Sinomac Developer Connection
渤麥堂 本番坊開發者聯盟

Cocoa编程之新手上路：第二课

我们在上一课里制作了一个简单的程序，我们学习了：

用Project Builder (PB) 建立新的 project
用 Interface Builder (IB) 设计程序界面
生成类，定义 outlet（插头）和action （动作）.
连接界面和程序的执行代码 (connection).

现在我们来籍此分析一下其背后的Cocoa 面向对象编程思想。

图2-1 界面(interface) 和 内部构造（implementation) 的隔离的例子

来看一个现实世界中的一个机械钟表 (图2-1)，对于一个使用者来说，他所看到的的只是一个时间显示，一个界面（interface), 他看不到也不关心这个钟表的内部是如何构造来实现报时的功能，钟表的界面（interface) 把内部机构（implementation) 屏蔽隔离开来，只有钟表的制作者知道并关心内部的构造，他的责任就是让钟表提供给用户一个实现报时的功能的界面，并把内部的实际构造（implementation) 封装起来，使外界看不到。

拿到程序设计中，我们就是钟表匠，我们用程序语言来描述钟表这样这样一个能报时的物体，（这个过程叫抽象(abstraction)，就是抛开表面的细节，抓住事物的本质，因果关系，这是我们人类认识世界的方法）。我们要使用户能通过与这个“物体”的交互得到需要的功能（指示时间），而我们作为制作者要把内部的实施部分隔离开来，因为用户不关心内部的构造。这时我们可以把这样的一个物体叫做”对象“ （object).

一个对象（object) 包含一定的行为或功能(function)和实现这些功能的内部结构(structure)，内部的结构（数据结构）是封装起来的，外界看不到，在这个对象的内部，外界只能通过得到这个对象的行为或功能。这些功能叫做这个对象的方法（method) , 而内部的数据结构就叫做他的实例变量（instance variable). 方法把内部的实例变量（数据部分）包裹起来，使之与外界隔离。

图 2-2 一个对象 (object)

object = instance variables + instance methods

一个程序里有很多的object, 他们之间的通讯是相互之间传递消息（message）. 当一个对象接受到其他对象传递来的消息时，就会相形的执行自己内部的方法（method).

Objective-C 不同于其他面向对象语言（如 C++)的一个显著特点是，它不需要你事先知道要传递给消息的对象是哪个类的，这叫做动态绑定（动态捆绑）或运行时绑定，也就是说消息会在运行的时候才被和对象绑到一块（结合），而不是在编译的时候去做绑定。这样的好处是非常的灵活，给予程序员极大的自由度。比如，我们要在一个文本编辑程序中截取（cut) 文本或图片或表格等，程序不必事先知道要发送 cut:消息给那一类的目标对象。

objective－C 里面， 用一个id变量来表示一个对象，（既 id object 这样的格式），实际上就是一个指向内存中的对象数据。objective－C运行的时候会检查核对 这个id 指针到底指向那一类的对象。就是我们所说的动态绑定。

消息的传递是这样格式表示

［消息接收者 消息］

假如我有一条狗叫阿花， 我传递给他一个指令（消息）：趴下！， 那摸就应该写成

［阿花 趴下］

那摸阿花就会根据受到的消息，来启动它内部的机构，用它自己的方法， 来产生趴下的动作，来回应我。

我们会在接下来的代码讲解中逐步了解更多的 objective－C 语法和结构。

我们用interface builder (IB) 不仅是制作了界面，而且我们还定义个一个类（class)
-Controller, 这是根据 NSObject 这个”祖宗“ 类”而生出的“孩子“（子类）。

在Objective-C 里面定义一个类有些似与在 C里面定义一个结构。

在C语言里定义一个结构是这样：
```
struct key {
char *word;
int count;
```

};

但 在Objective-C 里面的类定义的不同之处是类里面不仅要有数据部分还要要定义方法（method)， 即使数据部分相同而对数据的操作,既方法（ method)不同，也属于两个不同的类。

在Objective-C 里面类定义有两个部分一个是界面（interface)，, 另一个是实施部分（implementation)，这反映了我们上面提到的界面 / 实施的分隔思想。

在界面的部分我们只是声明类的实例变量和方法，以及它的超类（super class).
在实施部分我们才会具体的定义类的方法，既写方法的代码。

controller类的界面部分－controller.h

```
/* Controller.h */

#import

@interface Controller : NSObject
{
IBOutlet id textAtTop;
IBOutlet id textBox;
}
- (IBAction)setTextButtonPressed:(id)sender;
- (IBAction)setURLPressed:(id)sender;
@end
```

Obeject-C 的 ＃import 语句就类似于 c语言的 ＃include 。但比#include 有极大改进， 如果＃import 里指定的文件被引入（imported) 一次之后，不会再来第二次。
在 c语言程序里经常会有这样的语句

```
／* a c # include file - sample.h*/

# ifndef __theSymbol__
# define __theSymbol__

...
```

/需要include的代码，只要include 一次*/

endif

这个c语言程序首先检查符号（theSymbol)是否有定义，如果没有定义（if not defined =ifndef) , 就定义这个符号并接着处理下面的代码。这种做法效率不高而且极其危险。以为include 文件一般都会多次执行，另外，如果其他文件定义了的符号（如这里的theSymbol)， 那磨这个sample.h里就不会再执行定义了。

