

# 前程贷接口文档 v1.3

修订记录表

序号	变更内容说明	版本号	版本日期	修改者
1	新建文档	1.0	2017-5-18	Happy
2	新加了用户相关接口	1.1	2017-5-24	Nick
3	文档优化	1.2	2018-1-31	Happy
4	切换到 Springboot、数据提交及鉴权方式调整、文档更新	1.3	2019-8-25	Happy

## 1 接口概述

为便于学习，文档描述的为前程贷简化版接口，业务与前程贷 web、app 存在一定差异，接口测试时相关业务需求以本文档为准。

### 1.1 接口 URL 格式

URL 的格式如下：

**http://ip:port/futureloan/apiName**

如部署接口的主机 ip 地址为 192.168.0.188，端口号为 8080，以注册接口为例，注册接口名称为 /member/register，则接口完整 URL 为：

http://**192.168.0.188:8080/futureloan/member/register**

### 1.2 请求头

注：红色字体标记的请求头必须设置

请求头	可选值	备注
X-Lemonban-Media-Type	lemonban.v1	接口无鉴权
	lemonban.v2	token 鉴权
	lemonban.v3	timestamp+token+sign 鉴权 其中 sign= RSA(token 前 50 位+时间戳)
Content-Type	application/json	POST、PATCH 请求必须设置 GET 请求不设置
Authorization	Bearer Token	X-Lemonban-Media-Type 请求头的值为 lemonban.v2 或 lemonban.v3 时必须添加此请求头，lemonban.v1 时无需添加 值的格式为 "Bearer token_value"，其中 token_value 是登录成功后返回 token_info 中的 token 值，注意 Bearer 后有空格

### 1.3 响应体

响应体为 json 对象，包含 code、msg、data 三个字段

```
{
  "code": "返回码",
  "msg": "发生错误时返回的错误信息",
  "data": {
    //数据，对象或数组，为空统一设置为 NULL
  }
}
```

## 1.4 通用返回码

返回码 code=0 表示成功, code>0 表示失败

code 说明表	
0	成功
1	必填参数为空
2	参数错误 如格式错误、类型转换出错、内容无法解析、数值类型格式化异常、超出范围、用户已存在、项目已存在等
1001	账号信息错误
1002	账户余额不足
1003	token 或 sign 验证不通过或 token 过期
1004	URL 错误
1005	服务器繁忙, 通常是后端代码运行异常
1006	缺少必须的请求头
1007	无权限访问, 如进行其他用户相关业务操作

## 1.5 接口鉴权

为方便学习, 提供如下两种鉴权方式:

- 1) token 方式
- 2) timestamp+token+sign 方式

### 1.5.1 token 方式

当 **X-Lemonban-Media-Type** 请求头值为 lemonban.v2 时, 接口使用 token 鉴权。除注册、登录和项目列表接口, 其它接口必须设置 **Authorization** 请求头, 值为 Bearer token 值。

用户登录成功时返回 token\_info, 包括如下三个字段

参数	变量名	类型	说明
token 类型	token_type	string	Bearer token
过期时间	expires_in	long	token 过期时间为 5 分钟
token	token	string	服务器签发的 token

