

# 移动应用开发实践

项目实战一



## 本节内容

---

- **选择器**
- **自己的选择器**
- **利用框架实现自定义选择器**
- **自定义日期选择器（作业）**

# 选择器

---

## 1. 日期选择器 UIDatePicker

2013年	12月	5日
2014年	1月	6日
2015年	2月	7日
2016年	3月	8日
2017年	4月	9日
2018年	5月	10日
2019年	6月	11日

日期

3月5日 周六	5	10
3月6日 周日	6	11
3月7日 周一 上午	7	12
今天 下午	8	13
3月9日 周三	9	14
3月10日 周四	10	15
3月11日 周五	11	16

日期时间

	5	11
	6	12
上午	7	13
下午	8	14
	9	15
	10	16
	11	17

时间

17	9
18	10
19	11
20 hours	12 min
21	13
22	14
23	15

倒计时定时器

# 选择器

---

## 2. 普通选择器 UIPickerView

是UIDatePicker的父类，需要两个重要的协议：

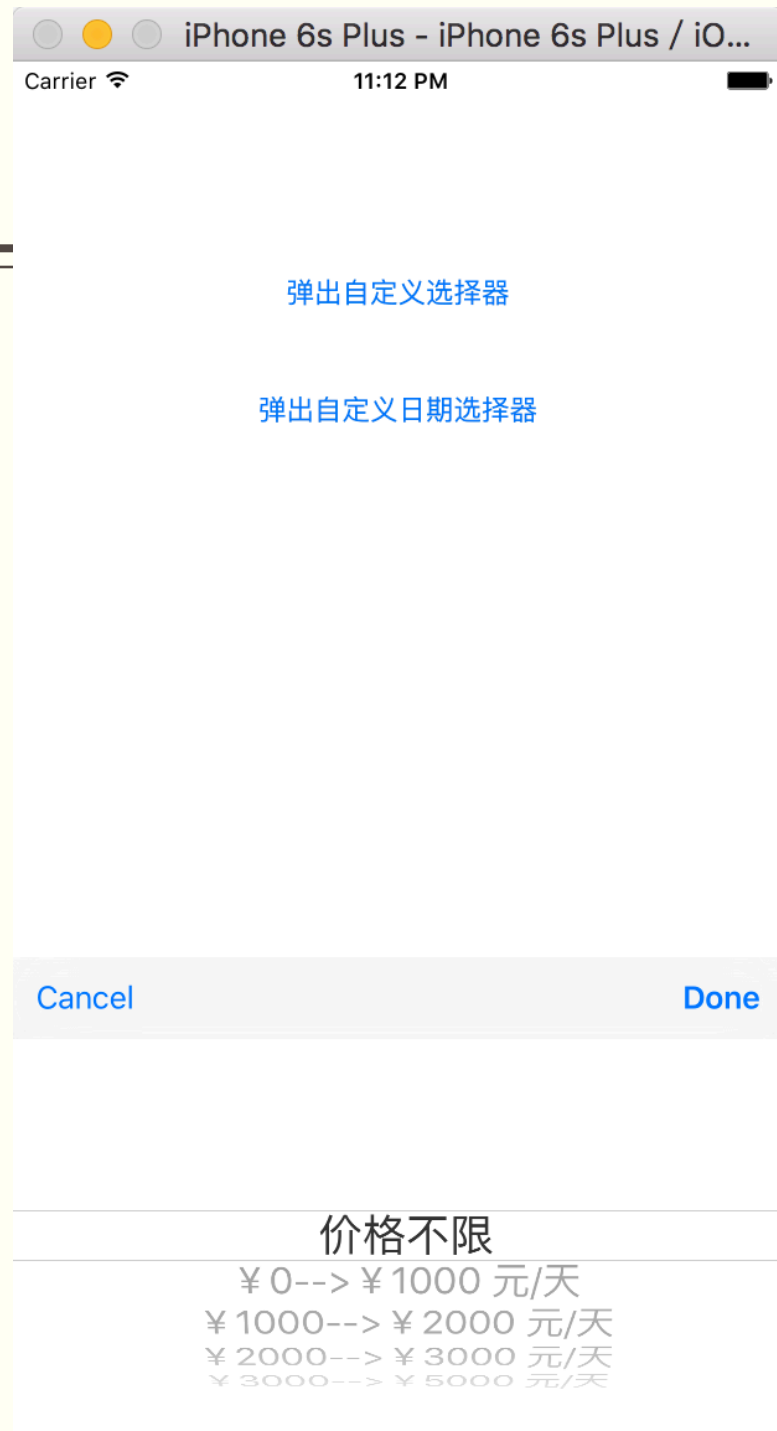
(1) UIPickerViewDataSource

(2) UIPickerViewDelegate

吉林省	沈阳
