Programación Orientada a Objetos

Ejemplos de uso de clases en Flash Lite

Desde la versión 2, ActionScript es un lenguaje totalmente orientado a objetos. En esta práctica vamos a realizar unos sencillos ejemplos que ilustren la filosofía de trabajo basada en clases. Las clases son librerías externas que agrupan propiedades (variables) y métodos (funciones). La gran ventaja es que al trabajar con clases podemos reutilizarlas de un proyecto a otro, y las modificaciones que hagamos sobre una clase afectarán a todas las aplicaciones basadas en dicha clase; imaginemos un proyecto de N ficheros SWF cada uno con una función interna idéntica en todos ellos, sería una locura modificar N veces la misma función.

Creación de clases

Vamos a crear una clase que se llame `Persona`. Lo que hará esta clase será crear instancias con un nombre determinado, el cual podremos modificar mediante el atributo de clase `nombre`. Para estas pruebas crearemos un campo de texto en nuestra aplicación, llamado `estado_txt`.

Así, comenzamos creando un nuevo fichero de texto, o, desde el mismo IDE Flash, creando un nuevo *ActionScript File*.

Todas las clases se declaran así en ActionScript:

```
class NombreDeLaClase {
 // definición de la clase
}
```

Nótese que el nombre de la clase empieza por una letra mayúscula (convención usual), y el fichero de texto debe guardarse tal y como se ha llamado a la clase, comenzando en letra mayúscula también, con la extensión AS (**.as**).

Siguiendo con nuestro ejemplo, vamos a definir una propiedad y un método constructor para nuestra clase `Persona`. El constructor es necesario para crear posteriormente instancias de la clase. Se llama así porque cada vez que usemos en nuestra aplicación la instrucción `new Persona`, se creará un objeto `Persona`, que es una instancia de dicha clase.

La propiedad será el nombre, una variable del tipo `String` y pública, porque vamos a acceder a ella desde fuera del ámbito de la clase (desde nuestra aplicación). Así, escribimos dentro de la definición de clase:

```
public var nombre:String;
```

Ahora escribiremos el constructor, una función pública[1] que tendrá como argumento el nombre de la persona creada:

```
public function Persona(nom:String) {
 // definición del constructor
}
```

1 En ActionScript los constructores de clase deben ser públicos. Tampoco pueden devolver un valor – ni siquiera `Void`.

Llamamos al argumento `nom` para no confundirlo con la propiedad pública `nombre` anteriormente definida. Para acceder a la propiedad `nombre` desde dentro de la clase usaremos el apuntador `this`. La función constructora quedará así:

```
public function Persona(nom:String) {
 this.nombre = nom;
 trace("Una nueva Persona ha nacido");
};
```

Ahora vamos a escribir dos métodos para esta clase. Uno será público, pues se podrá llamar desde fuera de la clase, y otro privado (sólo podrá ser invocado desde la propia clase).

```
public function saludar():String {
 checkName();
 return "saluda " + this.nombre;
};
private function checkName():Void {
 if (this.nombre == undefined)          →
this.nombre = "sin nombre";
};
```

Empleo de clases en la aplicación

Una vez creado el fichero **Persona.as**, en el mismo directorio que nuestra nueva aplicación Flash Lite, podemos utilizar la clase en el editor de código. La importamos y creamos dos objetos, `per1` y `per2`, del tipo `Persona`:

```
import Persona;
var per1:Persona = new Persona("Pepe");
var per2:Persona = new Persona;
```

También podríamos haber hecho la asignación e instanciación de los dos objetos así:

```
var per1,per2:Persona;
per1 = new Persona();
per2 = new Persona("Pepe");
```

Ahora haremos unas pruebas:

```
estado_txt.text += per1.nombre + "\n";
estado_txt.text += per1.saludar() + "\n";
estado_txt.text += per2.nombre + "\n";
estado_txt.text += per2.saludar() + "\n";
per2.nombre = "María";
estado_txt.text += per2.saludar() + "\n";
```

Al publicar nuestra aplicación veremos en la ventana de Salida estas líneas:

```
Una nueva Persona ha nacido
Una nueva Persona ha nacido
```

Y en el campo de texto de nuestra aplicación:

```
Pepe
saluda Pepe
undefined
saluda sin nombre
saluda María
```

Y aquí termina nuestro sencillo ejemplo para explicar las bases de la programación orientada a objetos en ActionScript.

Hay mucho todavía por conocer, ya que sólo hemos hecho referencia a las cosas más básicas. Existen mecanismos como la herencia y el polimorfismo, que son unas de las posibilidades más potentes de la programación orientada a objetos.