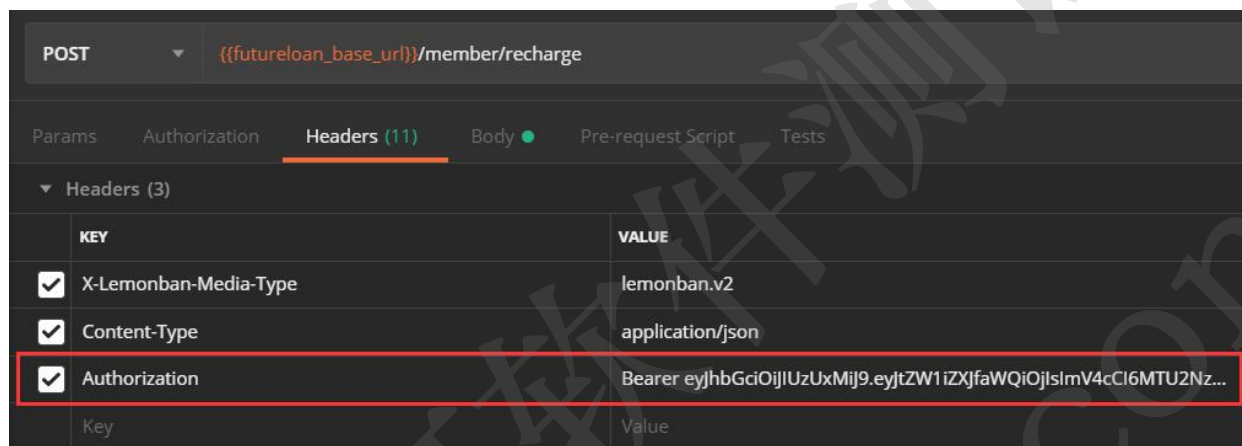
登录成功的返回示例:

```
{
  "code": 0,
  "msg": "OK",
  "data": {
    "id": 2,
    "leave_amount": 0.0,
    "mobile_phone": "13888888811",
    "reg_name": "小柠檬",
    "reg_time": "2019-08-31 20:17:31.0",
    "type": 1,
    "token_info": {
      "token_type": "Bearer",
      "expires_in": "2019-09-01 20:39:03",
      "token": "eyJhbGciOiJIUzUxMiJ9.eyJtZW1iZXJfYWQiOiJlImV4cCI6MTU2NmZMTU0M30.SW3zgoBwU2YI_aaHZLv2CwKXgOncD-RPE1SfCH_SQegsFnjT5qvtILeIdqnVK0Mwm_VNTdPUufbxnub0kisZQ"
    }
  },
  "copyright": "Copyright 柠檬班 © 2017-2019 湖南省零檬信息技术有限公司 All Rights Reserved"
}
```

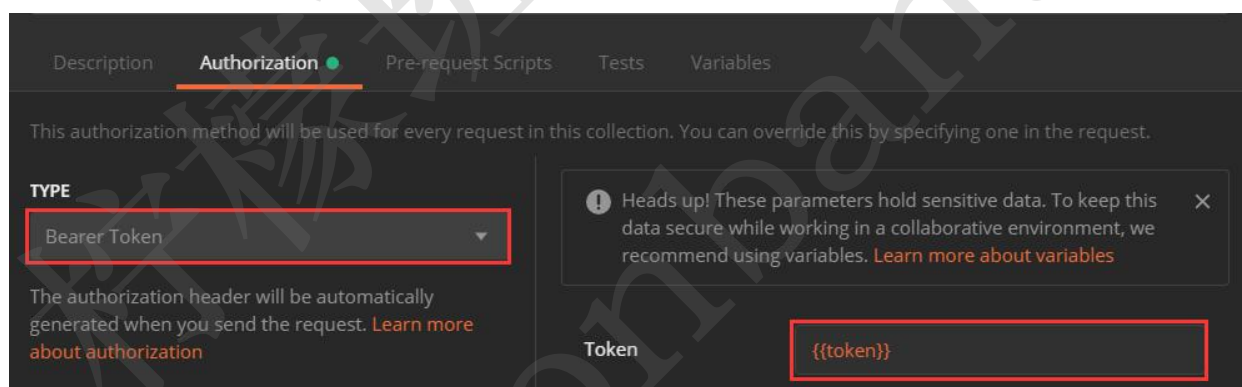
设置 Authorization 请求头时，注意对应值必须是 **Bearer + 空格 + token** 值，**Bearer** 和 **空格**均不能省略。如 token 值为：

```
eyJhbGciOiJIUzUxMiJ9.eyJtZW1iZXJfaWQiOiJlImV4cCI6MTU2Nm0MTU0M30.SW3zgoBwU2YI_aaHZLvy2CWkXgOncD-RPE1SfCH_SQegsFnjT5qvtILeIdqnVK0Mwm_VNTdPUufbxnub0kisZQ
```

对应 Authorization 请求头的值设置形式如下：



Postman 可以直接设置 Authorization 鉴权方式为 Bear token，值为签发的 token 值，无须显式拼接 Bearer 和空格前缀，Postman 会自动添加 Authorization 请求头。



## 1.5.2 timestamp+token+sign 方式

当 **X-Lemonban-Media-Type** 请求头值为 **lemonban.v3** 时，接口使用 timestamp+token+sign 鉴权。除注册、登录和项目列表接口，其它接口需要进行如下设置：