黑龙江省	大连
辽宁省	鞍山
	抚顺
	本溪
	丹东

# 自定义选择器



- 自定义控件
- 自定义空间的发布
  - ① 源代码
  - ② 静态链接库
  - ③ 框架



# 自定义选择器

## ■ 框架

Choose a template for your new project:

iOS		
Application	Cocoa Touch Framework	Cocoa Touch Static Library
Framework & Library		
watchOS		
Application		
Framework & Library		
tvOS		
Application		
Framework & Library		
OS X		
Application		
Framework & Library		
System Plug-in		
Other	Cocoa Touch Framework This template creates a framework that uses UIKit.	

Cancel Previous Next

# 自定义选择器

- 为工程添加文件

勾选 XIB File

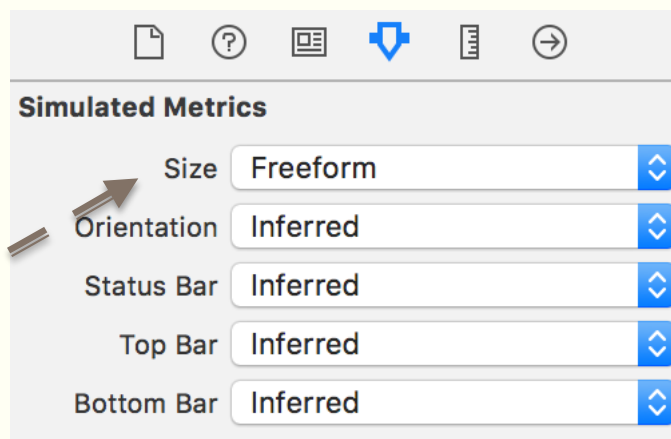
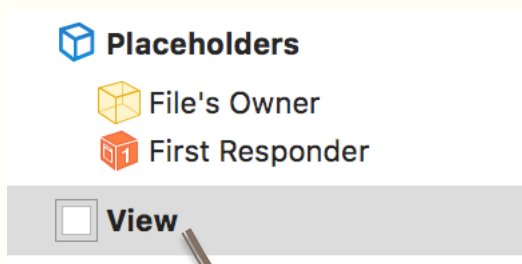
Class:

