EBook Gratis

APRENDIZAJE

OSX

Free unaffiliated eBook created from
Stack Overflow contributors.

#osx

Tabla de contenido

Acerca de

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: osx

It is an unofficial and free osx ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official osx.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Capítulo 1: Empezando con osx

Observaciones

Esta etiqueta es para la documentación de las API de programación específicas para Mac creadas por Apple, como AppKit.

Examples

Descripción general de los marcos

Las aplicaciones escritas para macOS generalmente se escriben con los marcos de Apple. Los marcos que casi todas las aplicaciones usarán son:

- AppKit - para crear y administrar elementos de UI
- Fundación - para objetos y operaciones comunes que no son UI

Hay otros marcos comunes que se utilizan en muchas, pero no en todas las aplicaciones:

- CoreData - para el almacenamiento de datos
- Despacho - para la gestión de múltiples hilos
- CoreGraphics - para dibujar tareas relacionadas con gráficos
- CoreAnimation - para la animación de elementos UI

Lea Empezando con osx en línea: https://riptutorial.com/es/osx/topic/2818/empezando-con-osx

Capítulo 2: Asociación de archivos

Examples

Establecer mi aplicación como aplicación predeterminada para un tipo de archivo

```
- (NSString *) UTIforFileExtension:(NSString *) extension {
    NSString * UTIString = (NSString
*)UTTypeCreatePreferredIdentifierForTag(kUTTagClassFilenameExtension,
                                                                    (CFStringRef)extension,

                                                                    NULL);

    return [UTIString autorelease];
}

- (BOOL) setMyselfAsDefaultApplicationForFileExtension:(NSString *) fileExtension {
    OSStatus returnStatus = LSSetDefaultRoleHandlerForContentType (
                                                                   (CFStringRef) [self
UTIforFileExtension:fileExtension],
                                                                   kLSRolesAll,
                                                                   (CFStringRef) [[NSBundle
mainBundle] bundleIdentifier]
                                                                   );

    if (returnStatus != 0) {
        NSLog(@"Got an error when setting default application - %d", returnStatus);
        // Please see the documentation or LSInfo.h
        return NO;
    }

    return YES;
}
```

fuente

Cree una asociación con tipos de archivos nuevos / personalizados a través de Info.plist

```
<key>CFBundleDocumentTypes</key>
<array>
    <dict>
        <key>CFBundleTypeIconFile</key>
        <string>Icon file for associated file</string>
        <key>CFBundleTypeName</key>
        <string>My file format</string>
        <key>CFBundleTypeRole</key>
        <string>Viewer</string> <!-- The value can be Editor, Viewer, Shell, or None. This key
is required.  -->
        <key>LSItemContentTypes</key>
        <array>
            <string>UTI of the file</string> <!-- Existing UTI or create a UTI for your new
```

```
file type -->
        </array>
        <key>LSHandlerRank</key>
        <string>Owner</string>
    </dict>
</array>
```

fuente

Lea Asociación de archivos en línea: https://riptutorial.com/es/osx/topic/10926/asociacion-de-archivos

Capítulo 3: Establecer variables de entorno

Introducción

En Mac OS X, puede establecer las variables de entorno en uno de los siguientes archivos:

~ / .bashrc

~ / .bash_profile

~ / .profile

De forma predeterminada, Mac OS X no tiene los archivos anteriores, debe crearlo manualmente.

Examples

Agregar un camino

1. `vim ~/.bash_profile`

Es posible que el archivo no exista (si no, simplemente puede crearlo).

2.tipo en esto y guarda el archivo:

```
export PATH=$PATH:YOUR_PATH_HERE
```

Lea Establecer variables de entorno en línea: https://riptutorial.com/es/osx/topic/10162/establecer-variables-de-entorno

Capítulo 4: NSFont

Introducción

NSFont es el objeto que proporciona a las aplicaciones de Mac información de glifos y características de fuente para ser utilizadas principalmente para la visualización. Aprenderá a crear y usar objetos NSFont de varias maneras, tanto comunes como poco comunes.