1. 设置 Authorization 请求头，值为 Bearer token 值，设置方式同上 1.5.1 章节
2. 请求体 json 设置 timestamp 参数，值为当前时间戳（注意是秒级时间戳），类型为 long
3. 请求体 json 设置 sign 参数，值为取 token 前 50 位再拼接上 timestamp 值，然后通过 RSA 公钥加密得到的签名字符串。

如 token 值为：

```
eyJhbGciOiJIUzUxMiJ9.eyJtZW1iZXJfaWQiOiJlImV4cCI6MTU2Nm0MTU0Mz0.E0NH0.gc_U13I8fr-6K3x3MFAUD_Kc8xXzgB8qg8MG4FGV1cifTchhFGGZu0E3sZ0pBCHgYizLbt4TlRnJSyhnz5uEcQ
```

timestamp 值：

```
1567342860
```

取 token 前 50 位拼接 timestamp，得到如下字符串：

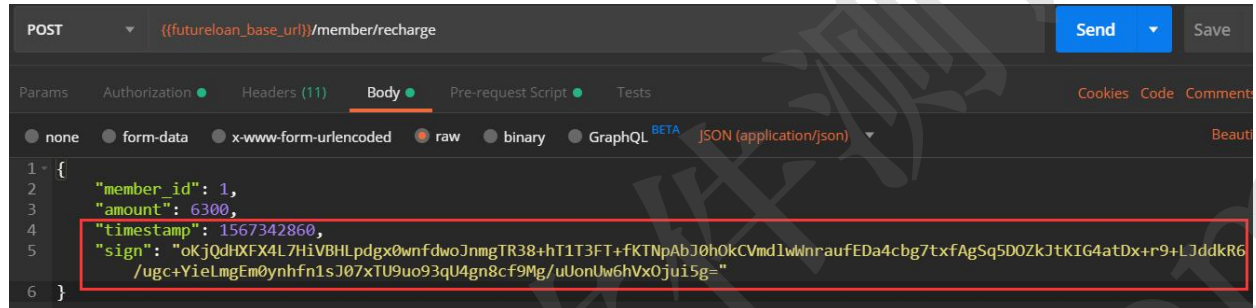
```
eyJhbGciOiJIUzUxMiJ9.eyJtZW1iZXJfaWQiOiJlImV4cCI6MTU2Nm0MTU0M30.1567342860
```

对如上字符串进行 RSA 加密，得到加密的签名字符串：

```
oKjQdHFX4L7HiVBHLpdgx0wnfdwoJnmgTR38+hT1T3FT+fKTnpAbJ0hOkCVmdlwWnraufEDa
4cbg7txfAgSq5DOZkJtKIG4atDx+r9+LJddkR6/ugc+YieLmgEm0ynhfn1sJ07xTU9uo93qU4
gn8cf9Mg/uUonUw6hVxOjui5g=
```

将该加密字符串作为 sign 参数的值设置到请求体 json 对象中。

如当使用 timestamp+token+sign 鉴权方式时，充值接口的请求头设置如下所示：



## 1.6 各终端加密方式

sign 签名加密方式采用 RSA 非对称加密，公钥为：

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDQENQujKLFZfc5Tu9Z1LprzedE
O3F7gs+7bzrgPsM129LX8UoPYvIG8C604CprBQ4FkfnJpnhWu21vUB0WZyLq6sBr
tuPorOc42+gLnFfyhJAwdZB6SqWfDg7bW+jNe5Ki1DtU7z8uF6Gx+b1EMGo8Dg+S
kKlZFc8Br7SHtbL2tQIDAQAB
-----END PUBLIC KEY-----
```

### 1.6.1 JAVA

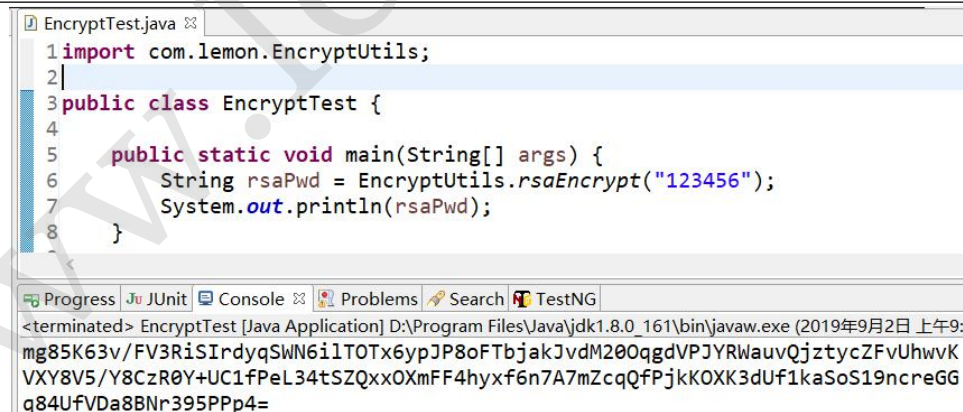
#### 1.6.1.1 普通 Java 工程或 Jmeter

1. 在 Java 项目或者 Jmeter 中添加 lemoner.jar 包
2. 导入工具类

