# 目录



---

## 自定义配置文件

**模版配置文件位于:**
DiffHelper/User/UserConfig/Demo-CONFIG.json

**建议保留模版配置文件，复制一份模版配置文件另行自定义配置**
**配置文件需要是-CONFIG.json 结尾(使用过程中可以自行选择配置文件)**

```
例如新命名为:projectName-CONFIG.json
```

## 功能开关配置部分:

**开启设置成 true**
**关闭设置成 false**

### RemoveNSLog

> *删除代码中 NSLog 日志输出*

```
配置成true则移除
配置成false则不移除
```

### ModifyResourceMd5

> *修改资源文件 md5*

```
开启则md5会被差异化，不会修改文件尺寸，文件体积会细微变化
```

### CheckSensitiveWord

> *检查项目中的敏感词*

```
项目中如果存在"敏感词.txt"中的词，混淆后就没了
默认的"敏感词.txt"可能会存在覆盖不完全的情况，
"敏感词.txt"在User/UserConfig文件夹下面，可按照格式自行删减
```

### IgnoreOCString

> *忽略 OC 字符串(类似@"xxx"这种)*

**例如网络请求方法:**

```
[[DemoManager defaultManager] getAccountSuccess:^(NSDictionary *repsonse) {
        NSString *name =  [repsonse objectForKey:@"name"];
        NSString *pwd = [repsonse objectForKey:@"password"];
    } failure:nil];
```

**开启则不会处理 @"name"，@"password"**

**这个开关不会影响下面这种情况**
开启或者关闭，demo 这个图片都会被处理成其他，**所有资源文件**适用这种情况，一切为了混淆的更多

```
UIImage *img = [UIImage imageNamed:@"demo"];
```

## 白名单配置部分

### ExcludeFolderList

> *项目中需要排除的文件夹(不是路径)*

**配置在该字段的文件夹，其中所有文件名和文件夹名完全不会被修改，其中的资源 md5 会被处理，创建时间和修改时间会同步处理，Pods 默认被忽略，不需要配置在这**

**下面这些情况需要将文件夹设置为排除:**

**1.项目中 C/C++开源库所在的文件夹名**

例如:

```
iOSTest/OpenCV
```

则配置为:

```
ExcludeFolderList : ["OpenCV"]
```

**2.第三方 SDK 以及其 bundle 文件所在的文件夹名**
例如:

```
iOSTest/Bugly.framwork
iOSTest/Bugly.bundle
iOSTest/Bugly/Buglybridge.a
iOSTest/Bugly/Header/xxxx.h
```

则配置为:

```
ExcludeFolderList : ["Bugly.framwork", "Bugly.bundle", "Bugly"]
```

**3.不想混淆的文件夹(并且该文件夹内的文件只被外部引用，而没有引用外部的文件)**
例如:

```
iOSTest/iOSNetworking/xxx.h
iOSTest/iOSNetworking/xxx.m
```

则配置为:

```
ExcludeFolderList : ["iOSNetworking"]
```

## IgnoreClassList

> *设置忽略的类(不带后缀)*

**该类自身不变，引用的类和调用的外部方法同步混淆**

例如:

```
iOSTest/AppDelegate.h
iOSTest/AppDelegate.m
iOSTest/ViewController.h
iOSTest/ViewController.m
iOSTest/XXX.h
iOSTest/XXX.m
```

假设 `AppDelegate.h引用了XXX.h`
配置:

---

```
IgnoreFolderList : ["AppDelegate", "ViewController"]
```

**表示 AppDelegate.h、AppDelegate.m 中自身的方法和属性等符号都不会被混淆**
**表示 AppDelegate.h、AppDelegate.m 中引用的类和调用的外部方法等会被混淆**
**如果 XXX 被混淆成了 YYY，AppDelegate.h 中引用的 XXX.h 会同步修改为 YYY.h**

## IgnoreFolderList

> *设置忽略的文件夹(不是路径)，对于 IgnoreClassList 的补充(一个个类名配置太麻烦)*

例如:

```
iOSTest/Net.h
iOSTest/Net.m
iOSTest/Hello/Name.h
iOSTest/Hello/Name.m
iOSTest/Hello/Util/User.h
iOSTest/Hello/Util/User.m
```

配置:

```
IgnoreFolderList : ["Hello"]
```

**表示 Hello 文件夹下一切(包括了 Util 下的 User)文件不会被混淆**
**如果 Name 类中引用了 Net 类，Net 类的修改在 Name 类中会同步修改**
**如果 User 类中引用了 Net 类，Net 类的修改在 Name 类中会同步修改**

