[IOS之基本UI控件](http://www.cnblogs.com/syxchina/archive/2012/09/04/2669844.html)

3.1 Button控件

3.2 开关控件

3.3 滑块控件

3.4 工具栏

3.5 WebView

**3.1 Button控件**

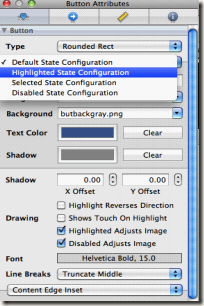
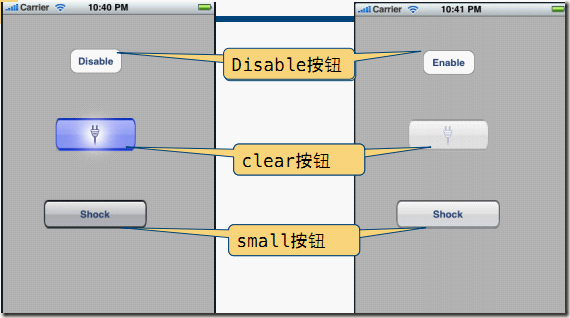
iPhone的Button控件可以做的很绚丽，Button可以有多种状态：

" Default State

" Highlighted State

" Selected State

" Disabled State

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732269785.png)[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732277244.png)

实现上图的效果：新建ButtonsBackground项目：

ButtonsBackgroundViewController.h文件

[](javascript:void(0);)

@interface ButtonsBackgroundViewController : UIViewController {

UIButton \* clearButton;

UIButton \* smallButton;

}

@property (nonatomic, retain) IBOutlet UIButton \* clearButton;

@property (nonatomic, retain) IBOutlet UIButton \* smallButton;

- (IBAction) disableBut: (id) sender;

@end

[](javascript:void(0);)

ButtonsBackgroundViewController.m文件

[](javascript:void(0);)

@synthesize clearButton;

@synthesize smallButton;

- (IBAction) disableBut: (id) sender {

if(clearButton.enabled == YES) {

clearButton.enabled = NO;

smallButton.enabled = NO;

[((UIButton \*) sender) setTitle:@"Enable" forState:UIControlStateNormal];

}

else {

clearButton.enabled = YES;

smallButton.enabled = YES;

[((UIButton \*) sender) setTitle:@"Disable" forState:UIControlStateNormal];

}

}

- (void)dealloc {

[clearButton release];

[smallButton release];

[super dealloc];

}

[](javascript:void(0);)

点击Disable按钮时候，调用disableBut方法，在该方法中实现了clearButton按钮和smallButton按钮“可用”状态和“不可用”状态的切换。在状态发生切换时候还要改变Disable按钮的上面的“标题”和“状态”：

[((UIButton \*) sender) setTitle:@"Enable"forState:UIControlStateNormal];

sender是事件源即点击的按钮对象本身。

Interface Builder设计页面 ：

点击Disable按钮的时候，会改变clearButton和smallButton的标题，因此需要连接File’s Owner到两个按钮（clearButton和smallButton）的输出口。

为了响应Disable按钮的事件需要，从Disable按钮连接到File’s Owner定义的disableBut事件。

clearButton属性设定

属性框，使Shows Touch on Highlight is checked 选项被选中。

Default State Configuration选中时候，设置按钮图片power.png背景图片butbackgray.png。

Highlighted State Configuration 选中时候，设置按钮图片power.png ,背景图butbackbluegray.png。

Disabled State Configuration 选中时候，设置按钮图片powerdisabled.png背景图片,butbackgraydisabled.png。

smallButton属性设定

Default State Configuration选中时候，设置按钮图片butbackgray.png背景图片， 设置title“Shock”。

Highlighted State Configuration 选中时候，设置图片butbackbluegray.png背景图片，设置title“Shocking”。

**3.2 开关控件**

开关控件（Switch），有些相windows中的checkbox，它只有两种状态，true和false。

可以通过该方法改变开关控件的状态。

-(void) setOn: (BOOL) on animated: (BOOL) animated

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732288017.png)

