

# Desarrollo de Apps para iOS Persistencia

IWEB,LSWC 2013-2014

Santiago Pavón

ver: 2013.11.09

# Persistencia

- Conservar datos
  - Aunque se pare y relance la aplicación.
  - Aunque apague y encienda el terminal.
- Existen varias formas de guardar datos:
  - User Defaults (preferencias de usuario)
  - Sistema de Ficheros
  - SQLite3
  - Core Data
  - Cloud

# Preferencias de Usuario

# Preferencias de Usuario

- Las preferencias de usuario son valores persistentes usados por la aplicación.
- Pueden modificarse:
  - desde la propia aplicación.
  - desde la aplicación **Ajustes (Settings)**.
- Para poder modificar las preferencias desde **Ajustes**:
  - La aplicación debe tener un **settings bundle**.
    - conjunto de ficheros describiendo los datos de preferencias.
  - **Ajustes** crea un GUI para editar los datos.
- **NSUserDefaults**
  - Es la clase usada para almacenar/recuperar los valores de las preferencias.
  - Cada valor está asociado a una clave.

# Acceder desde nuestra aplicación

- **NSUserDefaults** implementa un singleton

```
NSUserDefaults *def =  
[NSUserDefaults standardUserDefaults];
```

- Pueden guardarse combinaciones de:
  - NSData, NSString, NSNumber, NSDate, NSArray o NSDictionary.
- Se usa como un diccionario:
  - para obtener datos:
    - **objectForKey:**    **intForKey:**    **boolForKey:** ...
  - para salvar datos:
    - **setObject:forKey:**    **setFloat:forKey:** ...
  - Antes de salir invocar **synchronize** para salvar datos de la cache.

# Cuando cargar / salvar datos

- Cuando la pantalla va a mostrarse o ocultarse.
  - viewWillAppear: viewWillDisappear:
- Al cargar una pantalla en memoria.
  - viewDidLoad
- Cuando la aplicación vaya a terminar:
  - Apuntarse para recibir una notificación del centro de notificaciones (NSNotificationCenter).
- ...

# Ejemplo

- Cuando se carga el VC recupero los valores de las preferencias:

```
- (void)viewDidLoad {
    [super viewDidLoad];

    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    self.carModel = [defaults objectForKey:@"model"];
    self.carSpeed = [defaults floatForKey:@"speed"];
}
```

- Salvo las preferencias cuando la pantalla desaparece:

```
- (void) viewWillDisappear:(BOOL)animated {
    [super viewWillDisappear:animated];

    NSUserDefaults *defaults = [NSUserDefaults standardUserDefaults];
    [defaults setObject:self.carModel forKey:@"model"];
    [defaults setFloat:self.carSpeed forKey:@"speed"];
    [defaults synchronize]; // Forzar la sincronización ahora
}
```

# Ejemplo

- Usar notificaciones:

- Cuando la aplicación va a terminar, se envía a **self** el mensaje **appWillTerminate**.

```
- (void)viewDidLoad
{
    UIApplication *app = [UIApplication sharedApplication];
    [[NSNotificationCenter defaultCenter]
        addObserver:self
        selector:@selector(appWillTerminate:)
        name:UIApplicationWillTerminateNotification
        object:app];
    ...
}
```

- El método registrado para la notificación:

```
- (void)appWillTerminate:(NSNotification*)notification {
    // Salvar los datos aqui.
    [[NSUserDefaults standardUserDefaults] synchronize];
}
```

# Acceso desde la Aplicación Ajustes

- Las preferencias de usuario pueden editarse desde la aplicación **Ajustes (Settings)**.
  - Solo algunos tipos de valores.
- Para ello, hay que crear en nuestra aplicación un **Settings Bundle**:  
    New File > iOS > Resource > Settings Bundle
  - Para manipular el contenido de este fichero hay que usar Finder.
- **Root.plist**
  - define la primera vista de las preferencias.
- Para crear subvistas adicionales (nuevas pantallas) hay que crear nuevos ficheros de listas de propiedades.
  - Un fichero plist para cada pantalla adicional.
- Consultar la guía:
  - **Preferences and Settings Programming Guide: About Preferences and Settings.**

The image shows three sequential screenshots of an iPhone displaying a custom settings application. The device has a black case and a home button at the bottom.