Subclass of:  ▼

☒ Also create XIB file

▼

Language:  ▼

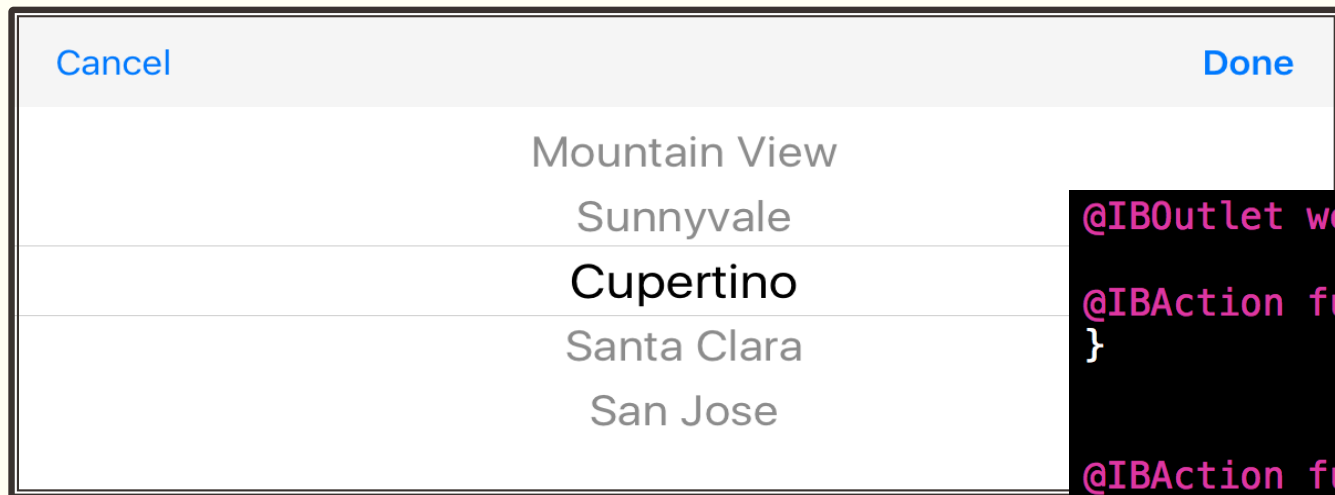


自定义大小

# 自定义选择器

---

- 在Xib里，拖入 导航栏、导航条按钮、普通选择器
- 在对应的文件里，为左右上角按钮和普通选择器创建事件处理方法和对象
- 实现普通选择器协议的方法



```
@IBOutlet weak var picker: UIPickerView!  
  
@IBAction func done(sender: UIBarButtonItem) {  
}  
  
@IBAction func cancel(sender: UIBarButtonItem) {  
}
```



# 自定义选择器

---

- 编写视图控制器委托协议

```
public protocol MyPickerControllerDelegate {  
  
    func myPickViewClose(selected : String)  
  
}
```

# 自定义选择器

---

- 视图控制器初始化

```
//1. 定义构造器,  
public init() {  
    //获取的资源目录是MyPickerViewController类所在的目录  
    let resourcesBundle = NSBundle(forClass:MyPickerViewController.self)  
    //通过Xib文件名调用父类构造器  
    super.init(nibName: "MyPickerViewController", bundle: resourcesBundle)  
  
    self.pickerData = ["价格不限", "¥0-->¥1000 元/天", "¥1000-->¥2000 元/天", "¥2000-->¥3000  
        元/天", "¥3000-->¥5000 元/天"]  
}
```

# 自定义选择器

---

## ■ 显示和隐藏视图

```
//2. 显示视图
public func showInView(superview : UIView) {

    if self.view.superview == nil {
        superview.addSubview(self.view)
    }

    //调整控件的中心位置, 让它在屏幕外
    self.view.center = CGPointMake(self.view.center.x, 900)
    //重新调整控件的大小, 根据运行屏幕的大小
    self.view.frame = CGRectMake(self.view.frame.origin.x , self.view.frame.origin.y ,
        superview.frame.size.width, self.view.frame.size.height)

    //以动画方式显示视图, 动画持续时间, 延迟时间, 效果, 执行动画动用时闭包, 动画完成时调用闭包
    UIView.animateWithDuration(0.3, delay: 0.3, options: UIViewAnimationOptions.CurveEaseInOut,
        animations: { () -> Void in

            self.view.center = CGPointMake(superview.center.x, superview.frame.size.height - self.view.frame.size.height/2) //保证控件下边界与父视图下边界对齐

        }, completion: nil)
}
```

# 自定义选择器

---

- 显示和隐藏视图

//3. 隐藏视图

```
public func hideInView() {  
    UIView.animateWithDuration(0.3, delay: 0.0, options: UIViewAnimationOptions.CurveEaseInOut,  
        animations: { () -> Void in  
  
        self.view.center = CGPointMake(self.view.center.x, 900)  
  
    }, completion: nil)  
}
```

# 自定义选择器

---

- 按钮事件处理方法

//4. Done按钮

```
@IBAction func done(sender: UIBarButtonItem) {  
    self.hideInView()  
    //获取第一个拨轮中的索引  
    let selectedIndex = self.picker.selectedRowInComponent(0)  
    //调用委托对象的方法  
    self.delegate?.myPickViewClose(self.pickerData[selectedIndex] as! String)  
}
```

