

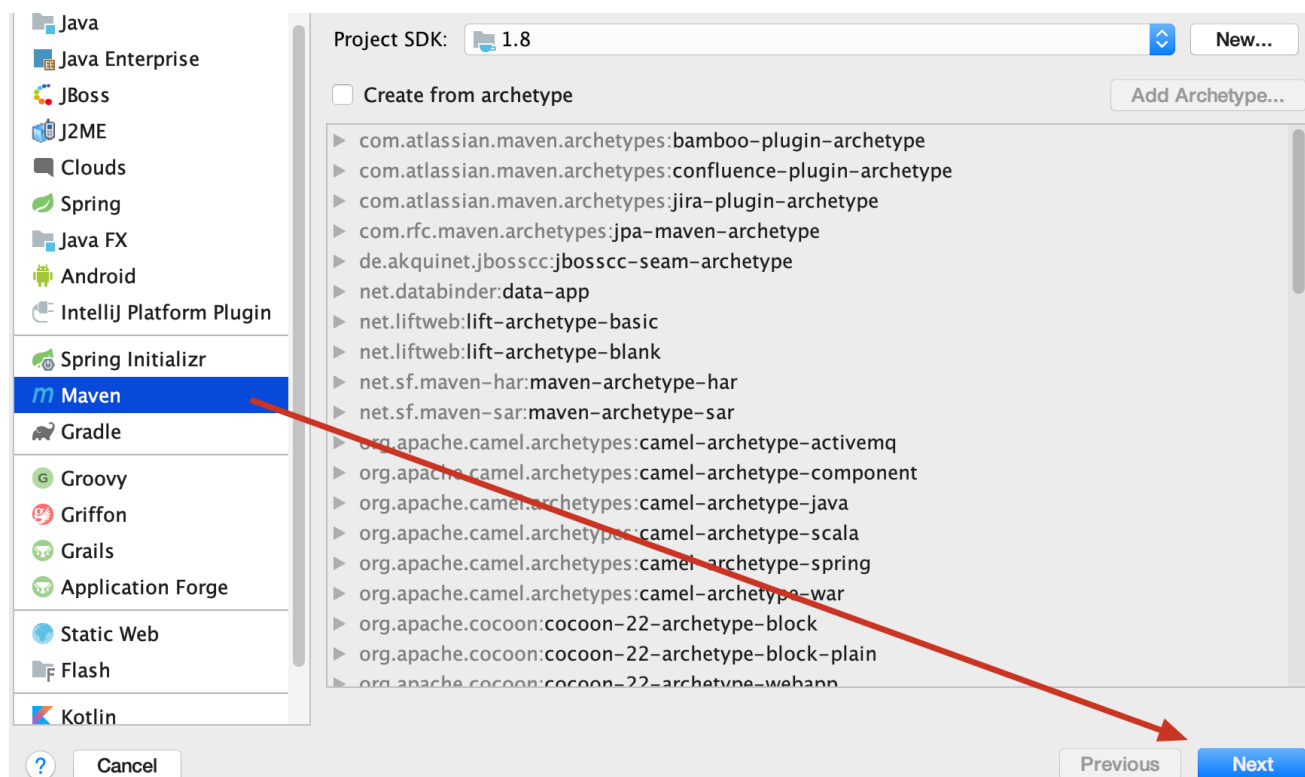
springboot 3.0

mybatis-plus

第一章 会员管理项目父模块搭建

1.1 创建模块 mengxuegu-member

mengxuegu-member 作为所有工程的父工程，用于管理项目的所有依赖。



GroupId	com.mengxuegu
ArtifactId	mengxuegu-member
Version	1.0-SNAPSHOT

然后点击右下角 `import Changes`

闪电PDF编辑器

<https://www.callmysoft.com/>

1.2 添加依赖 pom.xml

文件位于：会员管理系统/03-配套资料/pom文件/member-pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mengxuegu</groupId>
    <artifactId>mengxuegu-member</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>

    <!-- springboot依赖 -->
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.0.RELEASE</version>
    </parent>

    <!-- 依赖版本管理-->
    <properties>
        <mybatis-plus.version>3.3.2</mybatis-plus.version>
        <druid.version>1.1.21</druid.version>
        <fastjson.version>1.2.8</fastjson.version>
        <commons-lang.version>2.6</commons-lang.version>
        <commons-collections.version>3.2.2</commons-collections.version>
        <commons-io.version>2.6</commons-io.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <java.version>1.8</java.version>
    </properties>

    <!-- 实际依赖-->
    <dependencies>
        <!-- mybatis-plus启动器-->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-boot-starter</artifactId>
            <version>${mybatis-plus.version}</version>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
        </dependency>
        <!-- 生成setter,getter-->
        <dependency>
            <groupId>org.projectlombok</groupId>
```

闪电PDF编辑器

```

        <artifactId>lombok</artifactId>
    </dependency>
    <!--druid连接池-->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>druid-spring-boot-starter</artifactId>
        <version>${druid.version}</version>
    </dependency>

    <!--Spring Seucrity 加密模块-->
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-crypto</artifactId>
    </dependency>

    <!-- yml配置处理器 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-configuration-processor</artifactId>
        <optional>true</optional>
    </dependency>

    <!-- 工具类依赖 -->
    <dependency>
        <groupId>com.alibaba</groupId>
        <artifactId>fastjson</artifactId>
        <version>${fastjson.version}</version>
    </dependency>
    <dependency>
        <groupId>commons-lang</groupId>
        <artifactId>commons-lang</artifactId>
        <version>${commons-lang.version}</version>
    </dependency>
    <dependency>
        <groupId>commons-collections</groupId>
        <artifactId>commons-collections</artifactId>
        <version>${commons-collections.version}</version>
    </dependency>
    <dependency>
        <groupId>commons-io</groupId>
        <artifactId>commons-io</artifactId>
        <version>${commons-io.version}</version>
    </dependency>
</dependencies>

</project>

```

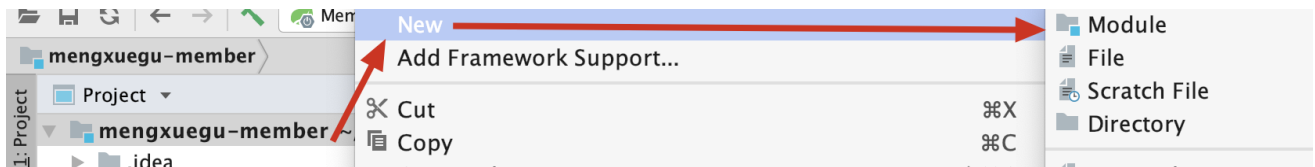
第二章 公共工具模块搭建

闪电PDF编辑器

<https://www.callmysoft.com/>

作用: `mengxuegu-member-util` 用于管理通用的工具类

2.1 创建模块 `mengxuegu-member-util`



右击然后点击右下角 `import Changes`

2.2 添加自定义日志文件

1. 将 `logback.xml` 日志配置文件添加到 `resources` 目录下,

位于: 会员管理系统/03-配套资料/logback.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--梦学谷 www.mengxuegu.com -->

<configuration>
    <!-- 彩色日志 -->
    <!-- 彩色日志依赖的渲染类 -->
    <conversionRule conversionWord="clr"
converterClass="org.springframework.boot.logging.logback.ColorConverter" />
    <conversionRule conversionWord="wex"
converterClass="org.springframework.boot.logging.logback.WhitespaceThrowableProxyConverter"
/>
    <conversionRule conversionWord="wEx"
converterClass="org.springframework.boot.logging.logback.ExtendedWhitespaceThrowableProxyCo
nverter" />
    <!-- 彩色日志格式 -->
    <property name="CONSOLE_LOG_PATTERN" value="${CONSOLE_LOG_PATTERN:-
%clr(%d{HH:mm:ss.SSS}){faint} %clr(${LOG_LEVEL_PATTERN:-%5p}) %clr(${PID:-  }){magenta}
%clr(---){faint} %clr([%15.15t]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint}
%m%n${LOG_EXCEPTION_CONVERSION_WORD:-%wEx}"/>
    <!-- ch.qos.logback.core.ConsoleAppender 表示控制台输出 -->
    <appender name="stdout" class="ch.qos.logback.core.ConsoleAppender">
        <layout class="ch.qos.logback.classic.PatternLayout">
            <pattern>${CONSOLE_LOG_PATTERN}</pattern>
        </layout>
    </appender>
    <root level="info">
        <appender-ref ref="stdout" />
    </root>
</configuration>
```

闪电PDF编辑器

<https://www.callmysoft.com/>

2.3 整合 Lombok

Lombok 介绍

官方网址: <https://www.projectlombok.org/features/all>

Lombok 工具提供一系列的注解, 使用这些注解可以不用定义getter、setter、equals、构造方法等, 可以消除java代码的臃肿, 它会在编译时在字节码文件自动生成这些通用的方法, 简化开发 人员的工作。

- `@Getter` 生成 getter 方法。
- `@Setter` 生成 setter 方法。
- `@ToString` 生成 toString 方法。
- `@NoArgsConstructor` 生成无参构造方法。
- `@AllArgsConstructor` 生成包含所有属性的构造方法。
- `@RequiredArgsConstructor` 会一个包含常量, 和标识了NotNull的变量的构造方法。生成的构造方法是私有的private。

主要使用 `@NoArgsConstructor` 和 `@AllArgsConstructor` 两个注解, 这样就不需要自己写构造方法, 代码简洁规范。

- `@Data` 生成 `setter`、`getter`、`toString`、`hashCode`、`equals` 和 `@RequiredArgsConstructor` 实现方法。
- `@Accessors(chain = true)` 生成的 setter 方法返回当前对象, 如下:
 - 类上加了 `@Accessors(chain = true)`, 对应生成的 setter 方法有返回值 `this`, 如下:

```
public Category setName(String name) {  
    this.name = name;  
    return this; // 会返回 this 当前对象  
}
```

- 类上没加 `@Accessors(chain = true)`, void 无返回值, 如下:

```
public void setName(String name) {  
    this.name = name;  
}
```

Lombok 使用

1. 在 mengxuegu-member 的 pom.xml 中添加依赖, 然后点击右下角 `import`

在上面已经添加过了

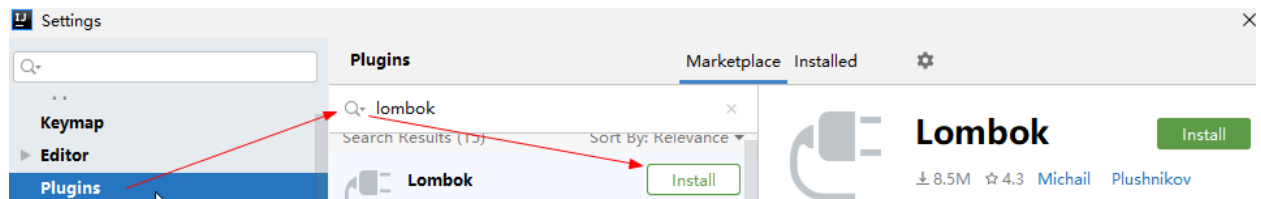
闪电PDF编辑器

<https://www.callmysoft.com/>

```
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
</dependency>
```

2. IDEA 安装 lombok 插件，

作用: 使用IDEA开发时，使用 Lombok 注解生成方法不报错。



2.4 规范统一响应枚举 ResultEnum

ResultEnum 枚举类是为了搭配 Result 规范响应的结果。

在 mengxuegu-member-util 模块创建 com.mengxuegu.member.base.ResultEnum 响应结果枚举。

文件位于：会员管理系统/03-配套资料/工具类/ResultEnum.java

```
package com.mengxuegu.member.base;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public enum ResultEnum {

    // 成功
    SUCCESS(2000, "成功"),
    // 错误
    ERROR(999, "错误");

    private Integer code;

    public String desc;

}
```

2.5 规范统一响应结果 Result

闪电PDF编辑器

<https://www.callmysoft.com/>

1. 说明：为了规范响应的结果，创建一个 `Result` 类来统一响应JSON格式：

code 操作代码、flag 是否成功、message 提示信息、data 自定义数据。

```
{
  "code": 2000,
  "flag": true,
  "message": "成功",
  "data": null
}
```

2. 在 mengxuegu-member-util 创建 `com.mengxuegu.member.base.Result` 用于封装接口统一响应结果。

文件位于：会员管理系统/03-配套资料/工具类/Result.java

```
package com.mengxuegu.member.base;

import com.alibaba.fastjson.JSON;
import lombok.Data;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.Serializable;

/**
 * 用于封装接口统一响应结果
 */
@Data
public class Result implements Serializable {

    private static final Logger logger = LoggerFactory.getLogger(Result.class);

    private static final long serialVersionUID = 1L;

    /**
     * 响应业务状态码
     */
    private Integer code;

    /**
     * 是否正常
     */
    private Boolean flag;

    /**
     * 响应信息
     */
    private String message;

    /**
     * 响应中的数据
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```

    */
    private Object data;

    public Result(Integer code, String message, Object data) {
        this.code = code;
        this.message = message;
        this.data = data;
        this.flag = code == ResultEnum.SUCCESS.getCode() ? true: false;
    }

    public static Result ok() {
        return new Result(ResultEnum.SUCCESS.getCode(), ResultEnum.SUCCESS.getDesc(),
null);
    }

    public static Result ok(Object data) {
        return new Result(ResultEnum.SUCCESS.getCode(), ResultEnum.SUCCESS.getDesc(),
data);
    }

    public static Result ok(String message, Object data) {
        return new Result(ResultEnum.SUCCESS.getCode(), message, data);
    }

    public static Result error(String message) {
        logger.debug("返回错误: code={}, message={}", ResultEnum.ERROR.getCode(), message);
        return new Result(ResultEnum.ERROR.getCode(), message, null);
    }

    public static Result build(int code, String message) {
        logger.debug("返回结果: code={}, message={}", code, message);
        return new Result(code, message, null);
    }

    public static Result build(ResultEnum resultEnum) {
        logger.debug("返回结果: code={}, message={}", resultEnum.getCode(),
resultEnum.getDesc());
        return new Result(resultEnum.getCode(), resultEnum.getDesc(), null);
    }

    public String toString() {
        return JSON.toJSONString(this);
    }
}

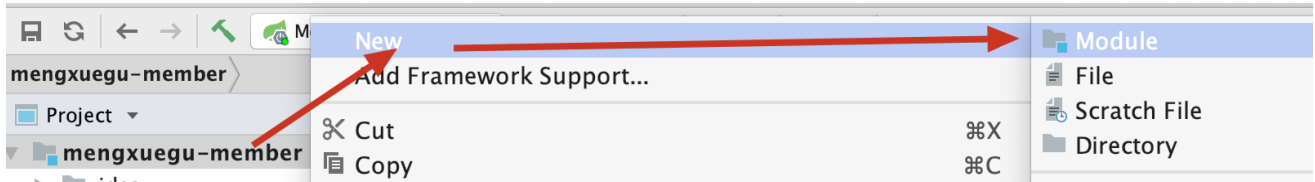
```

第三章 Api 接口模块搭建 mengxuegu-member-api

闪电PDF编辑器

<https://www.callmysoft.com/>

编写会员管理系统业务逻辑并向外提供 *RESTful* 风格接口给前端调用。



3.1 配置 pom.xml

mengxuegu-member-api 的 pom.xml 中添加 工具模块 和 web启动器依赖。

位于：会员管理系统/03-配套资料/pom文件/api-pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>mengxuegu-member</artifactId>
        <groupId>com.mengxuegu</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>mengxuegu-member-api</artifactId>

    <dependencies>
        <!--依赖工具模块-->
        <dependency>
            <groupId>com.mengxuegu</groupId>
            <artifactId>mengxuegu-member-util</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>

        <!--web启动器-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!--打包插件-->
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <!--指定启动类-->
```

```
        <mainClass>com.mengxuegu.member.MemberApplication</mainClass>
      </configuration>
    </plugin>
  </plugins>
</build>

</project>
```

3.2 创建启动类

在 mengxuegu-member-api 模块的src/main/java下创建 com.mengxuegu.member.MemberApplication

```
package com.mengxuegu.member;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MemberApplication {

    public static void main(String[] args) {
        SpringApplication.run(MemberApplication.class, args);
    }

}
```

3.3 创建 application.yml

```
server:
  port: 6666

# 数据源配置
spring:
  datasource:
    username: root
    password: root
    url: jdbc:mysql://127.0.0.1:3306/mxg_member?
    useUnicode=true&characterEncoding=utf8&useSSL=false&serverTimezone=GMT%2B8
    #mysql8版本以上驱动包指定新的驱动类
    driver-class-name: com.mysql.cj.jdbc.Driver
    # 数据源其他配置, 在 DruidConfig配置类中手动绑定
    initialSize: 8
    minIdle: 5
    maxActive: 20
    maxWait: 60000
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
timeBetweenEvictionRunsMillis: 60000
minEvictableIdleTimeMillis: 300000
validationQuery: SELECT 1 FROM DUAL
```

3.4 创建数据库

1. 创建 `mxg_member` 数据库
2. 导入数据库脚本：会员管理系统/03-配套资料/`mxg_member.sql`

3.5 创建会员管理启动类 Controller

```
package com.mengxuegu.member.controller;

import com.mengxuegu.member.base.Result;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

