

# 万信金融 第1章 讲义-开发环境搭建

## 1 服务端搭建

### 1.1 数据库环境

本项目使用MySQL数据存储数据。

1) 安装MySQL数据库(v5.6.5+)

端口号默认使用3306，请自行安装并启动MySQL数据库

2) 执行下列SQL脚本导入数据

执行wanxinp2p-init.sql 创建P2P平台数据库并导入初始数据

执行wanxindepository-init.sql 创建银行存管系统数据库并导入初始数据

数据库清单如下：

数据库名称	数据内容
p2p_uaa	统一认证数据
p2p_account	统一账户数据
p2p_consumer	用户中心数据
p2p_transaction_0	交易中心数据库1
p2p_transaction_1	交易中心数据库2
p2p_undo_log	分布式事务框架Hmily数据库
p2p_repayment	还款中心数据
p2p_file	文件存储服务
p2p_bank_depository	银行存管系统
p2p_depository_agent	银行存管代理服务数据
p2p_reconciliation	对账数据

备注：表、表中的字段以及表关系会在后续开发过程中随用随讲。

### 1.2 微服务基础工程

#### 1.2.1 开发工具配置

服务端工程使用IntelliJ IDEA开发(请使用2018以上的版本)。

1、直接用IDEA打开课件提供的基础工程wanxinp2p(在day01课件的代码文件夹中)。

## 2、配置maven环境

关于maven仓库有以下配置方法：

### 1) 在setting.xml中配置私服地址(企业最常用)

在setting.xml中的<servers>段增加：

```
<server>
  <id>maven-releases</id>
  <username>私服账号</username>
  <password>私服密码</password>
</server>
<server>
  <id>maven-snapshots</id>
  <username>私服账号</username>
  <password>私服密码</password>
</server>
```

在<mirrors>添加：

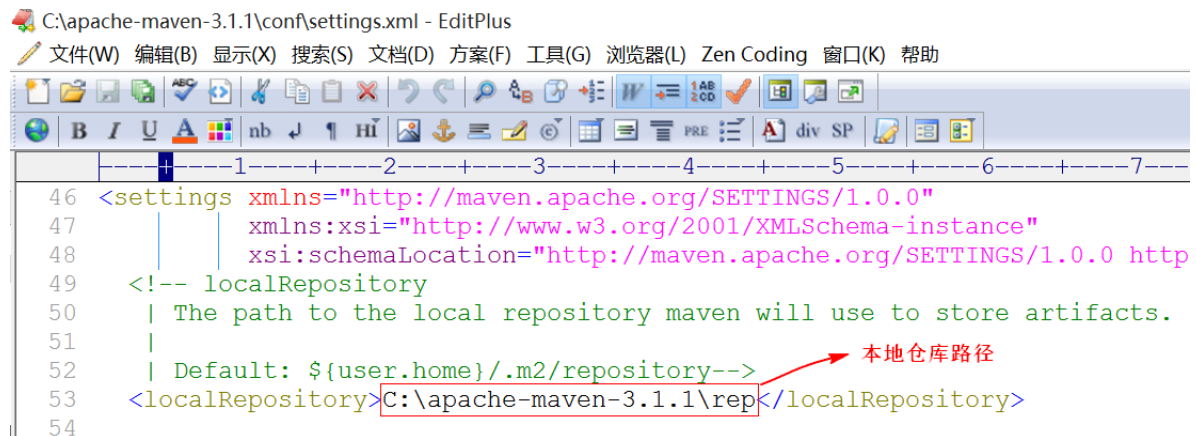
```
<mirror>
  <id>nexus</id>
  <mirrorOf>*</mirrorOf>
  <url>私服地址</url>
</mirror>
```

### 2) 从中央仓库下载

不作任何配置，maven自行从中央仓库下载。

### 3) 使用本地仓库(学习期间使用)

本课程使用本地仓库配置。拷贝老师提供的maven仓库到自己的电脑上（路径中不能包含有中文），先在setting.xml文件中进行如下配置：



```
C:\apache-maven-3.1.1\conf\settings.xml - EditPlus
文件(W) 编辑(B) 显示(X) 搜索(S) 文档(D) 方案(F) 工具(G) 浏览器(L) Zen Coding 窗口(K) 帮助

46 <settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
47     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
48     xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http
49     <!-- localRepository
50     | The path to the local repository maven will use to store artifacts.
51     |
52     | Default: ${user.home}/.m2/repository-->
53     <localRepository>C:\apache-maven-3.1.1\repo</localRepository>
54
```