在Objective－ C里面，＃import帮你搞定，他只会导入没有读过的文件。你的程序只要简单的一句：
＃ import

接下来的@interface Controller : NSObject 是告诉编译器我们要定义的类（class)叫做 Controller， 这个类的是从NS Object继承迩来。既controller 是NSObject的孩子（子类）， NSObject 是祖先（超类）。也也就是说controller的类的每个对象中会有NSObject 对象的相同变量的拷贝。

Controller类的实施部分－Controller.m

```
/*Controller.m*/

#import "Controller.h"

@implementation Controller

- (IBAction)setTextButtonPressed:(id)sender
{
[textAtTop setStringValue:[textBox stringValue]];

}

- (IBAction)setURLPressed:(id)sender
{

[textAtTop setStringValue:[sender title]];
}

@end
```

接着我们还给Controller这个类添加 了outlet 和 action.

我们现在来看看 outlet 和action 到底是甚末。

OUTLET CONNECTION 插座连接

我们 有了Controller 这个类并生成了它的实例对象(Object)，我们需要它来控制我们的程序界面上的对象， 如按钮，对话框，并使按钮和对话框之间传递文本，达到我们需要的目的， 但如何来实现这些操作呢？

在 Cocoa里面我们用一个叫做“插座和连接”的机构来实现。

插座（ outlet) ，可以理解为日常生活中的电源插座，在早期的开发工具版本里，outlet 就是用一个电源插座的图标表示，）

在cocoa 里面，一个插座就是一个实例变量（instance varibale), 他的类型是id,
我们知道 id实际上是一个指向对象的指针（pointer). 所以，我们就可以用这个id 指针来指想另外一个我们程序界面（nib)中的对象, 如按钮，文本框。也就是说，在插座里面储存另一个对象的 id. 这就叫做连接（connection).

我们在程序界面建立的了两个 outlets, textBox 指向了输入文本框，textAttop 则指向了另一个上面的显示文本框 。

我们 在 IB 里面用Control -拖动的方法建立了 outlet connection. 我们注意到拖动的方向永远是消息传递的方向，比如这里我们建立outlet connection 时，是从Controller 拖向文本框，因为 controller对象要发送消息给他们，让他们读取和显示文本。

在Interface Buidler （IB) 的界面文件（nib)里可以储存我们设定的插座和连线，当 IB读取一个nib 文件时，这些信息也是自动恢复的。

TARGET AND ACTION 目标和动作

当一个对象接受到一个消息被要求产生一系列的动作(action)，既执行它的方法（ method)时，我们称这个对象叫做目标对象（ target object), 而发出动作消息的对象叫做控制对象（control object),.

这个被要求执行的动作方法(action method), 比较特殊。 因为动作方法只有一个参数 sender， 也就是发送消息出来的对象的 id.

当我们建立了两个action method, 在代码中用 IBAction 表示一个动作。我们采取control＋拖动的方法使他们和两个按钮分别连接起来，注意拖动的方向是消息传递的方向，即从按钮拖向 controller 对象。

当程序运行时，如果按钮被按动的时候就产生了一个事件，这会自激发动作， 发送动作消息action messge给 controller对象， 让它执行 action method.

在我们的代码里面， 我们用 IBOutlet 关键字表示 outlet, 用 IBAction 表示一个动作

controller类的界面部分－controller.h

```
/* Controller.h */

#import

@interface Controller : NSObject
{
IBOutlet id textAtTop;
IBOutlet id textBox;
}
- (IBAction)setTextButtonPressed:(id)sender;
- (IBAction)setURLPressed:(id)sender;
@end
```

在 interface 文件中，只是声明了两个outlet, 两个actiion.

Controller类的实施部分－Controller.m

```
/*Controller.m*/

#import \"Controller.h\"

@implementation Controller

- (IBAction)setTextButtonPressed:(id)sender
{
[textAtTop setStringValue:[textBox stringValue]];
```

```
}

- (IBAction)setURLPressed:(id)sender
{

[textAtTop setStringValue:[sender title]];
}

@end
```

分别来看 action method 里面，
－(IBAction)setTextButtonPressed:(id)sender

这个名字叫做setTextButtonPressed 的action method, 只有一个参数（冒号后面的是参数）,sender, sender 的类型是id, 既 sender是一个发出消息的对象。

```
{
[textAtTop setStringValue:[textBox stringValue]];
```

这个擦提哦内 actionmethod 的具体作甚磨实在这一行代码里定义的， 它首先要求取得BtextBox 这个 outlet所指的输入文本框里的字符串，这是string Value 这个 method 来实现的，去取得字符串后返回给 textAtTop 这个 outlet 所指向的显示文本框，然后特性台调用setStringValue方法在当前文本框里 写出这个字符串。

下面的method , 取得url 按钮的title ,按钮上的文字，作为参数返回到 textAttop 的setString Value 方法将会显示出来这个string .

好了，代码将完了。

稍后我们在做练习，巩固 target.action, outlet connection 这些基本概念。！

日期: 2004/6/17
章节: MacNuts 讲 Cocoa
这篇文章的网址在:
http://www.sinomac.com/modules/sdcection/article.php?articleid=8