/**
 * <p>
 * 会员信息表 前端控制器
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
@RestController
@RequestMapping("/member")
public class MemberController {

    Logger logger = LoggerFactory.getLogger(getClass());

    @PostMapping("/list/search/{page}/{size}")
    public Result search(@PathVariable("page") long page,
                        @PathVariable("size") long size ) {
        logger.info("分页查询会员列表: page={}, size={}", page, size);
        return Result.ok();
    }
}
```

3.6 测试

启动项目，使用 postman 发送 POST 请求，访问 `localhost:6666/member/list/search/1/20`

第四章 整合 Mybatis-plus

参考: <https://mp.baomidou.com/>

4.1 创建 Mybatis-Plus 配置类

添加 Mybatis-Plus 配置类开启事务管理、Mapper接口扫描、分页功能。

```
package com.mengxuegu.member.config;

import com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor;
import org.mybatis.spring.annotation.MapperScan;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.transaction.annotation.EnableTransactionManagement;

@EnableTransactionManagement // 开启事务管理
@MapperScan("com.mengxuegu.member.mapper") // 扫描Mapper接口
@Configuration
public class MybatisPlusConfig {
    /**
     * 分页插件
     */
    @Bean
    public PaginationInterceptor paginationInterceptor() {
        return new PaginationInterceptor();
    }
}
```

4.2 修改 application.yml 扫描实体类与xxxMapper.xml

```
server:
  port: 6666

# 数据源配置
spring:
  datasource:
    username: root
    password: root

    url: jdbc:mysql://127.0.0.1:3306/mxg_member?
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```

useUnicode=true&characterEncoding=utf8&useSSL=false&serverTimezone=GMT%2B8
#mysql8版本以上驱动包指定新的驱动类
driver-class-name: com.mysql.cj.jdbc.Driver
# 数据源其他配置, 在 DruidConfig配置类中手动绑定
initialSize: 8
minIdle: 5
maxActive: 20
maxWait: 60000
timeBetweenEvictionRunsMillis: 60000
minEvictableIdleTimeMillis: 300000
validationQuery: SELECT 1 FROM DUAL

mybatis-plus:
  type-aliases-package: com.mengxuegu.member.entity
  # xxxMapper.xml 路径
  mapper-locations: classpath:com/mengxuegu/member/mapper/xml/**/*.xml

# 日志级别, 会打印sql语句
logging:
  level:
    com.mengxuegu.member.mapper: debug

```

创建 com.mengxuegu.member.entity 和 com.mengxuegu.member.mapper 包

4.3 编译 xxxMapper.xml 文件

我们将 xxxMapper.xml 会放到 src/main/java 目录下, 当文件编译时, 默认情况下不会将 Mapper.xml文件编译到 classes 中, 需要指定 `**/*.xml` 编译打包时, 把xml文件也一起打包。

在 mengxuegu-member-api/pom.xml 添加如下 resources 标签:

```

<build>

  <resources>
    <resource>
      <!--编译时, 默认情况下不会将 mapper.xml文件编译进去,
      src/main/java 资源文件的路径,
      **/*.xml 需要编译打包的文件类型是xml文件 -->
      <directory>src/main/java</directory>
      <includes>
        <include>**/*.xml</include>
      </includes>
    </resource>
    <resource>
      <directory>src/main/resources</directory>
    </resource>
  </resources>

  <plugins>
    <!--打包插件-->

```

闪电PDF编辑器

<https://www.callmysoft.com/>

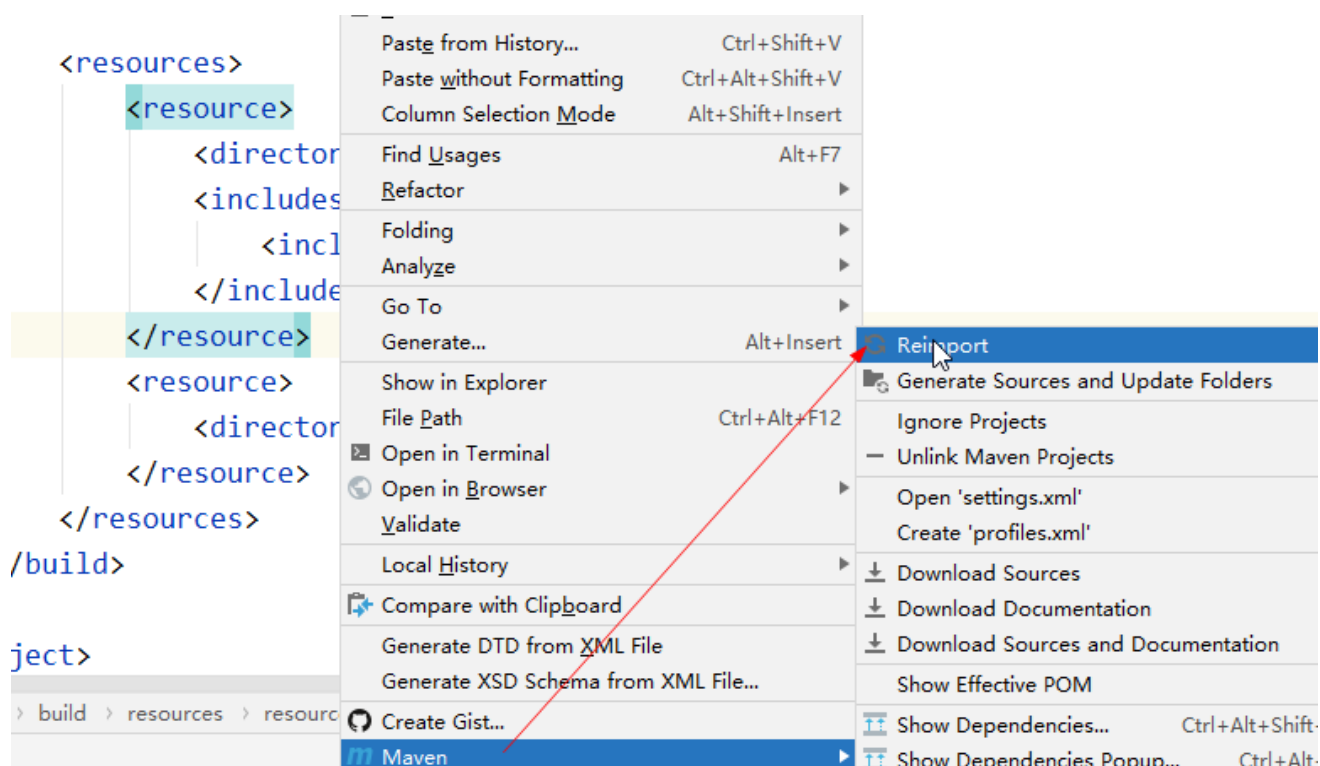
```

        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
            <configuration>
                <!--指定启动类-->
                <mainClass>com.mengxuegu.member.MemberApplication</mainClass>
            </configuration>
        </plugin>
    </plugins>

</build>

```

添加后记得pom.xml 文件中任意地方右击，**Maven > reimport** 才会生效



第五章 会员管理服务端-分页条件查询

需求：通过会员姓名、卡号、支付类型、会员生日 条件查询列表数据，并实现分页功能。

5.1 创建会员实体类 Member

```

package com.mengxuegu.member.entity;

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.annotation.TableId;

```

闪电PDF编辑器

```
import com.baomidou.mybatisplus.annotation.TableName;
import lombok.Data;
import lombok.experimental.Accessors;

import java.io.Serializable;
import java.util.Date;

/**
 * 会员信息表对应实体类
 */
@Accessors(chain = true)
@Data
@TableName("tb_member")
public class Member implements Serializable {

    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    /**
     * 会员卡号
     */
    private String cardNum;

    /**
     * 会员名字
     */
    private String name;

    /**
     * 生日
     */
    private Date birthday;

    /**
     * 手机号
     */
    private String phone;

    /**
     * 可用积分
     */
    private Integer integral;

    /**
     * 可用金额
     */
    private Double money;

    /**
     * 支付类型 ('1' 现金, '2' 微信, '3' 支付宝, '4' 银行卡)
     */
    private String payType;

    /**
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
    * 会员地址
    */
    private String address;
}
```

5.2 创建会员请求类 MemberREQ

REQ：作为 request 简写，主要作用是把将查询条件请求参数封装为一个对象。

比如：会员姓名、卡号、支付类型、会员生日 作为条件，查询出对应分类数据。

1. 创建 `com.mengxuegu.member.req.MemberREQ` 类

```
package com.mengxuegu.member.req;

import lombok.Data;

import java.io.Serializable;
import java.util.Date;

/**
 * 会员查询条件请求类
 */
@Data
public class MemberREQ implements Serializable {

    /**
     * 会员姓名
     */
    private String name;

    /**
     * 卡号
     */
    private String cardNum;

    /**
     * 支付类型 ('1'现金, '2'微信, '3'支付宝, '4'银行卡)
     */
    private String payType;

    /**
     * 会员生日
     */
    private Date birthday;
}
```

闪电PDF编辑器

<https://www.callmysoft.com/>

5.3 编写 MemberMapper

1. 创建接口 `com.mengxuegu.member.mapper.MemberMapper` 继承 `BaseMapper<Member>` 接口。

MyBatis-Plus 的 `BaseMapper<T>` 接口提供了很多对 `T` 表的数据操作方法

```
package com.mengxuegu.member.mapper;

import com.mengxuegu.member.entity.Member;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;

/**
 * <p>
 * 会员信息表 Mapper 接口
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
public interface MemberMapper extends BaseMapper<Member> {

}
```

2. java 目录下创建映射文件 `com/mengxuegu/member/mapper/xml/MemberMapper.xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.mengxuegu.member.mapper.MemberMapper">

</mapper>
```

5.4 创建业务层

1. 创建接口 `com.mengxuegu.member.service.IMemberService` 继承 `IService<Member>` 接口

实现 `IService<T>` 接口, 提供了常用更复杂的对 `T` 数据表的操作, 比如: 支持 Lambda 表达式, 批量删除、自动新增或更新操作等方法

2. 定义一个通过分页条件查询方法 `search`

```
package com.mengxuegu.member.service;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.entity.Member;
import com.baomidou.mybatisplus.extension.service.IService;
import com.mengxuegu.member.req.MemberREQ;
```

```
/**
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```

* <p>
* 会员信息表 服务类
* </p>
*
* @author 梦学谷-www.mengxuegu.com
*/
public interface IMemberService extends IService<Member> {

    Result search(long page, long size, MemberREQ req);

}

```

3. 创建实现类 `com.mengxuegu.member.service.impl.MemberServiceImpl` 继承 `ServiceImpl<MemberMapper, Member>` 类, 并且实现 `IMemberService` 接口。

`ServiceImpl<M extends BaseMapper<T>, T>` 是对 `IService` 接口中方法的实现

- 第1个泛型 M 指定继承了 BaseMapper 接口的子接口
- 第2个泛型 T 指定实体类

注意: 类上不要少了 `@Service`

baseMapper 引用的就是 MemberMapper 实例

```

package com.mengxuegu.member.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.entity.Member;
import com.mengxuegu.member.mapper.MemberMapper;
import com.mengxuegu.member.req.MemberREQ;
import com.mengxuegu.member.service.IMemberService;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import org.apache.commons.lang.StringUtils;
import org.springframework.stereotype.Service;

/**
* <p>
* 会员信息表 服务实现类
* </p>
* @author 梦学谷-www.mengxuegu.com
*/
@Service
public class MemberServiceImpl extends ServiceImpl<MemberMapper, Member> implements
IMemberService {

    @Override
    public Result search(long page, long size, MemberREQ req) {
        // 封装查询条件
        QueryWrapper<Member> query = new QueryWrapper<>();
        if (req != null) {
            if (StringUtils.isNotBlank(req.getName())) {

```

```

        query.like("name", req.getName());
    }
    if(StringUtils.isNotBlank(req.getCardNum())) {
        query.like("card_num", req.getCardNum());
    }
    if(StringUtils.isNotBlank(req.getPayType())) {
        query.eq("pay_type", req.getPayType());
    }
    if(req.getBirthday() != null) {
        query.eq("birthday", req.getBirthday());
    }
}

// 封装分页对象
IPage<Member> p = new Page<>(page, size);
IPage<Member> data = baseMapper.selectPage(p, query);
return Result.ok(data);
}

}

```

5.5 创建控制层

1. 创建控制层类 com.mengxuegu.member.controller.MemberController

```

package com.mengxuegu.member.controller;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.req.MemberREQ;
import com.mengxuegu.member.service.IMemberService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

/**
 * <p>
 * 会员信息表 前端控制器
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
@RestController
@RequestMapping("/member")
public class MemberController {

    Logger logger = LoggerFactory.getLogger(getClass());

    @Autowired // 不要少了注解
    private IMemberService;
}

```

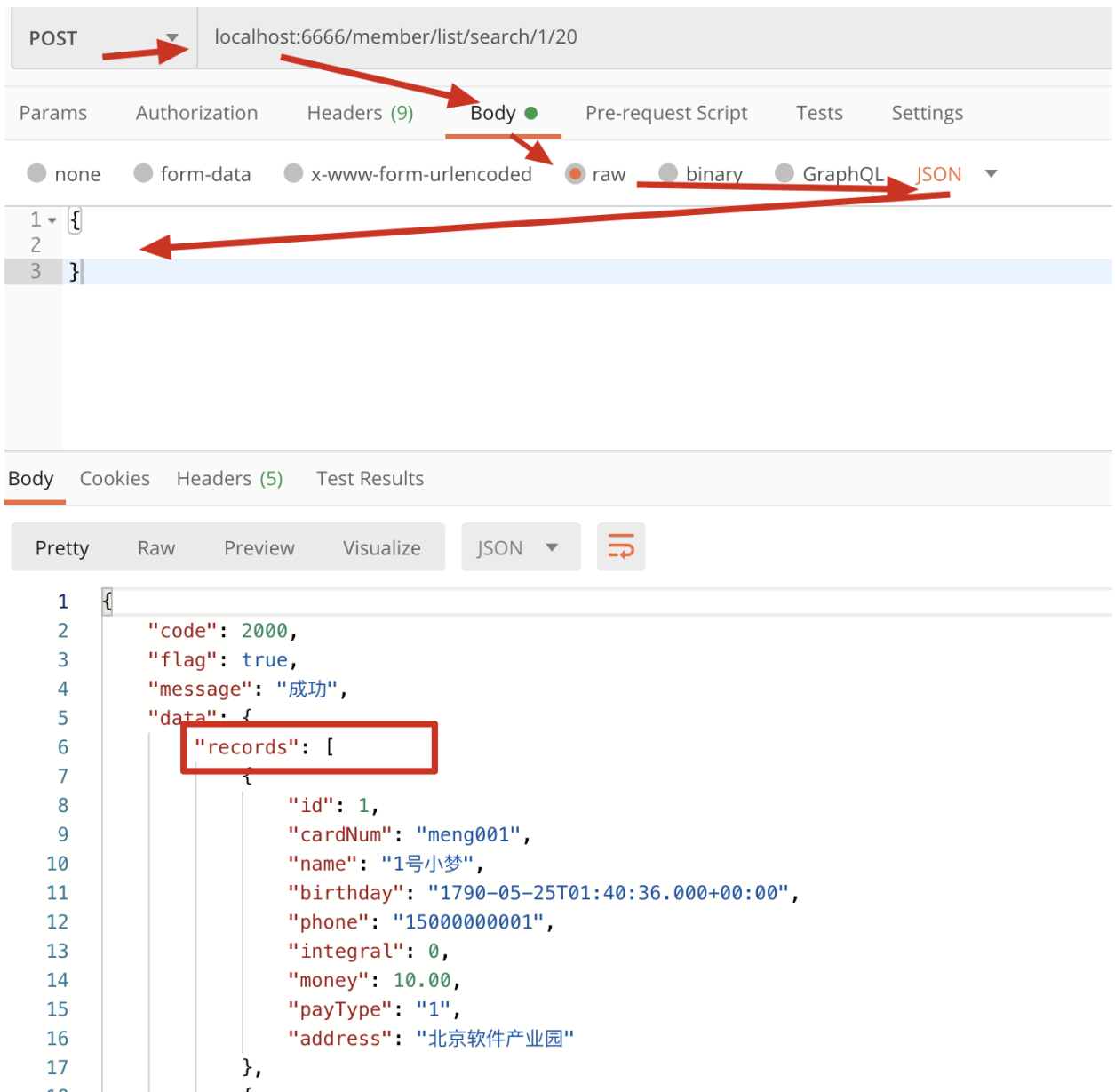
```
/**
 * 分页条件查询会员列表
 * @param page 页码
 * @param size 每页显示记录数
 * @param req 查询条件
 * @return
 */
@PostMapping("/list/search/{page}/{size}")
public Result search(@PathVariable("page") long page,
                    @PathVariable("size") long size,
                    @RequestBody(required=false) MemberREQ req) {
    logger.info("分页查询会员列表: page={}, size={}", page, size);
    return memberService.search(page, size, req);
}
}
```