C:\apache-maven-3.1.1\conf\settings.xml - EditPlus

文件(W) 编辑(B) 显示(X) 搜索(S) 文档(D) 方案(F) 工具(G) 浏览器(L) Zen Coding 窗口(K) 帮助

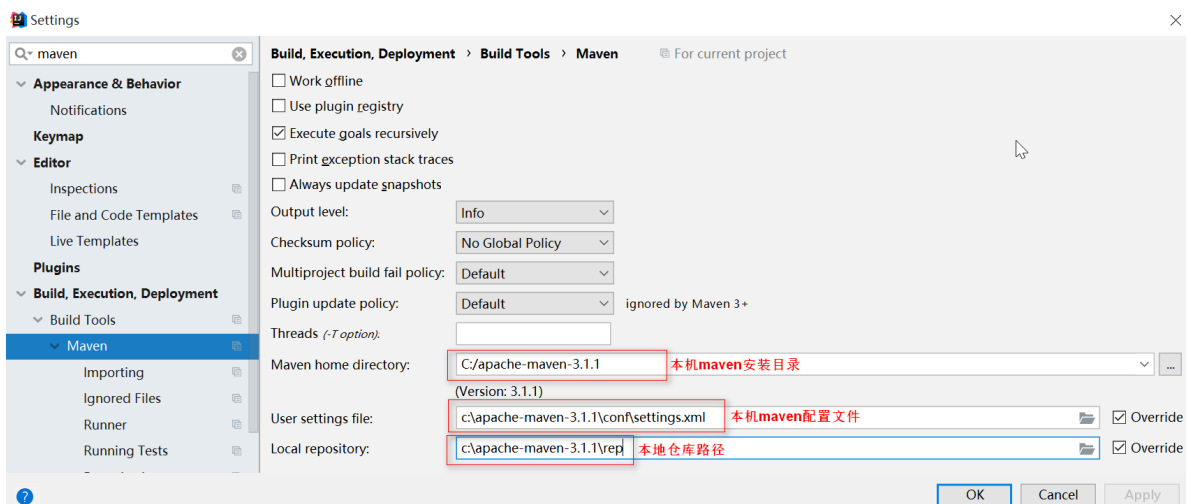
```

146 <mirrors>
147   <!-- mirror
148   | Specifies a repository mirror site to use instead of a given repository
149   | this mirror serves has an ID that matches the mirrorOf element of
150   | for inheritance and direct lookup purposes, and must be unique across
151   |
152   <mirror>
153     <id>mirrorId</id>
154     <mirrorOf>repositoryId</mirrorOf>
155     <name>Human Readable Name for this Mirror.</name>
156     <url>http://my.repository.com/repo/path</url>
157   </mirror>
158   -->
159   <mirror>
160     <id>alimaven</id>
161     <name>aliyun maven</name>
162     <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
163     <mirrorOf>central</mirrorOf>
164   </mirror>
165 </mirrors>

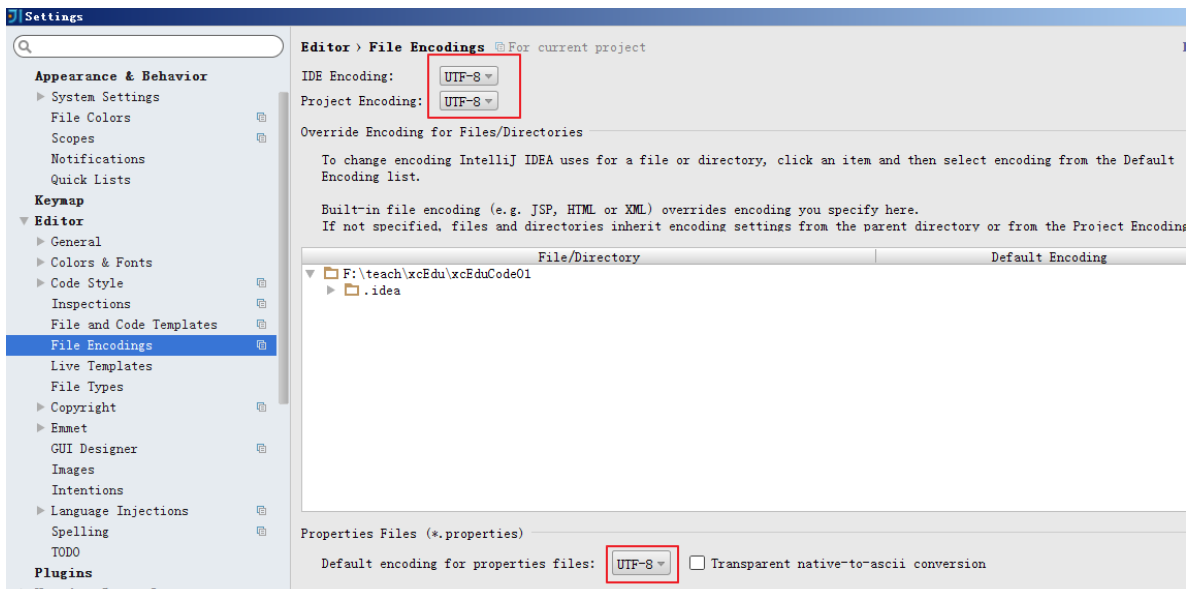
```

