YYModel了解

前言

揭秘YYModel的魔法

早期开发过程中，接口传过来的数据（一般是JSON类型）需要业务方手写代码，随着业务发展，很多第三方库出现了。这些库的神奇之处在于它们提供了模型与 JSON 数据的自动转换功能。

使用

先了解一下常用方法的使用，然后再等有时间去阅读源码解析。

```
// 字典转模型
+ (nullable instancetype)modelWithDictionary:(NSDictionary *)dictionary;
// json转模型
+ (nullable instancetype)modelWithJSON:(id)json;
// 模型转NSObject
- (nullable id)modelToJSONObject;
// 模型转NSData
- (nullable NSData *)modelToJSONData;
// 模型转json字符串
- (nullable NSString *)modelToJSONString;
// 模型深拷贝
- (nullable id)modelCopy;
// 判断模型是否相等
- (BOOL)modelIsEqual:(id)model;
// 属性数据映射，用来定义多样化数据时转换声明
+ (nullable NSDictionary<NSString *, id> *)modelCustomPropertyMapper;
// 属性自定义类映射，用来实现自定义类的转换声明
+ (nullable NSDictionary<NSString *, id> *)modelContainerPropertyGenericClass;
// 属性黑名单，该名单属性不转换为model
+ (nullable NSArray<NSString *> *)modelPropertyBlacklist;
// 属性白名单，只有该名单的属性转换为model
+ (nullable NSArray<NSString *> *)modelPropertyWhitelist;
// JSON 转为 Model 完成后，该方法会被调用，返回false该model会被忽略
// 同时可以在该model中做一些，转换不能实现的操作，如NSDate类型转换
- (BOOL)modelCustomTransformFromDictionary:(NSDictionary *)dic;
// Model 转为 JSON 完成后，该方法会被调用，返回false该model会被忽略
// 同时可以在该model中做一些，转换不能实现的操作，如NSDate类型转换
- (BOOL)modelCustomTransformToDictionary:(NSMutableDictionary *)dic
```

- 简单的数据交换

只需调用modelWithDictionary:便可，dictionary里面按照key-value分别对应model的属性名。

```
// ViewController.m
NSDictionary *dic = @{
                    @"name":@"张三",
                    @"age":@(12),
                    @"sex":@"男"
                      };
// 将数据转模型
YYPersonModel *model = [YYPersonModel modelWithDictionary:dic];
// 将模型转数据
NSDictionary *dics = [model modelToJSONObject];
```

- 自定义数据key和属性名的对应关系

```
// YYPersonModel.m
+ (NSDictionary *)modelCustomPropertyMapper {
   // 将personId映射到key为id的数据字段
    return @{@"personId":@"id"};
   // 映射可以设定多个映射字段
   //  return @{@"personId":@[@"id",@"uid",@"ID"]};
}
```

- 多样化的数据类型交换

支持NSArray和NSDictionary。支持嵌套层级的自定义数据key

```
// YYPersonModel.m
+ (NSDictionary *)modelCustomPropertyMapper {
  return @{
         @"personId":@"id",
         @"sex":@"sexDic.sex" // 声明sex字段在sexDic下的sex
         };
}
```

在数据中依然可以找到NSArray和NSDictionary和sexDic下的sex字段并转化为模型

```
// ViewController.m
NSDictionary *dic = @{
                 @"id":@"123",
                 @"name":@"张三",
                 @"age":@(12),
                 @"sexDic":@{@"sex":@"男"},
                 @"languages":@[
                         @"汉语",@"英语",@"法语"
                         ],
                 @"job":@{
                         @"work":@"iOS开发",
                         @"eveDay":@"10小时",
                         @"site":@"软件园"
                         }
                 };
YYPersonModel *model = [YYPersonModel modelWithDictionary:dic];
```

- 自定义类的数据转换

项目过程中肯定会有一些自定义实现的类。关于自定义类的声明，YYModel提供给我们另外一个方法 `modelContainerPropertyGenericClass`

```
// 声明自定义类参数类型
+ (NSDictionary *)modelContainerPropertyGenericClass {
  // value使用[YYEatModel class]或YYEatModel.class或@"YYEatModel"没有区别
  return @{@"eats" : [YYEatModel class]};
}
```

- 其他处理
  - 黑白名单

```
// YYPersonModel.m
// 黑白名单不同时使用
// 如果实现了该方法，则处理过程中会忽略该列表内的所有属性
+ (NSArray *)modelPropertyBlacklist {
  return @[@"sex", @"languages"];
}
// 如果实现了该方法，则处理过程中不会处理该列表外的属性。
+ (NSArray *)modelPropertyWhitelist {
  return @[@"eats"];
}
```

  - 校验

    有时候转换后的model并不是我们最终想要的，可以在这里做一些格式转换

```
// YYPersonModel.m
// 当 JSON 转为 Model 完成后，该方法会被调用。
- (BOOL)modelCustomTransformFromDictionary:(NSDictionary *)dic {
  // 可以在这里处理一些数据逻辑，如NSDate格式的转换
  return YES;
}

// 当 Model 转为 JSON 完成后，该方法会被调用。
- (BOOL)modelCustomTransformToDictionary:(NSMutableDictionary *)dic {
    return YES;
}
```

  - 深拷贝 `Model newModel = [model modelCopy];`