Examples

Creando un objeto NSFont

La forma preferida y más común de hacer un objeto NSFont es la siguiente:

C objetivo

```
// Name is PostScript name of font; size is in points.
NSFont *essayFont = [NSFont fontWithName:@"Times New Roman" size:12.0];
```

Lea NSFont en línea: https://riptutorial.com/es/osx/topic/8881/nsfont

Capítulo 5: NSMenuItem

Observaciones

Consulte la documentación de Apple aquí:
https://developer.apple.com/library/mac/documentation/Cocoa/Reference/ApplicationKit/Classes/NSMenu

Examples

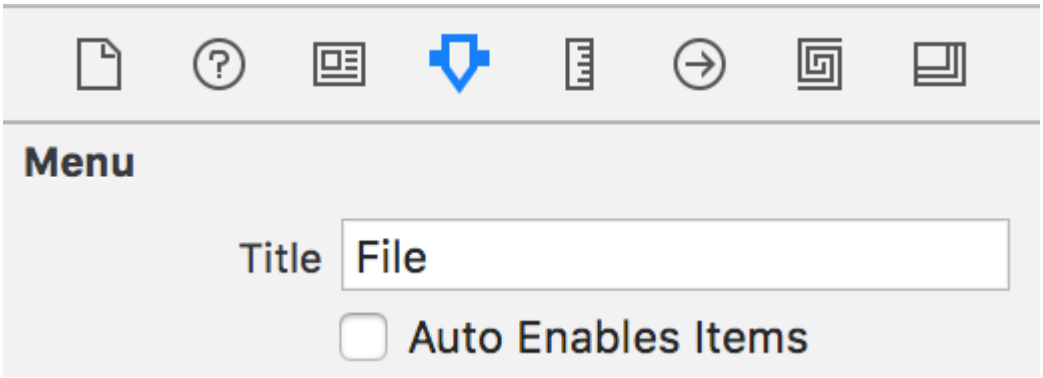
Habilitar elementos de menú

Habilitando manualmente los elementos del menú

Para controlar manualmente el estado habilitado de los elementos de un menú, el menú que lo contiene debe desactivar la habilitación automática de sus elementos.

Los menús pueden desactivar la habilitación automática de una de dos maneras:

1. En el Interface Builder



2. En codigo

```
menu.autoenablesItems = false
```

Tanto de los mecanismos de ajuste el autoenablesItems propiedad en NSMenu .

Una vez que el menú es el menú, ya no se habilita ni deshabilita los elementos del menú, los elementos del menú se pueden configurar mediante programación de una o dos formas

1. En Interface Builder
2. En codigo

```
menuItem.enabled = true
```

Activar automáticamente los elementos del menú.

Los elementos del menú se pueden habilitar automáticamente conectando las acciones de los elementos del menú al primer respondedor e implementando la acción entregada en un objeto en la cadena del respondedor según lo descrito por los documentos de Apple.

Apoyar acciones de menú por defecto

Los menús actúan como todos los elementos de control estándar. Tienen una acción que es la función que debe llamarse y un objetivo que es el objeto al que se envía la función. Si el objetivo se establece en un objeto, cuando un usuario selecciona un elemento del menú, el método de acción se enviará al objeto de destino. Si el elemento del menú tiene una acción, pero no un objetivo, el objetivo se seleccionará dinámicamente del primer objeto entre los siguientes que responden a la acción:

1. El primer respondedor
2. La jerarquía de vistas.
3. Ventana
4. Controlador de ventana
5. `NSApplication`
6. `NSApplication.delegate`
7. `NSApplication.nextResponder`

La implementación del elemento de menú Abrir (O) predeterminado se puede lograr implementando el método `openDocument` en cualquier objeto en la lista anterior.

```
- (IBAction)openDocument:(id)sender {

}
```

Agregar y eliminar elementos a un menú

```
// add an item to a menu
menu.addItem(item)

// remove and item from a menu
menu.removeItem(item)
```