//5. Cancel按钮

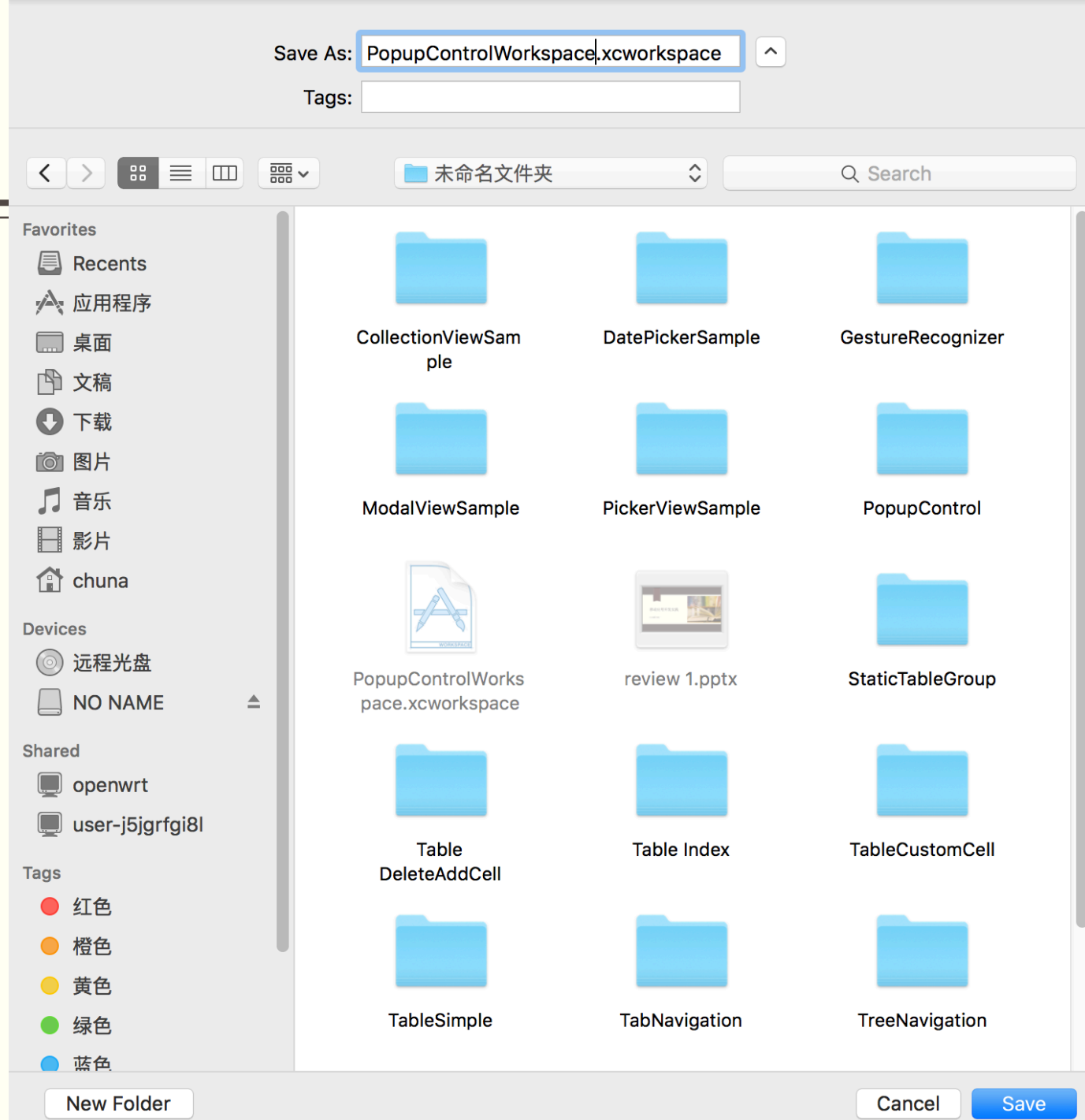
```
@IBAction func cancel(sender: UIBarButtonItem) {  
    self.hideInView()  
}
```