5.6 启动测试

1. 发送 POST 请求，访问 localhost:6666/member/list/search/1/20

闪电PDF编辑器

<https://www.callmysoft.com/>



上面响应的列表数据在 records 中。

2. 而我们前端在 easymock 上模拟响应的列表数据是在 rows 中，这样显然有问题，应该想办法把列表数据放到 rows 中才行。

闪电PDF编辑器

<https://www.callmysoft.com/>

```

{
  "code": 2000,
  "flag": true,
  "message": "查询成功",
  "data": {
    "total": "@integer(100,200)", // 查询出来的总记录数
    "rows": 10: [{ // 返回当前页的记录数 10 条, 即每页显示 10 条记录
      "id+1": 10,
      "cardNum": "@integer(10000)", //大于1000的正整数
      "name": "@cname",
      "birthday": "@date",
      "phone11": "@integer(0,9)", // 11个位数字
      "integral": "@integer(0, 500)",
      "money": "@float(0, 100, 1, 3)", // 0-1000小数,1-3位小数位
      "payType1": ['1', '2', '3', '4'],
      "address": "@county(true)"
    }]
  }
}

```

下面我们就自定义一个 Page 类

5.7 封装 Page 分页类

因为前端要求返回 rows, 而默认是在 records 中, 下面自定义一个 Page 类继承 Mybatis-Plus 的提供的Page类,

1. 在 mengxuegu-member-util 模块中创建 com.mengxuegu.member.base.Page

```

package com.mengxuegu.member.base;

import lombok.Data;

import java.util.List;

/**
 * 因为前端分页数据是 rows , 不是records,在这里转换下
 * @param <T>
 */
@Data
public class Page<T> extends
    com.baomidou.mybatisplus.extension.plugins.pagination.Page<T> {

    /**
     * 因为前端分页数据是 rows , 不是records
     * @return
     */
    public List<T> getRows() {
        // 调用父类的
        return super.getRecords();
    }
}

```

```

    }

    // 把它设置为null,不然records也有数据
    public List<T> getRecords() {
        return null;
    }

    public Page(long current, long size) {
        super(current, size);
    }
}

```

2. 将 MemberServiceImpl 类导入的 Page 类, 改为导入 `import com.mengxuegu.member.base.Page;`

```

MemberServiceImpl.java x
1 package com.mengxuegu.member.service.impl;
2
3 import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
4 import com.baomidou.mybatisplus.core.metadata.IPage;
5 //import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
6 import com.mengxuegu.member.base.Page;
7 import com.mengxuegu.member.base.Result;
8 import com.mengxuegu.member.entity.Member;

```

3. 重启项目, 重新发送 POST 请求 `localhost:6666/member/list/search/1/20`

闪电PDF编辑器

<https://www.callmysoft.com/>

POST localhost:6666/member/list/search/1/20

Params Authorization Headers (9) Body Pre-request Script Tests Settings

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

```
1 {
2
3 }
```

body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "code": 2000,
3   "flag": true,
4   "message": "成功",
5   "data": {
6     "records": null,
7     "total": 5,
8     "size": 20,
9     "current": 1,
10    "orders": [],
11    "hitCount": false,
12    "rows": [
13      {
14        "id": 1,
15        "cardNum": "meng001",
16        "name": "1号小梦",
17        "birthday": "1790-05-25T01:40:36.000+00:00",
18        "phone": "15000000001",
19        "integral": 0,
20        "money": 10.00,
21        "payType": "1",
22        "address": "北京软件产业园"
```

被为 null

数据在 rows 中

第六章 会员管理服务端-增删改查

6.1 新增与删除会员

闪电PDF编辑器

<https://www.callmysoft.com/>

新增与删除会员只要对 tb_member 单表操作，并且我们可以直接使用 mybatis-plus 提供的 IMemberService 方法进行操作即可。

添加控制层方法

在 com.mengxuegu.member.controller.MemberController 类中添加 add 和 delete 方法：

```
/**
 * 新增会员
 * @param member
 * @return
 */
@PostMapping // /member
public Result add(@RequestBody Member member) {
    boolean b = memberService.save(member);
    if(b) {
        return Result.ok();
    }
    return Result.error("新增会员信息失败");
}

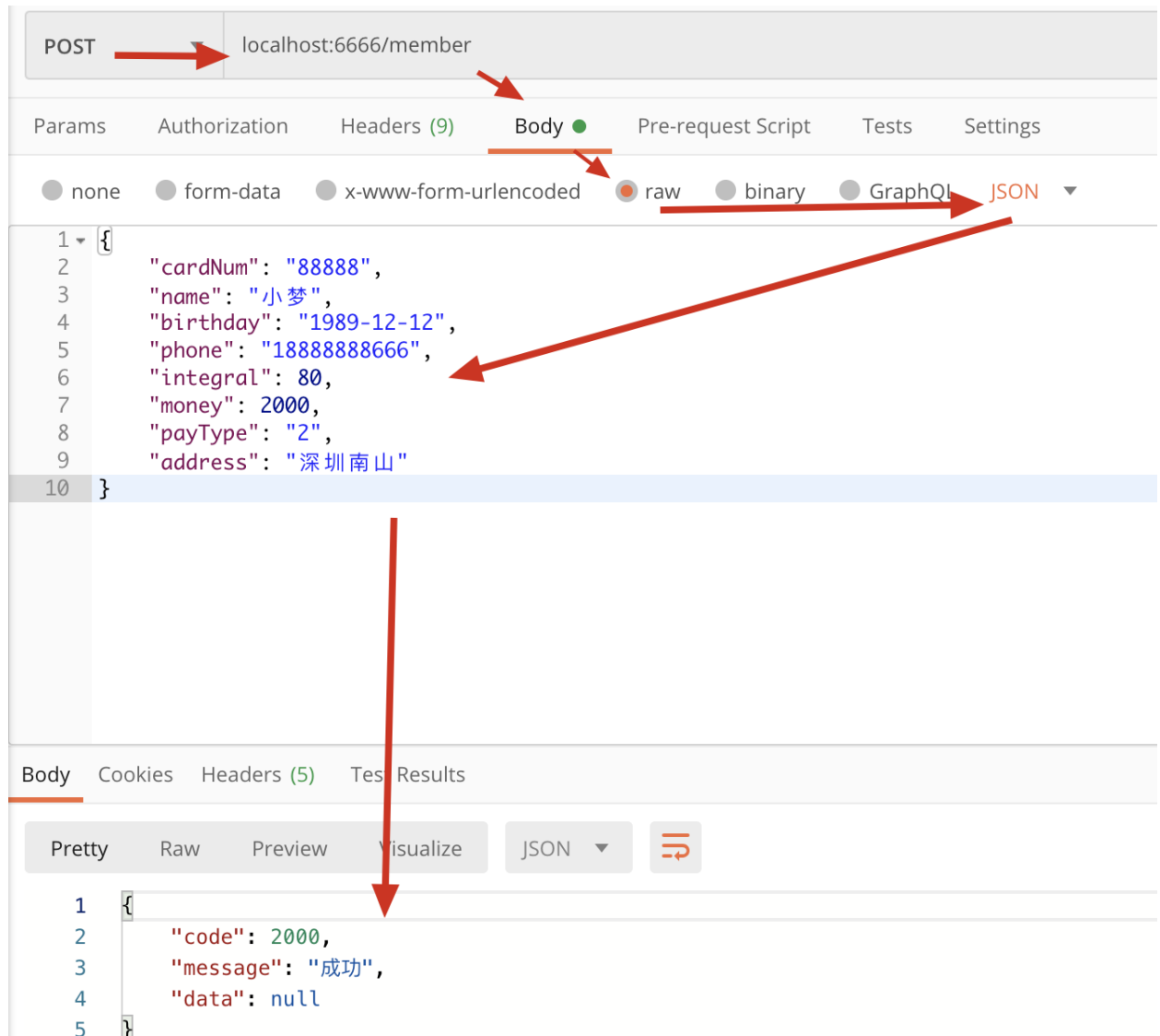
/**
 * 删除会员
 * @return
 */
@DeleteMapping("/{id}")
public Result delete(@PathVariable("id") int id) {
    boolean b = memberService.removeById(id);
    if(b){
        return Result.ok();
    }
    return Result.error("删除会员信息失败");
}
```

测试

1. 新增会员，发送 POST 请求 localhost:6666/member

闪电PDF编辑器

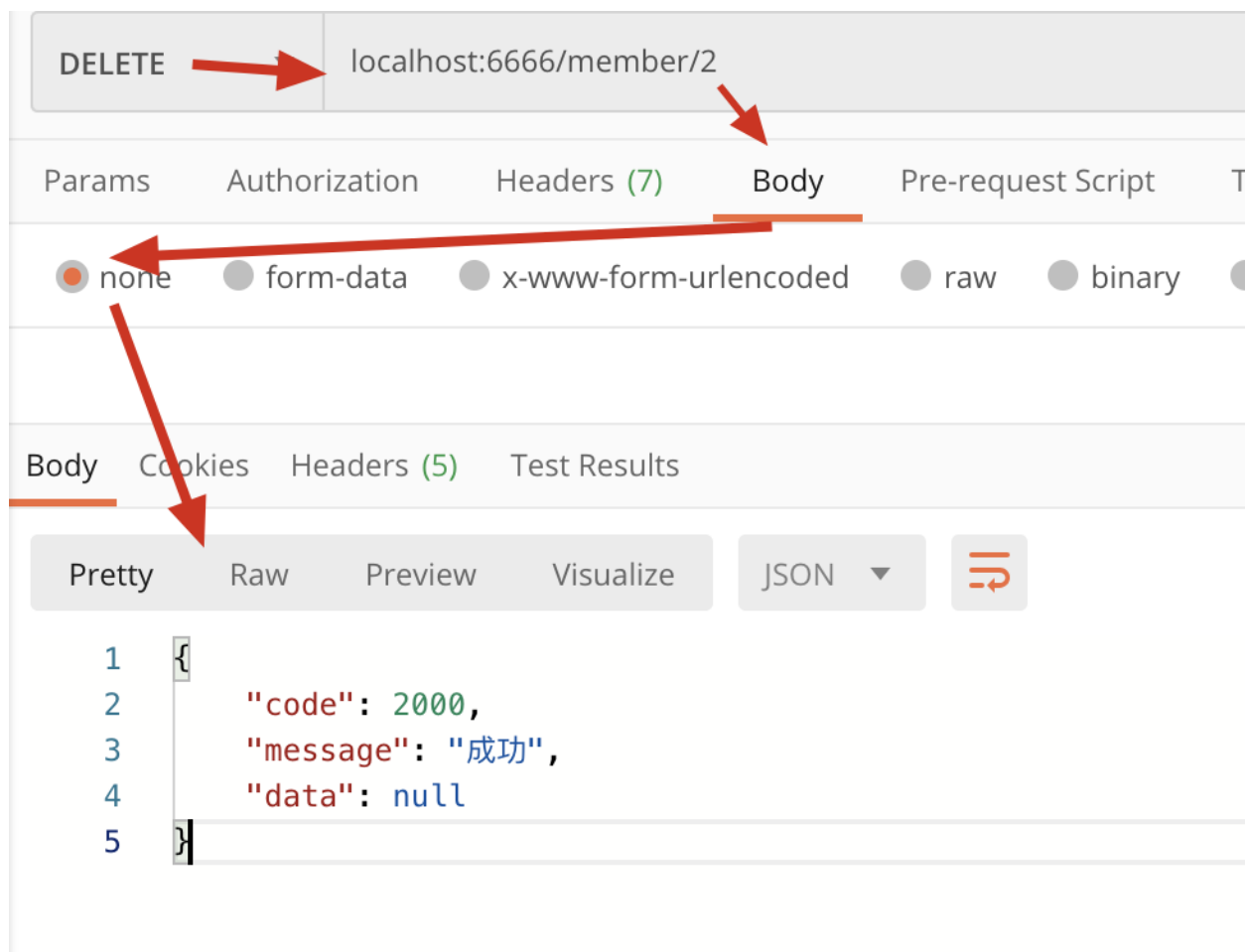
<https://www.callmysoft.com/>



2. 删除会员, 发送 DELETE 请求 localhost:6666/member/2

闪电PDF编辑器

<https://www.callmysoft.com/>



6.2 查询会员详情与修改会员

修改会员信息，要先查询会员详情回显，然后再将修改后的数据提交更新

添加控制层方法

在 `com.mengxuegu.member.controller.MemberController` 类中添加 `get` 和 `update` 方法：

```
/**
 * 通过id查询详情
 * @param id
 * @return
 */
@GetMapping("/{id}") // /member/{id}
public Result get(@PathVariable("id") int id) {
    Member member = memberService.getById(id);
    return Result.ok(member);
}

/**
 * 修改会员
```

闪电PDF编辑器

```

    * @param member
    * @return
    */
    @PutMapping("/{id}") // /member/{id}
    public Result update(@PathVariable("id") int id,
                        @RequestBody Member member) {
        return memberService.update(id, member);
    }

```

添加业务层方法

在com.mengxuegu.member.service.IMemberService添加 update 接口方法

```
Result update(int id, Member member);
```

在 com.mengxuegu.member.service.impl.MemberServiceImpl 实现 update 方法

```

@Override
public Result update(int id, Member member) {
    if(member.getId() == null) {
        member.setId(id);
    }
    //更新操作
    int size = baseMapper.updateById(member);
    if(size < 1) {
        return Result.error("修改会员信息失败");
    }
    return Result.ok();
}

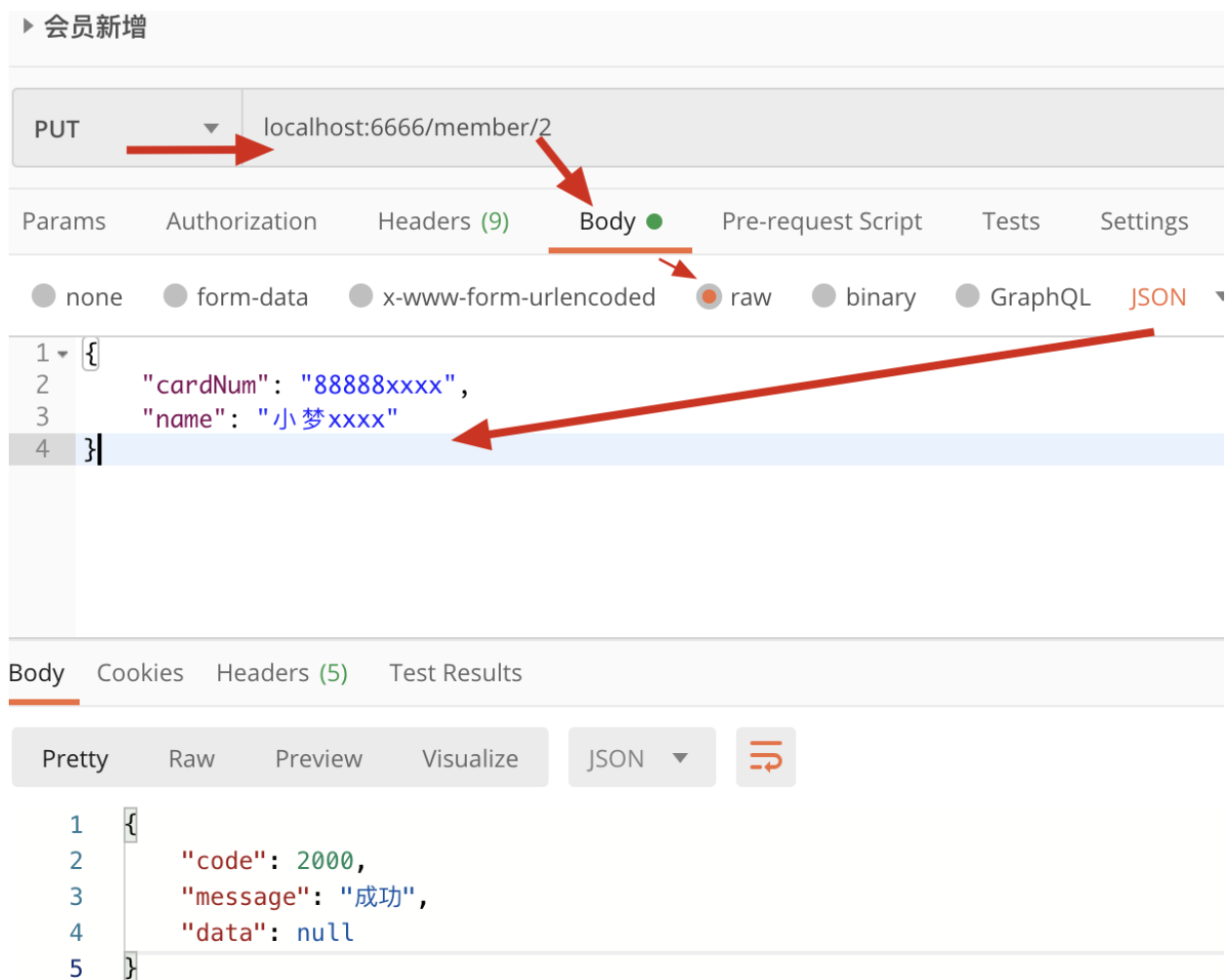
```

测试

1. 查询会员详情，发送 GET 请求 localhost:6666/member/2
2. 修改会员，发送 PUT 请求 localhost:6666/member/2

闪电PDF编辑器

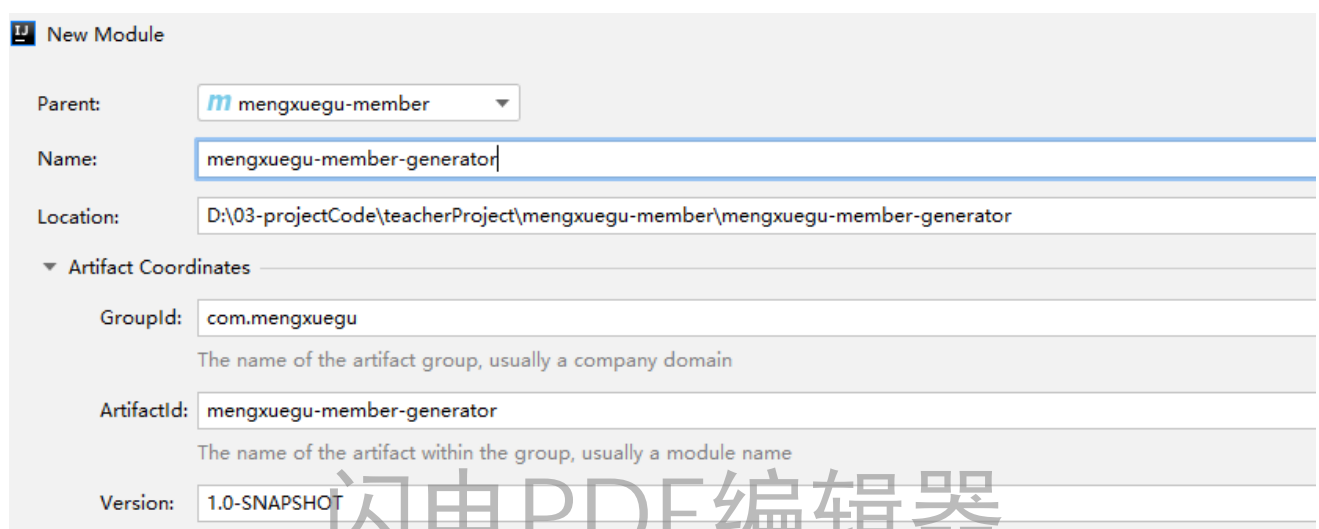
<https://www.callmysoft.com/>



第七章 MyBatis-plus 代码生成器

参考官网: <https://mybatis.plus/guide/generator.html>

7.1 创建代码生成器模块 mengxuegu-member-generator



7.2 添加依赖 pom.xml

文件位于：会员管理系统/03-配套资料/pom文件/generator-pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>com.mengxuegu</artifactId>
        <groupId>mengxuegu-member</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>mengxuegu-member-generator</artifactId>

    <dependencies>
        <!-- 代码生成器核心依赖 -->
        <dependency>
            <groupId>com.baomidou</groupId>
            <artifactId>mybatis-plus-generator</artifactId>
            <version>3.3.1</version>
        </dependency>
        <!-- mybatis-plus必须要配置模板引擎 -->
        <dependency>
            <groupId>org.freemarker</groupId>
            <artifactId>freemarker</artifactId>
        </dependency>
    </dependencies>

</project>
```

