iOS开发·KVC:字典转模型,防止因本地未定义字段（后台的字段与本地字符串名不一致）导致数据转换过程中的奔溃

> 将后台JSON数据中的字典转成本地的模型，我们一般选用部分优秀的第三方框架，如SBJSON、JSONKit、MJExtension、YYModel等。但是，一些简单的数据，我们也可以尝试自己来实现转换的过程。

更重要的是，有时候在iOS面试的时候，部分面试官会不仅问你某种场景会用到什么框架，更会问你如果要你来实现这个功能，你有没有解决思路？所以，自己实现字典转模型还是有必要掌握的。有了这个基础，在利用运行时runtime的动态特性，你也可以实现这些第三方框架。

笔者的KVC系列为：

- iOS开发·KVC:字典转模型,防止因本地未定义字段（后台的字段与本地字符串名不一致）导致数据转换过程中的奔溃
- iOS开发·runtime+KVC实现字典模型转换

1. 建模

假设网络请求图片信息并在APP的界面上展示，这里新建一个图书图片的模型，id表示图书的ID，imgUrl是图书的封面地址（可以用SDWebImage加载该图），nameStr时图书的名字，introduceStr是图书的简介。

我们建立一个模型如下（暂不先管 `id` 这个含有关键字的属性，后面会讲）：

- BookModel.h

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

@interface BookModel : NSObject
@property (nonatomic,strong) NSString *id;
@property (nonatomic,strong) NSString *imgUrlStr;
@property (nonatomic,strong) NSString *nameStr;
@property (nonatomic,strong) NSString *introduceStr;

+ (instancetype)getBookModelWithDict:(NSDictionary *)dict;

@end
```

- BookModel.m

```
#import "BookModel.h"
@implementation BookModel
+ (instancetype)getBookModelWithDict:(NSDictionary *)dict
{
    BookModel *bookModel = [[self alloc] init];
    [bookModel setValuesForKeysWithDictionary:dict];
    return bookModel;
}
@end
```

当然，你也可以一个一个地为每个属性分别写setValue，不嫌麻烦的话

```
#import "BookModel.h"
@implementation BookModel
+ (instancetype)getBookModelWithDict:(NSDictionary *)dict
{
    BookModel *bookModel = [[self alloc] init];
    [bookModel setValue:[dict valueForKey:@"id"] forKey:@"id"];
    [bookModel setValue:[dict valueForKey:@"imgUrlStr"]
forKey:@"imgUrlStr"];
    [bookModel setValue:[dict valueForKey:@"nameStr"]
forKey:@"nameStr"];
    [bookModel setValue:[dict valueForKey:@"introduceStr"]
forKey:@"introduceStr"];
    return bookModel;
}
@end
```

2. 含有模型未定义属性同名字段的字典

字典转模型过程中也会遇到一些问题，比如，字典里面有多余的keyValue，但是模型没有定义同名属性，使用`setValuesForKeysWithDictionary`就会崩溃了。

但是，只需要重写`- (void)setValue:(id)value forUndefinedKey:(NSString *)key`方法即可防止未定义的字段与本地字符串名不一致导致的奔溃。

- BookModel.m

```
-(void)setValue:(id)value forUndefinedKey:(NSString *)key{
    //空的什么都不写都可以
}
```

3. 含有系统关键字同名字段的字典

如上所示，许多JSON数据里面会有一个`id`的字段, 而`id`是iOS的一个关键字，不能用关键字定义属性名，此时我们就需要在model类中修改这个属性的名字，并在`- (void)setValue:(id)value forUndefinedKey:(NSString *)key`的方法体中重写该方法，以针对`id`字段作特殊处理。

- BookModel.h

```
@property (nonatomic,strong) NSString *bookId;
```

- BookModel.m

```
- (void)setValue:(id)value forUndefinedKey:(NSString *)key  {
    if([key isEqualToString:@"id"]){
        //self.bookId = value;//不推荐
        [self setValue:value forKey:@"bookId"]; // 推荐
    }
}
```

4. 示例

假设，APP本地里面用plist写了一个字典数组，然后写一个CustomerListModel模型。现在，需要将这个plist字典数组转换成CustomerListModel模型数组，并在VC取值出来赋给表单元cell的模型数组，用于展示数据。示例的用法参考如下：

- CustomerListModel.m

```
#import "CustomerListModel.h"

@implementation CustomerListModel

//kvc实现字典转模型
- (instancetype)initWithDict:(NSDictionary *)dict{
    if (self = [super init]) {
        [self setValuesForKeysWithDictionary:dict];
    }
    return self;
}

//防止与后台字段不匹配而造成崩溃
- (void)setValue:(id)value forUndefinedKey:(NSString *)key{}

//类方法：实现字典转模型，返回模型对象
+ (instancetype)customerListModelWithDict:(NSDictionary *)dict;{
    return [[self alloc]initWithDict:dict];
}

//类方法：实现字典转模型，返回模型对象数组
+ (NSArray<CustomerListModel *> *)customerListModelsWithPlistName:
(NSString *)plistName;{
    //获取路径
    NSString *path = [[NSBundle
mainBundle]pathForResource:plistName ofType:@"plist"];
    //读取plist
    NSArray *dictArr = [NSArray arrayWithContentsOfFile:path];
    //字典转模型
    NSMutableArray *modelArr = [NSMutableArray array];
    [dictArr enumerateObjectsUsingBlock:^(NSDictionary *dict,
NSUInteger idx, BOOL * _Nonnull stop) {
        [modelArr addObject:[self customerListModelWithDict:dict]];
    }];
    return modelArr.copy;
}

@end
```

- 调用处的VC中
-
  - 1).导入模型对象头文件，并声明模型对象数组属性：

```
@property (nonatomic, strong) NSArray<GloryListModel *> *
customerListModelArr;
```

- 2).重写懒加载，并在tableView的代理方法调取模型数组用于显示：

```
#pragma mark - 懒加载模型数组
- (NSArray< CustomerListModel *> *)customerListModelArr{
    if (!_customerListModelArr) {
        _customerListModelArr = [CustomerListModel
customerListModelsWithPlistName:@"GloryList"];
    }
    return _customerListModelArr;
}

//cell delegate
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath{
    GloryListCell *cell = [tableView
dequeueReusableCellWithIdentifier:NSStringFromClass([GloryListCell
class])];
    //定制表格单元分割线
    if ([self.tableView
respondsToSelector:@selector(setSeparatorInset:)]) {
        [self.tableView setSeparatorInset:UIEdgeInsetsMake(0, 0, 0,
0)];
    }
    cell.customerListModel =
self.customerListModelArr[indexPath.row];
    return cell;
}
```

5. 小结

划重点： `- (void)setValue:(id)value forUndefinedKey:(NSString *)key` 方法的作用

- 1、当实现这个方法以后，对未定义的keyValue的处理，防止奔溃。

```
-(void)setValue:(id)value forUndefinedKey:(NSString *)key{
    //空的什么都不写都可以
    //return nil;
}
```

- 2、如果服务返回的字符串有系统默认不能使用的关键字（例如：id，description等可以进行转换）

```
-(void)setValue:(id)value forUndefinedKey:(NSString *)key
{
    if ([key isEqualToString:@"id"]) {
        //self.id1 = value;// 不推荐
        [self setValue:value forKey:@"id1"]; // 推荐
    }
    if ([key isEqualToString:@"description"]) {
        //self.description1 = value;// 不推荐
        [self setValue:value forKey:@"description1"]; // 推荐
    }
}
```

- 3、除了自己实现字典转模型，可以考虑选用部分优秀的第三方框架，如MJExtension、YYModel等。