# 自定义选择器

## ■ 创建工作空间

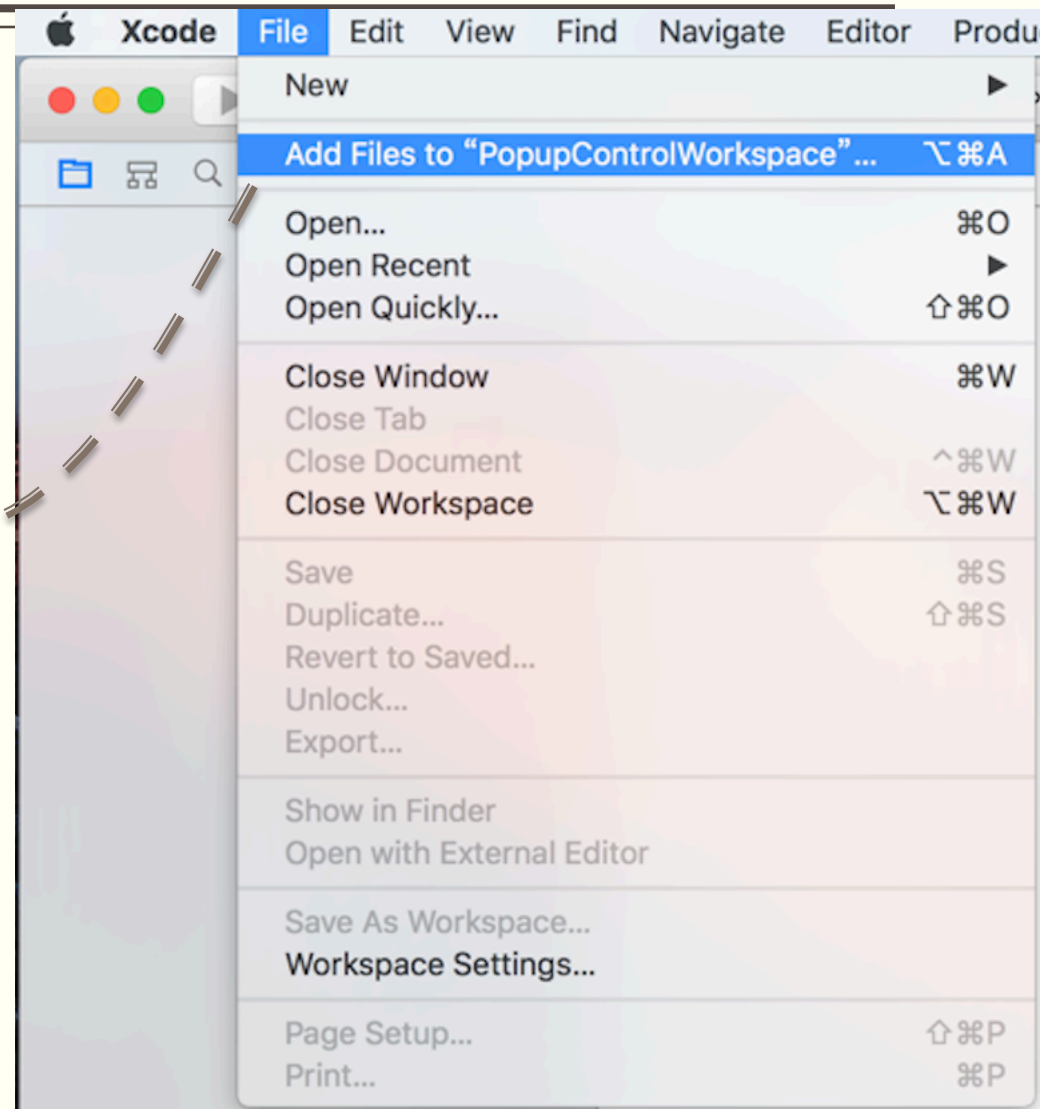
