A01-代码演示，思路分析

2014年10月14日 星期二　　11:40

1、演示做好的代码
2、根据界面讲解做的思路

3、思路
　　1、加载应用信息
　　2、根据应用的个数创建view
　　3、点击下载按钮，实现相应的操作

A02-创建项目

2014年10月14日 星期二　　11:41

1、创建项目
2、拖入使用到的图片和plist文件

A03-加载plist文件

2014年10月14日 星期二　　14:03

1、在默认的ViewController中创建NSArray属性，准备加载plist文件中的内容

```
//存储从plist中加载的数据
@property (nonatomic, strong) NSArray *appInfos;
```

2、加载完成后NSArray中就存储了所有的字典数据（应用信息的name和icon）

```
//1 懒加载plist中的数据
- (NSArray *)appInfos
{
    if (_appInfos == nil) {
        //获取当前应用的bunle
        NSBundle *bundle = [NSBundle mainBundle];
        NSString *path = [bundle pathForResource:@"app.plist" ofType:nil];
        //
        _appInfos = [NSArray arrayWithContentsOfFile:path];

    }
    return _appInfos;
}
```

A04-制作9宫格view

2014年10月14日 星期二　　21:32

1、手工添加视图

```
    //1 动态创建view
UIView *view1 = [[UIView alloc] init];
//添加到父view
[self.view addSubview:view1];
//2 设置view的属性
//设置背景颜色
view1.backgroundColor = [UIColor redColor];
//设置位置和大小
view1.frame = CGRectMake(10, 50, 90, 90);


//1 动态创建view
UIView *view2 = [[UIView alloc] init];
//添加到父view
[self.view addSubview:view2];
//2 设置view的属性
//设置背景颜色
view2.backgroundColor = [UIColor redColor];
//设置位置和大小
view2.frame = CGRectMake(110, 50, 90, 90);


//1 动态创建view
UIView *view3 = [[UIView alloc] init];
//添加到父view
[self.view addSubview:view3];
//2 设置view的属性
//设置背景颜色
view3.backgroundColor = [UIColor redColor];
//设置位置和大小
view3.frame = CGRectMake(210, 50, 90, 90);


//1 动态创建view
UIView *view4 = [[UIView alloc] init];
//添加到父view
[self.view addSubview:view4];
//2 设置view的属性
//设置背景颜色
view4.backgroundColor = [UIColor redColor];
//设置位置和大小
view4.frame = CGRectMake(10, 150, 90, 90);
```

2、找公式

子view的横向间距　=　（父view的宽度- 3 * 子view的宽度）/ 4

子view的纵向间距 =　20

当前子view的行号 = 当前遍历到得索引值 / 总列数

当前子view的列号 = 当前遍历到得索引值 % 总列数

子view横坐标的公式 = 子view的横向间距 + 列号 *（子view的横向间距+ 子view的宽度）
子view纵坐标的公式 = 50 + 行号 *（子view的纵向间距+ 子view的高度）

2、计算9宫格位置，循环添加

```
//定义子view的宽度和高度
    CGFloat viewW = 90;
    CGFloat viewH = 90;
    //总列数
    int totalColumns = 3;

    //横向间距
    CGFloat marginX = (self.view.frame.size.width - viewW * totalColumns)/(totalColumns + 1);
    CGFloat marginY = 20;

    // 1 遍历所有的appInfo 生成9宫格
    for (int i = 0; i < self.appInfos.count; i++) {
        //2 动态创建子view
        UIView *view = [[UIView alloc] init];
        //添加到父view上
        [self.view addSubview:view];
        view.backgroundColor = [UIColor redColor];

        //计算行号和列号
        int row = i / totalColumns;
        int column = i % totalColumns;
        //计算x和y
        CGFloat viewX = marginX + column * (viewW + marginX);
        CGFloat viewY = 50 + row * (viewH + marginY) ;

        //设置frame
        view.frame = CGRectMake(viewX, viewY, viewW, viewH);
    }
```