采用阿里云中央仓库

然后在IDEA中进行如下配置：

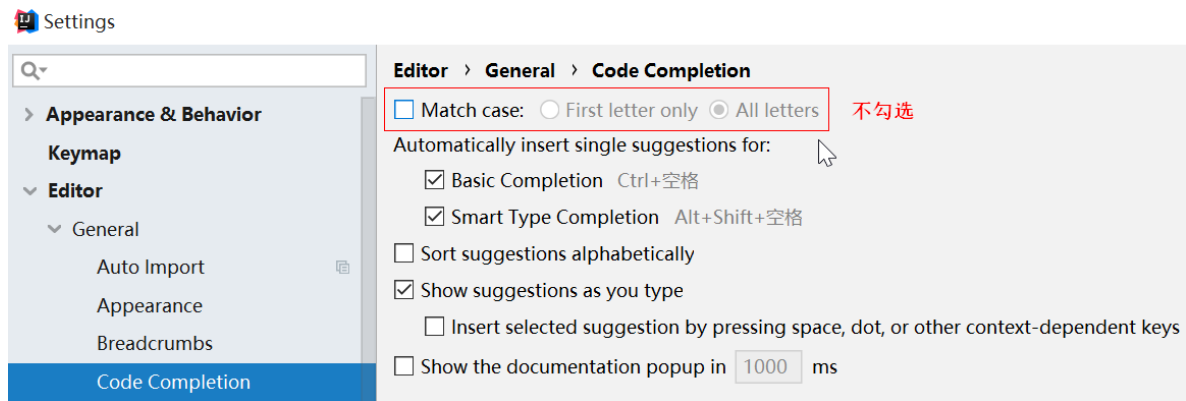


### 3、配置编码



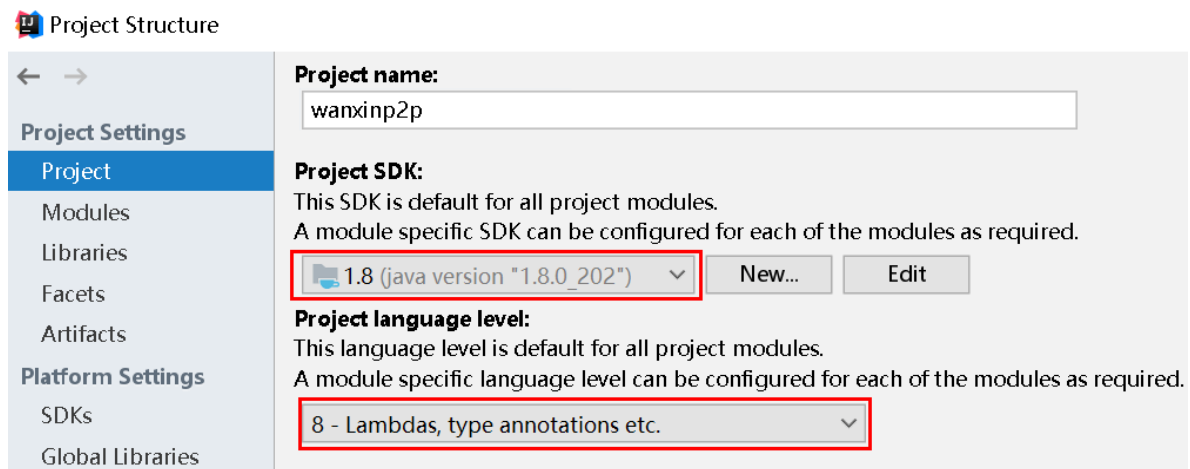
#### 4、代码提示忽略大小写

IDEA默认的代码提示是区分大小写的，这里设置为忽略大小写，编码更方便



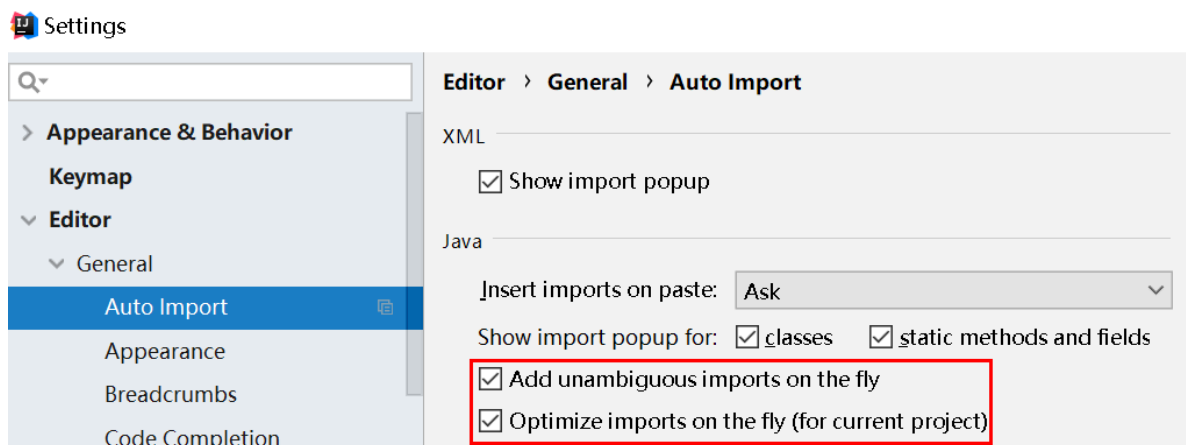
#### 5、配置JDK1.8

先在本机安装JDK1.8，并设置好环境变量(JAVA\_HOME)，然后在IDEA中配置JDK1.8



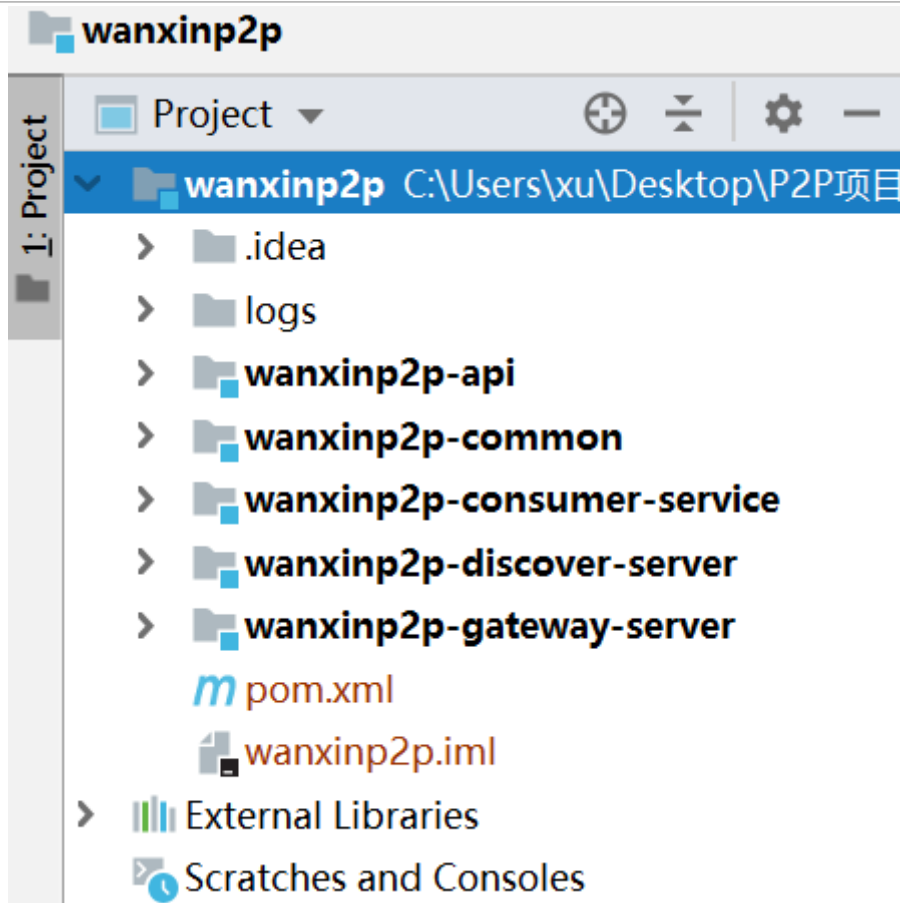
#### 6、自动导入包：

idea可以自动优化导入包，但是有多个同名的类调用不同的包，必须自己手动Alt+Enter设置