Lea NSMenuItem en línea: https://riptutorial.com/es/osx/topic/6038/nsmenuitem

Capítulo 6: NSRunLoop

Examples

aplicación de demonio simple

Un proceso de daemon ejecuta un programa en segundo plano, generalmente sin interacción del usuario. El siguiente ejemplo muestra cómo crear un demonio y registrar un oyente, que supervisa todas las aplicaciones abiertas. La parte principal es la función llamada `NSRunLoop.mainRunLoop().run()` , que inicia el daemon.

```
class MyObserver: NSObject
{
    override init() {
        super.init()

        // app listeners
        NSWorkspace.sharedWorkspace().notificationCenter.addObserver(self, selector:
"SwitchedApp:", name: NSWorkspaceDidActivateApplicationNotification, object: nil)
    }

    func SwitchedApp(notification: NSNotification!)
    {
        print(notification)
    }
}


let observer = MyObserver()

// simply to keep the command line tool alive - as a daemon process
NSRunLoop.mainRunLoop().run()
```

También puede utilizar este código como base para un proceso de servidor.

Lea NSRunLoop en línea: https://riptutorial.com/es/osx/topic/6667/nsrunloop

Capítulo 7: NSStoryBoard

Examples

Abra un nuevo controlador de ventana

Para abrir una nueva ventana, agregue el siguiente código en algún lugar donde pueda mantener una referencia a la nueva ventana (IE, el delegado de la aplicación).

Rápido

```
let storyboard:NSStoryboard = NSStoryboard(name: "Main", bundle: nil)
guard let controller:NSWindowController =
storyboard.instantiateControllerWithIdentifier("myWindowController") as? NSWindowController
else { return /*or handle error*/ }
controller.showWindow(self)
```

C objetivo

```
NSStoryboard *storyBoard = [NSStoryboard storyboardWithName:@"Main" bundle:nil]; // get a
reference to the storyboard
myController = [storyBoard instantiateControllerWithIdentifier:@"secondWindowController"]; //
instantiate your window controller
[myController showWindow:self];
```

Una vez que haya creado su `controller` asegúrese de mantener una referencia en algún lugar fuera de la llamada a la función. Esto se puede hacer creando una variable `NSWindowController` en el delegado de su aplicación y asignando su nuevo controlador a la variable.

Lea NSStoryBoard en línea: https://riptutorial.com/es/osx/topic/4287/nsstoryboard