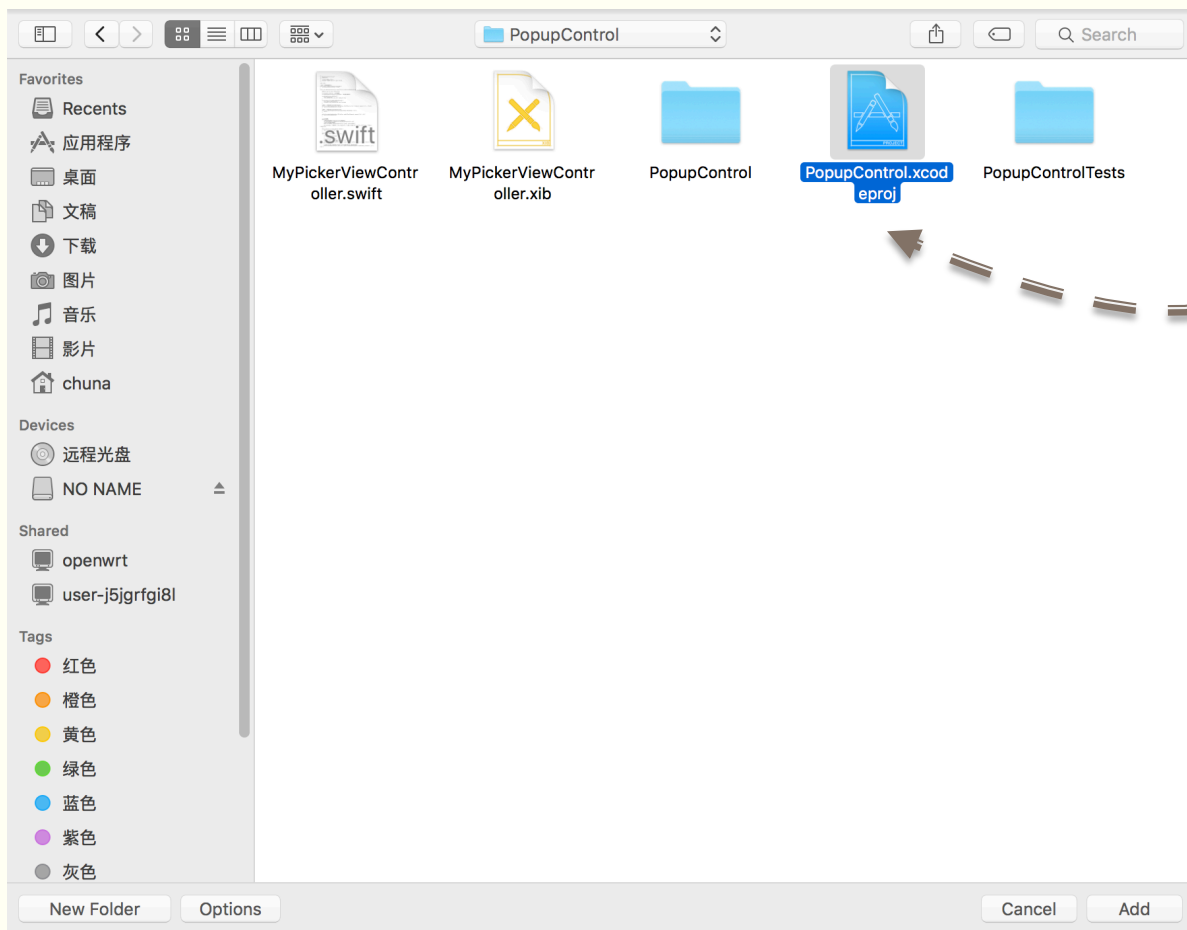
将应用程序的测试工程和自定义控件的框架工程加工作空间。