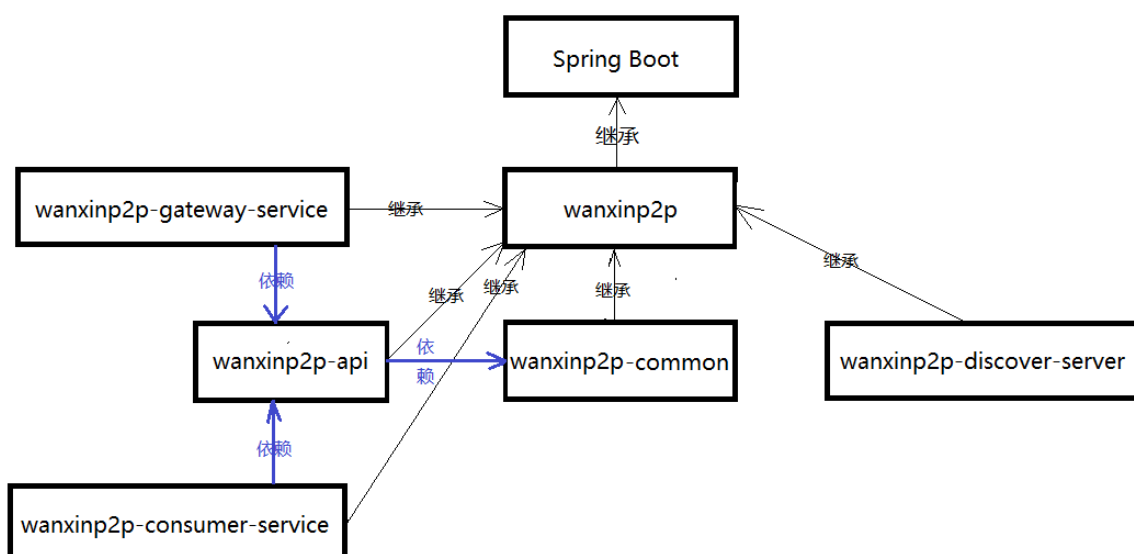
### 1.2.2 了解基础工程

1) 直接用IDEA打开课件提供的基础工程wanxinp2p(在day01课件的代码文件夹中)，打开后的基础工程结构如下图所示：



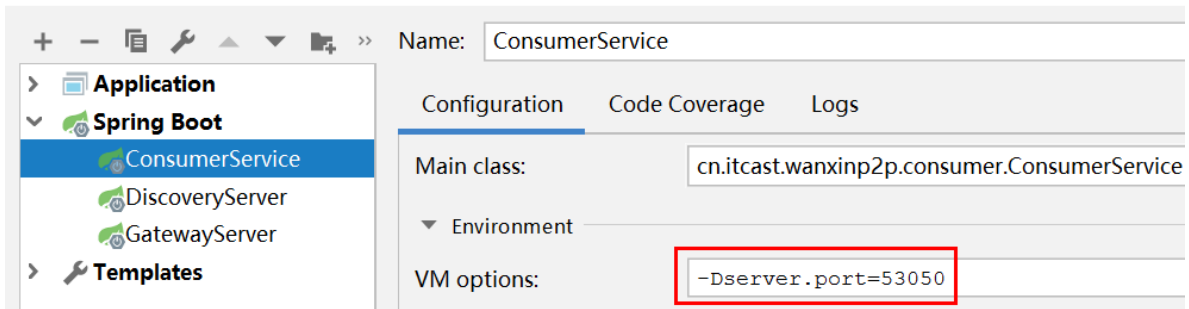
- wanxinp2p: 这是整个项目的父工程，管理依赖环境
- wanxinp2p-api: 存放整个项目的API(接口+各种实体类)
- wanxinp2p-common: 存放整个项目的通用组件(各种业务封装类+工具类)
- wanxinp2p-gateway-service: 网关微服务，端口号53010
- wanxinp2p-consumer-service: 用户中心微服务，端口号53050
- wanxinp2p-discover-server: 服务注册中心 (Eureka)，端口号53000