A05-显示应用信息

2014年10月14日 星期二　　21:34

动态创建控件可能会比较陌生，慢慢消化

设置控件的属性，对应storyboard来演示

讲解button中有两个子控件，一个UIImageView和一个UILabel，通过storyboard演示
按钮的几种状态

1、绘制完9宫格之后，继续在子view中添加3个子控件

```
//3 给子view添加子控件
//3.1 添加图标
UIImageView *iconView = [[UIImageView alloc] init];
//添加到子view
[view addSubview:iconView];
//设置图片
iconView.image = [UIImage imageNamed:dic[@"icon"]];
//设置frame
CGFloat iconW = 50;
CGFloat iconH = 50;
CGFloat iconX = (viewW - iconW) / 2;
CGFloat iconY = 0;
iconView.frame = CGRectMake(iconX, iconY, iconW, iconH);

//3.2 添加显示名称的label
UILabel *nameLabel = [[UILabel alloc] init];
[view addSubview:nameLabel];

//设置label上的文字
nameLabel.text = dic[@"name"];
//设置字体大小
nameLabel.font = [UIFont systemFontOfSize:13];
//设置文字居中
nameLabel.textAlignment = NSTextAlignmentCenter;
//设置label的frame
CGFloat nameW = viewW;
CGFloat nameH = 20;
CGFloat nameX = 0;
CGFloat nameY = iconH;
nameLabel.frame = CGRectMake(nameX, nameY, nameW, nameH);

//3.3 添加下载按钮
//此处不用UIButtonTypeSystem的原因是系统会提供一些样式，灰色的蒙版和字体样式
UIButton *downloadView = [UIButton buttonWithType:UIButtonTypeCustom];
[view addSubview:downloadView];
//设置按钮的文字
[downloadView setTitle:@"下载" forState:UIControlStateNormal];
//设置按钮的背景图片
[downloadView setBackgroundImage:[UIImage imageNamed:@"buttongreen"] forState:UIControlStateNormal];
[downloadView setBackgroundImage:[UIImage imageNamed:@"buttongreen_highlighted"]
forState:UIControlStateNormal];
//设置frame
CGFloat downW = iconW;
CGFloat downH = 20;
CGFloat downX = (viewW - downW) / 2;
//获得nameLabel最大的y值，作为下载按钮的y值
CGFloat downY = CGRectGetMaxY(nameLabel.frame);
downloadView.frame = CGRectMake(downX, downY, downW, downH);
//设置字体大小
```

```
downloadView.titleLabel.font = [UIFont systemFontOfSize:14];
```

2、回头展望viewDidLoad方法，已经惨不忍睹，把添加3个子控件的代码封装成一个方法，减少viewDidLoad的代码量

```
//添加3个子控件显示应用信息
[self displaySubViews:view dic:dic viewW:viewW];
```

A06-字典转模型对象

2014年10月14日 星期二　　22:10

1、使用字典的坏处

手敲字符串key，key容易写错

Key如果写错了，编译器不会有任何警告和报错，造成设错数据或者取错数据

不面向对象

2、使用对象的好处

面向对象，对象把显示世界中的内容抽象到程序的世界，更直观，更接近人类的语言

所谓模型，其实就是数据模型，专门用来存放数据的对象，用它来表示数据会更加专业

模型设置数据和取出数据都是通过它的属性，属性名如果写错了，编译器会马上报错，因此，保证了数据的正确性

使用模型访问属性时，编译器会提供一系列的提示，提高编码效率

```
@interface CZAppInfo : NSObject
//copy  字符串
//strong  oc对象
//weak ui控件
//assign 基本数据类型
@property (nonatomic, copy) NSString *icon;
@property (nonatomic, copy) NSString *name;
@end
```

```
//懒加载plist中的数据
- (NSArray *)appInfos
{
    if (_appInfos == nil) {
        //1 获取当前应用的bundle
        NSBundle *bundle = [NSBundle mainBundle];
        //2 plist文件的路径
        NSString *path = [bundle pathForResource:@"app.plist" ofType:nil];
        //3 从plist中加载 字典数组
        NSArray *dicArray = [NSArray arrayWithContentsOfFile:path];
        //4 存储appInfo的临时数组
        NSMutableArray *appInfos = [NSMutableArray array];
        //5 遍历字典数组，取出每一个字典转换成模型
        for (NSDictionary *dic in dicArray) {
            CZAppInfo *appInfo = [[CZAppInfo alloc] init];
            appInfo.name = dic[@"name"];
            appInfo.icon = dic[@"icon"];
            //把模型添加到模型数组中
            [appInfos addObject:appInfo];
        }
        _appInfos = appInfos;
    }
    return _appInfos;
}
```