7.3 编写自动生成器代码

常用配置项

相关代码生成器配置项，可参考文档：<https://mybatis.plus/config/>

创建生成器代码

创建类 com.mengxuegu.generator.CodeGenerator

文件位于：会员管理系统/03-配套资料/CodeGenerator.java

```
package com.mengxuegu.generator;
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```

import com.baomidou.mybatisplus.annotation.IdType;
import com.baomidou.mybatisplus.core.exceptions.MybatisPlusException;
import com.baomidou.mybatisplus.generator.AutoGenerator;
import com.baomidou.mybatisplus.generator.config.DataSourceConfig;
import com.baomidou.mybatisplus.generator.config.GlobalConfig;
import com.baomidou.mybatisplus.generator.config.PackageConfig;
import com.baomidou.mybatisplus.generator.config.StrategyConfig;
import com.baomidou.mybatisplus.generator.config.rules.DateType;
import com.baomidou.mybatisplus.generator.config.rules.NamingStrategy;
import com.baomidou.mybatisplus.generator.engine.FreemarkerTemplateEngine;
import org.apache.commons.lang.StringUtils;

import java.util.Scanner;

public class CodeGenerator {

    // 生成的代码放到哪个工程中
    private static String PROJECT_NAME = "mengxuegu-member-api";

    // 数据库名称
    private static String DATABASE_NAME = "mxg_member";

    public static void main(String[] args) {
        // 代码生成器
        AutoGenerator mpg = new AutoGenerator();
        // 数据源配置
        DataSourceConfig dsc = new DataSourceConfig();
        dsc.setUrl("jdbc:mysql://localhost:3306/" + DATABASE_NAME + "?
useUnicode=true&characterEncoding=utf8&useSSL=false&serverTimezone=GMT%2B8");
        dsc.setDriverName("com.mysql.cj.jdbc.Driver");
        dsc.setUsername("root");
        dsc.setPassword("root");
        mpg.setDataSource(dsc);

        // 全局配置
        GlobalConfig gc = new GlobalConfig();
        String projectPath = System.getProperty("user.dir") + "/";
        gc.setOutputDir(projectPath + PROJECT_NAME + "/src/main/java");
        gc.setIdType(IdType.AUTO); // 自增长id
        gc.setAuthor("梦学谷-www.mengxuegu.com");
        gc.setFileOverride(true); //覆盖现有的
        gc.setOpen(false); //是否生成后打开
        gc.setDateType(DateType.ONLY_DATE);
        mpg.setGlobalConfig(gc);

        // 包配置
        PackageConfig pc = new PackageConfig();
        pc.setParent("com.mengxuegu.member"); //父包名
        mpg.setPackageInfo(pc);

        // 策略配置

```

闪电PDF编辑器

<https://www.callmysoft.com/>

```

StrategyConfig strategy = new StrategyConfig();
strategy.setNaming(NamingStrategy.underline_to_camel);
strategy.setColumnNaming(NamingStrategy.underline_to_camel);
strategy.setEntityLombokModel(true); //使用lombok
strategy.setEntitySerialVersionUID(true); // 实体类的实现接口Serializable
strategy.setRestControllerStyle(true); // @RestController
strategy.setInclude(scanner("表名, 多个英文逗号分割").split(","));
strategy.setControllerMappingHyphenStyle(true);
strategy.setTablePrefix("tb_"); // 去掉表前缀
mpg.setStrategy(strategy);

mpg.setTemplateEngine(new FreemarkerTemplateEngine());
mpg.execute();
}

/**
 * <p>
 * 读取控制台内容
 * </p>
 */
public static String scanner(String tip) {
    Scanner scanner = new Scanner(System.in);
    StringBuilder help = new StringBuilder();
    help.append("请输入" + tip + ": ");
    System.out.println(help.toString());
    if (scanner.hasNext()) {
        String ipt = scanner.next();
        if (StringUtils.isNotBlank(ipt)) {
            return ipt;
        }
    }
    throw new MybatisPlusException("请输入正确的" + tip + " ! ");
}
}

```

7.4 生成供应商&商品&员工管理代码

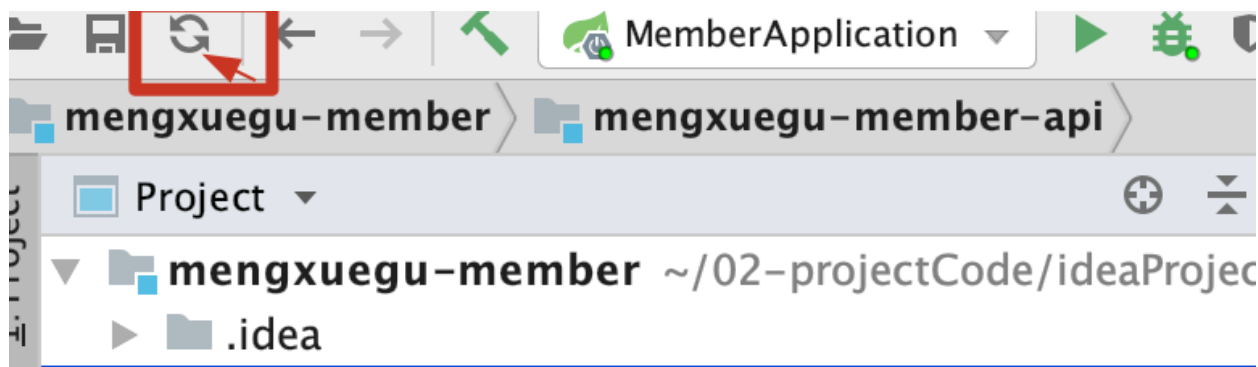
1. 执行 main 方法，控制台输入表名回车，会自动生成对应项目目录中



2. 点击IDEA 左上角刷新对应文件就会加载

闪电PDF编辑器

<https://www.callmysoft.com/>



3. 注意：如何重新生成会覆盖当前存在的，如果基于生成代码开发过功能，要谨慎，以防被覆盖了。

第八章 供应商管理服务端-分页条件查询

创建供应商请求类 SupplierREQ

1. 创建 `com.mengxuegu.member.req.SupplierREQ` 类

```
package com.mengxuegu.member.req;

import lombok.Data;

import java.io.Serializable;

/**
 * 供应商查询条件请求类
 */
@Data
public class SupplierREQ implements Serializable {

    /**
     * 供应商名称
     */
    private String name;

    /**
     * 联系人
     */
    private String linkman;

    /**
     * 联系电话
     */
    private String mobile;

}
```

闪电PDF编辑器

<https://www.callmysoft.com/>

编写业务层

1. 在接口 `com.mengxuegu.member.service.ISupplierService` 定义一个通过分页条件查询方法 `search`

```
package com.mengxuegu.member.service;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.entity.Supplier;
import com.baomidou.mybatisplus.extension.service.IService;
import com.mengxuegu.member.req.SupplierREQ;

/**
 * <p>
 * 供应商信息表 服务类
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
public interface ISupplierService extends IService<Supplier> {

    Result search(long page, long size, SupplierREQ req);

}
```

2. 在实现类 `com.mengxuegu.member.service.impl.SupplierServiceImpl` 实现 `search` 方法。

注意：Page导包要导入自定义的 `import com.mengxuegu.member.base.Page;`

```
package com.mengxuegu.member.service.impl;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.mengxuegu.member.base.Page;
import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.entity.Supplier;
import com.mengxuegu.member.mapper.SupplierMapper;
import com.mengxuegu.member.req.SupplierREQ;
import com.mengxuegu.member.service.ISupplierService;
import org.apache.commons.lang.StringUtils;
import org.springframework.stereotype.Service;

/**
 * <p>
 * 供应商信息表 服务实现类
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
@Service

public class SupplierServiceImpl extends ServiceImpl<SupplierMapper, Supplier> implements
```

闪电PDF编辑器

```

ISupplierService {

    @Override
    public Result search(long page, long size, SupplierREQ req) {
        // 封装查询条件
        QueryWrapper<Supplier> query = new QueryWrapper<>();
        if(req != null) {
            if(StringUtils.isNotBlank(req.getName())) {
                query.like("name", req.getName());
            }
            if(StringUtils.isNotBlank(req.getLinkman())) {
                query.like("linkman", req.getLinkman());
            }
            if(StringUtils.isNotBlank(req.getMobile())) {
                query.like("mobile", req.getMobile());
            }
        }
        // 封装分页对象
        IPage<Supplier> data =
            baseMapper.selectPage(new Page<Supplier>(page, size), query);
        return Result.ok(data);
    }
}

```

编写控制层

1. 在控制层类 com.mengxuegu.member.controller.SupplierController

```

package com.mengxuegu.member.controller;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.req.SupplierREQ;
import com.mengxuegu.member.service.ISupplierService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

/**
 * <p>
 * 供应商信息表 前端控制器
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
@RestController
@RequestMapping("/supplier")
public class SupplierController {

```

闪电PDF编辑器

```
Logger logger = LoggerFactory.getLogger(getClass());

@Autowired // 不要少了注解
private ISupplierService supplierService;

/**
 * 分页条件查询供应商列表
 * @param page 页码
 * @param size 每页显示记录数
 * @param req 查询条件
 * @return
 */
@PostMapping("/list/search/{page}/{size}")
public Result search(@PathVariable("page") long page,
                    @PathVariable("size") long size,
                    @RequestBody(required=false) SupplierREQ req) {
    logger.info("分页查询供应商列表: page={}, size={}", page, size);
    return supplierService.search(page, size, req);
}

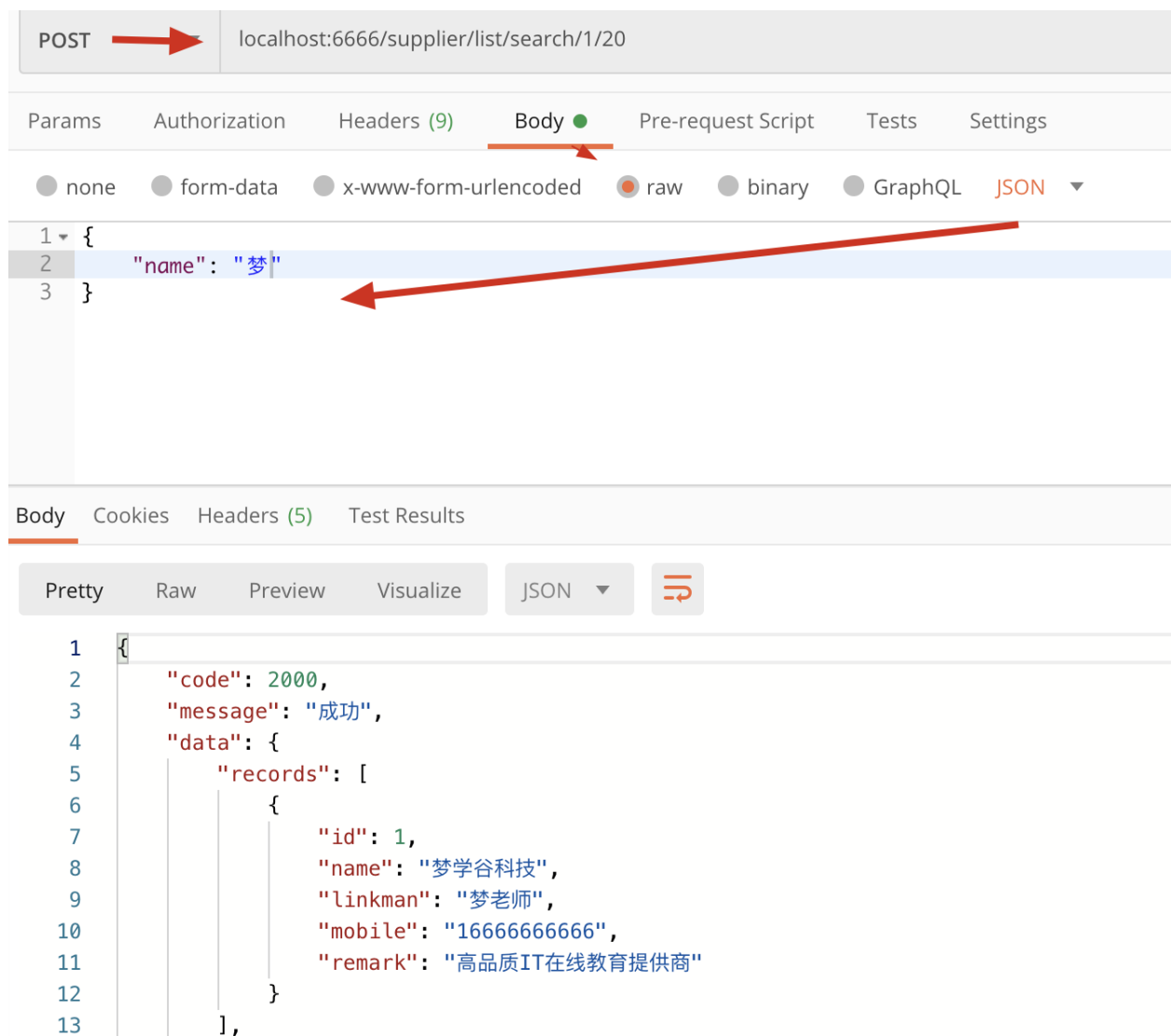
}
```