创建的工作空间最好与框架在同一个目录下。



# 自定义控件

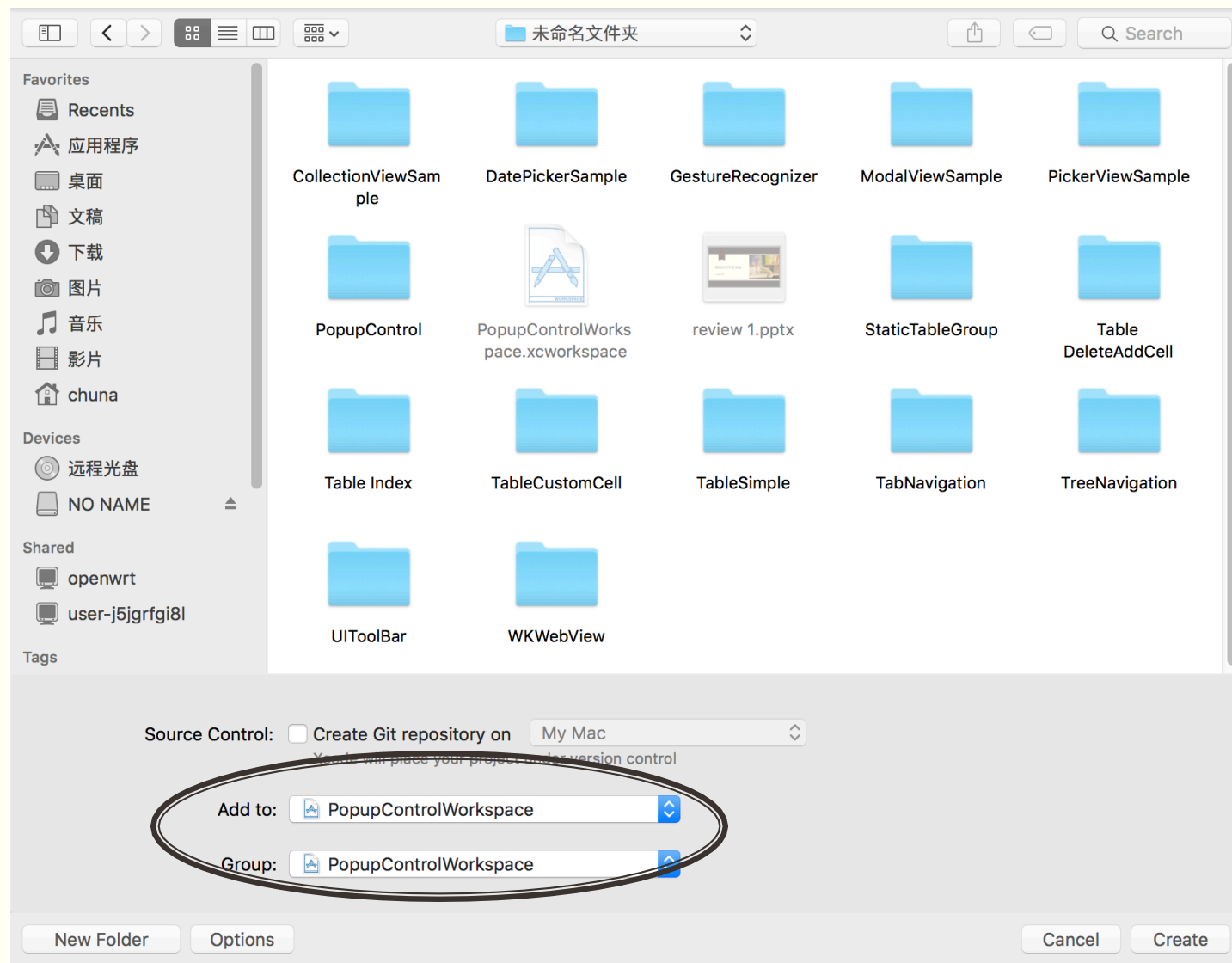
## ■ 将框架加入到空间中



# 自定义选择器

## ■ 创建测试程序工程到空间

### □ 在空间中新建工程



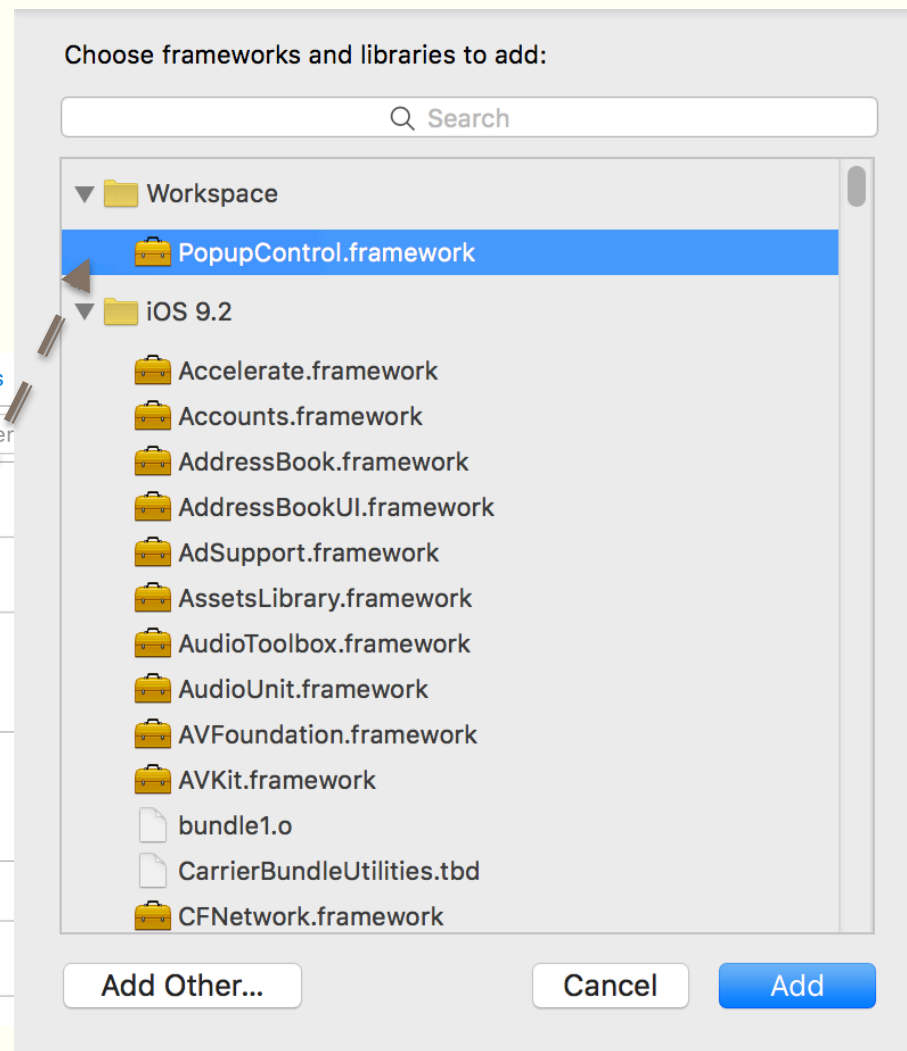