3、controller中的改造

要把之前用字典的地方都改成模型appInfo

```
//取得当前遍历到的应用信息
//NSDictionary *dic = self.appInfos[i];
CZAppInfo *appInfo = self.appInfos[i];
```

```objc
//设置图片
iconView.image = [UIImage imageNamed:appInfo.icon];
```

```objc
//设置label上的文字
nameLabel.text = appInfo.name;
```

A07-封装字典转对象

2014年10月14日 星期二　　22:33

1、instancetype和id

instancetype在类型表示上，跟id一样，都是表示任何对象的类型
instancetype只能用在返回类型上，不能跟id一样用在参数类型上
instancetype 比id多一个好处，编译器会检测instancetype的真实类型

id万能类型， NSString *s = [CZAppInfo appInfoWithDic:dic]; 是可以的，没有任何提示
instancetype 会返回特定的类型　　NSString *s = [CZApp appInfoWithDic:dic]; 编译会警告类型不匹配

2、封装字典转模型，在CZAppInfo的构造方法中，封装从字典中取数据并给模型属性赋值

```
@implementation CZAppInfo
//control + command + 上下  .m和.h之间切换
//构造方法 初始化类内部的name和icon属性
- (instancetype)initWithDic:(NSDictionary *)dic
{
    if (self = [super init]) {
        self.name = dic[@"name"];
        self.icon = dic[@"icon"];
    }
    return self;
}

//类方法，快速初始化类的对象
+ (instancetype)appInfoWithDic:(NSDictionary *)dic
{
    return [[self alloc] initWithDic:dic];
}
@end
```

3、controller中的改造

```
for (NSDictionary *dic in dicArray) {
      CZAppInfo *appInfo = [[CZAppInfo alloc] init];
      appInfo.name = dic[@"name"];
      appInfo.icon = dic[@"icon"];
    //调用类方法快速初始化对象
    CZAppInfo *appInfo = [CZAppInfo appInfoWithDic:dic];
    //把模型添加到模型数组中
    [appInfos addObject:appInfo];
}
```

4、在nb一些

1、在controller中

```
- (NSArray *)appInfos
{
    if (_appInfos == nil) {
        _appInfos = [CZAppInfo appInfosList];
```

```
    }
    return _appInfos;
}
```

2、在CZAppInfo中

```
+ (NSArray *)appInfosList
{
    //1 获取当前应用的bundle
    NSBundle *bundle = [NSBundle mainBundle];
    //2 plist文件的路径
    NSString *path = [bundle pathForResource:@"app.plist" ofType:nil];
    //3 从plist中加载  字典数组
    NSArray *dicArray = [NSArray arrayWithContentsOfFile:path];
    //4 存储appInfo的临时数组
    NSMutableArray *appInfos = [NSMutableArray array];
    //5 遍历字典数组，取出每一个字典转换成模型
    for (NSDictionary *dic in dicArray) {
        //调用类方法快速初始化对象
        CZAppInfo *appInfo = [CZAppInfo appInfoWithDic:dic];
        //把模型添加到模型数组中
        [appInfos addObject:appInfo];
    }
    return appInfos;
}
```