**Screenshot 1: Preferencias**

- Carrier: WiFi
- Time: 3:02 PM
- Battery: Low
- Section: PRIMER GRUPO
  - Nombre: Santiago
  - Password: ••••••••
- Section: SEGUNDO GRUPO
  - Desayuno: Churros >
  - Frio: A slider switch is turned on.
  - A temperature slider with a blue bar is set to the middle.
- Section: TERCER GRUPO
  - Mas cosas >

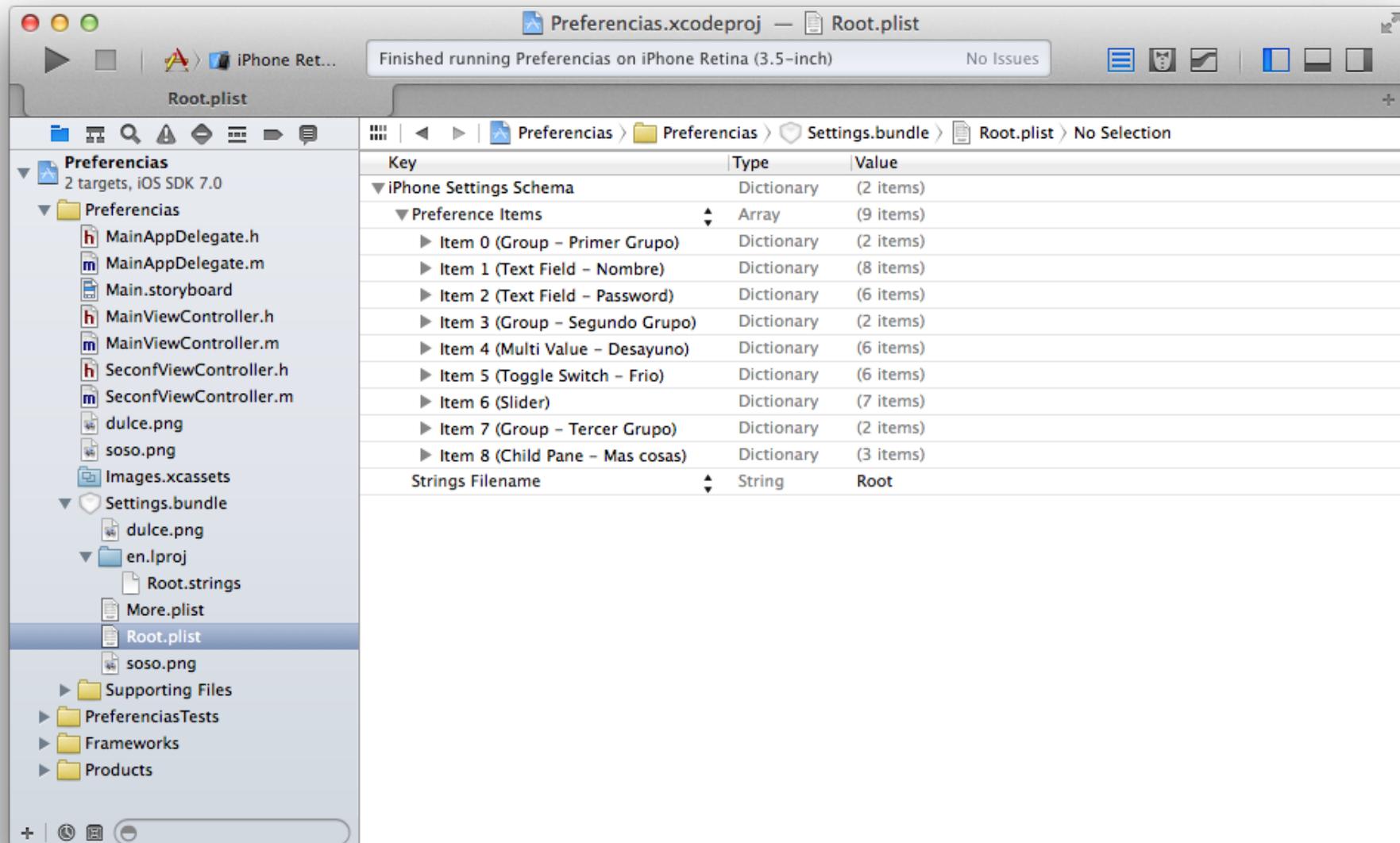
**Screenshot 2: Preferencias - Mas cosas**

- Carrier: WiFi
- Time: 3:02 PM
- Battery: Low
- Section: MARCAS
  - Leche: Pascual
  - Cafe: Nescafe
  - Galletas: Cuetara
  - Cereales: Krispis
- Section: Tipo de churros: Churro de lazo >