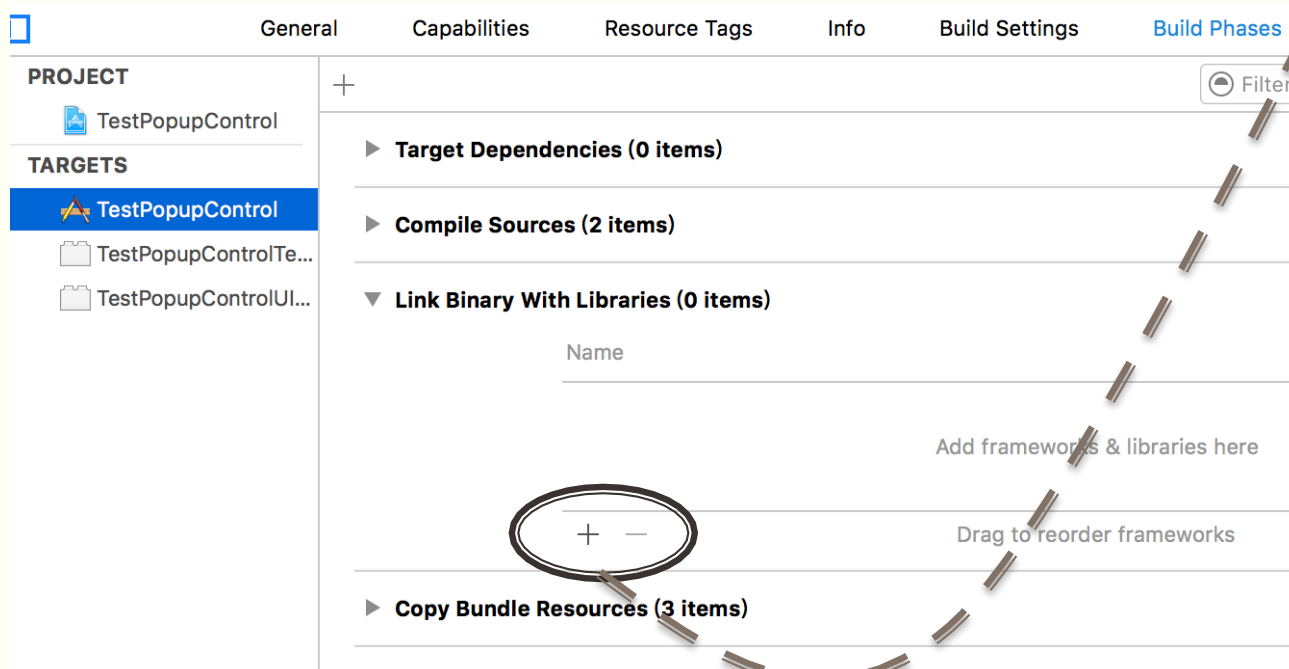


# 自定义选择器

## ■ 创建测试程序工程到空间

□ 在空间中新建工程

□ 添加依赖关系



# 自定义选择器

---

## 构建界面

□ 添加按钮

## 编写测试代码



# 作业

创建自定义日期选择器框架，并添加到空间中，进行测试调用

