**项目后台接口和规范**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 作者 | 日期 | 修改背景色 |
| V1.0 | 孙晓晨 | 2013-8-2 | 无 |
| V1.1 | 维旭光 | 2013-8- | 蓝色 |
| V1.2 | 黄明 | 2013-8-28 | 红色 |
|  |  |  |  |
|  |  |  |  |

* 客户端APP调用后台接口统一使用HTTP请求。
* 关于HTTP请求类型：
  + 只涉及到查询的接口，使用GET请求；
  + 涉及到后台数据库的插入、更新、删除等操作的，使用POST请求。
  + 有用户输入的内容作参数的请求，使用POST请求。
* 传入数据：
  + 对于可以由用户任意输入的内容(如场景描述等)，在存入数据库等操作之前，必须对其进行关键词转义，防止注入式攻击。
  + 对于POST请求，建议加上验证字段，防止恶意刷请求。
* 返回数据：
  + 统一使用JSON格式。
  + 返回数据的JSON应该包含的3个字段：
    1. code：返回码。0为成功，大于0为失败。如未登录、权限不够等
    2. msg：错误信息。展示给用户的失败信息。
    3. detail：请求返回的详细内容。

1. **可能用到的数据结构**

这个只是接口描述方便. 具体数据结构的定义取决于产品需求和后台实现

* User
  + string uid
  + string nickname
  + string headUrl 用户头像
* Story
  + Uint64 stid 故事ID
  + string title 故事名
  + int keysMask 故事类型(武侠/魔幻/都市等)的|值
  + string summary 故事摘要
  + int hot 玩过的人数
  + int support 赞的人数（当故事的章节被赞时此数加一）
  + int unsupport 踩的人数（当故事的章节被踩时此数加一）
  + User author 故事原创作者
  + int modeMask 权限类型的|值(是否允许其他人进行编辑或续写)
  + int chapNum 故事章节数
  + Chapter startChap 开始章节
* Chapter 章节(场景)
  + Uint64 cpid
  + string desc 章节内容
  + Uint64 parentId 父亲节点的ID
  + List<Uint64> children 所有的续写章节ID
  + User author 故事原创作者
  + User coauthor 章节作者
  + int support 赞
  + int unsupport 踩
  + int modeMask 权限类型的|值.(是否允许其他人进行编辑或续写)

下列接口中所说的返回值，指的都是返回的json结构中的‘detail’部分

1. **登陆接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口** | **方法** | **参数** | **返回值** | **备注** |
| **/qqlogin** | POST | access\_token | {} | qq登陆 |
| **/sinalogin** | POST | access\_token | {} | 微博登陆 |
| **/reg** | GET | 无 | 注册页 | 可选实现 |
| **/reg** | POST | 用户必选字段 | 首页 | 可选实现 |

1. **故事/章节(场景)接口**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **接口** | **方法** | **参数** | **返回值** | **备注** |
| **/story/create** | POST | string title  string summary  User author  int keysMask  int modeMask  string startChapDesc  int startChapModeMask | {“stid”: 故事ID} | 创建一个新故事.  创建要素包括故事名、故事类型、开头章节(场景)描述 |
| **/chapter/create** | POST | int storyId  int parentId  string desc  User coauthor  int modeMask | {“newcpid”:创建的章节ID,  “brothers”:当前章节的所有兄弟章节ID} | 创建一个新章节(场景). 章节依托于故事, 并需要指定上一个章节的ID.一个故事的开始章节，它的parentId为0 |
| **/story/{id}** | GET | int id | 返回故事的各项属性，这里只列出返回的key  {'title',‘keysMask’,  'summary',  'hot'，  'support',  'unsupport',  'author',  'startChap','modeMask'  } | 查看一个故事 |
| **/chapter/{id}** | GET | int id | id所指的章节具体内容 | 获取章节内容 |
| **/chapter/{id}/next** | GET | int id | 跳转到弟弟章节节点内容 | 获取下一个分支章节内容 |
| **/chapter/{id}/pre** | GET | int id | 跳转到哥哥章节节点内容 | 获取上一个分支章节内容.  建议使用本地缓存, 不再重复请求数据 |
| **/chapter/{id}/child** | GET | int id | 跳转到第一个字章节 |  |
| **/chapter/{id}/children** | GET | int id | 所有续写章节. |  |
| **/chapter/{id}/offspring** | GET | int id  int level | 以这个子节点为根节点的level层子树 | 客户端以子树为单位进行缓存 |