**Screenshot 3: Tipo de churros**

- Carrier: WiFi
- Time: 3:03 PM
- Battery: Low
- List of churro types:
  - Porra recta
  - Porra curvada
  - Churro recto
  - Churro de lazo
  - Porra con crema
  - Porra con chocolate
  - Porra light

# Root.plist



Preferencias.xcodeproj — Root.plist

Finished running Preferencias on iPhone Retina (3.5-inch) No Issues

Root.plist

Preferencias

2 targets, iOS SDK 7.0

Preferencias

MainAppDelegate.h

MainAppDelegate.m

Main.storyboard

MainViewController.h

MainViewController.m

SeconfViewController.h

SeconfViewController.m

dulce.png

soso.png

Images.xcassets

Settings.bundle

dulce.png

en.lproj

Root.strings

More.plist

Root.plist

soso.png

Supporting Files

PreferenciasTests

Frameworks

Products

Root.plist

Preferencias > Preferencias > Settings.bundle > Root.plist > No Selection

Key	Type	Value
iPhone Settings Schema	Dictionary	(2 items)
Preference Items	Array	(9 items)
Item 0 (Group – Primer Grupo)	Dictionary	(2 items)
Title	String	Primer Grupo
Type	String	Group
Item 1 (Text Field – Nombre)	Dictionary	(8 items)
Type	String	Text Field
Title	String	Nombre
Identifier	String	username
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection
Default Value	String	
Text Field Is Secure	Boolean	NO
Keyboard Type	String	Alphabet
Item 2 (Text Field – Password)	Dictionary	(6 items)
Type	String	Text Field
Title	String	Password
Identifier	String	password
Autocapitalization Style	String	None
Autocorrection Style	String	No Autocorrection
Text Field Is Secure	Boolean	YES
Item 3 (Group – Segundo Grupo)	Dictionary	(2 items)
Item 4 (Multi Value – Desayuno)	Dictionary	(6 items)
Item 5 (Toggle Switch – Frio)	Dictionary	(6 items)
Item 6 (Slider)	Dictionary	(7 items)