Capítulo 8: NSTextView

Introducción

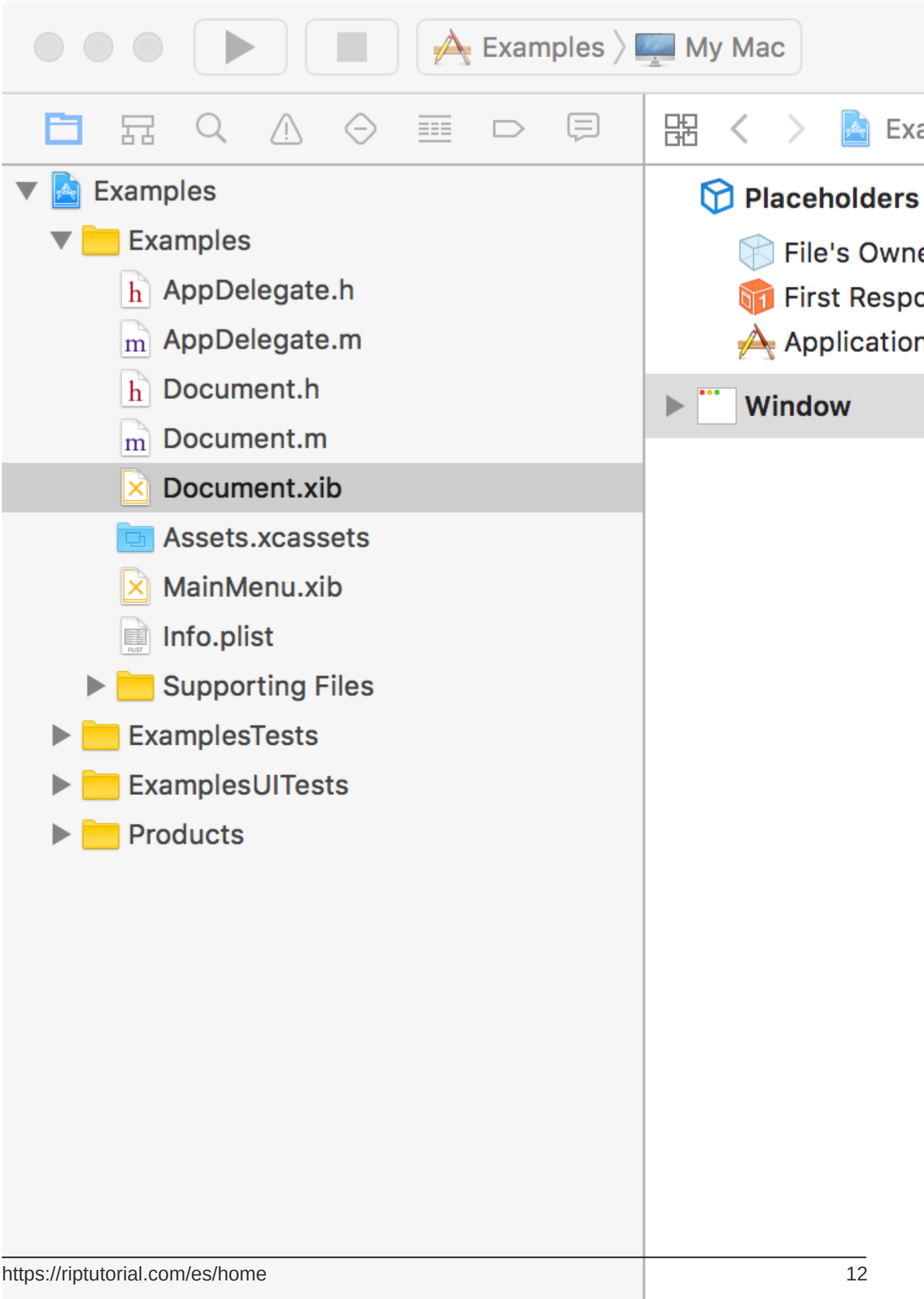
NSTextView es el controlador principal de Apple del sistema de texto de AppKit. Contiene todo lo que necesita para crear un visor de texto / espacio de editor en aplicaciones de OS X (con nombre de macOS).