A08-xib

2014年10月14日 星期二　　22:52

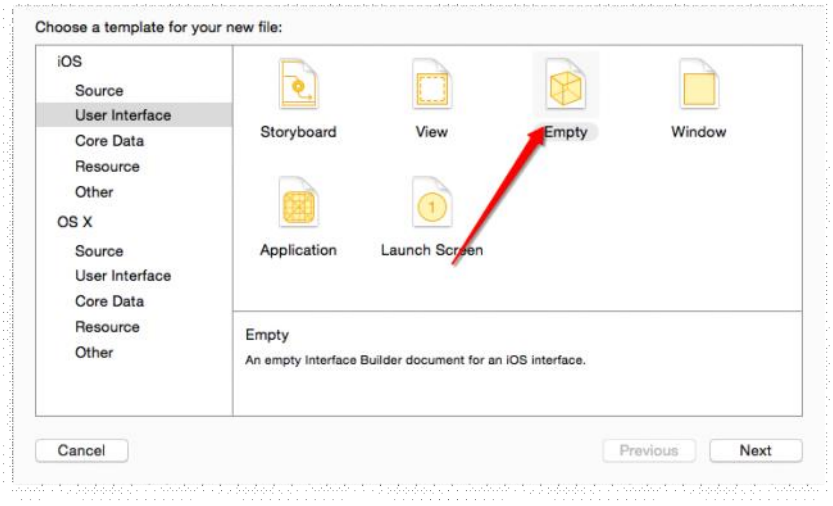
1、xib和storyboard

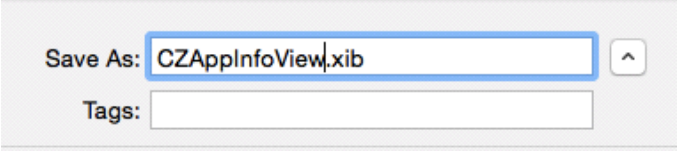
Storyboard　描述软件界面，相对于xib比较重量级，一个storyboard可以有多个场景。可以描述整个软件的所有界面

xib　描述软件界面，一般用来描述一个界面中的某一个部分

2、xib的改造(使用xib封装子view)

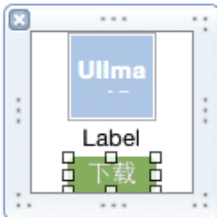
2.1 创建xib





在xib中拖拽一个view。设置xib中的view的大小可调整 设置大小为90*90 和之前的子view的大小一样

2.2 设置子view中的控件



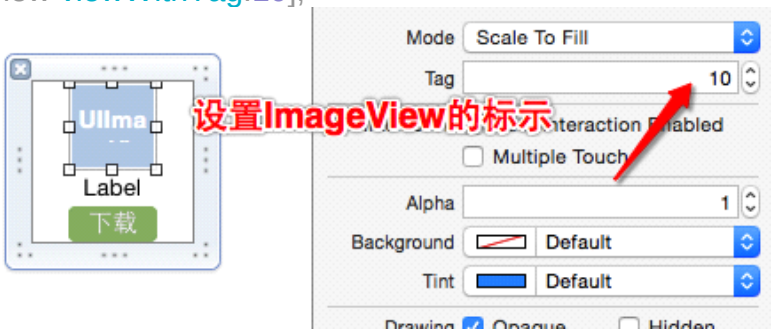
3、从xib中加载子view 改造controller

```
//输出当前应用所在的位置（找到mainBundle）
//NSLog(@"%@",NSHomeDirectory());
```

```
        //2 动态创建子view
//        UIView *view = [[UIView alloc] init];
        //2 从xib中加载子view
        NSBundle *bundle = [NSBundle mainBundle];
        UIView *view = [[bundle loadNibNamed:@"CZAppInfoView" owner:nil
            options:nil] lastObject];
```

4、给子view中的name和icon赋值

[view viewWithTag:10];



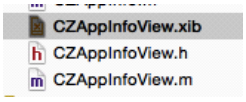

```
//设置icon、name的值
- (void)displaySubViews:(UIView *)view appInfo:(CZAppInfo *)appInfo viewW:
    (CGFloat)viewW
{
    //3 获取xib中的子view，并获取子view中的icon和name控件赋值
    //获取view中子控件的方式
    //view.subviews[0];
    //[view viewWithTag:10];

    //icon赋值
    UIImageView *iconView = view.subviews[0];
    iconView.image = [UIImage imageNamed:appInfo.icon];

    //name赋值
    UILabel *nameLabel = view.subviews[1];
    nameLabel.text = appInfo.name;
```