Preferencias.xcodeproj — Root.plist

Finished running Preferencias on iPhone Retina (3.5-inch) No Issues

Root.plist

Preferencias

2 targets, iOS SDK 7.0

Preferencias

MainAppDelegate.h

MainAppDelegate.m

Main.storyboard

MainViewController.h

MainViewController.m

SeconfViewController.h

SeconfViewController.m

dulce.png

soso.png

Images.xcassets

Settings.bundle

dulce.png

en.lproj

Root.strings

More.plist

Root.plist

soso.png

Supporting Files

PreferenciasTests

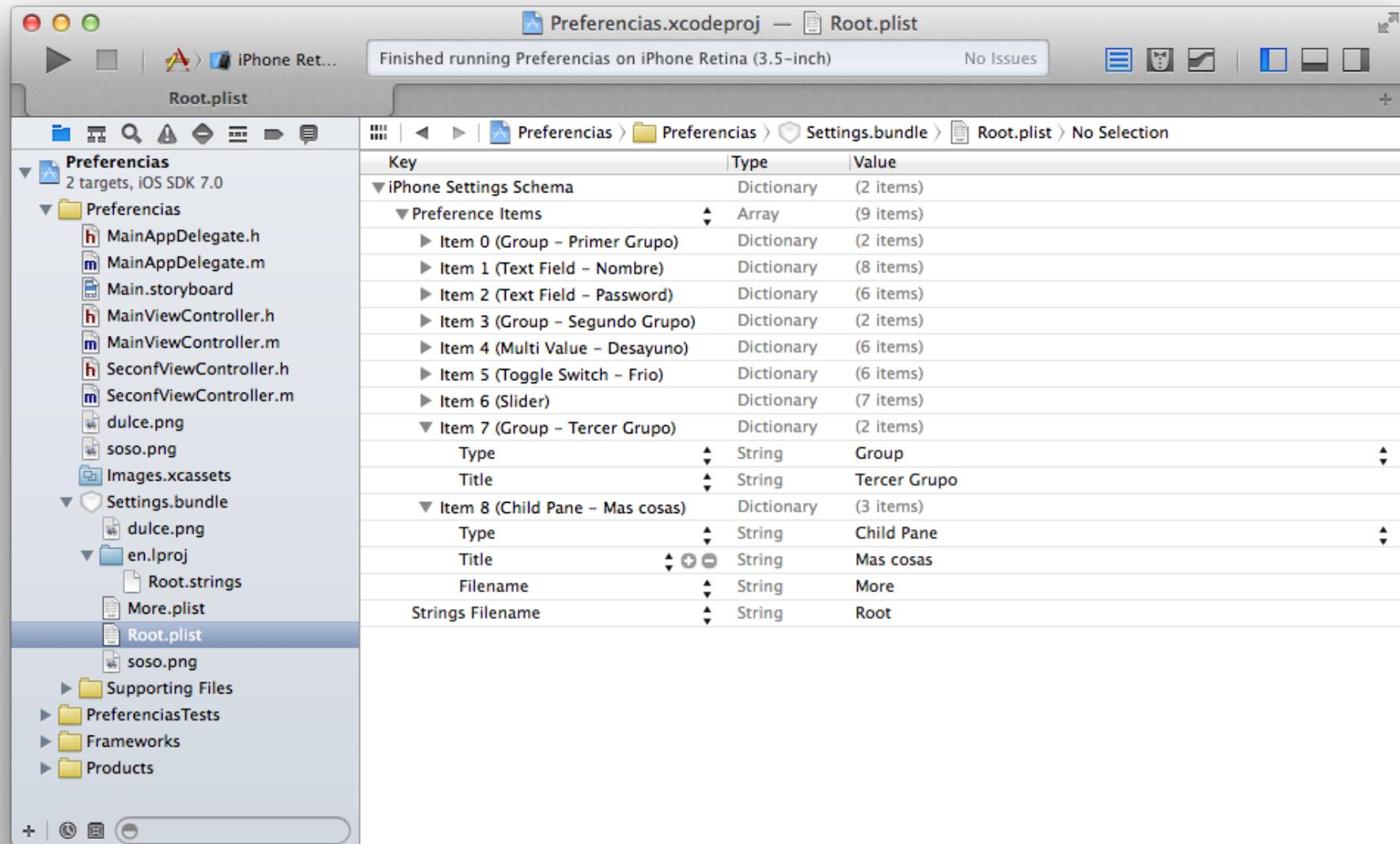
Frameworks

Products

Root.plist

Preferencias > Preferencias > Settings.bundle > Root.plist > No Selection

Key	Type	Value
Item 3 (Group – Segundo Grupo)	Dictionary (2 items)	
Type	String	Group
Title	String	Segundo Grupo
Item 4 (Multi Value – Desayuno)	Dictionary (6 items)	
Type	String	Multi Value
Title	String	Desayuno
Identifier	String	breakfast
Default Value	String	galletas
Values	Array (5 items)	
Titles	Array (5 items)	
Item 5 (Toggle Switch – Frio)	Dictionary (6 items)	
Type	String	Toggle Switch
Title	String	Frio
Identifier	String	temperatura
Value for ON	String	frio
Value for OFF	String	caliente
Default Value	Boolean	NO
Item 6 (Slider)	Dictionary (7 items)	
Type	String	Slider
Identifier	String	azucar
Default Value	Number	5
Maximum Value	Number	10
Max Value Image Filename	String	dulce.png
Minimum Value	Number	1
Min Value Image Filename	String	soso.png



# More.plist

The screenshot shows the Xcode interface with the project "Preferencias.xcodeproj" open. The main window displays the file "More.plist" located in the "Settings.bundle" folder under the "Preferencias" target. The left sidebar shows the project structure with files like MainAppDelegate.h, Main.storyboard, and various ViewControllers. The right pane shows the plist editor with the following data:

Key	Type	Value
Root	Dictionary	(2 items)
Title	String	More Settings
PreferenceSpecifiers	Array	(6 items)
Item 0	Dictionary	(2 items)
Type	String	PSGroupSpecifier
Title	String	Marcas
Item 1	Dictionary	(5 items)
Type	String	PSTextFieldSpecifier
Title	String	Leche
Key	String	marca_leche
AutocapitalizationType	String	No
AutocorrectionType	String	No
Item 2	Dictionary	(5 items)
Type	String	PSTextFieldSpecifier
Title	String	Cafe
Key	String	marca_cafe
AutocapitalizationType	String	None
AutocorrectionType	String	No
Item 3	Dictionary	(5 items)
Type	String	PSTextFieldSpecifier
Title	String	Galletas
Key	String	marca_galletas
AutocapitalizationType	String	No
AutocorrectionType	String	No
Item 4	Dictionary	(5 items)