Examples

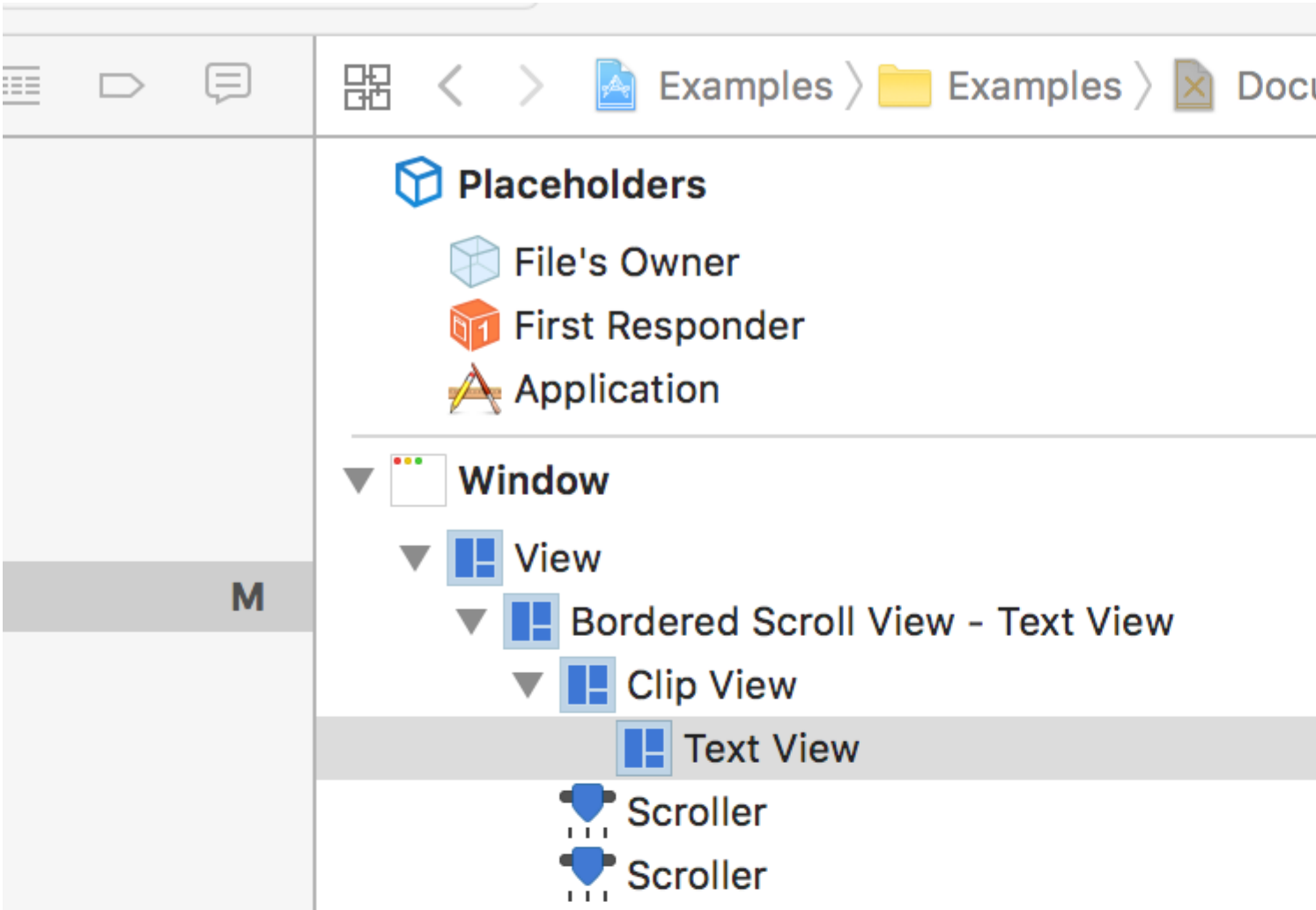
Creando un NSTextView

Gráficamente

En XCode se puede crear un simple NSTextView arrastrando y soltando uno de la biblioteca de objetos.


Examples
My Mac
Examples
Examples
AppDelegate.h
AppDelegate.m
Document.h
Document.m
Document.xib
Assets.xcassets
MainMenu.xib
Info.plist
Supporting Files
ExamplesTests
ExamplesUITests
Products
Placeholders
File's Owne
First Respo
Application
Window

que se configura automáticamente para expandirse verticalmente con la vista de texto. Asegúrese de que cuando presiona la opción (option) realice conexiones a la vista de texto y no a la vista de desplazamiento.



Drag out to expose

1. Un NSLayoutManager: realiza el diseño de glifos / caracteres.
2. Un NSTextContainer: controla el espacio gráfico que pueden albergar los glifos / caracteres.
3. Un NSTextStorage: contiene los datos de cadena reales que muestra NSTextView.

- Un NSTextStorage puede tener muchos NSLayoutManager, pero un NSLayoutManager solo puede tener un NSTextStorage. Esto es útil si desea mostrar los mismos datos de diferentes maneras al mismo tiempo.
- NSLayoutManager's puede tener muchos NSTextContainer's. Útil para el texto paginado.
- Los NSTextView solo pueden tener un NSTextContainer a la vez.
- Ciertas cosas incorporadas en NSTextView están fuera de los límites en el momento de la escritura. Por ejemplo, las funciones integradas de Buscar y Reemplazar no se pueden personalizar, pero se pueden anular con funciones personalizadas.

Más información sobre las formas de utilizar el sistema de texto se puede encontrar aquí .