**3.3 滑块控件**

滑块控件（Slider），水平放置，可以用手触摸改变它的值。

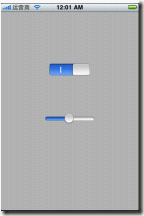
默认情况下它的最小值0最大值1.00，而.50是初始值，我们可以通过下面的方法设定值：

- (void) setValue:(float) value animated:(BOOL) animated

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732282128.png)

开关和滑块实例

我们通过在页面放在开关和滑块控件了解他们的使用情况。

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732285476.png)

新建项目SwitchSlider:

SwitchSliderViewController.h

[](javascript:void(0);)

@interface SwitchSliderViewController : UIViewController {

UISwitch \* mySwitch;

}

@property(nonatomic, retain)IBOutlet UISwitch \* mySwitch;

-(IBAction) handleSwitch: (id) sender;

-(IBAction) handleSlider: (id) sender;

@end

[](javascript:void(0);)

SwitchSliderViewController.m

[](javascript:void(0);)

@implementation SwitchSliderViewController

@synthesize mySwitch;

- (IBAction) handleSwitch: (id) sender {

if( [((UISwitch \*) sender) isOn] == YES){

NSLog(@"It's on");

} else {

NSLog(@"It's off");

}

}

- (IBAction) handleSlider: (id) sender {

NSLog(@"value: %f", ((UISlider \*)sender).value);

if( [((UISlider \*) sender) value] == ((UISlider \*) sender) .maximumValue) {

[mySwitch setOn:YES animated:YES];

}

}

- (void)dealloc {

[mySwitch release];

[super dealloc];

}

[](javascript:void(0);)

连接输出口和动作事件

连接开关控件到的handleSwitch: 动作。

连接滑块控件到的handleSlider: 动作。

制定开关控件输出口。

**3.4 分段控件**

分段控件（Segment），是有2个或更多段构成的组，它相当与独立的按钮。

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732298441.png)

新建项目Segment：

SegmentViewController.h

定义一个动作事件

[](javascript:void(0);)

@interface SegmentViewController : UIViewController {

}

- (IBAction) handleSegment: (id) sender;

[](javascript:void(0);)

SegmentViewController.m

[](javascript:void(0);)

- (IBAction) handleSegment: (id) sender {

UISegmentedControl \* myseg = (UISegmentedControl \*) sender;

if(myseg.selectedSegmentIndex == 0) {

NSLog(@"selected zero index...");

}

else if(myseg.selectedSegmentIndex == 1) {

NSLog(@"selected one index...");

}

else {

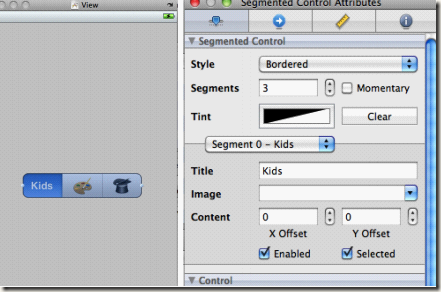
NSLog(@"selected two index...");

}

}

[](javascript:void(0);)

IB设计视图

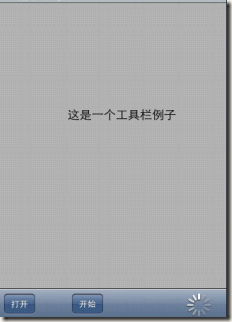
[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732303882.png)

连接动作事件

连接段控件到File’s Owner的handleSegment: 动作。

**3.4 工具栏**

工具栏（UIToolBar），一般是放置在屏幕的底部，在工具栏的内部可以放置多个按钮和控件。

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732301897.png)

新建项目ToolBar：

ToolBarViewController.h

定义两个动作事件

[](javascript:void(0);)

@interface ToolBarViewController : UIViewController {

IBOutlet UIActivityIndicatorView \* myActivityView;