```
import com.lemon.EncryptUtils;
```

3. 加密

```
String rsaPwd = EncryptUtils.rsaEncrypt("123456");
```



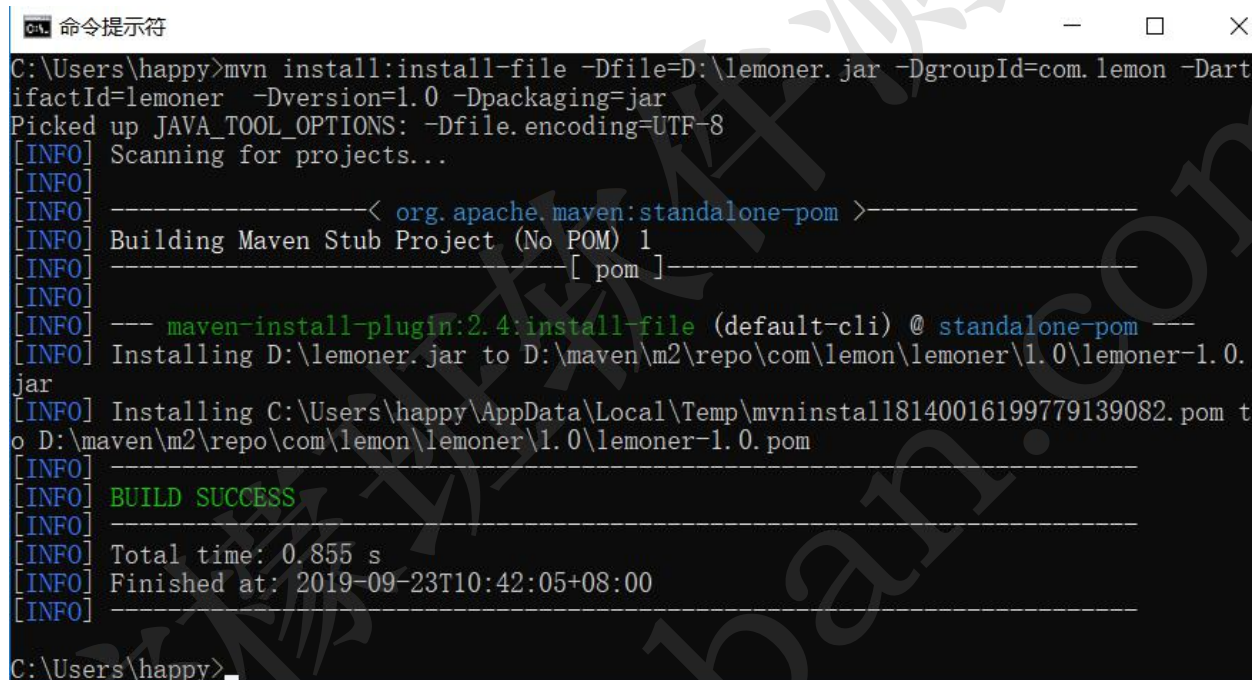


### 1.6.1.2 Maven 工程的引用

如果是 Maven 项目，请先将依赖安装到 Maven 本地仓库

1. 使用 mvn install 命令安装，控制台输入如下命令（注意替换 jar 包路径为你本地 jar 包路径）

```
mvn      install:install-file      -Dfile=D:\lemoner.jar      -DgroupId=com.lemon
-DartifactId=lemoner -Dversion=1.0 -Dpackaging=jar
```