Ahora para el código. Este código creará un simple NSTextView, sin siquiera desplazarse. Tales cosas como el desplazamiento y la paginación estarán en otro ejemplo.

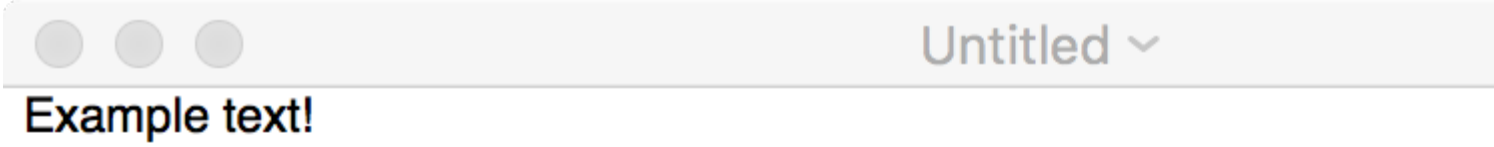
C objetivo

```
// This code resides in an NSDocument object's windowControllerDidLoadNib:(NSWindowController
*)windowController method.
// This is done simply because it is easy and automatically gets called upon.

// This method is also where the following NSRect variable gets size information. We need this
information for this example.
NSRect windowFrame = windowController.window.contentView.frame;
NSTextStorage *textStorage = [[NSTextStorage alloc] initWithString:@"Example text!"];
NSLayoutManager *manager = [[NSLayoutManager alloc] init];
NSTextContainer *container = [[NSTextContainer alloc]
initWithContainerSize:NSMakeSize(windowFrame.size.width, windowFrame.size.height)];
NSTextView *textView = [[NSTextView alloc] initWithFrame:windowFrame textContainer:container];

[textStorage addLayoutManager:manager];
[manager addTextContainer:container];
[windowController.window setContentView:textView];
```

¡Felicidades! ¡Has hecho un NSTextView programáticamente!



Lea NSTextView en línea: https://riptutorial.com/es/osx/topic/8880/nstextview

Capítulo 9: Pedir al usuario un archivo

Introducción

NSOpenPanel proporciona una API para solicitar al usuario que abra un archivo. Este menú es la IU estándar presentada por el elemento de menú Abrir (O).

Examples

Abriendo archivos

Abriendo cualquier archivo

```
NSOpenPanel *openPanel = [NSOpenPanel openPanel];
[openPanel beginWithCompletionHandler:^(NSInteger result) {
    NSURL *url = openPanel.URL;
    if (result == NSFileHandlingPanelCancelButton || !url) {
        return;
    }
    // do something with a URL
}];
```

Permitiendo abrir múltiples archivos

```
NSOpenPanel *openPanel = [NSOpenPanel openPanel];
openPanel.allowsMultipleSelection = YES;
[openPanel beginWithCompletionHandler:^(NSInteger result) {
    NSArray <NSURL *>*urls = openPanel.URLs;
    // do things
}];
```

Limitar a tipos de archivos específicos

```
NSOpenPanel *openPanel = [NSOpenPanel openPanel];
openPanel.allowedFileTypes = @[@".png", @".jpg"];
[openPanel beginWithCompletionHandler:^(NSInteger result) {
    NSURL *url = openPanel.URL;
    if (result == NSFileHandlingPanelCancelButton || !url) {
        return;
    }
    // do something with a picture
}];
```

Lea Pedir al usuario un archivo en línea: https://riptutorial.com/es/osx/topic/9438/pedir-al-usuario-

un-archivo

Creditos

S. No	Capítulos	Contributors
1	Empezando con osx	Andrew Hoos, Community, l'L'l, tbodt
2	Asociación de archivos	bikram990
3	Establecer variables de entorno	Kuhan
4	NSFont	malicedShade
5	NSMenuItem	Andrew Hoos, Barlow Tucker
6	NSRunLoop	Marco Pashkov
7	NSStoryBoard	Barlow Tucker
8	NSTextView	malicedShade
9	Pedir al usuario un archivo	Andrew Hoos