## 2) 基础工程关系图

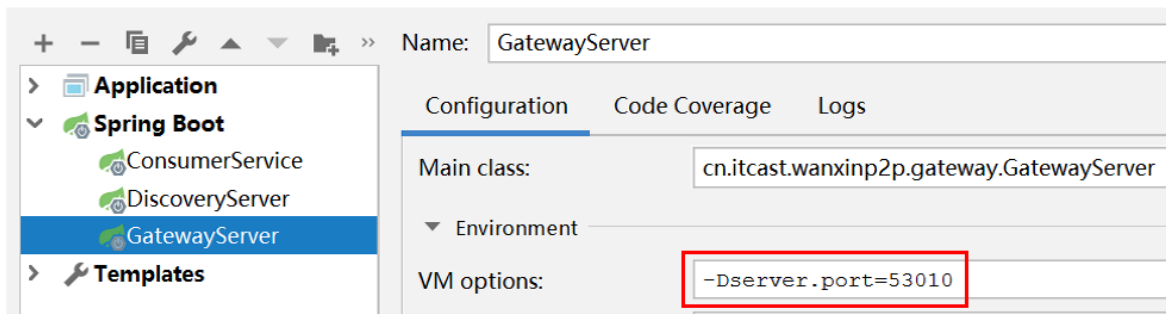


## 3) 基础工程测试

## Run/Debug Configurations



## Run/Debug Configurations



- 访问Eureka服务注册中心
- 直接访问一个Controller进行测试
- 通过网关访问微服务的一个Controller进行测试

## 1.3 Apollo配置中心

在动手编程之前需要先学习下Apollo配置中心，本项目中各个微服务的大部分配置信息由Apollo统一管理。

详见“应用配置中心Apollo-讲义.pdf”

## 1.4 微服务基础工程(Apollo环境)

### 1.4.1 Apollo环境搭建

1. 进入“资料\p2p\_apollo环境”文件夹中，执行p2p\_apollo.sql(包含有基础工程的初始配置)
2. 用记事本打开runApollo.bat，修改其中的MySQL密码
3. 双击runApollo.bat启动Apollo，启动完毕后访问<http://localhost:8070>

### 1.4.2 打开基础工程(Apollo环境)并测试

在“代码”文件夹中，有一个“wanxinp2p\_apollo”项目，这是使用Apollo环境的微服务基础工程，直接右键用IDEA打开即可，可以浏览一下这些基础工程与前面的变化。

分别启动这几个基础工程，然后进行测试：

- 1、访问Eureka服务注册中心
- 2、直接访问一个Controller进行测试

3、通过网关访问微服务的一个Controller进行测试

注意：在以后的开发过程中，每次都要先启动Apollo，再启动p2p后端微服务工程

## 2 接口相关工具

### 2.1 API接口文档利器：Swagger

#### 2.1.1 Swagger介绍

Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务(<http://swagger.io/>)。它的主要作用是：

1. 使得前后端分离开发更加方便，有利于团队协作
2. 接口的文档在线自动生成，降低后端开发人员编写接口文档的负担
3. 功能测试

Spring已经将Swagger纳入自身的标准，建立了Spring-swagger项目，现在叫Springfox。通过在项目中引入Springfox，即可非常简单快捷的使用Swagger。

#### 2.1.2 SpringBoot集成Swagger

1. 引入依赖

```
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger2</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>io.springfox</groupId>
  <artifactId>springfox-swagger-ui</artifactId>
  <version>2.9.2</version>
</dependency>
```

只需要在wanxinp2p\_common中进行配置即可，因为其他微服务工程都直接或间接依赖wanxinp2p\_common。

2. 在wanxinp2p\_consumer工程的application.properties中配置swagger的启动开关

```
# 应用程序名称
spring.application.name=consumer-service
# 微服务访问路径
server.servlet.context-path=/consumer
# 开启swagger
swagger.enable=true
```

3. 在wanxinp2p\_consumer工程的config包中添加一个配置类

```
@Configuration
@ConditionalOnProperty(prefix = "swagger",value = {"enable"},havingValue = "true")
```

```
@EnableSwagger2
public class SwaggerConfiguration {

    @Bean
    public Docket buildDocket() {
        return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(buildApiInfo())
            .select()
            // 要扫描的API(Controller)基础包
            .apis(RequestHandlerSelectors.basePackage("cn.itcast.wanxinp2p"))
            .paths(PathSelectors.any())
            .build();
    }

    private ApiInfo buildApiInfo() {
        Contact contact = new Contact("黑马程序员", "", "");
        return new ApiInfoBuilder()
            .title("万信金融P2P平台-用户服务API文档")
            .description("包含用户服务api")
            .contact(contact)
            .version("1.0.0").build();
    }
}
```

### 2.1.3 Swagger常用注解

在Java类中添加Swagger的注解即可生成Swagger接口文档，常用Swagger注解如下：

@Api：修饰整个类，描述Controller的作用 @ApiOperation：描述一个类的一个方法，或者说一个接口 @ApiParam：单个参数的描述信息

@ApiModel：用对象来接收参数

@ApiModelProperty：用对象接收参数时，描述对象的一个字段

@ApiResponse：HTTP响应其中1个描述

@ApiResponses：HTTP响应整体描述

@ApiIgnore：使用该注解忽略这个API

@ApiError：发生错误返回的信息

@ApiImplicitParam：一个请求参数

@ApiImplicitParams：多个请求参数的描述信息

@ApiImplicitParam属性：



属性	取值	作用
paramType		查询参数类型
	path	以地址的形式提交数据
	query	直接跟参数完成自动映射赋值
	body	以流的形式提交 仅支持POST
	header	参数在request headers 里边提交
	form	以form表单的形式提交 仅支持POST
dataType		参数的数据类型 只作为标志说明，并没有实际验证
	Long	
	String	
name		接收参数名
value		接收参数的意义描述
required		参数是否必填
	true	必填
	false	非必填
defaultValue		默认值