启动测试

1. 分页条件查询供应商列表，发送 POST 请求 localhost:6666/supplier/list/search/1/20

闪电PDF编辑器

<https://www.callmysoft.com/>



第九章 供应商管理服务端-增删改查

新增供应商

供应商只要对 tb_supplier 单表操作，并且我们可以直接使用 mybatis-plus 提供的 ISupplierService 方法进行操作即可。

添加控制层方法

在 com.mengxuegu.member.controller.SupplierController 类中添加 add 方法：

闪电PDF编辑器

<https://www.callmysoft.com/>

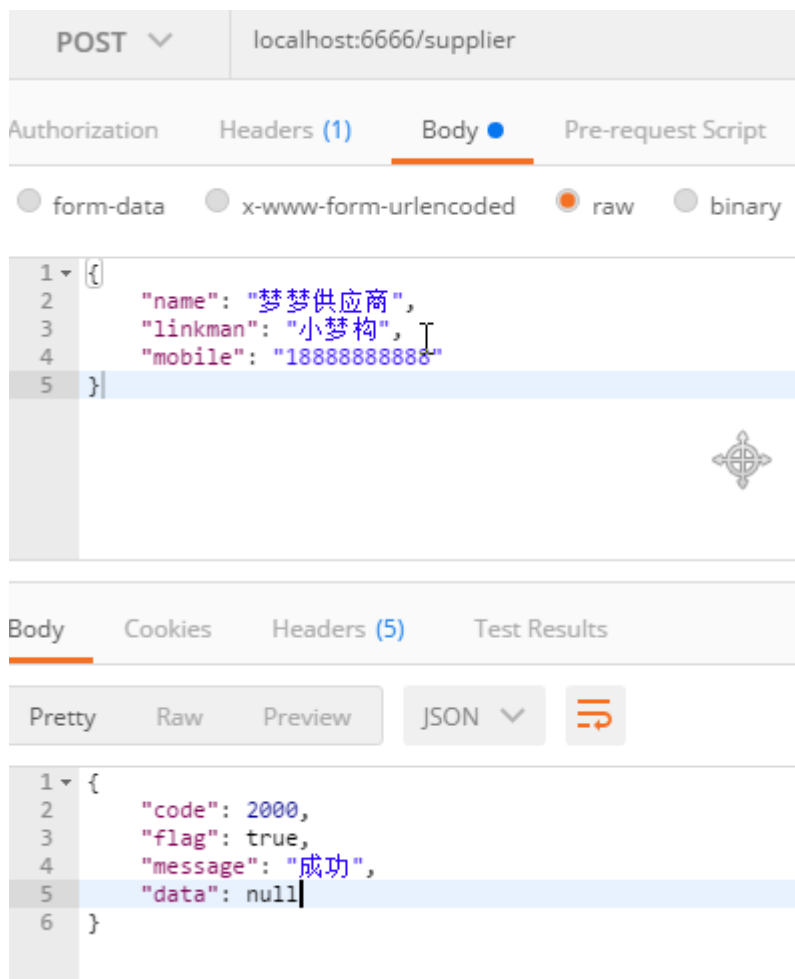
```

/**
 * 新增供应商
 * @param supplier
 * @return
 */
@PostMapping // /supplier
public Result add(@RequestBody Supplier supplier) {
    boolean b = supplierService.save(supplier);
    if(b) {
        return Result.ok();
    }
    return Result.error("新增供应商信息失败");
}

```

测试

1. 新增供应商，发送 POST 请求 localhost:6666/supplier



删除供应商

通过供应商id删除供应商数据，在删除前判断该供应商是否已经被 tb_goods 商品引用了，如果被引用则不允许删除。

闪电PDF编辑器

<https://www.callmysoft.com/>

编写商品业务层

通过供应商id查询商品表 tb_goods 是否存在数据，存在则供应商被引用

1. 在接口 `com.mengxuegu.member.service.IGoodsService` 定义一个 `selectBySupplierId` 方法

```
package com.mengxuegu.member.service;

import com.mengxuegu.member.entity.Goods;
import com.baomidou.mybatisplus.extension.service.IService;

import java.util.List;

/**
 * <p>
 * 商品信息表 服务类
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
public interface IGoodsService extends IService<Goods> {

    /**
     * 通过供应商id查询商品信息
     * @param supplierId
     * @return
     */
    List<Goods> selectBySupplierId(int supplierId);

}
```

2. 在实现类 `com.mengxuegu.member.service.impl.GoodsServiceImpl` 实现 `selectBySupplierId` 方法

```
@Override
public List<Goods> selectBySupplierId(int supplierId) {
    QueryWrapper<Goods> query = new QueryWrapper<>();
    query.eq("supplier_id", supplierId);
    return baseMapper.selectList(query);
}
```

编写供应商业务层

1. 在接口 `com.mengxuegu.member.service.ISupplierService` 定义一个 `deleteById` 方法

闪电PDF编辑器

<https://www.callmysoft.com/>

```
/**
 * 删除供应商
 * @param id 供应商id
 * @return
 */
Result deleteById(int id);
```

2. 在实现类 `com.mengxuegu.member.service.impl.SupplierServiceImpl` 实现 `deleteById` 方法

```
@Autowired
private IGoodsService goodsService;

@Override
public Result deleteById(int id) {
    // 1. 通过供应商id查询是否被商品引用,
    List<Goods> goodsList = goodsService.selectBySupplierId(id);
    // 2. 如果被商品引用, 则不让删除供应商
    if(CollectionUtils.isNotEmpty(goodsList)) {
        return Result.error("该供应商被商品引用, 不允许删除");
    }

    // 3. 如果没有被引用, 直接删除
    int i = baseMapper.deleteById(id);
    if(i < 1) {
        return Result.error("删除供应商失败");
    }

    return Result.ok();
}
```

编写控制层

1. 在控制层类 `com.mengxuegu.member.controller.SupplierController` 添加删除方法

```
/**
 * 删除供应商
 * @return
 */
@DeleteMapping("/{id}")
public Result delete(@PathVariable("id") int id) {
    return supplierService.deleteById(id);
}
```

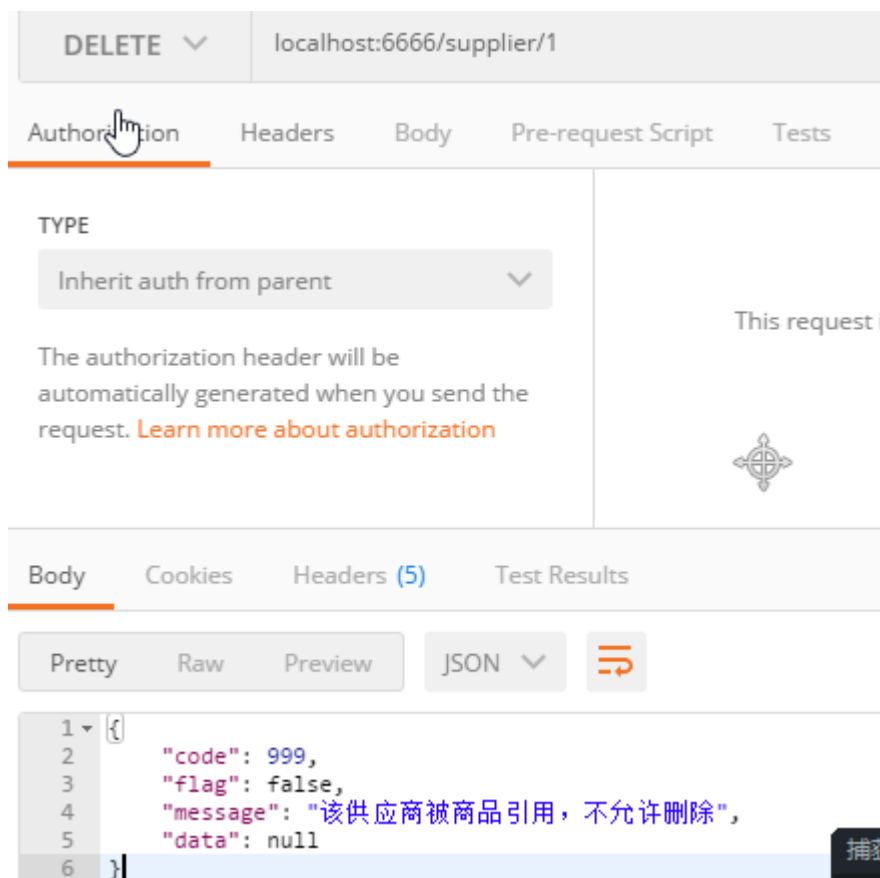
测试

闪电PDF编辑器

<https://www.callmysoft.com/>

1. 删除供应商，发送 DELETE 请求 localhost:6666/supplier/31

如果被引用则删除失败，会响应该供应商被商品引用，不允许删除。



查询和修改供应商

修改供应商信息，要先查询供应商详情回显，然后再将修改后的数据提交更新

添加控制层方法

在 com.mengxuegu.member.controller.SupplierController 类中添加 get 和 update 方法:

```
/**
 * 通过id查询详情
 * @param id
 * @return
 */
@GetMapping("/{id}") // /supplier/{id}
public Result get(@PathVariable("id") int id) {
    Supplier supplier = supplierService.getById(id);
    return Result.ok(supplier);
}
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
/**
 * 修改供应商
 * @param supplier
 * @return
 */
@PutMapping("/{id}") // /supplier/{id}
public Result update(@PathVariable("id") int id,
    @RequestBody Supplier supplier) {
    return supplierService.update(id, supplier);
}
```

添加业务层方法

1. 在 com.mengxuegu.member.service.ISupplierService 添加 update 接口方法

```
Result update(int id, Supplier supplier);
```

2. 在 com.mengxuegu.member.service.impl.SupplierServiceImpl 实现 update 方法

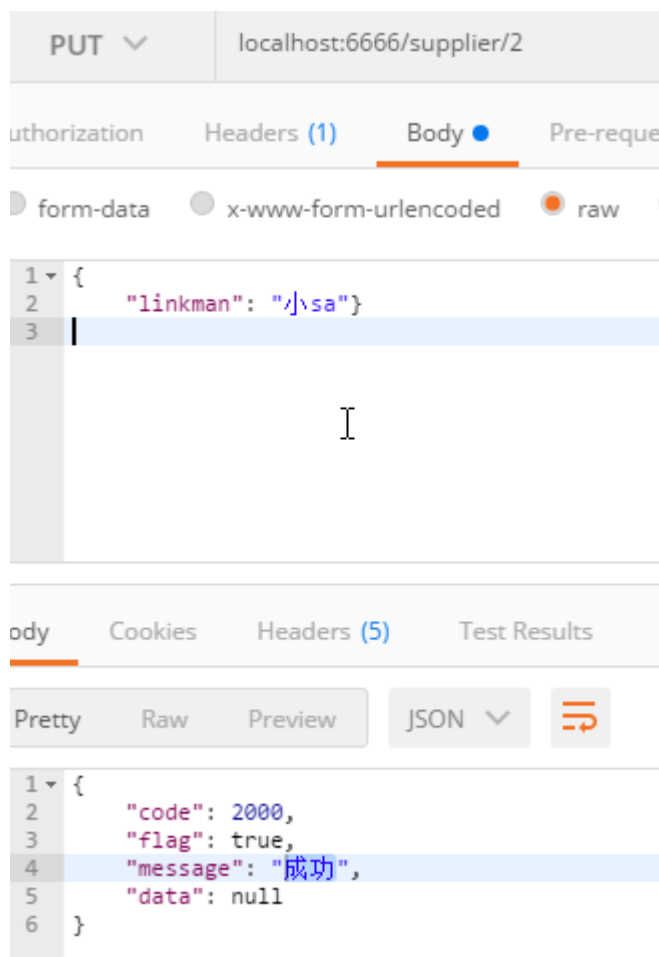
```
@Override
public Result update(int id, Supplier supplier) {
    if(supplier.getId() == null) {
        supplier.setId(id);
    }
    //更新操作
    int size = baseMapper.updateById(supplier);
    if(size < 1) {
        return Result.error("修改供应商信息失败");
    }
    return Result.ok();
}
```

测试

1. 修改供应商，发送 PUT 请求 localhost:6666/supplier/2

闪电PDF编辑器

<https://www.callmysoft.com/>



第十章 商品管理服务端-分页条件查询

商品列表要显示供应商名称，而在 tb_goods 表中只有供应商id，要显示供应商名称就要tb_goods 关联 tb_supplier 表查询。

创建商品请求类 GoodsREQ

1. 创建 `com.mengxuegu.member.req.GoodsREQ` 类

```
package com.mengxuegu.member.req;

import lombok.Data;

import java.io.Serializable;

/**
 * 商品查询条件请求类
 */
@Data
public class GoodsREQ implements Serializable {
```

闪电PDF编辑器

```

    /**
     * 商品名称
     */
    private String name;

    /**
     * 商品编码
     */
    private String code;

    /**
     * 供应商id
     */
    private String supplierId;
}

```

编写数据访问层Mapper

tb_goods 关联 tb_supplier 表条件查询商品分页数据。

1. 在 com.mengxuegu.member.mapper.GoodsMapper 接口中定义 searchPage 方法

```

package com.mengxuegu.member.mapper;

import com.baomidou.mybatisplus.core.metadata.IPage;
import com.mengxuegu.member.entity.Goods;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.mengxuegu.member.req.GoodsREQ;
import org.apache.ibatis.annotations.Param;

/**
 * <p>
 * 商品信息表 Mapper 接口
 * </p>
 * @author 梦学谷-www.mengxuegu.com
 */
public interface GoodsMapper extends BaseMapper<Goods> {

    /**
     * 不需要手动去分页，而mybait-plus会自动实现分页
     * 但是你必须第1个参数传入IPage对象，第2个参数通过 @Param 取别名，
     * 最终查询到的数据会被封装到IPage实现里面
     * @param page
     * @param req
     * @return
     */
    IPage<Goods> searchPage(IPage<Goods> page, @Param("req") GoodsREQ req);
}

```

闪电PDF编辑器

```
}
```

2. 在 com/mengxuegu/member/mapper/xml/GoodsMapper.xml 添加查询的sql实现

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.mengxuegu.member.mapper.GoodsMapper">

    <!--分页条件查询商品列表-->
    <select id="searchPage" resultType="Goods" >
        SELECT
            t1.*,
            t2.NAME AS supplierName
        FROM
            tb_goods t1
        LEFT JOIN tb_supplier t2 ON t1.supplier_id = t2.id
        WHERE 1=1
        <if test="req.name != null and req.name != ''">
            AND t1.name LIKE CONCAT('%', #{req.name}, '%')
        </if>
        <if test="req.code != null and req.code != ''">
            AND t1.code LIKE CONCAT('%', #{req.code}, '%')
        </if>
        <if test="req.supplierId != null">
            AND t1.supplier_id = #{req.supplierId}
        </if>
    </select>

</mapper>
```

3. 检查 application.yml 中的包名是否正确

```
mybatis-plus:
  type-aliases-package: com.mengxuegu.member.entity
  # xxxMapper.xml 路径
  mapper-locations: classpath:com/mengxuegu/member/mapper/xml/**/*.xml
  # 日志级别，会打印sql语句
  logging:
    level:
      com.mengxuegu.member.mapper: debug
```

4. 上面sql查询结果有 supplierName 商品名称，我们在 com.mengxuegu.member.entity.Goods 类添加一个 supplierName 属性

闪电PDF编辑器

<https://www.callmysoft.com/>

```
/**
 * 供应商名称
 */
// 标识它不是tb_goods表中字段，否则会报错
@TableField(exist = false)
private String supplierName;
```

编写业务层

1. 在接口 `com.mengxuegu.member.service.IGoodsService` 定义一个通过分页条件查询方法 `search`

```
Result search(long page, long size, GoodsREQ req);
```

2. 在实现类 `com.mengxuegu.member.service.impl.IGoodsServiceImpl` 实现 `search` 方法。

注意：Page导包要导入自定义的 `import com.mengxuegu.member.base.Page;`

```
@Override
public Result search(long page, long size, GoodsREQ req) {
    if(req == null) {
        req = new GoodsREQ();
    }
    // 在 GoodsMapper 已经实现了 searchPage 分页条件查询
    IPage<Goods> data =
        baseMapper.searchPage(new Page<Goods>(page, size), req);
    return Result.ok(data);
}
```

编写控制层

1. 在控制层类 `com.mengxuegu.member.controller.GoodsController`

```
package com.mengxuegu.member.controller;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.req.GoodsREQ;
import com.mengxuegu.member.service.IGoodsService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

/**
 * <p>
 * 商品信息表 前端控制器
 * </p>
```

闪电PDF编辑器

```
* @author 梦学谷-www.mengxuegu.com
*/
@RestController
@RequestMapping("/goods")
public class GoodsController {

    @Autowired
    private IGoodsService goodsService;

    /**
     * 分页条件查询商品列表
     * @param page
     * @param size
     * @param req
     * @return
     */
    @PostMapping("/list/search/{page}/{size}")
    public Result search(@PathVariable("page") long page,
                        @PathVariable("size") long size,
                        @RequestBody(required=false) GoodsREQ req) {
        return goodsService.search(page, size, req);
    }

