# 无限互联

**Infinite Interconnection**

无限互联是国内唯一一家**专注**于iPhone和iPad软件开发培训机构，到目前为止为各大公司输送了一大批优秀的iOS高级软件研发人才。随着iOS6系统的发布，我们也在当天陆续发布**国内首套完整**的iOS开发的视频教程，手把手教您写代码，从入门到熟练再到精通。

**高薪就业**是检验一家培训机构质量的唯一标准，我们的学员高薪就业是对我们最好的肯定，也是我们前进的最强烈的动力，我们感谢同学们的努力，感谢你们对我们的支持！我们也将免费为你们提供最好的就业后的技术支持！

**亲爱的同学们，你们的成功才是我们最大的成功！**

http://www.iphonetrain.com

# 无限互联

# Infinite Interconnection

## 类的扩展

主讲：周泉

http://www.iphonetrain.com

# 本课学习内容

- 类目的基本概念和用法
- 延展的基本概念和用法
- 协议的基本概念和用法
- 委托设计模式的概念和用法

# 1、类目的基本概念和用法

- 类目（Category）的基本概念

封装是面向对象的一个特征，OC也不意外，但是有的时候我们会碰到这样一种情况，比如我封装了一个类，不想再动它了，可是随着程序功能的增加，需要在那个类中增加一个小小的方法，这时我们就不必在那个类中做修改或者在定义一个子类，只需要在用到那个方法时随手添加一个该类的类别（category)即可。

（1）在类目定义的方法，会成为原始类的一部分，与其他方法的调用没有区别;

（2）通过给父类定义类目方法，其子类也会继承这些方法。如果子类添加类目方法，父类则不会拥有子类的类目方法;

# 类目的应用和局限

- 类目方法的应用
  - 对现有类进行扩展：比如，你可以扩展Cocoa touch框架中的类，你在类目中增加的方法会被子类所继承，而且在运行时跟其他的方法没有区别。
  - 作为子类的替代手段：不需要定义和使用一个子类，你可以通过类目直接向已有的类里增加方法。
  - 对类中的方法归类：利用category把一个庞大的类划分为小块来分别进行开发，从而更好的对类中的方法进行更新和维护。
- 类目方法的局限性
  - 无法向类目中，添加新的实例变量，类目没有位置来容纳实例变量。如果想增加类的实例变量，只能通过定义子类的方式。
  - 如若，在类目中覆盖现有类的方法。这样会引起super消息的断裂，因为类目中的方法具有更高的优先级，因此，一般不要覆盖现有类中的方法。

# 类目的定义

- 类目的命名与用法

  类目的命名规则：类名+扩展方法，如“NSString+Revert”。

  类目的接口声明与类的定义十分相似，但类目不继承父类，只需要带有一个括号，表明该类目的主要用途。

```
#import <Foundation/Foundation.h>
// .h文件 类目的定义
@interface NSString (Revert)
- (void)test;
@end
```

```
#import <Foundation/Foundation.h>
// .m文件 类的定义
@interface Revert : NSObject
- (void)test;
@end
```

```
#import "NSString+Revert.h"
// .h文件 类目方法的实现
@implementation NSString (Revert)
- (void)test {
} @end
```

```
#import "Revert.h"
// .m文件 类中方法的实现
@implementation Revert
- (void)test {
} @end
```

# 类目的定义

- 类目的命名与用法

  类目的命名规则：类名+扩展方法，如“NSString+Revert”。

  类目的接口声明与类的定义十分相似，但类目不继承父类，只需要带有一个括号，表明该类目的主要用途。

```
#import <Foundation/Foundation.h>
// .h文件 类目的定义
@interface NSString (Revert)
- (void)test;
@end
```

```
#import <Foundation/Foundation.h>
// .m文件 类的定义
@interface Revert : NSObject
- (void)test;
@end
```

```
#import "NSString+Revert.h"
// .h文件 类目方法的实现
@implementation NSString (Revert)
- (void)test {
} @end
```

```
#import "Revert.h"
// .m文件 类中方法的实现
@implementation Revert
- (void)test {
} @end
```

# 类目的定义

- 类目的命名与用法

  类目的命名规则：类名+扩展方法，如“NSString+Revert”。

  类目的接口声明与类的定义十分相似，但类目不继承父类，只需要带有一个括号，表明该类目的主要用途。

```
#import <Foundation/Foundation.h>
// .h文件 类目的定义
@interface NSString (Revert)
- (void)test;
@end
```

```
#import <Foundation/Foundation.h>
// .m文件 类的定义
@interface Revert : NSObject
- (void)test;
@end
```

```
#import "NSString+Revert.h"
// .h文件 类目方法的实现
@implementation NSString (Revert)
- (void)test {
} @end
```

```
#import "Revert.h"
// .m文件 类中方法的实现
@implementation Revert
- (void)test {
} @end
```