我们在ConsumerController中添加Swagger注解，代码如下所示：

```

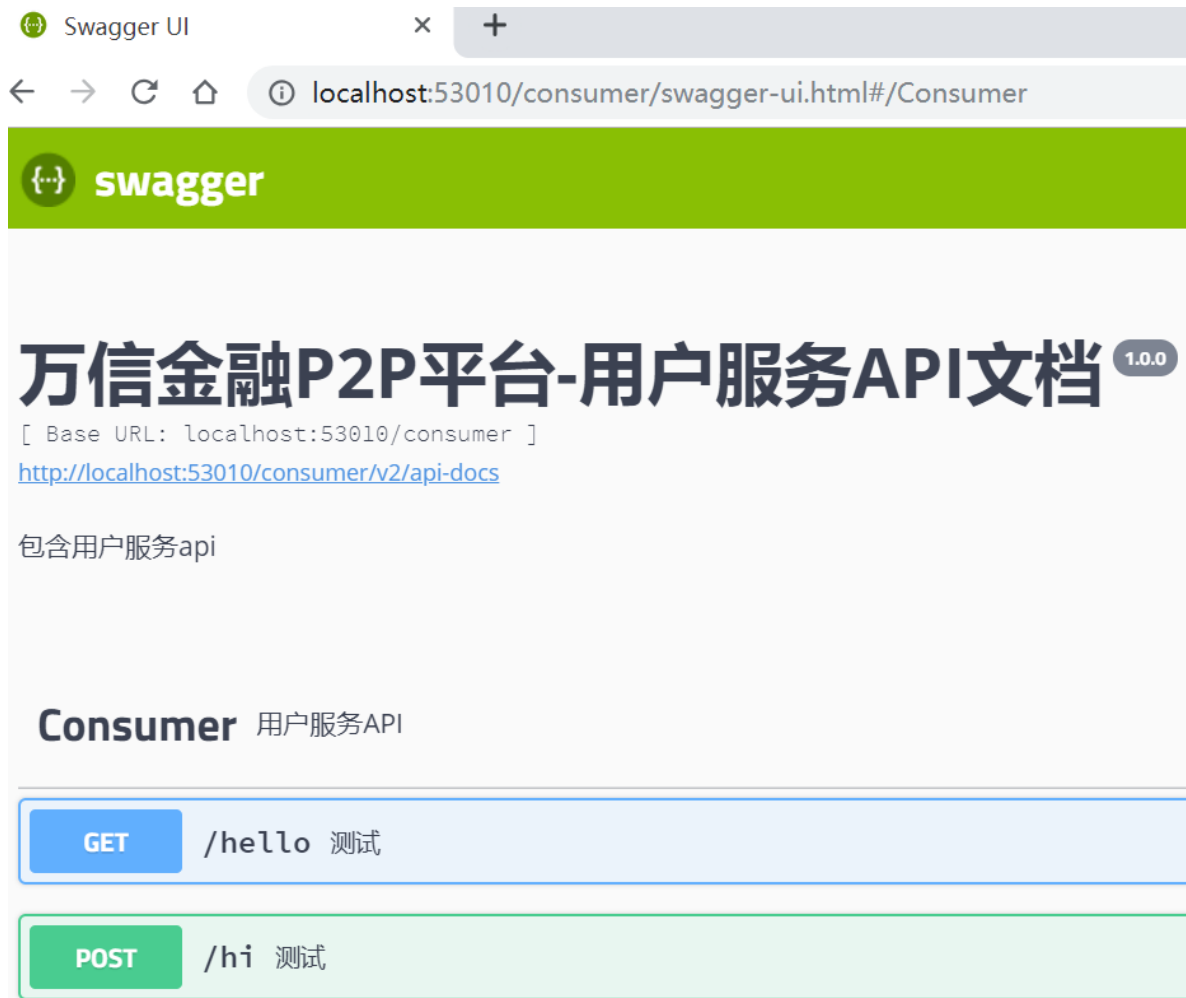
@RestController
@Api(value = "用户服务", tags = "Consumer", description = "用户服务API")
public class ConsumerController {

    @ApiOperation("测试")
    @GetMapping(path = "/hello")
    public String hello(){
        return "hello";
    }

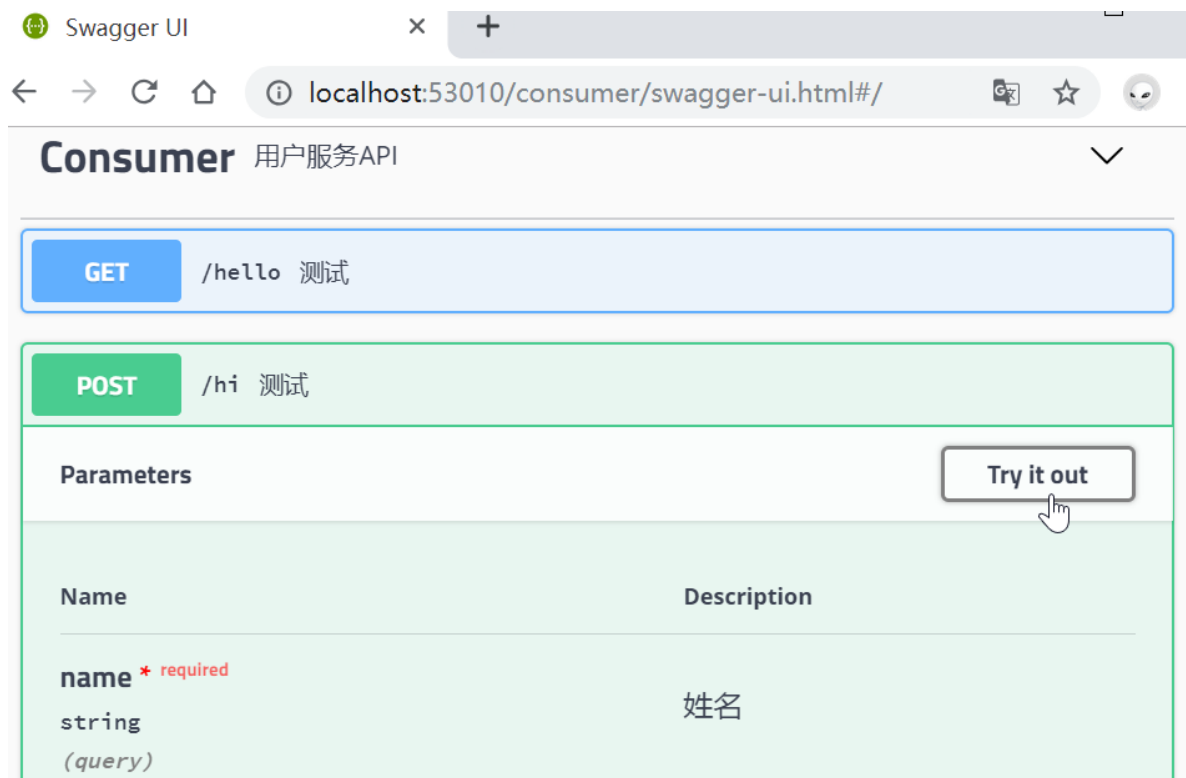
    @ApiOperation("测试")
    @ApiImplicitParam(name = "name", value = "姓名", required = true, dataType = "string")
    @PostMapping(value = "/hi")
    public String hi(String name) {
        return "hi,"+name;
    }
}
    
```