}

@property (nonatomic, retain) IBOutlet UIActivityIndicatorView \* myActivityView;

-(IBAction)onClickStartButton: (id)sender;

-(IBAction)onClickOpenButton: (id)sender;

@end

[](javascript:void(0);)

ToolBarViewController.m

[](javascript:void(0);)

@implementation ToolBarViewController

@synthesize myActivityView;

-(IBAction)onClickStartButton: (id)sender {

if ([myActivityView isAnimating]) {

[myActivityView stopAnimating];

} else {

[myActivityView startAnimating];

}

}

-(IBAction)onClickOpenButton: (id)sender {

UIAlertView \*alert = [[UIAlertView alloc] initWithTitle:@"提示信息"

message:@"您点击了打开按钮" delegate:self

cancelButtonTitle:@"Done"

otherButtonTitles:nil];

[alert show];

[alert release];

}

- (void)didReceiveMemoryWarning {

[super didReceiveMemoryWarning];

}

- (void)viewDidUnload {

}

- (void)dealloc {

[myActivityView release];

[super dealloc];

}

@end

[](javascript:void(0);)

IB设计视图

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732318210.png)

连接动作事件

连接到打开按钮的onClickOpenButton: 动作。

连接到开始按钮的onClickStartButton: 动作。

连接到UIActivityIndicatorView输出口。

**3.5 WebView**

WebView可以帮助我们构建Web的iPhone应用程序。 很多网站的iPhone和iPad客户端程序都是使用WebView开

发的。 WebView能够支持HTML5，不支持Flash等。

[](http://images.cnblogs.com/cnblogs_com/syxchina/201209/201209040732312637.png)

新建项目MyWeb：

MyWebViewController.h

[](javascript:void(0);)

@interface MyWebViewController : UIViewController <UIWebViewDelegate> {

IBOutlet UITextField \* myTextField;

IBOutlet UIWebView \* myWebView;

}

@property(nonatomic, retain) UIWebView \* myWebView;

@property(nonatomic, retain) UITextField \* myTextField;

- (IBAction) changeLocation: (id) sender;

@end

[](javascript:void(0);)

MyWebViewController 需要实现协议UIWebViewDelegate，它一个委托对象。委托是一种设计模式，在iPhone中主要用于回调事件。在委托中定义了一下方法，实现了该委托的对象，要提供该方法的实现。

UIWebViewDelegate的方法是webViewDidFinishLoad:  它上在异步情况一个网址，当应答回来后回调该方法。

MyWebViewController.m

[](javascript:void(0);)

@implementation MyWebViewController

@synthesize myWebView;

@synthesize myTextField;

- (void) viewDidLoad {

myWebView.delegate = self;

}

- (void)dealloc {

myWebView.delegate = nil;

[myTextField release];

[myWebView release];

[super dealloc];

}

- (IBAction) changeLocation: (id) sender {

[myTextField resignFirstResponder];

NSURL \* url = [NSURL URLWithString: myTextField.text];

NSURLRequest \* request = [NSURLRequest requestWithURL:url];

[myWebView loadRequest:request];

}

#pragma mark WebView 委托

#pragma mark --

- (void)webViewDidFinishLoad: (UIWebView \*) webView {

NSLog(@"%@", [webView stringByEvaluatingJavaScriptFromString:

@"document.body.innerHTML"]);

}

@end

[](javascript:void(0);)

在viewDidLoad 方法中的myWebView.delegate =self是指定为自身。

webViewDidFinishLoad方法中实现委托回调功能。

[myWebView stringByEvaluatingJavaScriptFromString:

@"document.documentElement.textContent”]

是运行一个JavaScript脚本程序，document.body.innerHTML获得页面中的HTML代码。

注：

1 本教程是基于关东升老师的教程

2 基于黑苹果10.6.8和xcode4.2

3 本人初学，有什么不对的望指教

4 教程会随着本人学习，持续更新

5 教程是本人从word笔记中拷贝出来了，所以格式请见谅