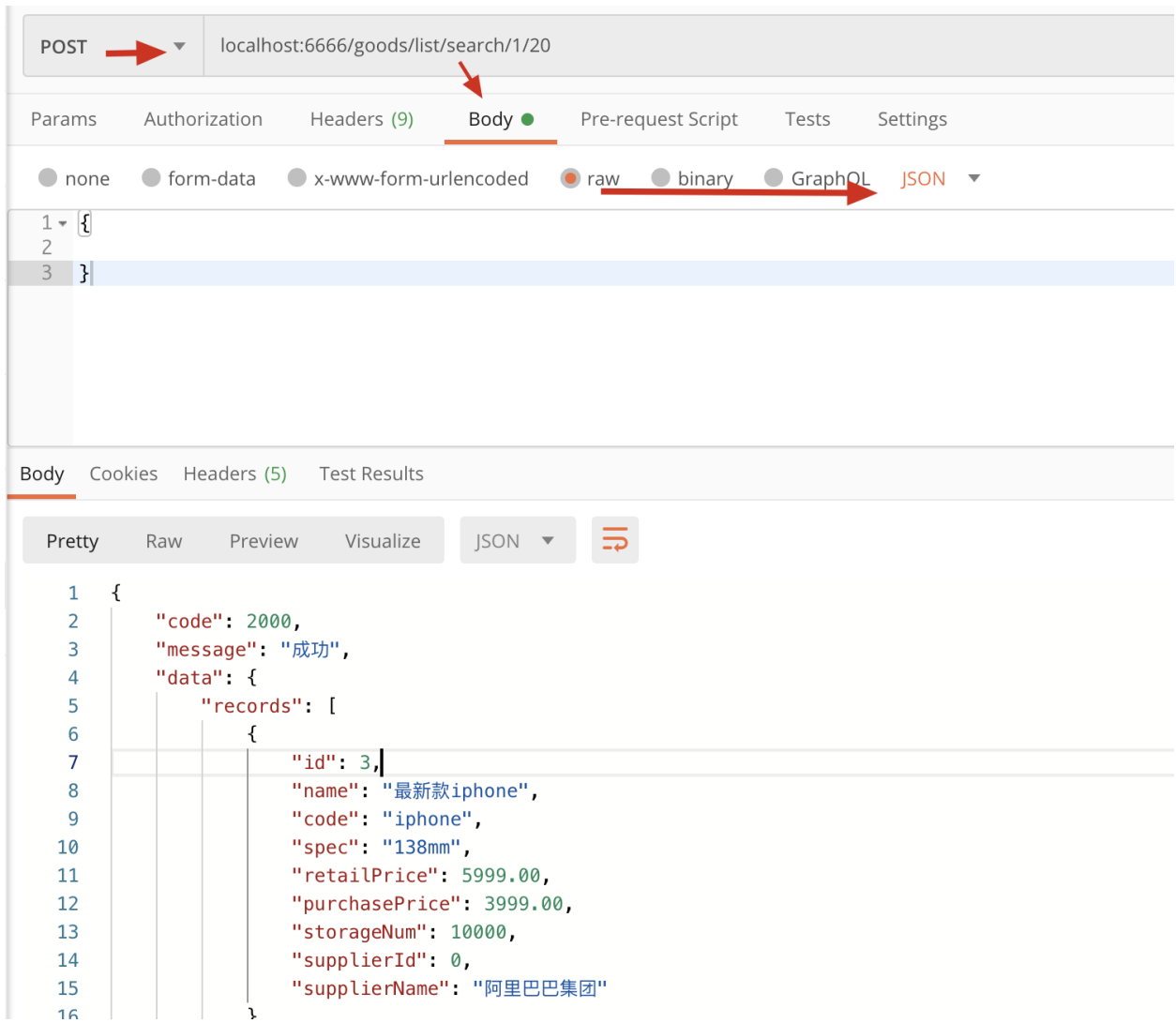
}
```

启动测试

1. 分页条件查询商品列表，发送 POST 请求 localhost:6666/goods/list/search/1/20

闪电PDF编辑器

<https://www.callmysoft.com/>



第十一章 商品管理服务端-增删改查

新增与删除商品

新增与删除商品只要对 tb_goods 单表操作，我们可以直接使用 mybatis-plus 提供的 IMemberService 方法进行操作即可。

添加控制层方法

在 com.mengxuegu.member.controller.GoodsController 类中添加 add 和 delete 方法：

```
/**
 * 新增商品
 * @param goods
 * @return
 */
@PostMapping // /goods
```

闪电PDF编辑器

<https://www.callmysoft.com/>


```

public Result add(@RequestBody Goods goods) {
    boolean b = goodsService.save(goods);
    if(b) {
        return Result.ok();
    }
    return Result.error("新增会员信息失败");
}

/**
 * 删除商品
 * @return
 */
@DeleteMapping("/{id}")
public Result delete(@PathVariable("id") int id) {
    boolean b = goodsService.removeById(id);
    if(b){
        return Result.ok();
    }
    return Result.error("删除商品信息失败");
}

```

测试

1. 新增商品，发送 POST 请求 localhost:6666/goods
2. 删除商品，发送 DELETE 请求 localhost:6666/goods/2

查询和修改商品

修改商品信息，要先查询商品详情回显，并且要同时查询出供应商名称。

然后再将修改后的数据提交更新。

添加业务层方法

1. 在 com.mengxuegu.member.service.ISupplierService 添加 findById 和 update 接口方法

```

Result findById(int id);

Result update(int id, Goods goods);

```

2. 在 com.mengxuegu.member.service.impl.GoodsServiceImpl 实现 findById 和 update 方法

```

@Autowired
private ISupplierService supplierService;

```

闪电PDF编辑器

```

@Override
public Result findById(int id) {
    // 查询商品详情
    Goods goods = baseMapper.selectById(id);

    // 查询供应商名称, 设置到商品对象中
    if(goods != null && goods.getSupplierId() != null) {
        Supplier supplier = supplierService.getById(goods.getSupplierId());
        if(supplier != null) {
            goods.setSupplierName(supplier.getName());
        }
    }

    return Result.ok(goods);
}

@Override
public Result update(int id, Goods goods) {
    if(goods.getId() == null) {
        goods.setId(id);
    }
    //更新操作
    int size = baseMapper.updateById(goods);
    if(size < 1) {
        return Result.error("修改商品信息失败");
    }
    return Result.ok();
}

```

添加控制层方法

在 com.mengxuegu.member.controller.GoodsController 类中添加 get 和 update 方法:

```

/**
 * 通过id查询详情
 * @param id
 * @return
 */
@GetMapping("/{id}") // /goods/{id}
public Result get(@PathVariable("id") int id) {
    return goodsService.findById(id);
}

/**
 * 修改商品
 * @param goods
 * @return

```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
*/
@PutMapping("/{id}") // /goods/{id}
public Result update(@PathVariable("id") int id,
    @RequestBody Goods goods) {
    return goodsService.update(id, goods);
}
```

测试

1. 查询商品详情，发送 GET 请求 localhost:6666/goods/2
2. 修改商品，发送 PUT 请求 localhost:6666/goods/2

第十二章 员工管理服务端-分页条件查询

创建商品请求类 StaffREQ

1. 创建 `com.mengxuegu.member.req.StaffREQ` 类

```
package com.mengxuegu.member.req;

import lombok.Data;
import java.io.Serializable;
import java.util.Date;

/**
 * 员工查询条件请求类
 */
@Data
public class StaffREQ implements Serializable {

    /**
     * 姓名
     */
    private String name;

    /**
     * 帐号
     */
    private String username;
}
```

闪电PDF编辑器

<https://www.callmysoft.com/>

编写业务层

1. 在接口 `com.mengxuegu.member.service.IStaffService` 定义一个通过分页条件查询方法 `search`

```
Result search(long page, long size, StaffREQ req);
```

2. 在实现类 `com.mengxuegu.member.service.impl.StaffServiceImpl` 实现 `search` 方法。

注意：Page导包要导入自定义的 `import com.mengxuegu.member.base.Page;`

```
@Override
public Result search(long page, long size, StaffREQ req) {
    // 封装查询条件
    QueryWrapper<Staff> query = new QueryWrapper<>();
    if(req != null) {
        if(StringUtils.isNotBlank(req.getName())) {
            query.like("name", req.getName());
        }
        if(StringUtils.isNotBlank(req.getUsername())) {
            query.like("username", req.getUsername());
        }
    }
    IPage<Staff> data =
        baseMapper.selectPage(new Page<Staff>(page, size), query);
    return Result.ok(data);
}
```

编写控制层

1. 在控制层类 `com.mengxuegu.member.controller.StaffController#search`

```
package com.mengxuegu.member.controller;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.req.StaffREQ;
import com.mengxuegu.member.service.IStaffService;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

/**
 * <p>
 * 员工信息表 前端控制器
 * </p>
 *
 * @author 梦学谷-www.mengxuegu.com
 */
```

闪电PDF编辑器

```
*/
@RestController
@RequestMapping("/staff")
public class StaffController {

    Logger logger = LoggerFactory.getLogger(getClass());

    @Autowired // 不要少了注解
    private IStaffService staffService;

    /**
     * 分页条件查询员工列表
     * @param page 页码
     * @param size 每页显示记录数
     * @param req 查询条件
     * @return
     */
    @PostMapping("/list/search/{page}/{size}")
    public Result search(@PathVariable("page") long page,
                        @PathVariable("size") long size,
                        @RequestBody(required=false) StaffREQ req) {
        logger.info("分页查询员工列表: page={}, size={}", page, size);

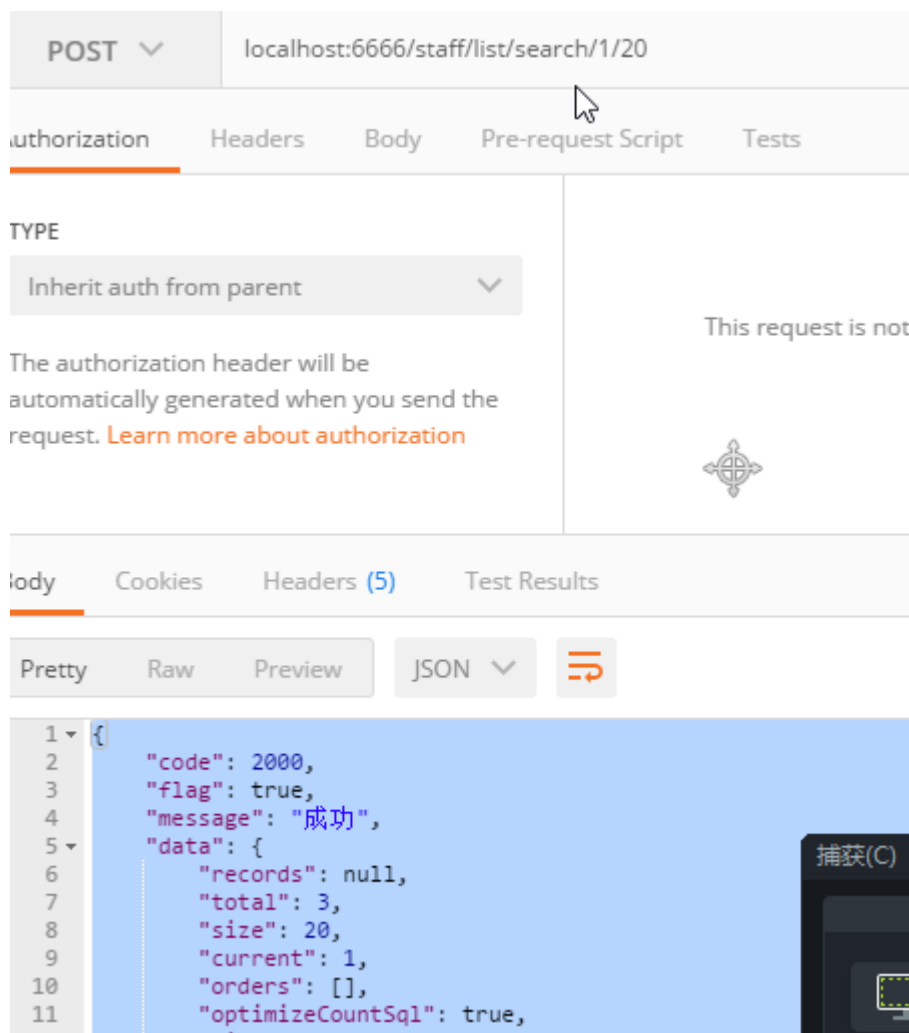
        return staffService.search(page, size, req);
    }
}
```

启动测试

1. 分页条件查询员工列表，发送 POST 请求 localhost:6666/staff/list/search/1/20

闪电PDF编辑器

<https://www.callmysoft.com/>



第十三章 员工管理服务端-增删改查

新增员工

需求

1. 查询员工用户名是否存在，存在不允许新增
2. 对密码进行加密保存
3. 提交数据到 tb_staff 表

编写业务层

1. 在接口 `com.mengxuegu.member.service.IStaffService` 定义一个 add 方法

```
Result add(Staff staff);
```

2. 在实现类 `com.mengxuegu.member.service.impl.StaffServiceImpl` 实现 add 方法

闪电PDF编辑器

```

public Result add(Staff staff) {
    if(staff == null || StringUtils.isEmpty(staff.getUsername())) {
        return Result.error("用户名不能为空");
    }

    // 1. 查询用户名是否存在
    Staff s = getByUsername(staff.getUsername());
    if(s != null) {
        return Result.error("用户名已存在");
    }

    // 2. 使用SpringSecurity提供的加密器加密
    String password = new BCryptPasswordEncoder().encode(staff.getPassword());
    staff.setPassword(password);

    // 3. 保存到数据库
    boolean b = this.save(staff);
    if(b) {
        return Result.ok();
    }
    return Result.error("新增失败");
}

public Staff getByUsername(String username) {
    QueryWrapper<Staff> query = new QueryWrapper<>();
    query.eq("username", username);
    return baseMapper.selectOne(query);
}

```

编写控制层

在 com.mengxuegu.member.controller.StaffController 类中添加 add 方法：

```

/**
 * 新增员工
 * @param staff
 * @return
 */
@PostMapping // /staff
public Result add(@RequestBody Staff staff) {
    return staffService.add(staff);
}

```

测试

1. 新增员工，发送 POST 请求 localhost:6666/staff

闪电PDF编辑器

<https://www.callmysoft.com/>

删除员工

删除员工只要对 tb_staff 单表操作，我们可以直接使用 mybatis-plus 提供的 IStaffService 方法进行操作即可。

编写控制层

1. 在控制层类 com.mengxuegu.member.controller.StaffController 添加 delete 删除方法

```
/**
 * 删除员工
 * @return
 */
@DeleteMapping("/{id}")
public Result delete(@PathVariable("id") int id) {
    boolean b = staffService.removeById(id);
    if(b){
        return Result.ok();
    }
    return Result.error("删除员工信息失败");
}
```

测试

1. 删除员工，发送 DELETE 请求 localhost:6666/staff/2

查询和修改员工

修改供应商信息，要先查询供应商详情回显，然后再将修改后的数据提交更新

添加控制层方法

在 com.mengxuegu.member.controller.SupplierController 类中添加 get 和 update 方法：

```
/**
 * 通过id查询详情
 * @param id
 * @return
 */
@GetMapping("/{id}") // /staff/{id}
public Result get(@PathVariable("id") int id) {
    Staff staff = staffService.getById(id);
    return Result.ok(staff);
}

/**
 * 修改员工
 * @param staff
```

闪电PDF编辑器

<https://www.callmysoft.com/>


```

    * @return
    */
    @PutMapping("/{id}") // /staff/{id}
    public Result update(@PathVariable("id") int id,
        @RequestBody Staff staff) {
        return staffService.update(id, staff);
    }

```

添加业务层方法

1. 在 com.mengxuegu.member.service.IStaffService 添加 update 接口方法

```
Result update(int id, Staff staff);
```

2. 在 com.mengxuegu.member.service.impl.StaffServiceImpl 实现 update 方法

```

@Override
public Result update(int id, Staff staff) {
    if(staff.getId() == null) {
        staff.setId(id);
    }
    //更新操作
    int size = baseMapper.updateById(staff);
    if(size < 1) {
        return Result.error("修改员工信息失败");
    }
    return Result.ok();
}

```

测试

1. 修改员工，发送 PUT 请求 localhost:6666/staff/2

修改密码

需求：

1. 校验原密码是否正确
2. 提交新密码

创建密码请求类 PasswordREQ

```
package com.mengxuegu.member.req;
```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
import lombok.Data;

/**
 * 修改密码请求类
 */
@Data
public class PasswordREQ {
    /**
     * 用户id
     */
    private Integer userId;
    /**
     * 原密码 or 新密码
     */
    private String password;
}
```

编写业务层

1. 在接口 `com.mengxuegu.member.service.IStaffService` 定义 `checkPassword` 和 `updatePassword` 方法

```
/**
 * 校验原密码是否正确
 * @param req
 * @return
 */
Result checkPassword(PasswordREQ req);

/**
 * 更新修改后的新密码
 * @param req
 * @return
 */
Result updatePassword(PasswordREQ req);
```

2. 在实现类 `com.mengxuegu.member.service.impl.StaffServiceImpl` 实现 `checkPassword` 和 `updatePassword` 方法

```
@Override
public Result checkPassword(PasswordREQ req) {
    if(req == null || StringUtils.isEmpty(req.getPassword())) {
        return Result.error("原密码不能为空");
    }

    Staff staff = baseMapper.selectById(req.getUserId());
    if(!new BCryptPasswordEncoder().matches(req.getPassword(), staff.getPassword())) {
        return Result.error("原密码错误");
    }
    return Result.ok();
}
```

闪电PDF编辑器

```

@Override
public Result updatePassword>PasswordREQ req) {
    if(req == null || StringUtils.isEmpty(req.getPassword())) {
        return Result.error("新密码不能为空");
    }

    // 更新密码
    Staff staff = baseMapper.selectById(req.getUserId());
    staff.setPassword(new BCryptPasswordEncoder().encode(req.getPassword()));
    baseMapper.updateById(staff);
    return Result.ok();
}

```

创建控制层

因为 修改密码URL是 /user 开头的，而StaffController 是 /staff 开头，我们就创建一个的控制类 AuthController。

创建 com.mengxuegu.member.controller.AuthController 类中：