## 2.1.4 Swagger生成文档

1. 启动wanxinp2p-consumer-service服务工程，用浏览器访问：<http://localhost:53010/consumer/swagger-ui.html>



2. 点击其中任意一项即可打开接口详情，如下图所示：



3. 点击“Try it out”开始测试，并录入参数信息，如下图所示：

Swagger UI interface for the Consumer API. The 'POST /hi' endpoint is selected. A parameter 'name' of type 'string (query)' is defined with a description '姓名'. The input field contains '黑马程序员'. The 'Execute' button is highlighted with a mouse cursor.

4. 点击“Execute”发送请求，执行测试，如下图所示：

Swagger UI interface showing the results of the POST /hi request. The 'Execute' button was clicked, and the response is displayed. The response code is 200, and the response body is 'hi, 黑马程序员'.

Swagger生成API文档的工作原理：

1、wanxinp2p-consumer-service启动时会扫描到SwaggerConfiguration类

2、在此类中指定了扫描包路径cn.itcast.wanxinp2p，会找到在此包下及子包下标记有@RestController注解的controller类

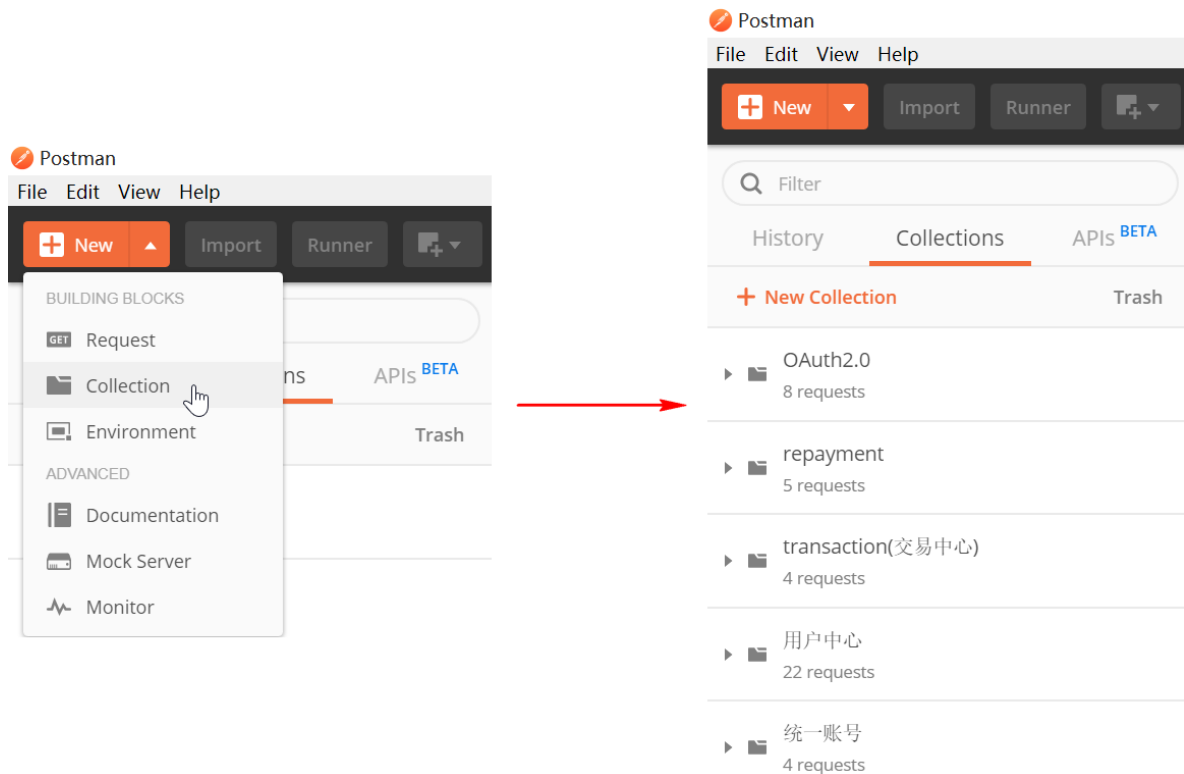
3、根据controller类中的Swagger注解生成API文档

## 2.2 接口调试利器：Postman

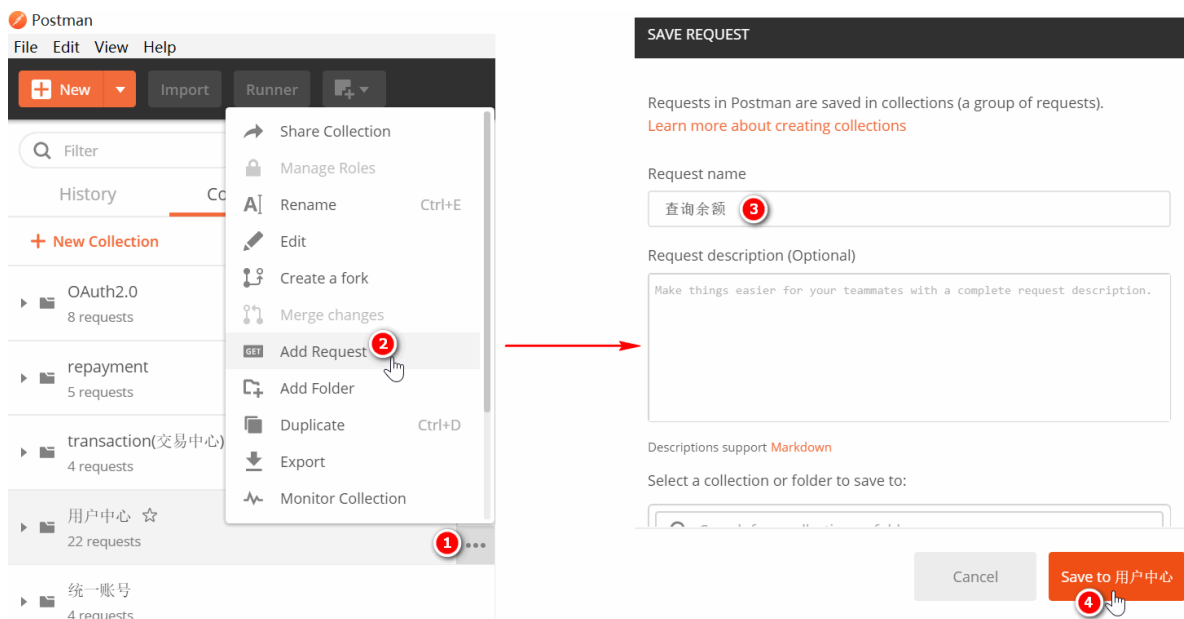
Postman是一款功能强大的http接口测试工具，使用Postman可以完成http各种请求的功能测试。作为服务器端开发人员，当一个业务功能开发完毕后，应该用Postman进行功能测试。

1、请自行在本机安装Postman，首次使用需要注册并登录。

2、新建集合(建议一个微服务新建一个对应的集合)



3、在某集合中新建请求，并录入请求信息



4、使用postman测试http接口

GET 查询余额

GET 1 http://localhost:53050/consumer/l/balances/USR\_0AB2431603BF4B258AB1366EB4EC1A4F 2 Send 4

Params 3 Authorization Headers (8) Body Pre-request Script Tests Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (3) Test Results Status: 200 OK Time: 2020 ms Size: 162 B Save