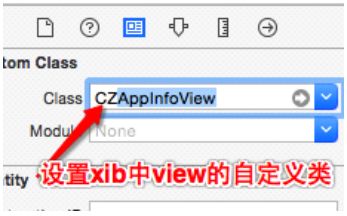
A09-Xib的封装，自定义view

2014年10月14日 星期二　　23:05

1、自定义CZAppInfoView

新建CZAppInfoView类，继承UIView





2、xib和自定义View产生关联后，进行连线

```
@property (weak, nonatomic) IBOutlet UIImageView *iconView;
@property (weak, nonatomic) IBOutlet UILabel *nameView;
```

3、在controller中给name和icon赋值　　测试

```
//取得当前遍历到的应用信息
//NSDictionary *dic = self.appInfos[i];
CZAppInfo *appInfo = self.appInfos[i];

view.nameView.text = appInfo.name;
view.iconView.image = [UIImage imageNamed:appInfo.icon];
```

4、继续自定义view的封装

4.1 添加类方法，快速创建自定义view的对象

```
+ (instancetype)appInfoView
{
    NSBundle *bundle = [NSBundle mainBundle];
    CZAppInfoView *view = [[bundle loadNibNamed:@"CZAppInfoView" owner:nil options:nil] lastObject];
    return view;
}
```

4.2 CZAppInfoView添加模型属性

```
@property (nonatomic, strong) CZAppInfo *appInfo;
```

4.3 重写属性的setter方法，给自定义view的内部的name和icon控件赋值

```
- (void)setAppInfo:(CZAppInfo *)appInfo
{
    _appInfo = appInfo;

    self.nameView.text = appInfo.name;
    self.iconView.image = [UIImage imageNamed:appInfo.icon];
}
```

4.4 改造controller

```
        //2 动态创建子view
//        UIView *view = [[UIView alloc] init];
        //2 从xib中加载子view
//        NSBundle *bundle = [NSBundle mainBundle];
//        CZAppInfoView *view = [[bundle loadNibNamed:@"CZAppInfoView"
    owner:nil options:nil] lastObject];
        //2 从xib中加载子view
        CZAppInfoView *view = [CZAppInfoView appInfoView];
```

```
        //取得当前遍历到的应用信息
        //NSDictionary *dic = self.appInfos[i];
        CZAppInfo *appInfo = self.appInfos[i];
        //给自定义view的模型属性赋值（setter方法中给zidingyiview内部的子控件赋值）
        view.appInfo = appInfo;

/         view.nameView.text = appInfo.name;
/         view.iconView.image = [UIImage imageNamed:appInfo.icon];
```

A10-xib自定义view的总结

2014年10月14日 星期二　　23:05

1、封装的好处：

A08-xib中完成的代码 controller得知道xib中具体的控件，产生依赖

为了减少依赖，把xib内部控件的赋值给封装起来

如果一个view内部的子控件比较多，一般会考虑自定义一个view，把它内部的子控件的创建屏蔽起来，不让外界关心，这样不管view内部怎么变化外界都不需要知道

外界可以传入对应的模型数据给自定义view，view拿到模型数据后给内部的子控件设置对应的数据

2、使用xib封装一个自定义view的步骤

1、新建一个继承UIView的自定义view，类名CZAppInfoView
2、创建xib（xib的名字和自定义view的名字一样CZAppInfoView），来描述CZAppInfoView的内部结构
3、修改xib中的UIView的类型改为CZAppInfoView
4、把xib中控件连线到CZAppInfoView
5、CZAppInfoView提供一个模型属性
6、重写模型的set方法，在set方法中给xib中对应的子控件赋值