# 2、延展基本概念和用法

- 延展（Extension）的基本概念和用法
  - 类的延展就如同是“匿名”的类目，延展中声明的方法在类本身的@implementation和它对应的@end之间实现
  - 类有时需要方法只有自己所见，我们可以通过延展的方式定义类的私有方法

```
#import "Person.h"
// .m文件中 延展的声明
@interface Person ()
- (void)private;
@end
```

```
@implementation Person
- (void)private {
    // .m文件中 doing something
}
@end
```

```
#import <Foundation/Foundation.h>
// .h文件中 类目的声明
@interface Person (Creation)
- (void)initPerson;
@end
```

```
@implementation Person (Creation)
- (id)initPerson {
    return nil; // .m文件中
}
@end
```

# 2、延展基本概念和用法

- 延展（Extension）的基本概念和用法
  - 类的延展就如同是“匿名”的类目，延展中声明的方法在类本身的@implementation和它对应的@end之间实现
  - 类有时需要方法只有自己所见，我们可以通过延展的方式定义类的私有方法

```
#import "Person.h"
// .m文件中 延展的声明
@interface Person ()
- (void)private;
@end
```

```
@implementation Person
- (void)private {
    // .m文件中 doing something
}
@end
```

```
#import <Foundation/Foundation.h>
// .h文件中 类目的声明
@interface Person (Creation)

- (void)initPerson;
@end
```

```
@implementation Person (Creation)
- (id)initPerson {
    return nil; // .m文件中
}
@end
```

# 2、延展基本概念和用法

- 延展（Extension）的基本概念和用法
  - 类的延展就如同是“匿名”的类目，延展中声明的方法在类本身的@implementation和它对应的@end之间实现
  - 类有时需要方法只有自己所见，我们可以通过延展的方式定义类的私有方法

```
#import "Person.h"
// .m文件中 延展的声明
@interface Person ()
- (void)private;
@end
```

```
@implementation Person
- (void)private {
    // .m文件中 doing something
}
@end
```

```
#import <Foundation/Foundation.h>
// .h文件中 类目的声明
@interface Person (Creation)
- (void)initPerson;
@end
```

```
@implementation Person (Creation)
- (id)initPerson {
    return nil; // .m文件中
}
@end
```

# 3、协议的基本概念和用法

- 协议（Protocol）的基本概念

  协议的声明看起来比较类似一个类的接口，不同的是协议没有父类也不能定义实例变量。协议是一种特殊的程序设计结构，用于声明专门被别的类实现的方法。协议在以下场合非常有用：

  - 需要由别的类实现的方法
  - 声明未知类的接口
  - 两个类之间的通信

- 协议的基本特点
  - 协议可以被任何类实现的方法
  - 协议本身不是类，它是定义了一个其他类可实现的接口
  - 类目也可以采用协议

# 协议的声明与实现

- 协议中的关键字
  - @required：表示必须强制实现的方法
  - @optional：表示可以有选择性的实现方法

```
/* 协议的声明 */
@protocol HelloProtocol <NSObject>
@optional
- (void)optionalMethod1;
- (void)optionalMethod2;
@required
- (void)requiredMethod1;
@end
/* 采用了该协议 */
@interface Person : NSObject
<HelloProtocol, OtherProtocol>
@end
```

```
@implementation Person
- (void)requiredMethod1 {
} // 实现了该协议中的方法，且方法必须实现

- (void)optionalMethod1 {

} // 实现了该协议中的方法，可以选择不实现

- (void)optionalMethod2 {

} // 实现了该协议中的方法，可以选择不实现
@end
```

# 4、代理（委托）设计模式

- 代理设计模式的基本概念

  代理是指一个对象提供机会对另一个对象中的行为发生变化时做出的反应。

  如，当你将一颗石子（对象1）丢入水中（行为发生变化，之前可能在你的手中）时，水面（对象2）泛起波纹（做出的反应）。

  总而言之，代理设计模式的基本思想——两个对象协同解决问题，通常用于对象之间的通信。

- 代理设计模式的基本特点
  - 简化了对象的行为，最小化了对象之间的耦合度
  - 使用代理，一般来说无需子类化
  - 简化了我们应用程序开发，既容易实现，又灵活

# 实例：“中介找房”

- 课堂实例

  假设有一个Jack的人（Person），他想租一套公寓（Apartment），由于他工作繁忙（或者其他原因），没有时间（或者自身没有能力）去租房。因此，他委托中介公司（Agent）帮他寻找房源，找到合适的房源告知Jack。

| Person 类 | 协议 | Agent 类 |
| --- | --- | --- |
| •name<br>•delegate<br>•houseRent | •lookingForApartment | •lookingForApartment |
| •init<br>•findHouse | | |

# 实例中涉及知识点

- 定时器（NSTimer）的基本概念

  一旦创建了一个定时器对象（NSTimer实例），它可以按照一定时间的间隔，将指定消息发送到目标对象，并更新某个对象的行为，你可以选择在未来的某个时候将它“开启”，或者将它停止乃至销毁。

