Shared Objects

Almacenamiento persistente de datos

Cuando cerramos una aplicación SWF, todos los objetos y variables se borran de la memoria del dispositivo. Para poder guardar datos y recuperarlos posteriormente se emplean los *shared objects*, unos pequeños ficheros de texto similares a las cookies de los navegadores Web. El espacio de almacenamiento para cada archivo SWF está limitado por el dispositivo. Puede determinarse utilizando el método `getMaxSize()` de la clase `SharedObject`. Lo bueno de los shared objects es que podemos almacenar prácticamente cualquier tipo de datos: `Number`, `String`, `Array`, `XML`, `Date`, `Boolean`, etc.

Consideraciones

En Flash Player (la versión de escritorio) la clase `SharedObject` permite que varios archivos SWF puedan compartir los datos guardados. Sin embargo, Flash Lite no permite compartir datos entre distintos archivos SWF.

En Flash Lite, se considera que un archivo SWF es de una versión diferente si se ha modificado desde su versión original, aunque conserve el mismo nombre. No ocurre lo mismo en Flash Player, donde se considera que un archivo SWF es igual si su URL y su nombre no cambian, aunque el archivo SWF sí se haya modificado. En Flash Lite, dos versiones distintas del mismo SWF no pueden acceder a los shared objects de cada una.

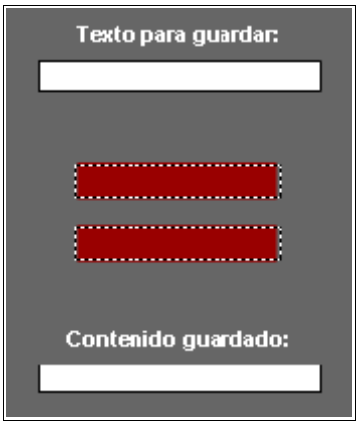
Los shared objects de Flash Lite sólo están disponibles en archivos SWF almacenados localmente. Los SWF que se reproducen desde la red no pueden utilizar objetos compartidos de Flash Lite < 2.

La lectura y escritura de los datos en un teléfono puede ser un proceso lento, dependiendo de los bytes que se vayan a escribir. Para garantizar la disponibilidad inmediata de los datos cuando la aplicación los solicite al dispositivo, necesitamos configurar un detector de eventos. El reproductor invoca al detector cuando el dispositivo ha cargado los datos del shared object. Los métodos que acceden a la instancia de `SharedObject` devuelta por la llamada a `getLocal()` tendrán que esperar hasta que se invoque al detector antes de poder realizar ninguna operación.

Aplicación

Vamos a ilustrar el uso de SharedObjects con un sencillo ejemplo. Se tiene un texto de entrada (`inputSo`) donde escribir cualquier palabra o frase y dos botones: uno para guardar y otro para cargar el texto guardado. En un segundo campo de texto dinámico (`contentSo`) se mostrará el contenido del shared object y los mensajes de información.

Los botones son dos instancias del mismo movieClip –- para reutilizar símbolos y optimizar así la aplicación en caso de ampliarla –-, el cual es un rectángulo y un campo de texto (`label_txt`).



Estos botones los llamaremos `insertBtn` y `testBtn`, respectivamente, donde escribir la etiqueta de cada uno de los botones:

```
insertBtn.label_txt.text = "GUARDAR";
testBtn.label_txt.text = "CARGAR";
```

La creación del SharedObject se realiza así:

```
var nombre:String = "unNombre";
var so:SharedObject =                    →
SharedObject.getLocal(nombre);
```

Con esto, si no existía antes, hemos creado un fichero llamado **unNombre.sol** en la memoria de almacenamiento del dispositivo. Si ya existía antes, entonces veamos qué hacer:

```
function so_listener
(the_so:SharedObject):Void {
 var que:Number = the_so.getSize();
 if (que > 0) {
  contentSo.text = "hay "+ que + " bytes";
  SharedObject.removeListener(nombre);
 } else contentSo.text = "no hay datos";
};
SharedObject.addListener(nombre,         →
so_listener);
```

Para escribir cualquier dato se usa la orden `xx_so.data.algo`, donde `algo` es la variable que queremos alojar y `xx_so` es el shared object que habíamos creado.

Así, por último, sólo falta asignar las acciones a cada botón:

```
insertBtn.onRelease = function():Void {
 if (que > 0) so.clear();
 so.data.maVariable = inputSo.text;
 so.flush();
```

```
 contentSo.text = inputSo.text;
};
testBtn.onRelease = function():Void {
 contentSo.text = so.data.maVariable;
};
```

Existe un ejemplo muy recomendable en la carpeta **Samples and Tutorials** del directorio de instalación de Flash8, donde podemos emplear una clase más específica para dispositivos móviles y de fácil manejo.

Si queremos más flexibilidad podemos usar el framework KuneriLite (http://www.kunerilite.net) o usar la aplicación **write2file**, escrita en C++, que se integra perfectamente con Flash Lite:

http://www.scriptamanentgroup.net/byte/?p=144