Pretty Raw Preview JSON 5

```

1 {
2   "code": -2,
3   "msg": "获取失败"
4 }
```

## 5、请求参数设置

### 1) get请求参数设置

POST 查询当登录用户充值记录

POST 1 http://localhost:53010/consumer/my/recharge-records/q?pageNo=1&pageSize=10&sortBy&order 2 Send 4

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Co

Query Params 3

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> pageNo	1	
<input checked="" type="checkbox"/> pageSize	10	
<input checked="" type="checkbox"/> sortBy		
<input checked="" type="checkbox"/> order		

### 2) post请求参数设置

- form表单数据：

POST 登录

POST 1 http://localhost:53010/uaa/oauth/token 2 Send 4

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Co

Body 3

none form-data x-www-form-urlencoded raw binary GraphQL BETA

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> grant_type	password	
<input checked="" type="checkbox"/> client_id	wanxin-p2p-web-h5	
<input checked="" type="checkbox"/> client_secret	itcasth5	
<input checked="" type="checkbox"/> username	admin	
<input checked="" type="checkbox"/> password	12345	

- JSON数据：

POST 注册

注册

POST 1 http://localhost:53050/consumer/reg 2 Send 4

Params Authorization Headers (1) Body Pre-request Script Tests Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL BETA JSON (application/json)

```
1 {  
2   "username": "createtest1",  
3   "password": "12345",  
4   "mobile": "133333433822",  
5   "role": "I"  
6 }
```

小技巧：每个测试都可以进行保存(Ctrl+S)，以便于后续使用。