- NSRunloop基本概念

  一个runloop就是一个事件处理的循环，用来不停的调度工作以及处理输入事件。使用 run loop 的目的是让你的线程在有工作的时候忙于工作，而没工作的时候处于休眠状态。

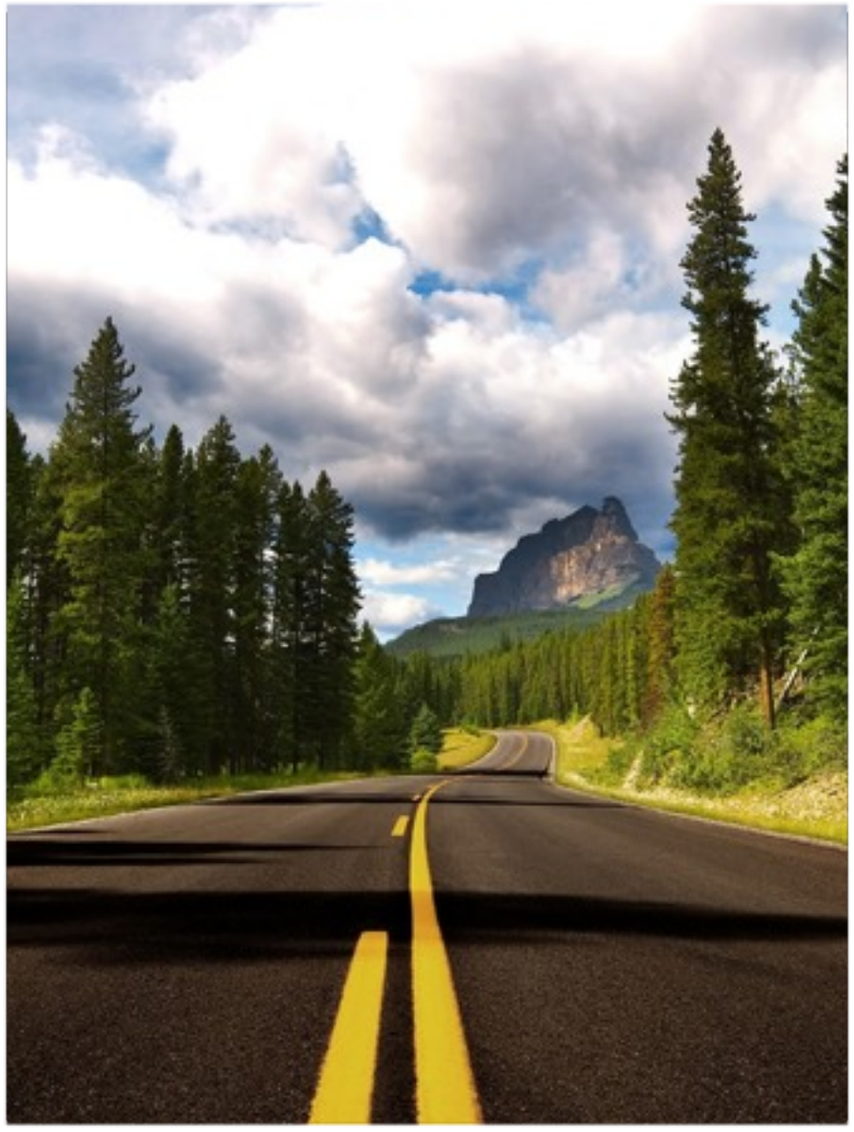
  在我们的应用程序中，你不需要创建NSRunloop对象。因为，在我们的主线程中（包含其他子线程）系统会自动创建NSRunloop对象。如果你需要访问当前线程中的runloop，你可以通过类方法“currentRunloop”

# 今日习题

- 掌握类目、延展、协议以及代理设计模式的基本概念
- 定义一个Children类和一个Nurse类，并实例化一个保姆对象和孩子对象。当孩子哭的时候，告诉保姆“我需要你陪我玩”，当孩子饿的时候，要告诉保姆“我需要吃饭”，当孩子不舒服的时候，需要告诉保姆“我需要吃药”，当孩子玩脏了时候，要告诉保姆“我需要洗澡”。保姆照顾孩子的行为，在控制台打印出来即可。

# 下节预告

- 下一节课将会学习“内存管理”
  - 变量在内存中的位置
  - 对象所有权和引用计数的基本概念
  - 点语法中的内存管理
  - 垃圾回收机制和ARC计数的基本概念



无限互联
Infinite
Interconnection





无限互联网站：http://www.iphonetrain.com

博客：http://blog.csdn.net/xbiii3s

http://hi.baidu.com/new/xiongbiao0327

公司E-mail：wxhl2805@gmail.com

老师E-mail：xbiii3s@gmail.com

视频讲解过程中如有不妥之处，欢迎大家将信息反馈到我的Email中，我们会努力完善！谢谢各位的支持。

# 视频持续更新中... 敬请期待！