# 15.3 Model类定义

本节我们先梳理一下APP中将用到的数据，然后生成相应的Dart Model类。Json文件转Dart Model的方案采用前面介绍过的 json\_model 包方案

### Github账号信息

登录Github后，我们需要获取当前登录者的Github账号信息，Github API接口返回Json结构如下：

{  
 "login": "octocat", //用户登录名  
 "avatar\_url": "https://github.com/images/error/octocat\_happy.gif", //用户头像地址  
 "type": "User", //用户类型，可能是组织  
 "name": "monalisa octocat", //用户名字  
 "company": "GitHub", //公司  
 "blog": "https://github.com/blog", //博客地址  
 "location": "San Francisco", // 用户所处地理位置  
 "email": "octocat@github.com", // 邮箱  
 "hireable": false,  
 "bio": "There once was...", // 用户简介  
 "public\_repos": 2, // 公开项目数  
 "followers": 20, //关注该用户的人数  
 "following": 0, // 该用户关注的人数  
 "created\_at": "2008-01-14T04:33:35Z", // 账号创建时间  
 "updated\_at": "2008-01-14T04:33:35Z", // 账号信息更新时间  
 "total\_private\_repos": 100, //该用户总的私有项目数(包括参与的其它组织的私有项目)  
 "owned\_private\_repos": 100 //该用户自己的私有项目数  
 ... //省略其它字段  
}

我们在“jsons”目录下创建一个“user.json”文件保存上述信息。

### API缓存策略信息

由于Github服务器在国内访问速度较慢，我们对Github API应用一些简单的缓存策略。我们在“jsons”目录下创建一个“cacheConfig.json”文件缓存策略信息，定义如下：

{  
 "enable":true, // 是否启用缓存  
 "maxAge":1000, // 缓存的最长时间，单位（秒）  
 "maxCount":100 // 最大缓存数  
}

### 用户信息

用户信息(Profile)应包括如下信息：

1. Github账号信息；由于我们的APP可以切换账号登录，且登录后再次打开则不需要登录，所以我们需要对用户账号信息和登录状态进行持久化。
2. 应用使用配置信息；没一个用户都应有自己的APP配置信息，如主题、语言、以及数据缓存策略等。
3. 用户注销登录后，为了便于用户在退出APP前再次登录，我们需要记住上次登录的用户名。

需要注意的是，目前Github有三种登录方式，分别是账号密码登录、oauth授权登录、二次认证登录；这三种登录方式的安全性依次加强，但是在本示例中，为了简单起见，我们使用账号密码登录，因此我们需要保存用户的密码。

注意：在这里需要提醒读者，在登录场景中，保护用户账号安全是一个非常重要且永恒的话题，在实际开发中应严格杜绝直接明文存储用户账密的行为。

我们在“jsons”目录下创建一个“profile.json”文件，结构如下：

{  
 "user":"$user", //Github账号信息，结构见"user.json"  
 "token":"", // 登录用户的token(oauth)或密码  
 "theme":5678, //主题色值  
 "cache":"$cacheConfig", // 缓存策略信息，结构见"cacheConfig.json"  
 "lastLogin":"", //最近一次的注销登录的用户名  
 "locale":"" // APP语言信息  
}

### 项目信息

由于APP主页要显示其所有项目信息，我们在“jsons”目录下创建一个“repo.json”文件保存项目信息。通过参考Github 获取项目信息的API文档，定义出最终的“repo.json”文件结构，如下：

{  
 "id": 1296269,  
 "name": "Hello-World", //项目名称  
 "full\_name": "octocat/Hello-World", //项目完成名称  
 "owner": "$user", // 项目拥有者，结构见"user.json"  
 "parent":"$repo", // 如果是fork的项目，则此字段表示fork的父项目信息  
 "private": false, // 是否私有项目  
 "description": "This your first repo!", //项目描述  
 "fork": false, // 该项目是否为fork的项目  
 "language": "JavaScript",//该项目的主要编程语言  
 "forks\_count": 9, // fork了该项目的数量  
 "stargazers\_count": 80, //该项目的star数量  
 "size": 108, // 项目占用的存储大小  
 "default\_branch": "master", //项目的默认分支  
 "open\_issues\_count": 2, //该项目当前打开的issue数量  
 "pushed\_at": "2011-01-26T19:06:43Z",  
 "created\_at": "2011-01-26T19:01:12Z",  
 "updated\_at": "2011-01-26T19:14:43Z",  
 "subscribers\_count": 42, //订阅（关注）该项目的人数  
 "license": { // 该项目的开源许可证  
 "key": "mit",  
 "name": "MIT License",  
 "spdx\_id": "MIT",  
 "url": "https://api.github.com/licenses/mit",  
 "node\_id": "MDc6TGljZW5zZW1pdA=="  
 }  
 ...//省略其它字段  
}

### 生成Dart Model类

现在，我们需要的Json数据已经定义完毕，现在只需要运行json\_model package提供的命令来通过json文件生成相应的Dart类：

flutter packages pub run json\_model

命令执行成功后，可以看到lib/models文件夹下会生成相应的Dart Model类：

├── models  
│   ├── cacheConfig.dart  
│   ├── cacheConfig.g.dart  
│   ├── index.dart  
│   ├── profile.dart  
│   ├── profile.g.dart  
│   ├── repo.dart  
│   ├── repo.g.dart  
│   ├── user.dart  
│   └── user.g.dart

### 数据持久化

我们使用shared\_preferences包来对登录用户的Profile信息进行持久化。shared\_preferences是一个Flutter插件，它通过Android和iOS平台提供的机制来实现数据持久化。由于shared\_preferences的使用非常简单，读者可以自行查看其文档，在此不再赘述。