```
C:\Users\happy>mvn install:install-file -Dfile=D:\lemoner.jar -DgroupId=com.lemon -DartifactId=lemoner -Dversion=1.0 -Dpackaging=jar
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
[INFO] Scanning for projects...
[INFO] -----< org.apache.maven:standalone-pom >-----
[INFO] Building Maven Stub Project (No POM) 1
[INFO] -----[ pom ]-----
[INFO] --- maven-install-plugin:2.4:install-file (default-cli) @ standalone-pom ---
[INFO] Installing D:\lemoner.jar to D:\maven\m2\repo\com\lemon\lemoner\1.0\lemoner-1.0.jar
[INFO] Installing C:\Users\happy\AppData\Local\Temp\mvninstall18140016199779139082.pom to D:\maven\m2\repo\com\lemon\lemoner\1.0\lemoner-1.0.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 0.855 s
[INFO] Finished at: 2019-09-23T10:42:05+08:00
[INFO] -----
C:\Users\happy>
```

2. 在 Maven 工程中引入依赖

```
<dependency>
  <groupId>com.lemon</groupId>
  <artifactId>lemoner</artifactId>
  <version>1.0</version>
</dependency>
```

3. 导入包

```
import com.lemon.EncryptUtils;
```

4. 加密

```
String rsaPwd = EncryptUtils.rsaEncrypt("123456");
```

## 1.6.2 Python

### 1. 安装 Crypto 模块

```
pip3 install Crypto
```

**注意：**该库 python2 名为 Crypto，模块内部导入全部是基于 Crypto，python3 中模块名虽然改为 crypto，内部导入依然使用的是 Crypto，因此，安装好了之后，需要找到该模块，手动修改包名为 Crypto，才可以正常使用。

### 2. 加密

```
import base64
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_v1_5 as Cipher_pkcs1_v1_5
def rsaEncrypt(msg):
    """
    公钥加密
    :param msg: 要加密内容
    :type msg:str
    :return: 加密之后的密文
    """
    key = '-----BEGIN PUBLIC
KEY-----\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDQENQujkLfZfc5Tu9Z1LprzedE\nO3F7g
s+7bzrgPsMl29LX8UoPYvIG8C604CprBQ4FkfnJpnhWu2lvUB0WZyLq6sBr\ntuPorOc42+gLnFfyhJAw
dZB6SqWfDg7bW+jNe5Ki1DtU7z8uF6Gx+b1EMGo8Dg+S\nkK1ZFc8Br7SHtbL2tQIDAQAB\n-----END
PUBLIC KEY-----\n'
    publickey = RSA.importKey(key)
    cipher = Cipher_pkcs1_v1_5.new(publickey)
    # 分段加密
    cipher_text = []
    for i in range(0, len(msg), 80):
        cont = msg[i:i + 80]
        cipher_text.append(cipher.encrypt(cont.encode()))

    # base64 进行编码
    cipher_text = b''.join(cipher_text)
    cipher_result = base64.b64encode(cipher_text)
    # 返回密文
    return cipher_result.decode()

if __name__ == '__main__':
    # 待加密内容
    pwd = "123qwe"
    # 加密操作
    en_msg = rsaEncrypt(msg=pwd)
    print('加密密文: ', en_msg)
```

### 1.6.3 JavaScript (Postman 中使用)

登录接口 Tests 选项卡中, 将 token 保存到环境变量

```
//获取响应数据 json 对象
var jsonData = pm.response.json();
//从响应数据提取出 token
var token = jsonData.data.token_info.token;
//设置 token
pm.environment.set("token", token);
//设置用户 id
pm.environment.set("member_id", jsonData.data.id);
```

在需要鉴权接口的 Pre-request-Script 中, 设置时间戳和 sign:

