Ejercicios - Aspectos avanzados de Sencha Touch

Índice

En esta sesión vamos a continuar con el ejercicio del editor de notas de la sesión anterior. En primer lugar añadiremos los elementos necesarios para poder guardar las notas y visualizarlas en un listado. También crearemos el formulario y los botones necesarios para crear, editar y borrar notas.

Si necesitas ayuda puedes descargar la plantilla para los ejercicios sesion06-ejercicios.zip. Al descomprimirlo aparecerán tres carpetas para cada uno de los ejercicios.

1. Ejercicio 1 - Modelo de datos y Listado

En este primer ejercicio vamos a crear el modelo de datos a utilizar y el almacén donde se van a guardar. Además crearemos una primera versión del listado con datos de prueba.

Nuestro modelo de datos (con identificador 'modeloNotas') tendrá cuatro campos: 'id' de tipo 'int', 'date' de tipo 'date' y formato 'c', 'title' de tipo 'string' y 'text' de tipo 'string'. Además deberemos definir 'id' como identificador único (ver sección "Data Model") y las siguientes validaciones: los campos 'id', 'title' y 'text' serán requeridos, y para los campos 'title' y 'text' modificaremos el mensaje de error por defecto, poniendo "Introduzca un título/texto".

A nuestro almacén de datos le pondremos como identificador 'storeNotas' y le indicaremos que tiene que usar el modelo 'modeloNotas'. Como proxy vamos a usar el almacenamiento en local, indicando como identificador 'misNotas-app-localstore'. Además indicaremos que se ordenen los datos de forma descendente (DESC) por fecha (campo 'date'), y que se carguen los datos del repositorio al inicio (`autoLoad: true`).

De forma temporal y para poder ver los resultados vamos a insertar datos en el Store, añadiendo las siguientes líneas:

```
data: [
        { id: 1, date: new Date(), title: 'Test 1', text: 'texto de
prueba' },
        { id: 2, date: new Date(), title: 'Test 2', text: 'texto de
prueba' },
        { id: 3, date: new Date(), title: 'Test 3', text: 'texto de
prueba' },
        { id: 4, date: new Date(), title: 'Test 4', text: 'texto de
prueba' }
]
```

Por último vamos a crear el listado donde se visualizarán los datos. Le pondremos como identificador 'panelLista' y le diremos que utilice el store 'storeNotas' que hemos definido previamente.

Además le indicaremos en el `itemTpl` que muestre el campo `{title}` dentro de una capa con el estilo "list-item-title", y que el campo `{text}` lo muestre a continuación en otra capa que use el estilo "list-item-text". Este listado lo tendremos que añadir a nuestro panel 'panelContenedorLista' en su sección `items` para poder visualizarlo.

Ahora tenemos que añadir esos estilos al fichero *app.css*. Para ambas clases definiremos los mismos estilos (ver código siguiente), salvo para el "list-item-text" que añadiremos el color gris al texto.

```
        float:left;
        width:100%;
        font-size:80%;
        white-space: nowrap;
        overflow: hidden;
        text-overflow: ellipsis;
```

2. Ejercicio 2 - Formulario, Crear y Editar notas

En primer lugar vamos a crear el formulario. Para esto editamos el panel 'panelFormulario' que ya teníamos hecho, y le cambiamos su constructor de panel por uno de tipo formulario (ver sección de formularios). Además tenemos que quitar también los campos `layout: fit` y el texto HTML que teníamos puesto de prueba. En este formulario vamos a utilizar dos campos:

- Un campo de texto con el nombre 'title' (debe de coincidir con el modelo), con la etiqueta "Título:" y activaremos la opción de requerido.
- Un área de texto con nombre 'text', etiqueta "Texto:" y que también sea requerido.

Ahora vamos a modificar la función `handler` del botón "Nueva nota" para que al pulsarlo, **antes de realizar la transición**, cree una nueva nota y la asigne al formulario. Para esto vamos a añadir una llamada a la función `crearNuevaNota()`, la cual crearemos de forma separada.

En la función `crearNuevaNota()` en primer lugar nos guardaremos la fecha actual `var now = new Date();`, y a continuación obtendremos el identificador del registro a crear `var noteID = now.getTime();` (*con esta función transformamos la fecha en milisegundos, con lo que obtenemos un número que no se repite que podemos usar como ID*). A continuación creamos un registro del modelo 'modeloNotas' (ver sección "Data Model") y lo cargamos en nuestro formulario (`panelFormulario.load( note );`).

Por último vamos a añadir la funcionalidad de editar las notas creadas. Para esto vamos hasta el 'panelLista', y definimos su función `onItemDisclosure`. Esta función recoge un parámetro (*record*) que tenemos que cargar en el 'panelFormulario' (`panelFormulario.load( record );`). A continuación realizaremos una transición de tipo 'slide' hacia la izquierda y con una duración de 1 segundo, para mostrar el 'panelFormulario' (ver sección "Listados").

3. Ejercicio 3 - Guardar y Borrar notas

En este último ejercicio vamos a añadir las acciones de guardar y borrar nota. En las

funciones `handler` de los botones "Guardar" y "Borrar" añadiremos llamadas a las funciones `guardarNota()` y `borrarNota()` respectivamente. Las llamadas a estas funciones las deberemos de incluir antes de realizar la transición. A continuación definiremos las acciones a realizar en estas funciones (que estarán definidas de forma separada).

En la función `guardarNota()` realizaremos los siguientes pasos:

1. En primer lugar tendremos que cargar los datos introducidos en el formulario (usaremos la función `getRecord()` sobre la variable que contiene el formulario), y a continuación actualizaremos la instancia del formulario (ver sección "Guardar los datos de un formulario").
2. En segundo lugar comprobaremos si hay algún error de validación y mostraremos los errores (ver apartado "Comprobar validaciones" de la sección "Formularios").
3. Una vez validado el registro obtenido procederemos a guardar los datos. Obtenemos el *Store* usado por el listado y añadimos el registro solo si este no está repetido (función `findRecord()`, ver el apartado "Guardar los datos de un formulario").
4. Por último actualizamos el *Store* (`sync()`), lo reordenamos por fecha, y refrescamos el listado (`refresh()`).

La función `borrarNota()` es más sencilla:

1. Asignamos a variables el registro actual del formulario (`getRecord()`) y el *Store* usado por el listado (`getStore()`).
2. A continuación comprobaremos si existe el registro actual en el *store* (usando su "id") y si es así lo eliminaremos (ver apartado "Eliminar registros" de la sección "Data Store").
3. Por último actualizaremos los datos del *Store* (`sync()`) y refrescamos el listado (`refresh()`).