Preferencias.xcodeproj — More.plist

Finished running Preferencias on iPhone Retina (3.5-inch) No Issues

More.plist

Preferencias > Preferencias > Settings.bundle > More.plist

Key	Type	Value
Item 4	Dictionary	(5 items)
Type	String	PSTextFieldSpecifier
Title	String	Cereales
Key	String	marca_cereales
AutocapitalizationType	String	None
AutocorrectionType	String	No
Item 5	Dictionary	(6 items)
Type	String	PSMultiValueSpecifier
Title	String	Tipo de churros
Key	String	tipo_churros
DefaultValue	String	porra
Values	Array	(7 items)
Item 0	String	Porra recta
Item 1	String	Porra curva
Item 2	String	Churro recto
Item 3	String	Churro de lazo
Item 4	String	Porra de crema
Item 5	String	Porra de chocolate
Item 6	String	Porra light
Titles	Array	(7 items)
Item 0	String	Porra recta
Item 1	String	Porra curvada
Item 2	String	Churro recto
Item 3	String	Churro de lazo
Item 4	String	Porra con crema
Item 5	String	Porra con chocolate
Item 6	String	Porra light

# Sistema de Ficheros

# Sistema de Ficheros

- Las aplicaciones ven un sistema de ficheros UNIX.
- Las aplicaciones corren en un Sandbox.
  - La ejecución de un programa no daña a otros.
  - Proteger acceso a los datos de una aplicación.
  - Fácil borrar datos al desinstalar una aplicación.
- Contenido del sandbox:
  - directorio del bundle de aplicación. (sólo lectura).
  - directorio Documents. (donde salvar los datos permanentes).
  - directorio de caches. (temporales sin backup de itunes).
  - ...

# Obtener Rutas a Directorios

- Directorio Home:

```
NSString *home = NSHomeDirectory();
```

- Directorio Temporal:

```
NSString *tmp = NSTemporaryDirectory();
```

- Directorio Documents:

```
NSArray *paths =  
    NSSearchPathForDirectoriesInDomains(  
        NSDocumentDirectory,  
        NSUserDomainMask, YES);  
NSString *docs = [paths objectAtIndex:0];
```

# Manipular Rutas

- Consultar la documentación de las clases **NSString** y **NSURL**.

```
NSArray *paths = NSSearchPathForDirectoriesInDomains(  
    NSDocumentDirectory, NSUserDomainMask, YES);  
NSString *docsPath = [paths objectAtIndex:0];  
  
NSString *datosPath = [docsPath stringByAppendingPathComponent:@"g.dat"];  
  
NSString *extension = [datosPath pathExtension];  
NSString *fileName = [[datosPath lastPathComponent]  
    stringByDeletingPathExtension];  
  
NSString *basedir = [datosPath stringByDeletingLastPathComponent];  
  
NSURL* docsURL = [NSURL fileURLWithPath:docsPath];  
  
NSURL *datosURL = [docsURL URLByAppendingPathComponent:@"g.dat"];  
  
NSString *absPath = [datosURL absoluteString];
```

# NSBundle

- Un objeto **NSBundle** representa un lugar del sistema de ficheros.
  - Carpeta donde se guardan recursos, código, etc.
- Las aplicaciones y frameworks son bundles.
- El **main bundle** de una aplicación permite acceder a los recursos que se añadieron en el proyecto.
- Estos objetos están firmados,
  - no puede modificarse.

# Usar el Main Bundle

```
NSBundle *bundle = [NSBundle mainBundle];  
  
NSString *path =[bundle pathForResource:@"pokemons"  
                      ofType:@"plist"];  
  
NSURL *url =[bundle URLForResource:@"pokemons"  
                      withExtension:@"plist"];  
  
NSImage *img =[NSImage imageNamed:@"foto.png"];
```

# NSFileManager

- Proporciona métodos para:
  - ver si un fichero existe.
  - crear y examinar directorios.
  - manipular ficheros: copiar, mover, borrar.
  - comparar ficheros.
  - obtener URL de directorios del sistema.
  - ...
- Consultar la documentación para ver todos los métodos disponibles.

