返回目录:全网各大厂 iOS 面试题-题集大全

反射是什么？可以举出几个应用场景么？

系统 Foundation 框架为我们提供了一些方法反射的 API，我们可以通过这些 API 执行将字符串转为 SEL 等操作。由于 OC 语言的动态性，这些操作都是发生在运行时的。

```
// SEL 和字符串转换
FOUNDATION_EXPORT NSString *NSStringFromSelector(SEL aSelector);
FOUNDATION_EXPORT SEL NSSelectorFromString(NSString *aSelectorName);
// Class 和字符串转换
FOUNDATION_EXPORT NSString *NSStringFromClass(Class aClass);
FOUNDATION_EXPORT Class __nullable NSClassFromString(NSString *aClassNa
me);
// Protocol 和字符串转换
FOUNDATION_EXPORT NSString *NSStringFromProtocol(Protocol *proto) NS_AV
AILABLE(10_5, 2_0);
FOUNDATION_EXPORT Protocol * __nullable NSProtocolFromString(NSString *
namestr) NS_AVAILABLE(10_5, 2_0);
```

通过这些方法，我们可以在运行时选择创建那个实例，并动态选择调用哪个方法。这些操作甚至可以由服务器传回来的参数来控制，我们可以将服务器传回来的类名和方法名，实例为我们的对象。

```
// 假设从服务器获取 JSON 串，通过这个 JSON 串获取需要创建的类为 ViewController，
并且调用这个类的 getDataList 方法。
Class class = NSClassFromString(@"ViewController");
ViewController *vc = [[class alloc] init];
SEL selector = NSSelectorFromString(@"getDataList");
[vc performSelector:selector];
```

反射机制使用技巧

假设有一天公司产品要实现一个需求：根据后台推送过来的数据，进行动态页面跳转，跳转到页面后根据返回到数据执行对应的操作。

遇到这样奇葩的需求，我们当然可以问产品都有哪些情况执行哪些方法，然后写一大堆 if else 判断或 switch 判断。 但是这种方法实现起来太 low 了，而且不够灵活，假设后续版本需求变了，还要往其他已有页面中跳转，这不就傻眼了吗.... 这种情况反射机制就派上用场了，我们可以用反射机制动态的创建类并执行方法。当然也可以通过 runtime 来实现这个功能，但是我们当前需求反射机制已经足够满足需求了，如果遇到更加复杂的需求可以考虑用 runtime 来实现。 这时候就需要和后台配合了，我们首先需要和后台商量好返回的数据结构，以及数据格式、类型等，

返回后我们按照和后台约定的格式，根据后台返回的信息，直接进行反射和调用即可。

假设和后台约定格式如下：

```
@{
     // 类名
     @"className" : @"UserListViewController",
     // 数据参数
     @"propertys" : @{ @"name": @"liuxiaozhuang",
                       @"age": @3 },
     // 调用方法名
     @"method" : @"refreshUserInformation"
 };
```

定义一个 UserListViewController 类，这个类用于测试，在实际使用中可能会有多个这样的控制器类。

```
#import <UIKit/UIKit.h>
// 由于使用的 KVC 赋值，如果不想把这两个属性暴露出来，把这两个属性写在.m 文件也可以
@interface UserListViewController : UIViewController
@property (nonatomic,strong) NSString *name;/*!< 用户名 */
@property (nonatomic,strong) NSNumber *age;/*!< 用户年龄 */
/** 使用反射机制反射为 SEL 后，调用的方法 */
- (void)refreshUserInformation;
@end
```

下面通过反射机制简单实现了控制器跳转的方法，在实际使用中再根据业务需求进行修改即可。因为这篇文章主要是讲反射机制，所以没有使用 runtime 代码。

简单封装的页面跳转方法，只是做演示，代码都是没问题的，使用时可以根据业务需求进行修改。

```
- (void)remoteNotificationDictionary:(NSDictionary *)dict {
    // 根据字典字段反射出我们想要的类，并初始化控制器
    Class class = NSClassFromString(dict[@"className"]);
    UIViewController *vc = [[class alloc] init];
    // 获取参数列表，使用枚举的方式，对控制器属性进行 KVC 赋值
    NSDictionary *parameter = dict[@"propertys"];
    [parameter enumerateKeysAndObjectsUsingBlock:^(id  _Nonnull key, id
  _Nonnull obj, BOOL * _Nonnull stop) {
        // 在属性赋值时，做容错处理，防止因为后台数据导致的异常
        if ([vc respondsToSelector:NSSelectorFromString(key)]) {
            [vc setValue:obj forKey:key];
        }
    }];
    [self.navigationController pushViewController:vc animated:YES];
    // 从字典中获取方法名，并调用对应的方法
```

```
    SEL selector = NSSelectorFromString(dict[@"method"]);
    [vc performSelector:selector];
}
```

返回目录:全网各大厂 iOS 面试题-题集大全

更多精选大厂 · iOS 面试题答案 PDF 文集

*