```

package com.mengxuegu.member.controller;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.req.PasswordREQ;
import com.mengxuegu.member.service.IStaffService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/user")
public class AuthController {

    @Autowired
    private IStaffService staffService;

    /**
     * 校验原密码是否正确
     * @return
     */
    @PostMapping("/pwd")
    public Result checkPwd(@RequestBody PasswordREQ req) {
        return staffService.checkPassword(req);
    }

    /**
     * 修改密码
     * @return
     */
    @PutMapping("/pwd")

```

闪电PDF编辑器

<https://www.callmysoft.com/>

```
public Result updatePwd(@RequestBody PasswordREQ req) {  
    return staffService.updatePassword(req);  
}  
  
}
```

测试

1. 校验原密码，发送 POST 请求 localhost:6666/user/pwd

数据库中现在的密码都是 123456

2. 修改新密码，发送 PUT 请求 localhost:6666/user/pwd

第十四章 重构会员管理客户端代码

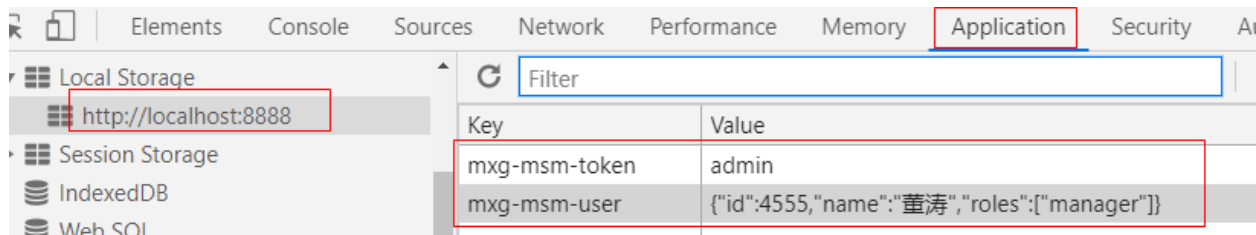
接口代理配置

1. 修改 mxg-msm-vuex/.env.development 文件里的 VUE_APP_SERVICE_URL = 'http://localhost:6666'



2. 测试是否调用实际接口。

如果在登录页面，而当前登录没有后台接口，则会报404，下面在 localStorage 中添加下 token 和 user，来模拟用户已登录。



修复新增数据问题

问题

每个模块的新增和修改模块代码是公用的，目前在新增会员、供应商、商品、员工信息时，如果先点击了修改，然后再点击新增时，提交新增数据会作为更新，发送PUT请求，而不是发送POST请求。

闪电PDF编辑器

<https://www.callmysoft.com/>

原因与解决

原因：

在提交数据时，是提交新增还是修改，我们是通过 pojo.id 是否有值还判断的，

```
<!-- <el-button type="primary" @click="addData('pojoForm')">确定</el-button> -->
<el-button type="primary" @click="pojo.id === null ? addData('pojoForm') : updateData('pojoForm')">确定</el-button>
```

如果先点击修改，pojo.id 就会有值，会先查询数据回显。当你再点击新增时，原修改时 pojo.id 没有设置为null，这样就会被认为还是提交修改数据。

解决：

在打开新增窗口时，把 this.pojo.id 设置为 null 即可。

找到 src/views 下在每个模块里的 handleAdd() 方法，在方法中加上 `this.pojo.id = null` 即可解决。

```
// 弹出新增窗口
handleAdd() {
  // this.pojo = {}
  this.dialogFormVisible = true
  this.$nextTick(() => {
    // this.$nextTick()它是一个异步事件，当渲染结束
    // 弹出窗口打开之后，需要加载Dom，就需要花费一点时
    this.$refs['pojoForm'].resetFields()
    this.pojo.id = null
  })
},
```

第十五章 基于 JWT 令牌实现身份认证

JSON Web Token(JWT)是一个非常轻巧的规范。这个规范允许我们使用JWT在用户和服务器之间传递安全可靠的信息。其中 JWT 中可以包含用户信息。

闪电PDF编辑器

<https://www.callmysoft.com/>

使用 JJWT 实现 JWT

什么是JJWT

JJWT 是一个提供端到端的JWT创建和验证的Java库。永远免费和开源(Apache License, 版本2.0), JJWT很容易使用和理解。

JJWT快速入门

1. 在 mengxuegu-member-util/pom.xml 添加jjwt依赖 (jdk8以下)

```
<dependencies>
  <!--jwt令牌-->
  <dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.6.0</version>
  </dependency>
</dependencies>
```

如果使用JDK9还要添加以下依赖

默认情况下, 在java SE 9.0 中 将不再包含java EE 的Jar包, 而 JAXB API是java EE 的API, 因此我们要手动导入这个 Jar 包。

```
<dependencies>
  <!--jwt令牌-->
  <dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.6.0</version>
  </dependency>
  <dependency>
    <groupId>javax.xml.bind</groupId>
    <artifactId>jaxb-api</artifactId>
    <version>2.3.0</version>
  </dependency>
  <dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-impl</artifactId>
    <version>2.3.0</version>
  </dependency>
  <dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-core</artifactId>
    <version>2.3.0</version>
  </dependency>
  <dependency>
    <groupId>javax.activation</groupId>
    <artifactId>activation</artifactId>
    <version>1.1.1</version>
```

闪电PDF编辑器

```
</dependency>
</dependencies>
```

2. 创建普通测试类 TestJwt , 用于生成 token

```
import io.jsonwebtoken.JwtBuilder;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class TestJwt {
    public static void main(String[] args) {
        String token = createJwt();
        System.out.println("生成的令牌: " + token);
    }

    // 生成令牌
    public static String createJwt () {
        JwtBuilder builder= Jwts.builder().setId("11111") // 是字符串
            .setSubject("admin") // 主题 如用户名
            .setIssuedAt(new Date()) // 签发时间
            .signWith(SignatureAlgorithm.HS256,"mengxuegu"); // 签名密钥
        return builder.compact();
    }
}
```

执行结果:

```
eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiIxMTExMSIsInN1YiI6ImFkbWluIiwiaWF0IjoxNTkwNDYyNDUyYQ.EbF6wKi
KRzFrJVShGurQnfhK36mDp2_ykiYTj5J0u0c
```

再次运行, 会发现每次运行的结果是不一样的, 因为我们每次的签发时间是不一样的。

解析 jwt 令牌

当服务端生成 token 后发给客户端, 客户端在下次向服务端发送请求时需要携带这个token (这就好像是拿着一张门票一样), 那服务端接到这个token 应该解析出token中的信息(例如用户名), 根据这些信息 查询数据库返回相应的结果。

在 TestCreateJwt 类添加一个方法 parserJwt 来解析jwt令牌中信息

闪电PDF编辑器

<https://www.callmysoft.com/>

```

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.JwtBuilder;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;

import java.util.Date;

public class TestCreateJwt {
    public static void main(String[] args) {
        String token = createJwt();
        System.out.println("生成的令牌: " + token);
        parserJwt(token);
    }

    // 生成令牌
    public static String createJwt () {
        JwtBuilder builder= Jwts.builder().setId("11111") // 是字符串
            .setSubject("admin") // 主题 如用户名
            .setIssuedAt(new Date()) // 签发时间
            .signWith(SignatureAlgorithm.HS256,"mengxuegu"); // 签名密钥
        return builder.compact();
    }

    // 解析令牌
    public static void parserJwt(String token) {
        Claims claims = Jwts.parser()
            .setSigningKey("mengxuegu") // 签名密钥要一致
            .parseClaimsJws(token).getBody();
        System.out.println("id:" + claims.getId());
        System.out.println("subject:" + claims.getSubject());
        // System.out.println("IssuedAt:"+claims.getIssuedAt());
    }
}

```

如果将 token 或签名密钥篡改一下，会发现运行时就会报错，所以解析 token 也就是验证 token

```

8 public class TestCreateJwt {
9     public static void main(String[] args) {
10         String token = createJwt();
11         System.out.println("生成的令牌: " + token);
12         parserJwt(token+xxxx);
13     }

```

TestCreateJwt > createJwt()

```

shines/jdk1.8.0_231.jdk/Contents/Home/bin/java ...
.J9.eyJqdGkiOiIxMTExMSIsInN1YiI6ImFkbWluIiwiaWF0IjoxNTkwNDYzMjEyfQ.Cv0BlIF
io.jsonwebtoken.SignatureException: JWT signature does not match locally c
.DefaultJwtParser.parse(DefaultJwtParser.java:342)
.DefaultJwtParser.parse(DefaultJwtParser.java:458)
.DefaultJwtParser.parseClaimsJws(DefaultJwtParser.java:518)

```

闪电PDF编辑器

<https://www.callmysoft.com/>

校验 Jwt令牌是否过期

一般我们并不希望签发的token是永久生效的，所以我们可以为token添加一个过期时间。

```
// 生成令牌
public static String createJwt () {
    // 当前时间毫秒数
    long now = System.currentTimeMillis();
    // 过期时间为10秒
    long exp = now + 1000*10;

    JwtBuilder builder= Jwts.builder().setId("11111") // 是字符串
        .setSubject("admin") // 主题 如用户名
        .setIssuedAt(new Date()) // 签发时间
        .signWith(SignatureAlgorithm.HS256,"mengxuegu") // 签名密钥
        .setExpiration(new Date(exp)); // 过期时间 ++++++
    return builder.compact();
}
```

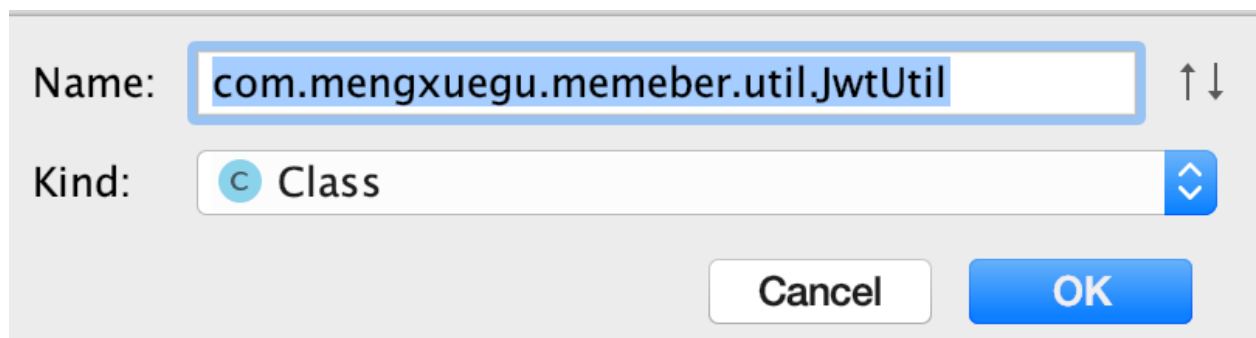
测试：先生成一个token，然后等待10秒后再去解析它，发现报如下错

Exception in thread "main" io.jsonwebtoken.ExpiredJwtException: JWT expired at

```
...hines/jdk1.8.0_231.jdk/Contents/Home/bin/java ...
.J9.eyJqdGkiOiIxMTExMSIsInN1YiI6ImFkbWluIiwiaWF0Ijox
.o.jsonwebtoken.ExpiredJwtException: JWT expired at
.DefaultJwtParser.parse(DefaultJwtParser.java:365)
```

创建JWT工具类

1. 在 mengxuegu-member-util 模块中创建工具类 **com.mengxuegu.member.util.JwtUtil**



2. JwtUtil 代码实现如下：

！ 位于：会员管理系统/03-配套资料/工具类/JwtUtil.java

闪电PDF编辑器

<https://www.callmysoft.com/>

```

package com.mengxuegu.member.util;

import io.jsonwebtoken.Claims;
import io.jsonwebtoken.JwtBuilder;
import io.jsonwebtoken.Jwts;
import io.jsonwebtoken.SignatureAlgorithm;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
@ConfigurationProperties(prefix = "mengxuegu.jwt.config")
public class JwtUtil {

    // 密钥
    private String secretKey;

    //单位秒, 默认7天
    private long expires = 60*60*24*7;

    public String getSecretKey() {
        return secretKey;
    }

    public void setSecretKey(String secretKey) {
        this.secretKey = secretKey;
    }

    public long getExpires() {
        return expires;
    }

    public void setExpires(long expires) {
        this.expires = expires;
    }

    /**
     * 生成JWT
     * @param id
     */
    public String createJWT(String id, String subject, Boolean isLogin) {
        long nowMillis = System.currentTimeMillis();
        Date now = new Date(nowMillis);
        JwtBuilder builder = Jwts.builder().setId(id)
            .setSubject(subject)
            .setIssuedAt(now)
            .signWith(SignatureAlgorithm.HS256, secretKey)
            .claim("isLogin", isLogin);
        if (expires > 0) {
            // expires乘以1000是毫秒转秒
            builder.setExpiration(new Date(nowMillis + expires*1000));
        }
    }
}

```

```

    }
    return builder.compact();
}

/**
 * 解析JWT
 * @param jwtToken
 */
public Claims parseJWT(String jwtToken){
    return Jwts.parser().setSigningKey(secretKey)
        .parseClaimsJws(jwtToken).getBody();
}
}

```

3. 在 mengxuegu-member-api 模块的 application.yml 中添加配置

```

mengxuegu:
  jwt:
    config:
      secretKey: mengxuegu # jwt令牌密钥
      expires: 604800 # 单位秒, 7天

```

业务层添加 login 方法

校验登录用户名密码，登录成功后，生成 token 响应给客户端

1. 在 com.mengxuegu.member.service.impl.IStaffService 接口添加 login 抽象方法

```

Result login(String username, String password);

```

2. 在 com.mengxuegu.member.service.impl.StaffServiceImpl 实现 login 方法

```

@Autowired
JwtUtil jwtUtil;

@Override
public Result login(String username, String password) {
    Result error = Result.error("用户名或密码错误");

    if(StringUtils.isBlank(username)
        || StringUtils.isBlank(password)) {
        return error;
    }
}

```

闪电PDF编辑器

<https://www.callmysoft.com/>

```

// 1. 通过用户名查询
Staff staff = getByUsername(username);
if(staff == null) {
    return error;
}

// 2. 存在, 判断密码是否正确(输入的密码, 数据库加密的密码)
if( !new BCryptPasswordEncoder().matches(password, staff.getPassword())) {
    return error;
}

// 3. 生成token 响应
String jwt = jwtUtil.createJWT(staff.getId() + "", staff.getUsername(), true);
// 手动封装个json对象 {token: jwt}
Map<String, String> map = new HashMap<>();
map.put("token", jwt);
return Result.ok(map);
}

```

控制层添加 login 方法

在 com.mengxuegu.member.controller.AuthController 控制类中添加 login 方法

```

/**
 * 登录
 * @return
 */
@PostMapping("/login")
public Result login(@RequestBody Staff staff) {
    return staffService.login(staff.getUsername(), staff.getPassword());
}

```

测试登录

登录请求, 发送 POST 请求 localhost:6666/user/login

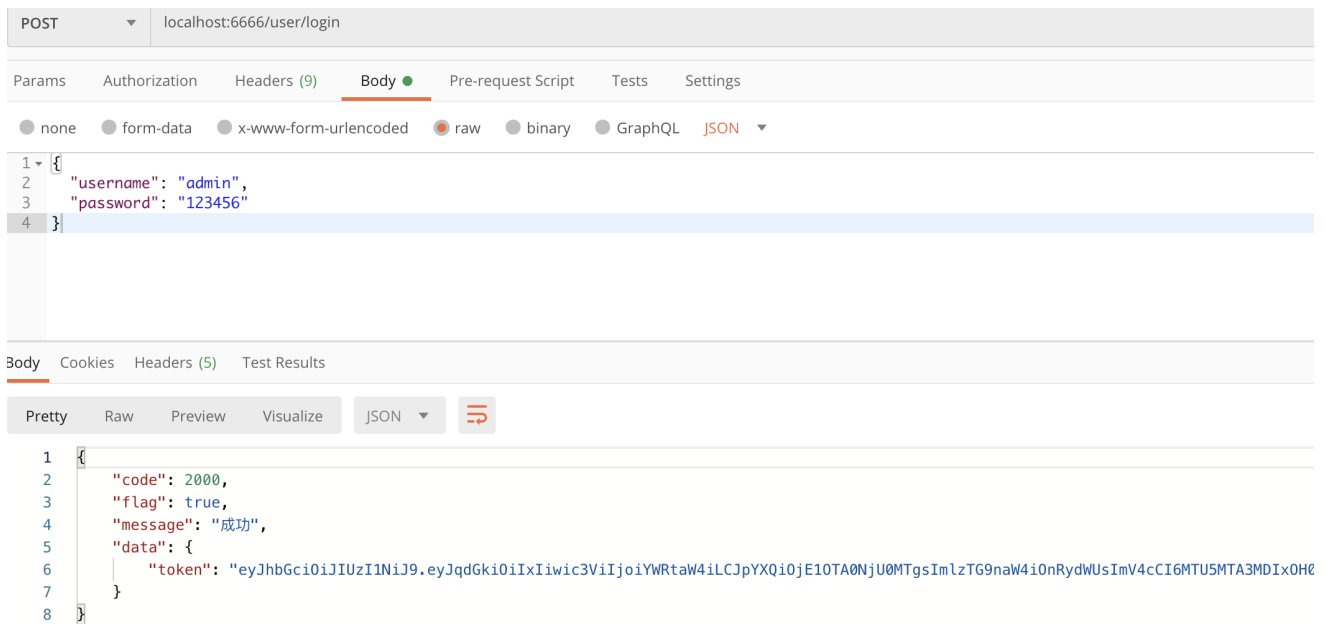
```

