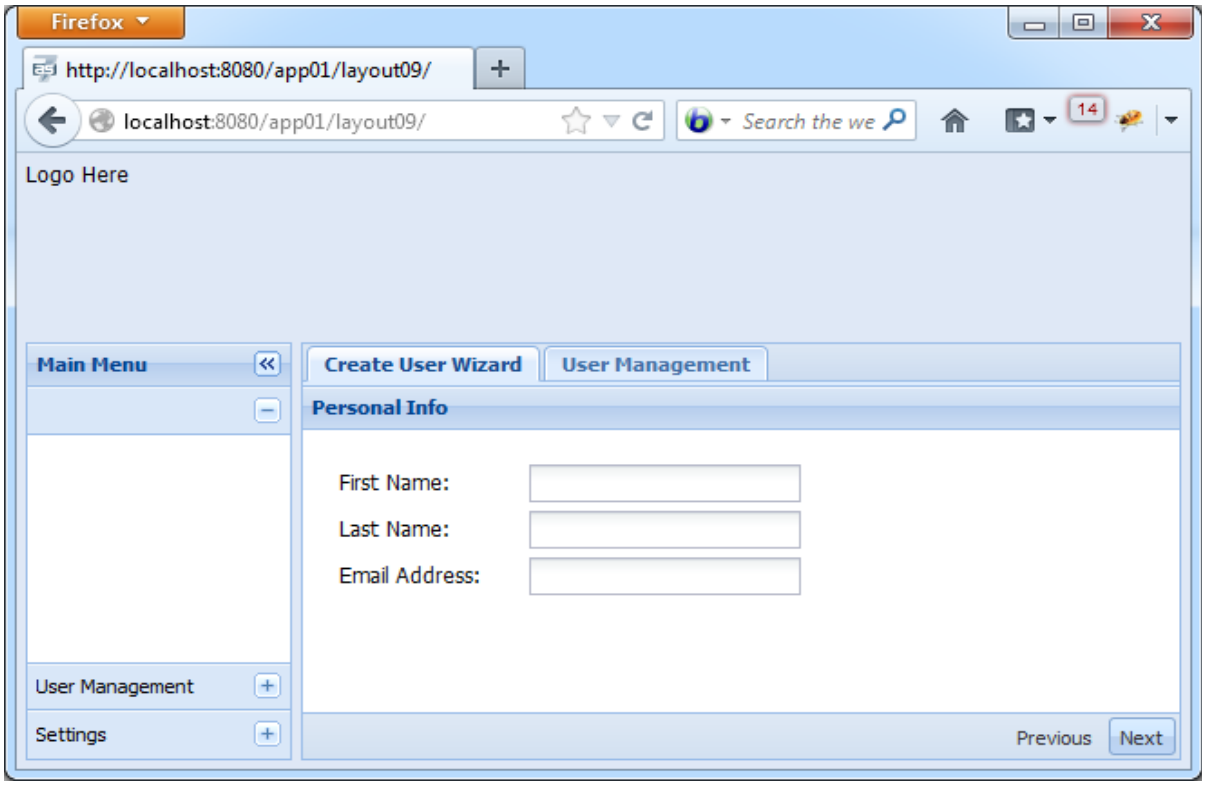


Código

```
Ext.onReady(function () {
    Ext.create('Ext.container.Viewport', {
        layout: 'border',
        items: [{
            region: 'north',
            margins: 5,
            height: 100,
        }, {
            title: 'West - Occidente',
            region: 'west',
            margins: '0 5 0 5',
            flex: .3,
            collapsible: true,
            split: true,
            titleCollapse: true
        }, {
            title: 'Center - Centro',
            region: 'center'
        }, {
            title: 'East - Oriente',
            region: 'east',
            margins: '0 5 0 5',
            width: 200,
            collapsible: true,
            collapsed: true
        }, {
            title: 'South - Sur',
            region: 'south',
            margins: '0 5 5 5',
            flex: .3,
            split: true
        }]
    });
});
```

3.9 Combinando múltiples layouts

Texto



Código

```
Ext.onReady(function () {
	var mainMenu = Ext.create('Ext.panel.Panel', {
		title: 'Main Menu',
		region: 'west',
		margins: '0 5 5 5',
		flex: .3,
		collapsible: true,
		titleCollapse: true,
		layout: 'accordion',
		layoutConfig: {
		animate: false,
		multi: true
		},
		items: [{

		}, {
			title: 'User Management'
		}, {
			title: 'Settings'
		}]
	});

	var card1 = new Ext.panel.Panel({
		bodyStyle: 'padding: 20px',
		title: 'Personal Info',
		border: false,
		items: [{
			xtype: 'textfield',
			fieldLabel: 'First Name'
		}, {
			xtype: 'textfield',
			fieldLabel: 'Last Name'
		}, {
			xtype: 'textfield',
			fieldLabel: 'Email Address',
			vtype: 'email'
		}]
	});
	var card2 = new Ext.panel.Panel({
		bodyStyle: 'padding: 20px',
		title: 'Account Info',
		border: false,
		items: [{
			xtype: 'textfield',
			fieldLabel: 'Username'
		}, {
			xtype: 'textfield',
			fieldLabel: 'Password',
			inputType: 'password'
		}]
	});
	var card3 = new Ext.panel.Panel({
		bodyStyle: 'padding: 20px',
		title: 'Account Creation Successful!',
		border: false,
		html: 'Success!'
	});
```

```
var createUserWizard = Ext.create('Ext.panel.Panel', {
	title: 'Create User Wizard',
	layout: 'card',
	deferredRender: true,
	items: [card1, card2, card3],
	bbar: ['->', {
		xtype: 'button',
		text: 'Previous',
		handler: function(btn){
			var layout = cardPanel.getLayout();
			if (layout.getPrev()) {
			layout.prev();
			}
		}
	}, {
		xtype: 'button',
		text: 'Next',
		handler: function(btn){
			var layout = cardPanel.getLayout();
			if (layout.getNext()) {
			layout.next();
			}
		}
	}]
});

var userManagementPanel = Ext.create('Ext.panel.Panel', {
	title: 'User Management',
	layout: {
		type: 'hbox',
		align: 'stretch',
		padding: 10
	},
	defaults: {
		flex: 1
	},
	items: [{
		xtype: 'container',
		margins: '0 5 0 0',
		layout: {
			type: 'vbox',
			align: 'stretch',
			animate: true
		},
		defaults: {
			flex: 1,
			frame: true
		},
		items: [{
			title: 'User Contact Information',
			margins: '0 0 5 0'
		}, {
			title: 'Session Log'
		}]
	}, {
		xtype: 'container',
		layout: {
			type: 'vbox',
			align: 'stretch',
			animate: true
```

```
                        },
                        defaults: {
                                flex: 1,
                                frame: true
                        },
                        items: [{
                                title: 'Account Privileges',
                                margins: '0 0 5 0'
                        }, {
                                title: 'Purchase History',
                        }]
                }]
        });

        var contentPanel = Ext.create('Ext.tab.Panel', {
                region: 'center',
                margins: '0 5 5 0',
                items: [createUserWizard, userManagementPanel]
        });

        Ext.create('Ext.container.Viewport', {
                layout: 'border',
                items: [{
                        region: 'north',
                        margins: 5,
                        height: 100,
                        xtype: 'container',
                        html: 'Logo Here'
                }, mainMenu, contentPanel]
        });

});
```