Sencha Touch 2

Controladores

Desarrollo de Aplicaciones Multiplataforma

Introducción

- Los controladores son las piezas de unión entre vistas, modelo y almacenamiento (stores).

- Sus principales funciones son:

    - Reaccionar ante eventos.

    - Tomar decisiones.

    - Cambiar el estado de la aplicación.

Introducción

- Los manejadores de eventos (event handlers) nos permiten reaccionar ante eventos de controles de nuestra aplicación.

- Pueden situarse en la configuración de cualquier control utilizando la etiqueta ***listeners***.

- Esta forma de manejar los eventos es poco mantenible y el código no es reutilizable.

- Por ello vamos a ubicarlos dentro de los controladores.

Creación

- Podemos tener tantos controladores como queramos.
- Los controladores son instancias de ***Ext.app.Controller***.

```
Ext.define('Aplicacion.controller.IndexController', {
    extend: 'Ext.app.Controller',
    config: {
        refs: {},
        control: {},
        routes: {}
    }
    // Manejadores de eventos
});
```

Creación

- Los controladores se ubican en la carpeta ***app/controller***.
- La sección ***config*** tiene las siguientes etiquetas:
  - **refs**: contiene un listado de elementos de la interfaz de usuario. Se invocan con la sintaxis **Query** y automáticamente se crea un **getter**.
  - **control**: relaciona una o más de las entradas de ***refs*** con los manejadores de eventos.
  - **routes**: provee enrutamiento a la aplicación.

Enlazado

- Para que los controladores se enlacen automáticamente al inicio de la aplicación debemos incluirlos en la llamada ***Ext.application()***.

```
Ext.application({
        name: 'Aplicacion',
        models: //…
        views: //…
        controllers: ['IndexController'],
        stores: //…
        launch: function() {

        }
});
```

Ejemplo

```
Ext.define('Aplicacion.controller.IndexController', {
        extend: 'Ext.app.Controller',
        config: {
                refs: {
                        listView: 'indexview > #listView'
                },
                control: {
                        listView: {
                                itemtap: 'pulsacion'
                        }
                },
                pulsacion: function(list, index, target, record, e, eOpts) {
                        console.log('Elemento de lista pulsado: ' + record.get('text'));
                }
        }
});
```

Ejemplo

- En la sección ***refs*** utilizamos una expresión para obtener la lista dentro del componente de *xtype indexview*.

- El identificador ***listView*** es arbitrario, podemos poner cualquier identificador válido en JavaScript. Nos permite dar un nombre a la expresión.

- En la ***sección*** control indicamos que para los elementos identificador por listView queremos gestionar el evento de pulsación mediante la función ***pulsacion***.

- Por último implementamos la función ***pulsacion***.

```
Ext.define('Aplicacion.controller.IndexController', {
	extend: 'Ext.app.Controller',
	config: {
		refs: {
			indexView: 'indexview',
			listView: 'indexview > #listView'
		},
		control: {
			listView: {
				itemsingletap: 'pulsacion',
				itemdoubletap: 'doblePulsacion'
			},
			indexView: {
				whatever: 'seleccionado'
			}
		},
		pulsacion: function(list, index, target, record, e, eOpts) {
			console.log('Elemento de lista pulsado: ' + record.get('text'));
		},
		doblePulsacion: function(list, index, target, record, e, eOpts) {
			console.log('Elemento de lista pulsado: ' + record.get('text'));
		},
		seleccionado: function(view) {
			console.log('Selección');
		}
	}
});
```

Inicialización

- A veces es necesario inicializar los controladores antes de uso.

- Para ello disponemos de las funciones ***init()*** y ***launch()***.

```
Ext.define('Aplicacion.controller.IndexController', {
        extend: 'Ext.app.Controller',
        config: {
                //…
        },
        init: function() {

        },
        launch: function() {

        }
});
```

Inicialización

- Cuando una aplicación Sencha comienza su ejecución se ejecutan los controladores en orden:
    - Todos los controladores ejecutan su método ***init()***.
    - Si existen perfiles, el perfil seleccionado ejecuta su función ***launch()***.
    - Después, el método ***launch()*** de ***app.js*** se ejecuta.
    - Por último se ejecutan los métodos ***launch()*** de cada uno de los controladores.

Enrutamiento

- A través del enrutamiento podemos usar URLs para mantener el estado de las aplicaciones.

- Esto nos permite utilizar el botón atrás del navegador y el historial.

- Podemos compartir las URLs manteniendo el estado de la aplicación.

Enrutamiento

```
Ext.define('Aplicacion.controller.IndexController', {
    extend: 'Ext.app.Controller',
    config: {
        refs: {
        },
        routes: {
            'usuarios': 'mostrarUsuarios'
        }
    },
    mostrarUsuarios: function() {
        var widget = Ext.widget('usuarios', {
            title: 'Usuarios'
        });

        this.showWidget(widget);
    }
});
```

Perfiles

- Sencha Touch nos permite ofrecer aplicaciones con experiencias de usuario diferenciadas en función de la categoría del dispositivo.

- Los perfiles se implementan a través de la clase ***Ext.app.Profile***. Las aplicaciones usan esta clase en el fichero *app.js* como entradas de la propiedad ***profiles***.

```
Ext.application({
        name: 'Aplicacion',
        profiles: ['Phone', 'Tablet', 'Desktop'],
        //…
        launch: function() {
                //…
        }
});
```

Perfiles

- Cuando Sencha arranca, carga automáticamente los ficheros ***app/profiles/Phone.js***, ***app/profiles/Tablet.js*** y ***app/profiles/Desktop.js***
- Cada uno de estos ficheros tiene que tener una clase de nombre similar y que herede de ***Ext.app.Profile***:

```
Ext.define('Aplicacion.profile.Phone', {
        extend: 'Ext.app.Profile',
        config: {
                name: 'EjemploTelefono'
        },
        isActive: function() {
                return Ext.os.is.Phone;
        },
        launch: function() {
                //…
        }
});
```

Perfiles

- Los perfiles deben implementar el método ***isActive*** que determina si aplica o no el perfil.

- Los perfiles se cargan todos a la vez y se invocan los métodos ***isActive*** en cualquier orden. El primero en devolver *true* será el que se aplique, y el resto dejan de evaluarse.