7、写一个appInfoView类方法，封装加载xib创建CZAppInfoView的过程

3、封装的结果

代码结构清晰
简单的mvc

A11-简单MVC

2014年10月14日 星期二 23:05

Model
View
Controller

A12-xib加载过程

2014年10月14日 星期二　　23:05

xib的加载过程,从xml中加载进来对界面的描述，并以此创建出来

```
JSAppView *view = [[JSAppView alloc] init];
View.frame = CGRectMake(0,0,85,90);

UIImageView *imageView = …….;
[view addSubView:imageView];

……
```

xib和storyboard的共同点

都用来描述软件界面

都用Interface Builder工具来编辑

都使用xml来存储 对界面的描述

xib和storyboard的区别，查看xib和storyboard的xml代码发现他们的区别仅仅是xib少了Scenes和viewController

xib是轻量级的，用来描述局部的UI界面

stroryboard是重量级的，用来描述整个软件的多个界面，并且能展示多个界面之间的跳转关系

storyboard是ios5以后才有的

X01-掌握

2014年10月14日 星期二　　20:10

UIView的常见属性和方法

九宫格的计算方法

字典转对象

xib的使用

自定义view （view的封装）

简单的MVC

X02-xcode中的常用快捷键

2014年10月14日 星期二　　20:10

把快捷键组合起来用，就跟游戏中的各种大招一样

各种新建
shift + comand + n　　新建xcode项目
option + command + n 新建分组
command + n 新建文件

搜索
shift + command + o
command + f

控制tab
command + t 新建tab
command + w 关闭tab

控制窗口的显示隐藏
comand + 0 隐藏显示导航窗口
comand + 1 。。。。n 显示切换导航窗口中的几个内容

option + command + 0 控制工具窗口
option + command + 1.。。。。n

文件之间的跳转
control + comand + 上/下　　.m/.h文件切换
control + comand + 左/右　　上一个和下一个文件之间的切换

代码操作
控制光标
shift + command + 方向键的左右 选中正行　上下 选中所有
shift + option + 方向键的左右 选中单词　上下选中行
去掉shift 实验

control + command + j 查看定义

control + 6 快速切换当前文档中的方法，支持智能输入

编译运行项目
　　command + r
编译不运行
　　command + b

停止

command + .

折叠代码块

option + command + 左/右　　折叠/展开代码块

X03-常见问题及解决

2014年10月15日 星期三　　9:17

1、手贱把

```
#import "JSAppInfo.h"
```

写成了

```
#import "JSAppInfo.m"
```

```
 4 duplicate symbols for architecture x86_64
clang: error: linker command failed with exit code 1 (use -v to see invocation)
```

2、 xx.h has been modified since the precompiled header

1、删除 /Users/xxxx/Library/Developer/Xcode/DerivedData

2、clean 下项目 重新编译

X04-容易忘记的

2014年10月14日 星期二　　21:57

//使控件产生圆角

```
nameView.layer.cornerRadius = 5;
nameView.layer.masksToBounds = YES;
```

类扩展Extensions

```
@interface JSAppInfoView ()

@end
```

分类Categories

```
@interface JSAppInfoView (ex)

@end
```

X05-上课笔记

2014年10月24日 星期五　　上午9:35

UI基础9天
1，2 学基础控件

3，4 用基础控件做练习

5 UIScrollView
6，7，8，9 UITableView

学习之美

玩什么
- 1、思路
- 2、代码实现
- 3、解决问题的方法

玩方法
- 1、做笔记
- 2、敲代码 3遍
- 3、勤问
- 4、不要去看视频
- 5、上课哪走神了，从周公那回来后把记录下时间
- 带着问题去学

工欲善其事，必先利其器
- xmind
- xcode
- onenote
- Snip

- …….

上课纪律
- 多说话，但不说废话
- 完成作业

App

Web app

Native App

Hybrid App

1、setter getter（字段(成员变量) 和属性）

```
- (NSString *)name
{
   return _name;
}

- (void)setName:(NSString *)name
{
   _name = name;
}
```

2、方法和函数 方法==函数

方法：类里面的函数：方法

函数：c语言中的函数：函数

3、方法的封装

把一段代码封装到一个方法中

一个方法只做一件事情

4、面向对象：封装、继承、多态

多态：里氏替换原则 CZPerson *person = [[CZStudent alloc] init];

X06-作业

2014年10月24日 星期五　　下午4:48

1、代码动态生成控件的方式实现 应用管理

2、使用xib的方式 -应用管理

3、自己实现点击下载按钮*