Runtime 示例代码 (v1#nxb)

动态方法解析

```
#import "Foo.h"
#include <objc/runtime.h>

void dynamicMethodIMP(id self, SEL _cmd) {
    NSLog(@" >> dynamicMethodIMP");
}

@implementation Foo

-(void)Bar
{
    NSLog(@" >> Bar() in Foo");
}

+ (BOOL)resolveInstanceMethod:(SEL)name
{
    NSLog(@" >> Instance resolving %@", NSStringFromSelector(name));

    if (name == @selector(MissMethod)) {
        class_addMethod([self class], name, (IMP)dynamicMethodIMP, "v@:");
        return YES;
    }

    return [super resolveInstanceMethod:name];
}

+ (BOOL)resolveClassMethod:(SEL)name
{
    NSLog(@" >> Class resolving %@", NSStringFromSelector(name));

    return [super resolveClassMethod:name];
}

@end
```

消息转发

```
- (id)forwardingTargetForSelector:(SEL)aSelector{
    NSLog(@"%s",__func__);
    return [super forwardingTargetForSelector:aSelector];
}

- (NSMethodSignature *)methodSignatureForSelector:(SEL)aSelector{
    NSLog(@"%s",__func__);
    if (aSelector == @selector(run)) {
        // forwardingTargetForSelector 没有实现 就只能方法签名了
        Method method    = class_getInstanceMethod(object_getClass(self), @selector(readBook));
        const char *type = method_getTypeEncoding(method);
        return [NSMethodSignature signatureWithObjCTypes:"v@:@"];
    }
    return [super methodSignatureForSelector:aSelector];
}

- (void)forwardInvocation:(NSInvocation *)anInvocation{
    NSLog(@"%s",__func__);
    }
}
```

添加属性(分类, 关联方法)

.h

```
#import <UIKit/UIKit.h>

@interface UIControl (RYButton)
// 声明一个时间间隔
@property (assign,nonatomic)NSTimeInterval  ry_time;

@end
```

.m 需手动实现set和get方法

```
#import "UIControl+RYButton.h"
#import <objc/runtime.h>
static const char * RY_CLICKKEY = "ry_clickkey";
@implementation UIControl (RYButton)

- (void)setRy_time:(NSTimeInterval)ry_time{
    objc_setAssociatedObject(self, RY_CLICKKEY, @(ry_time),
OBJC_ASSOCIATION_RETAIN_NONATOMIC);

}
- (NSTimeInterval)ry_time{
    return [objc_getAssociatedObject(self, RY_CLICKKEY) doubleValue];

}

@end
```

```
UIButton * button = [UIButton buttonWithType:UIButtonTypeCustom];
button.ry_time = 1.0f;
```

移除关联 :objc_removeAssociatedObjects(self)

动态添加方法

```
Person *p = [[Person alloc]init];
[p performSelector:@selector(eat:)withObject:@"log"]
```

```
+(BOOL)resolveInstanceMethod:(SEL)sel
{
    if (sel == @selector(eat:)) {
        class_addMethod(self, sel, (IMP)aaaa , "v@:@");
        return YES;
    }
    return [super resolveInstanceMethod:sel];
}
void aaaa(id self ,SEL _cmd,id Num)
{
    // 实现内容
    NSLog(@"%@的%@方法动态实现了,参数
为%@",self,NSStringFromSelector(_cmd),Num);
}
```

交换方法

将UIImageView的 initWithImage:方法换成一个自 定义的方法.

```
#import "UIImageView+category.h"
#import <objc/runtime.h>
 + (void)initialize{
    Method m1 = class_getInstanceMethod([UIImageView class],
@selector(initWithImage:));
    Method m2 = class_getInstanceMethod([UIImageView class],
@selector(initWithResizableImage:));
    method_exchangeImplementations(m1, m2);
}

- (instancetype)initWithResizableImage:(UIImage *)image {
    NSLog(@"新方法");
    return [[UIImageView alloc]init];
}
@end
```

遍历属性

1 遍历代码:

```
unsigned int count = 0;
    Ivar *ivars = class_copyIvarList([UIPageControl class], &count);
    for (int i = 0; i < count; i++) {
        Ivar ivar = ivars[i];
        //获取所有属性
        const char *property = ivar_getName(ivar);
        NSLog(@"%@",[[NSString alloc]initWithCString:property
encoding:NSUTF8StringEncoding]);
    }
```

2 利用KVC给私有属性赋值setValue:forKey:

```
[self.pageControl setValue:[UIImage imageNamed:@"compose_keyboard_dot_normal"]
forKey:@"_pageImage"];
    [self.pageControl setValue:[UIImage imageNamed:@"compose_keyboard_dot_selected"]
forKey:@"_currentPageImage"];
```