```
if(!pm.globals.has("forgeJS")){

pm.sendRequest("https://raw.githubusercontent.com/loveiset/RSAForPostman/master/forge.js", function (err, res) {
    if (err) {
        console.log(err);
    } else {
        pm.globals.set("forgeJS", res.text());
    }
})
}

eval(postman.getGlobalVariable("forgeJS"));

const public_key = '-----BEGIN PUBLIC KEY-----\n'+
'MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDQENQujkLfZfc5Tu9Z1LprzedE\n'+
'O3F7gs+7bzrgPsMl29LX8UoPYvIG8C604CprBQ4FkfnJpnhWu2lvUB0WZyLq6sBr\n'+
'tuPorOc42+gLnFfyhJAwdZB6SqWfDg7bW+jNe5Ki1DtU7z8uF6Gx+b1EMGo8Dg+S\n'+
'kKlZFc8Br7SHtbL2tQIDAQAB\n'+
'-----END PUBLIC KEY-----' + '\r';

var publicKey = forge.pki.publicKeyFromPem(public_key); //公钥
var timestamp = Math.round(new Date().getTime()/1000); //时间戳
var token =pm.environment.get("token"); //获取 token
var tempStr = token.substring(0,50)+timestamp; //拼接 token 和时间戳: token 前 50 位+timestamp
//rsa 加密得到签名
var sign = forge.util.encode64(publicKey.encrypt(tempStr, 'RSAES-PKCS1-V1_5', {
    md: forge.md.sh1.create(),
    mgf: forge.mgf.mgf1.create(forge.md.sh1.create())
}));
pm.environment.set("timestamp", timestamp); //设置到环境变量
pm.environment.set("sign", sign);
```



## 2 接口说明

### 2.1 用户

#### 2.1.1 注册

**业务描述:** 注册成功后, 会在 member 表中新增一条记录, 为此用户分配唯一的 id, 密码 md5 加密方式保存, 初始可用余额为 0.0, 注册时间为接口请求成功时间

**接口名称:** /member/register

**请求方法:** POST

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
手机号	mobile_phone	int	新用户的手机号	是
密码	pwd	string	8 到 16 位 (最少 8 位, 最长 16 位)	是
类型	type	int	0 管理员 1 普通用户, 缺省默认为 1	否
注册名	reg_name	string	昵称, 长度最多为 10 位, 缺省为小柠檬儿	否

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
数据	data	object	
	id	int	用户 id
	reg_name	string	用户名
	mobile_phone	string	手机号
描述信息	msg	string	返回码对应的描述信息

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/member/register
请求头	Content-Type:application/json
请求体	{"mobile_phone": "13888888888", "pwd": "123456"}
响应体	{ "code": 0, "msg": "OK", "data": { "id": 101, "reg_name": "小柠檬儿", "mobile_phone": "13888888888" } }

## 2.1.1.2 登录

**业务描述:** 输入正确的手机号及对应的密码才可登录成功

**接口名称:** /member/login

**请求方法:** POST

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
手机号	mobile_phone	string	用户手机号	是
密码	pwd	string	用户密码	是

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	返回码对应的描述信息
数据	data	object	
	id	int	用户 id
	leave_amount	double	余额
	mobile_phone	string	手机号
	reg_name	string	用户名
	reg_time	string	注册时间
	type	int	用户类型
	token_info	object	当 X-Lemonban-Media-Type 值为 lemonban.v2 或 lemonban.v3 时会返回 token_info 对象
	token_type	string	类型
	expires_in	string	过期时间
	token	string	token

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/member/login
请求头	Content-Type:application/json
请求体	{"mobile_phone": "13888888888", "pwd": "123456"}
响应体	{ "code": 0, "msg": "OK", "data": { "id": 101, "leave_amount": 0.02, "mobile_phone": "13888888888", "reg_name": "小柠檬儿", "reg_time": "2019-08-27 09:24:03.0", "type": 1 } }

### 2.1.3 充值

**业务描述:** 充值金额必须大于 0 且小于等于 50 万, 充值成功后, 用户的可用余额增加, member 表对应记录的 leaveAmount 增加, 并将新增一条流水记录, 保存到 financeLog 表

**接口名称:** /member/recharge

**请求方法:** POST

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
用户 id	member_id	int	用户 id	是
金额	amount	double	0 到 50 万之间的正数金额, 最多精确到小数点后两位。	是

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	原因描述
数据	data	object	
	id	int	用户 id
	leave_amount	double	余额
	mobile_phone	string	手机号
	reg_name	string	用户名