{
  "username": "admin",
  "password": "123456"
}

```

闪电PDF编辑器

<https://www.callmysoft.com/>



修改员工，发送 PUT 请求 localhost:6666/staff/2

通过token获取用户信息

业务层添加 getUserInfo 方法

登录成功后，通过 token 来获取用户信息

1. 在 com.mengxuegu.member.service.impl.IStaffService 接口添加 getUserInfo 抽象方法

```
/**
 * 通过token获取用户信息
 * @param token
 * @return
 */
Result getUserInfo(String token);
```

2. 在 com.mengxuegu.member.service.impl.StaffServiceImpl 实现 getUserInfo 方法

```
@Override
public Result getUserInfo(String token) {
    // 解析jwt
    Claims claims = jwtUtil.parseJWT(token);
    if(claims == null || StringUtils.isBlank(claims.getSubject())) {
        return Result.error("获取用户信息失败");
    }

    // 获取用户名
    String username = claims.getSubject();
    // 1. 通过用户名查询
```

```

        Staff staff = getByUsername(username);

        if(staff == null) {
            return Result.error("用户不存在");
        }

        // 2. 将密码设置为null, 不响应给前端
        staff.setPassword(null);

        return Result.ok(staff);
    }

```

控制层添加 getUserInfo 方法

在 com.mengxuegu.member.controller.AuthController 控制类中添加 getUserInfo 方法

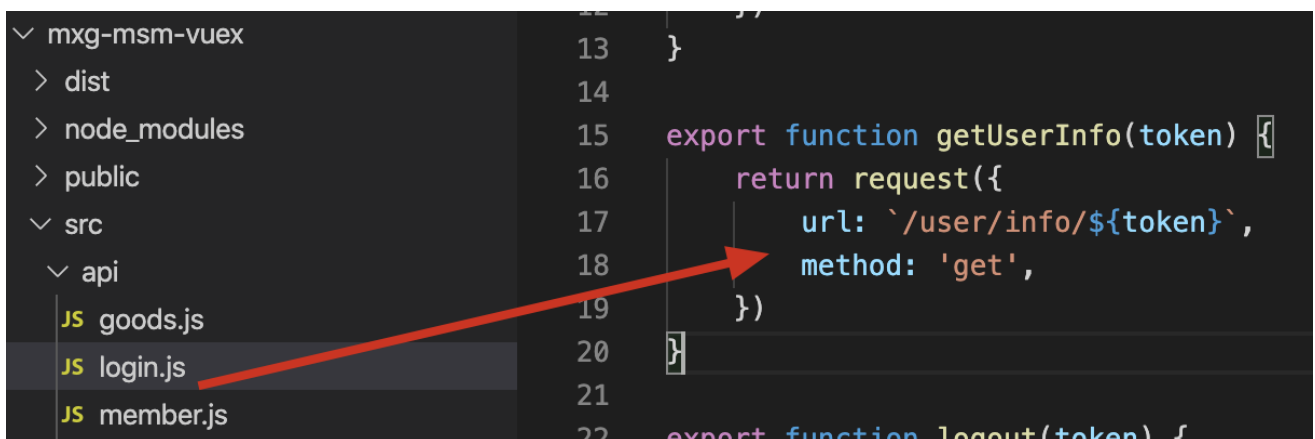
```

/**
 * 通过token获取用户信息
 * @param token
 * @return
 */
@GetMapping("/info/{token}")
public Result getUserInfo(@PathVariable("token") String token) {
    return staffService.getUserInfo(token);
}

```

路径变量传递 token 不成功

前端获取用户信息是在路径变量传递的 token, 即: /user/info/{token}



1. 当前存在问题:

前端发送以下请求, 有两个分隔点, 但是在后端获取路径变量的 token 时, 后面那个点以后都会丢失, 无法获取到:

闪电PDF编辑器

<https://www.callmysoft.com/>

/user/info/eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiIiXliwIiwic3ViOiYWRtaW4iLCJpYXQiOiE1OTA0MDA1MjAsImZlZG9naW4iOiOnRydWUsImV4cCI6MTU5MTAwNTMyMH0.B5FgFpdcOeulcdGqUAEWVdyMj8AR8_nkiOxPhfCN-PQ

后台获取时只获取到以下（可以Debug查看）

eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiIiXliwIiwic3ViOiYWRtaW4iLCJpYXQiOiE1OTA0MDA1MjAsImZlZG9naW4iOiOnRydWUsImV4cCI6MTU5MTAwNTMyMH0

2. 从而后台解析jwt时会报错：JWT strings must contain exactly 2 period characters. Found: 1
必须有两个分隔点，但是只发现1个

```
io.jsonwebtoken.MalformedJwtException: JWT strings must contain exactly 2 period characters. Found: 1
at io.jsonwebtoken.impl.DefaultJwtParser.parse(DefaultJwtParser.java:223) ~[jjwt-0.6.0.jar:0.6.0]
at io.jsonwebtoken.impl.DefaultJwtParser.parse(DefaultJwtParser.java:458) ~[jjwt-0.6.0.jar:0.6.0]
at io.jsonwebtoken.impl.DefaultJwtParser.parseClaimsJws(DefaultJwtParser.java:518) ~[jjwt-0.6.0.jar:0.6.0]
at com.mengxuegu.member.util.JwtUtil.parseJWT(JwtUtil.java:63) ~[classes/:na]
at com.mengxuegu.member.service.impl.StaffServiceImpl.getUserInfo(StaffServiceImpl.java:133) ~[classes/:na]
```

3. 解决问题：

修改前端调用接口方式，token 以请求参数方式传递就不会被丢失，即 /user/info?token=xxxxxx .

4. 修改 src/api/login.js

```
export function getUserInfo(token) {
  return request({
    url: '/user/info',
    method: 'get',
    params: { token }
  })
}
```

5. 修改获取用户信息代码 com.mengxuegu.member.controller.AuthController#getUserInfo

```
@GetMapping("/info")
public Result getUserInfo(@RequestParam("token") String token) {
    return staffService.getUserInfo(token);
}
```

退出系统

jwt令牌无法手动让它失效，在前端点击退出时直接删除localStorage中的数据，如果需要可以使用将jwt通过redis存储，每次请求时从redis查询是否存在，如果不存在，则认为未登录已退出。

```
/**
 * 退出
 * @return
 */
@PostMapping("/logout")
public Result logout() {
    return Result.ok();
}
```

闪电PDF编辑器

<https://www.callmysoft.com/>

拦截器方式实现 token 鉴权

我们自定一个拦截器，只有当用户登录后，才可以访问资源接口（会员、商品、供应商、员工），没有登录则要求登录。

其中判断是否登录，要求客户端请求接口时，在请求头上带上 token，然后在拦截器拦截到请求后，校验 token 是否有效，有效才让访问，否则无法访问。

请求头信息 Authorization: Bearer jwtToken

创建拦截器

1. 创建一个 HandlerInterceptorAdapter 拦截适配器的子类，并且重写它里面的 preHandle 方法，在请求目标接口时进行拦截。
2. 创建自定义拦截器类 com.mengxuegu.member.filter.AuthenticationFilter
类上不要少了 @Component

```
package com.mengxuegu.member.filter;

import com.mengxuegu.member.base.Result;
import com.mengxuegu.member.util.JwtUtil;
import io.jsonwebtoken.Claims;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;
import org.springframework.web.servlet.handler.HandlerInterceptorAdapter;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@Component // 不要少了
public class AuthenticationFilter extends HandlerInterceptorAdapter {

    @Autowired
    private JwtUtil jwtUtil;

    @Override
    public boolean preHandle(HttpServletRequest request,
                             HttpServletResponse response,
                             Object handler) throws Exception {

        // 是否登录
        boolean isLogin = false;

        // 获取请求头 Authorization: Bearer jwtToken
        final String authHeader = request.getHeader("Authorization");
        // 判断是否有token, 注意 Bearer 后面有空格
```



```

        if (authHeader != null && authHeader.startsWith("Bearer ")) {
            // 截取获取jwtToken
            final String token = authHeader.substring(7);
            // 解析
            Claims claims = jwtUtil.parseJWT(token);
            if (claims != null) {
                if ( (Boolean) claims.get("isLogin") ) {
                    // 已登录
                    isLogin = true;
                }
            }
        }

        if(!isLogin) {
            response.setContentType("application/json;charset=UTF-8");
            response.setStatus(401);
            response.getWriter().write("未通过身份认证");
        }

        return isLogin;
    }
}

```

配置拦截器类

1. 创建一个配置类 com.mengxuegu.member.config.WebMvcConfig

其中要放行登录请求 /user/login

```

package com.mengxuegu.member.config;

import com.mengxuegu.member.filter.AuthenticationFilter;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurationSupport;

/**
 * 自定义Web配置类
 */
@Configuration
public class WebMvcConfig extends WebMvcConfigurationSupport {

    @Autowired
    private AuthenticationFilter authenticationFilter;

    @Override
    protected void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(authenticationFilter);
        // 拦截所有请求
    }
}

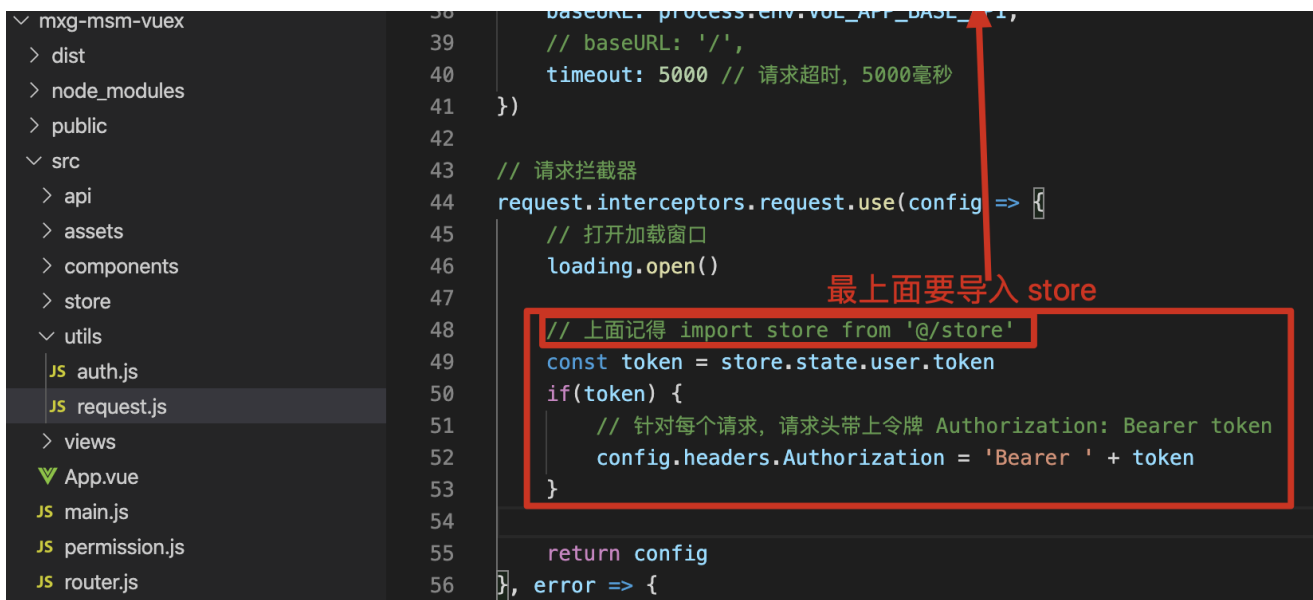
```

```

    addPathPatterns("/**").
    // 登录请求排除，不被拦截
    excludePathPatterns("/user/login");
  }
}

```

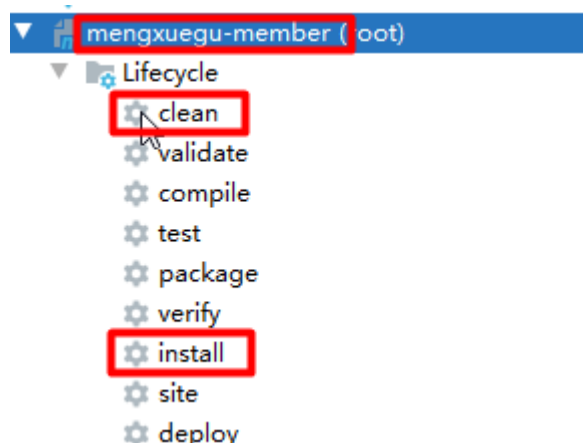
客户端请求头带上 token



部署

服务端部署

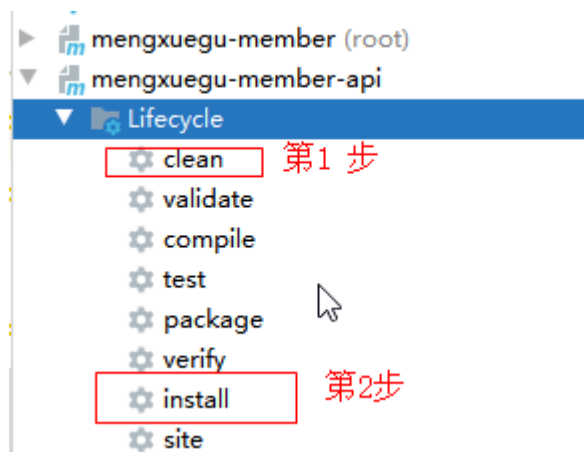
1. 打包前将 mengxuegu-member-api 工程下的 application.yml 中的数据库连接信息改为服务器的信息
2. 针对 mengxuegu-member 执行 clean 和 install



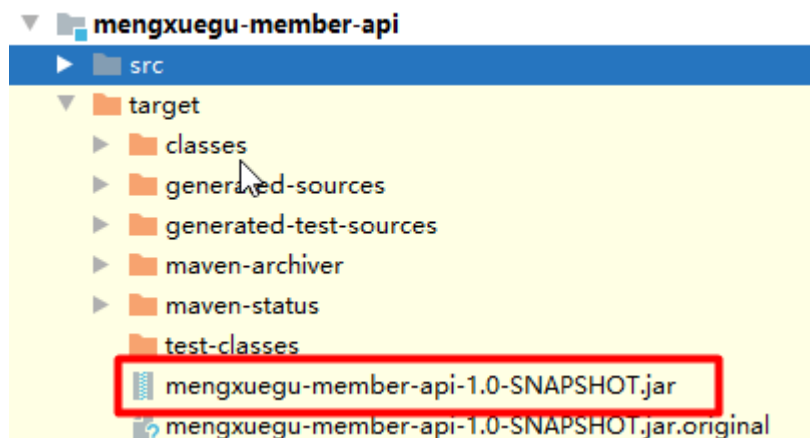
3. 针对 mengxuegu-member-api 执行 clean 和 install

闪电PDF编辑器

<https://www.callmysoft.com/>



3. 找到生成的项目jar包，上传到服务器目录



4. 先执行 `java -jar mengxuegu-member-api-1.0-SNAPSHOT.jar` 启动是否报错。

5. 不报错，则以后台进程方式启动

```
nohup java -jar mengxuegu-member-api-1.0-SNAPSHOT.jar &
```

然后执行 `exit` 退出

```
[root@izwz9e73kbnnm5u9mww2m1z member]# nohup java -jar mengxuegu-member-api-1.0-SNAPSHOT.jar &
[1] 4637
[root@izwz9e73kbnnm5u9mww2m1z member]# nohup: ignoring input and appending output to 'nohup.out'
exit
logout
```

查看进程是否存在 `ps -ef|grep java`

```
[root@izwz9e73kbnnm5u9mww2m1z ~]# ps -ef|grep java
root      4637      1  35  11:54 ?        00:00:17 java -jar mengxuegu-member-api-1.0-SNAPSHOT.jar
root      4841    4823  0  11:55 pts/8    00:00:00 grep --color=auto java
```

客户端部署

打包前端 `npm run build`

将 dist 上传到服务器 `/usr/local/nginx/html/mxg-mms`

修改 `/usr/local/nginx/conf/nginx.conf` 配置文件，

闪电PDF编辑器

<https://www.callmysoft.com/>

```
location /pro-api {  
    # 代理转发后台服务接口http://mengxuegu.com:7300/mock/5e9557822a2f716419f886dd;  
    proxy_pass http://127.0.0.1:6666/; # 最后不要少了 / 斜杠  
}
```

一定不要少了 / , 不然不会去掉 /pro-api

执行重新加载nginx配置:

```
[root@izwz9e73kbnnm5u9mww2m1z ~]# cd /usr/local/nginx/  
[root@izwz9e73kbnnm5u9mww2m1z nginx]# ./sbin/nginx -s reload
```

将 浏览器缓存清空, 然后再访问<http://vue.mengxuegu.com/login>, 不然可能获取用户信息时, 还是调用 /user/userInfo/{token} 而不是 /user/userInfo?token=xxxxx

闪电PDF编辑器

<https://www.callmysoft.com/>