淘宝开放平台自动化SDK介绍

## 简要介绍

淘宝开放平台自动化SDK是开放平台业界一项首创的技术。它的出现解决了长期以来：SDK维护困难、人工编写笔误、SDK更新速度慢、SDK无法多语言化的难题。

## API数据类型介绍

淘宝API的数据类型是一种简化版的JAVA类型，它适用于互联网传输，兼容各种流行的开发语言，简单、方便、容易理解。

|  |  |  |
| --- | --- | --- |
| 类型名称 | 类型描述 | 取值范围 |
| Number | 整型数据 | ~ |
| String | 字符串类型 |  |
| Boolean | 布尔型 | true, false |
| Date | 日期类型（使用北京时区） | 格式：yyyy-MM-dd HH:mm:ss |
| Field List | 以半角逗号分隔的字符串列表 |  |
| Price | 人民币金额 | 精确到分 |
| byte[] | 字节数组类型 |  |

## API元数据介绍

API[元数据](http://my.open.taobao.com/apidoc/metadata.do)是淘宝开放API最基本最抽象的数据，利用它可以自动生成各种语言的SDK，API在线测试工具，API离线文档等。

API元数据是以XML的形式展现的，它主要包含数据结构和API两个部分：

### 数据结构

主要包含：名称、描述和属性列表。

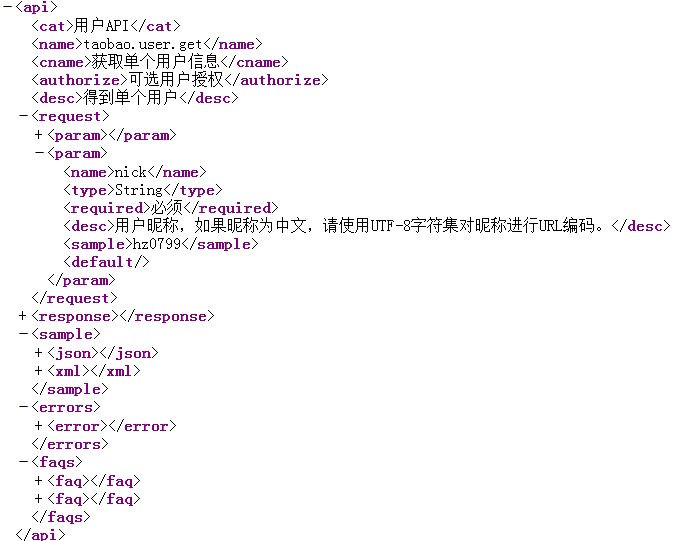


特殊字段解释

|  |  |
| --- | --- |
| **字段名称** | **字段描述** |
| level | 数据结构属性的类型级别，可选值为：   * Basic: 基本类型 * Basic Array: 基本类型数组 * Object: 对象类型 * Object Array: 对象类型数组 |
| private | 字段是否隐私，隐私字段需要用户登录才能返回，可选值为：true/false |

### API

主要包含：名称、授权、请求参数、响应参数，响应示例、错误码、FAQ。



|  |  |
| --- | --- |
| **字段名称** | **字段描述** |
| authorize | 调用接口的授权类型，直接表现为是否需要传入Session Key，可选值为：   * 必须用户授权 * 可选用户授权 * 不需用户授权 |

## SDK自动化设计

### 总体设计

整个系统基于状态和策略模式来设计以达到快速扩展的目的，它主要分为以下五个部分：

**Domain**：API元数据的层次结构展现

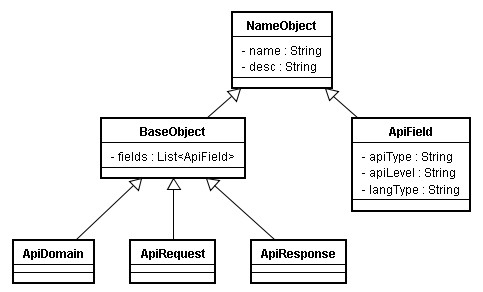
**TypeMapper**: 淘宝API数据类型与特定语言数据类型之间的映射

**SdkParser**: 淘宝API元数据的解释器，通过它可以把元数据解释为Domain结构

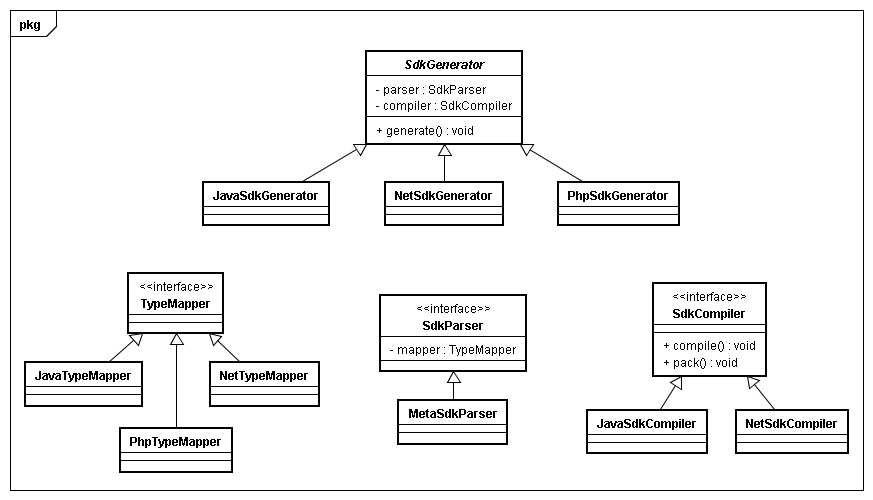
**SdkCompiler**：SDK的编译器，通过它可以把源代码编译成特定语言的字节码或二进制码

**SdkGenerator**：SDK的生成器，它主要负责生成源代码和对字节码进行打包发布

*模型类图*



*系统类图*



## 扩展方式

### 第一步：编写基础框架

主要包含API请求的HTTP通讯、API响应的通用解释（XML和JSON）、API的签名及验证工具。当然还要包括把这三者结合起来的框架。

存放位置：把基础框架打包压缩（ZIP包），并存放在src/main/resources目录下（如taobao-sdk-java-source.zip）

### 第二步：编写模板

主要包含数据结构模板、API请求模板，API响应模板。如果需要依赖第三方IDE来编译的话，还需要提供工程文件模板。模板以文本文件的方式提供，动态内容通过占位符的方式来替换。

存放位置：放在src/main/resources/template目录下

### 第三步：扩展类库

主要有3个扩展点：实现类型映射器（TypeMapper）、实现源码编译器（SdkCompiler）、实现源码生成器（SdkGenerator）

注：

1. 如果是弱类型语言（如PHP），是不需要实现类型映射器的；
2. 如果是动态语言（如PHP），是不需要实现源码编译器的。

存放位置：分别放在放在

com/taobao/top/autosdk/mapper,

com/taobao/top/autosdk/compiler,

com/taobao/top/autosdk/generator,

三个目录下

## 使用方式

生成JAVA SDK：

File target = **new** File("C:/Sdk/Java");

SdkParser parser = **new** MetaSdkParser(**new** JavaTypeMapper());

SdkGenerator sg = **new** JavaSdkGenerator(parser, **new** JavaSdkCompiler());

sg.generate(target);

代码示例请参考：SdkGeneratorTest.java