	reg_time	string	注册时间
	type	int	用户类型

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/member/recharge
请求头	Content-Type:application/json
请求体	<pre>{   "member_id": 101,   "amount": 6300 }</pre>
响应体	<pre>{   "code": 0,   "msg": "OK",   "data": {     "id": 101,     "leave_amount": 6300.02,     "mobile_phone": "13888888888",     "reg_name": "小柠檬儿",     "reg_time": "2019-08-25 21:18:30.0",     "type": 1   } }</pre>

### 2.1.1.4 提现

**业务描述:** 提现金额必须大于 0 且小于等于 50 万, 提现金额不能大于用户可用余额, 提现成功后, 用户的可用余额减少, member 表对应记录的 leaveAmount 减少, 并将新增一条流水记录, 保存到 financeLog 表

**接口名称:** /member/withdraw

**请求方法:** POST

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
用户 id	member_id	string	用户 id	是
提现金额	amount	double	0 到 50 万之间的正数金额, 最多精确到小数点后两位。	是

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	原因描述
数据	data	object	
	id	int	用户 id
	leave_amount	double	余额
	mobile_phone	string	手机号
	reg_name	string	用户名
	reg_time	string	注册时间
	type	int	用户类型

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/member/withdraw
请求头	Content-Type:application/json
请求体	{"member_id": 101, "amount": 500}
响应体	{ "code": 0, "msg": "OK", "data": { "id": 101, "leave_amount": 5800.02, "mobile_phone": "13888888888", "reg_name": "小柠檬儿", "reg_time": "2019-08-25 21:18:30.0", "type": 1 } }

### 2.1.5 更新昵称

**业务描述:** 更新用户信息

**接口名称:** /member/update

**请求方法:** PATCH

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
用户 id	member_id	int	用户 id	是
昵称	reg_name	string	昵称, 长度最多 10 位	是

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	原因描述
数据	data	object	
	id	int	用户 id
	leave_amount	double	余额
	mobile_phone	string	手机号
	reg_name	string	用户名
	reg_time	string	注册时间
	type	int	用户类型

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/member/update
请求头	Content-Type:application/json
请求体	<pre>{   "member_id": 101,   "reg_name": "柠檬" }</pre>
响应体	<pre>{   "code": 0,   "msg": "OK",   "data": {     "id": 101,     "leave_amount": 0.02,     "mobile_phone": "13888888888",     "reg_name": "柠檬",     "reg_time": "2019-08-25 21:21:19.0",     "type": 1   } }</pre>

## 2.1.6 用户信息

**业务描述:** 获取单个用户信息

**接口名称:** /member/{member\_id}/info

**请求方法:** GET

**响应格式:** application/json

**请求参数:** 无

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	原因描述
数据	data	object	
	id	int	用户 id
	leave_amount	double	余额
	mobile_phone	string	手机号
	reg_name	string	用户名
	reg_time	string	注册时间
	type	int	用户类型

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/member/186/info
响应体	<pre>{   "code": 0,   "msg": "OK",   "data": {     "id": 186,     "leave_amount": 2100000.01,     "mobile_phone": "13888888879",     "reg_name": "小柠檬儿",     "reg_time": "2019-08-25 21:18:30.0",     "type": 1   } }</pre>



## 2.2 项目

### 2.2.1 新增项目

**业务描述:** 新增一个项目, 借款人 id 必须是 member 表中已经存在, 新增成功后, 会保存一条项目记录到 loan 表中。借款期限单位为月的项目, 其借款期限范围为 1 到 36 个月 (包含 1 个月和 36 个月), 借款期限单位为天的项目, 其借款期限范围为 10 到 45 天 (包含 10 天和 45 天),

**接口名称:** /loan/add

**请求方法:** POST

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
用户 ID	member_id	int		是
标题	title	string		是
借款金额	amount	double		是
年利率	loan_rate	double	如年化 18.0%, 传参为 18.0	是
借款期限	loan_term	int	如 6 个月为 6, 此时 loan_date_type 为 1 30 天为 30, 此时 loan_date_type 为 2	是
借款期限类型	loan_date_type	int	借款期限单位, 1-按月 2-按天	是
竞标天数	bidding_days	int	1-10 天	是

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	返回码对应的描述信息
数据	data	object	项目信息 json 对象
	id	int	项目 id
	member_id	int	借款人 id