# Ejemplo

- Copiar un fichero del Bundle de la aplicación en el directorio de documentos:

```
// crear un File Manager
NSFileManager *fm = [[NSFileManager alloc] init];

// fichero origen
NSBundle *bundle = [NSBundle mainBundle];
NSURL *origenURL = [bundle URLForResource:@"pokemons"
                                         withExtension:@"plist"];

// destino
NSArray *docsURLs = [fm URLsForDirectory:NSDocumentDirectory
                                         inDomains:NSUserDefaultsDomainMask];
NSURL *docsURL = [docsURLs firstObject];
NSURL *destinoURL = [docsURL URLByAppendingPathComponent:@"pokemons.plist"];

// copiar el fichero
NSError *error;
[fm copyItemAtURL:origenURL
            toURL:destinoURL
           error:&error];

// cargar el fichero copiado
NSMutableDictionary *dic =
    [NSMutableDictionary dictionaryWithContentsOfURL:destinoURL];
```

# Lista de Propiedades

- Es un dato formado por cualquier combinación de los tipos:
  - **NSArray, NSDictionary, NSData, NSString, NSNumber, NSDate.**
- Se pueden guardar y recuperar de ficheros **.plist**.

```
[miArray writeToFile:path1 atomically:YES];
```

```
NSDictionary *miDiccionario =  
[NSDictionary dictionaryWithContentsOfFile:path2];
```

# NSPropertyListSerialization

- La clase **NSPropertyListSerialization** proporciona métodos para:
  - serializar una lista de propiedades en un **NSData**.
    - + **(NSData\*) dataWithPropertyList: (id)plist  
format: (NSPropertyListFormat)format  
options: (NSPropertyListWriteOptions)opt  
error: (NSError\*\*)error**
  - y crear una lista de propiedades desde un **NSData**.
    - + **(id) propertyListWithData: (NSData\*)data  
options: (NSPropertyListReadOptions)opt  
format: (NSPropertyListFormat\*)format  
error: (NSError\*\*)error**
- Los objetos **NSData** se pueden leer y escribir en ficheros usando por ejemplo:
  - **(BOOL)writeToURL: (NSURL\*)aURL atomically: (BOOL)atomically**
  - **(id)initWithContentsOfURL: (NSURL\*)aURL**

# NSCoding

- El protocolo **NSCoding** declara los dos métodos que debe implementar una clase para que sus objetos puedan serializarse (encode) y des-serializarse (decode).
- Una clase debe ser conforme al protocolo **NSCoding**. para poder:
  - Guardar sus objetos en ficheros (encode),
  - Recuperar sus objetos guardados en ficheros (decode).
- Este protocolo se explica en las transparencias de **Introducción a Objective-C**.

# Usar funciones I/O de C

```
FILE *fp;
if ( (fp = fopen(path,"r")) == NULL) {
    NSLogs( "No puedo abrir el fichero de iconos");
    return;
}
char line[1024];
int code;
char name[1024];
while ( fgets(line,1024,fp) != NULL) {
    sscanf(line,"%i %s", &code, name);
    NSString* key = [NSString stringWithFormat:@"%@",code];
    NSString* val = [NSString stringWithFormat:@"%@",name];
    [dictionary setValue:val forKey:key];
}
fclose(fp);
```

# SQLite3

# SQLite3

- iOS incluye soporte para esta base de datos SQL.
- Es una BD contenida en un único fichero.
- Introducción a API C de SQLite 3  
<http://www.sqlite.org/cintro.html>
- Guía del lenguaje SQL SQLite:  
<http://www.sqlite.org/lang.html>

# Crear una base de datos

```
sqlite *database;
int res = sqlite3_open("path_a_db",&database);

char *errMsg;
char *cmd = "CREATE TABLE IF NOT EXISTS PEOPLE
(ID INTEGER PRIMARY KEY AUTOINCREMENT, NAME TEXT)";

res = sqlite3_exec(database,cmd,NULL,NULL,&errMsg);

sqlite3_close(database);
```

# Core Data

# Core Data

- Aplicación para el diseño visual de modelos de datos.
- Los datos se almacenan por defecto usando una base de datos SQLite.
  - Alternativas: ficheros binarios, memoria.
- Manejamos ese almacén de datos usando un contexto.
  - No vemos como se almacena.

- Con el editor creamos entities
  - Son los tipos de datos que creamos.
- En la aplicación creamos “managed objects”
  - son las instancias de las entities definidas.
  - Las entities definen propiedades
  - los managed objects usan KVC
    - para acceder al valor de una propiedad se usa su nombre como clave.