## IgnoreFolderList 和 ExcludeFolderList 的区别

**ExcludeFolderList：**

> *配置在这的文件夹完全被排除(混淆和它们无关了)*

```
混淆不了的文件夹(C/C++库)
不能混淆的文件夹(第三方SDK以及bundle)
不想混淆而又没有调用过其他的模块的文件夹
```

IgnoreFolderList:

> *配置在这的文件夹自身完全被忽略，引用的外部文件发生了混淆，这里会同步修改*

```
不想混淆这个文件夹，但是这个文件夹又引用了别的模块(除非把引用的模块一块排除)
```

### IgnorePropertyOfClassList

> *设置不需要混淆属性的类(类名，不带后缀)*

```
属性不会被混淆，类名、方法等其他都会混淆，主要针对和网络交互的model类
```

### IgnoreModelFolderList

> *设置不需要混淆属性的类的文件夹(类名，不带后缀)*

**对于 IgnorePropertyOfClassList 的补充**

```
该配置下的文件夹中所有的类属性不会混淆
```

---

## 前缀映射配置部分

### ClassNamePrefixMap

> *类名前缀映射设置*

**如果项目中的资源文件、类文件、文件夹有特殊的前缀，可以通过这里移除或者替换**
**如果没有需要处理的前缀，配置如下**

```
ClassNamePrefixMap: {}
```

**举个 🌰:**
现有类名: `AFXXX, AFYYY`

**移除 AF 前缀**

```
ClassNamePrefixMap:{
  "AF": "",
}
```

**修改 AF 前缀为 SD**

```
ClassNamePrefixMap:{
  "AF": "SD",
}
```

**修改 AF 前缀为 SD 或者 SDD**

```
ClassNamePrefixMap:{
  "AF": ["SD","SDD"],
}
```

**可按照格式无限添加新的键值对**

---

## PropertyPrefixMap

> *属性前缀映射设置*

```
如果项目中有属性包含特殊的前缀，可以通过这里移除或者替换
规则同ClassNamePrefixMap
```

## MethodPrefixMap

> *方法名前缀映射设置*

```
如果项目中有方法包含特殊的前缀，可以通过这里移除或者替换
规则同ClassNamePrefixMap
```

## EnumPrefixMap

> *枚举前缀映射设置*

```
如果项目中有枚举包含特殊的前缀，可以通过这里移除或者替换
规则同ClassNamePrefixMap
```

## 其他功能

### SymbolMaxCount

> *混淆后的符号最大单词数量(避免混淆结果过长)*

**例如方法:**
`AFURLSessionDidReceiveAuthenticationChallengeBlock` 混淆前有七个单词
按照下面的配置则混淆后不会超过 5 个单词

```
SymbolMaxCount: 5
```

### SplitOCStringPercent

> *OC 字符串被拆分的概率(0 到 1 之间的小数)*
> *大于 0 的情况下，http(s)开头的字符串一定会被拆分*

**不拆分，http(s)开头的链接也不会拆分**

```
SplitOCStringPercent: 0
```

**字符串有 50%的概率被拆分**

```
SplitOCStringPercent: 0.5
```

**字符串 100%被拆分**

```
SplitOCStringPercent: 1
```

**效果演示:**
例如:@"hello" 结果可能为:

```
[NSString stringWithFormat:@"%@%@", @"he", @"llo"]
```

或者

```
[NSString stringWithFormat:@"%@%@%@", @"he", @"l", @"lo"]
```

**结果会是 hello 被拆分的所有情况排列组合中随机的一种，每一段至少一个字母**

---

### ModifyProjectTime

> *修改项目中所有文件的创建时间和修改时间*

**开启则设置成 true，需要设置日期区间，开始日期和结束日期需要相差一天以上，修改后文件的时间会随机分布在区间内日期的工作时间中(09:00-21:00)，同一个文件的修改时间一定晚于创建时间**

```
ModifyProjectTime: true
```

**关闭则设置成 false，不需要设置 ProjectStartDate，ProjectEndDate**

```
ModifyProjectTime: false
```

### ProjectStartDate

> *项目中文件的起始日期*

**修改后文件的创建时间、修改时间一定晚于该日期**
例如:

```
ProjectStartDate: "20200101"
```

### ProjectEndDate

> *项目中文件的结束日期*

**修改后文件的创建时间、修改时间一定早于该日期**
例如：

```
ProjectStartDate: "20200601"
```