	title	string	标题
	amount	double	借款金额
	loan_rate	double	年利率
	loan_term	int	借款期限
	loan_date_type	int	借款期限类型
	bidding_days	int	竞标天数
	create_time	string	创建时间
	bidding_start_time	string	开始竞标时间
	full_time	string	满标时间
	status	int	状态

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/loan/add
请求头	Content-Type:application/json
请求体	<pre>{   "member_id":1,   "title":"报名 Java 全栈自动化课程",   "amount":6300.00,   "loan_rate":12.0,   "loan_term":12,   "loan_date_type":1,   "bidding_days":5 }</pre>
响应体	<pre>{   "code": 0,   "msg": "OK",   "data": {     "id": 4,     "member_id": 1,     "title": "报名 Java 全栈自动化课程",     "amount": 6300,     "loan_rate": 12,     "loan_term": 12,     "loan_date_type": 1,     "bidding_days": 5,     "create_time": "2019-08-27 17:28:30.0",     "bidding_start_time": null,     "full_time": null,     "status": 1   } }</pre>

### 2.2.2 审核项目

**业务描述:** 对项目进行审核操作, 只有**管理员账号**才有对项目进行审核的权限

**接口名称:** /loan/audit

**请求方法:** PATCH

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
项目 id	loan_id	int	项目 id	是
审核状态	approved_or_not	boolean	true 表示通过 false 表示审核不通过	是

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	原因描述
数据	data	object	

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/loan/audit
请求头	Content-Type:application/json
请求体	{ "loan_id": 1, "approved_or_not": true }
响应体	{ "code": 0, "msg": "OK", "data": null }

### 2.2.3 投资

**业务描述:** 用户选择合适项目进行投资 (可以投资自己发布的项目), 当前项目必须是竞标中状态, 用户投资金额必须是能被 100 整除的正整数, 并且投资金额必须小于项目的剩余可投金额, 否则投资不成功。投资成功后, 将生成一条投资记录保存到 invest 表, 投资用户可用余额减少, 并新增一条流水记录保存到 financeLog 表。当可投金额为 0 时, 会自动为该项目的所有的投资记录生成对应的回款计划列表。

**接口名称:** /member/invest

**请求方法:** POST

**请求类型:** application/json

**响应格式:** application/json

**请求参数:**

参数	变量名	类型	说明	是否必填
用户 ID	member_id	int		是
标 id	loan_id	double		是
投资金额	amount	double		是

**结果说明:**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	原因描述
数据	data	object	
	id	int	投资记录 id
	member_id	int	用户 id
	loan_id	int	项目 id
	amount	double	投资金额
	create_time	string	投资时间
	is_valid	int	是否有效

**接口示例 (仅供参考):**

URL	http://localhost:8080/futureloan/member/invest
请求头	Content-Type:application/json
请求体	{"member_id":101,"loan_id":1,"amount":400.00}
响应体	{       "code": 0,       "msg": "OK",       "data": {         "id": 34,         "member_id": 1,         "loan_id": 1,         "amount": 400,         "create_time": "2019-08-28 12:52:37",         "is_valid": 1       }     }

## 2.2.4 项目列表

**业务描述：**分页获取项目列表

**接口名称：**/loans

**请求方法：**GET

**响应格式：**application/json

**请求参数：**

参数	变量名	类型	说明	是否必填
当页索引	pageIndex	int	从 1 开始表示第 1 页，缺省为 1	否
每页大小	pageSize	int	每页大小，缺省为 10	否

**结果说明：**

参数	变量名	类型	说明
返回码	code	int	
描述信息	msg	string	返回码对应的描述信息
数据	data	array	
	id	int	项目 id
	member_id	int	借款人 id
	title	string	标题
	amount	double	借款金额
	loan_rate	double	年利率
	loan_term	int	借款期限
	loan_date_type	int	借款期限类型
	bidding_days	int	竞标天数
	create_time	string	创建时间
	bidding_start_time	string	开始竞标时间
	full_time	string	满标时间
	status	int	状态

**接口示例（仅供参考）：**

URL	http://localhost:8080/futureloan/loans?pageIndex=1&pageSize=1
响应体	<pre>{   "code": 0,   "msg": "OK",   "data": [     {       "id": 1,       "member_id": 1,       "title": "报名 Java 全栈自动化课程",       "amount": 6300,       "loan_rate": 12,       "loan_term": 3,       "loan_date_type": 1,</pre>

	<pre>"bidding_days": 5, "create_time": "2019-08-27 17:23:08.0", "bidding_start_time": null, "full_time": null, "status": 1 } ] }</pre>
--	--