# 1.mybatis

## 1.1.mybatisPlus

### 1.1.1 mybatisPlus的配置文件

|  |
| --- |
| # Mybatis-Plus 配置  mybatis-plus:  # mapper-locations: classpath:/mapper/\*Mapper.xml  #实体扫描，多个package用逗号或者分号分隔  typeAliasesPackage: com.tensquare.article.pojo  configuration:  log-impl: org.apache.ibatis.logging.stdout.StdOutImpl #打印sql语句  global-config:  id-type: 1 #0:数据库ID自增 1:用户输入id  db-column-underline: false  refresh-mapper: true  configuration:  map-underscore-to-camel-case: true #实体类属性驼峰命名规则  cache-enabled: true #配置的缓存的全局开关  lazyLoadingEnabled: true #延时加载的开关  multipleResultSetsEnabled: true #是否允许单一语句返回多结果集（需要兼容驱动） |

**提示**：

|  |
| --- |
| Mybatis Plus 提供的EntityWrapper对象封装查询条件  mybatisPlus就是设置好pojo类与表之间的映射关系--------》和通用mapper的区别是，通用mapper需要在pojo类中与数据库表建立一种全局映射关系  而mybatisPlus是在application.yml中配置一些定义  同时mybatis-plus还提供了一些较为常用的用法---》分页查询和条件查询等 |

### 1.1.2mybatisPlus的mapper接口

|  |
| --- |
| @Service  public class ArticleService {  } |

这里可以不用写接口，而直接用实体类。只是代理的方式不一样，有接口使用jdkproxy 没接口使用cglib

# 2.全局异常处理类

## 2.1.创建自定义异常

|  |
| --- |
| public class MyException extends RuntimeException {  public MyException(String message) {  super(message);  }  } |

## 2.2.异常捕获

|  |
| --- |
| @RestControllerAdvice  public class MyExceptionAdvice {  @ExceptionHandler(Exception.class)  public Result handException(Exception e){  e.printStackTrace();  return new Result(false, StatusCode.ERROR,"发生异常了，请联系管理员");  }    //捕获自定义异常  @ExceptionHandler(MyException.class)  public Result handMyException(Exception e){  e.printStackTrace();  return new Result(false, StatusCode.ERROR,e.getMessage());  }  } |

## 2.3.抛出自定义异常

|  |
| --- |
| //抛出自定义的异常  if (StringUtils.isEmpty(map.get("columnid"))) {  throw new MyException("这是我抛出的异常");  } |

# 3.跨域

## 3.1网关

### 3.1.1 GateWay网关微服务

### 3.1.2 Zuul网关微服务

1.添加依赖

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-zuul</artifactId>  </dependency>  </dependencies> |

2.配置文件

|  |
| --- |
| server:  port: 9013  spring:  application:  name: tensquare-encrypt  zuul:  routes:  tensquare-article: #文章微服务  path: /article/\*\* #配置请求URL的请求规则  serviceId: tensquare-article #指定Eureka注册中心中的服务id（指定要拦截那个微服务）  strip-prefix: true  sentiviteHeaders: #将指定路由的敏感头设置为空  customSensitiveHeaders: true #对指定路由开启自定义敏感头  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:6868/eureka/  instance:  prefer-ip-address: true |

3.启动类

|  |
| --- |
| @SpringBootApplication  @EnableEurekaClient  @EnableZuulProxy  public class EncryptApplication {  public static void main(String[] args) {  SpringApplication.run(EncryptApplication.class);  }  } |

## 3.2 @CrossOrigin

|  |
| --- |
| @RestController @RequestMapping***(*"/article"*)*** @CrossOrigin *//跨域* **public class** ArticleController ***{  …………***  } |

# 4.MongoDB

## 4.1 spring整合操作数据库

4.1.1 依赖

|  |
| --- |
| ***<*dependency*>  <*groupId*>***org.springframework.boot***</*groupId*>  <*artifactId*>***spring-boot-starter-data-mongodb***</*artifactId*> </*dependency*>*** |

4.1.2：配置文件

|  |
| --- |
| **data:  mongodb:  database:** commentdb *#数据库的名称* **host:** 192.168.211.131  **port:** 27017 |

4.1.3 操作数据库方式  **两种方式（spring-data整合以及原生的模板---很多第三方中间件都是这样，redisTemplate以及jdbcTemplate….）**

|  |
| --- |
| 方式一： spring封装  //持久层  **public interface** CommentRepository **extends** MongoRepository***<***Comment,String***> {*** *// springdata中通过方法命名方式查询  // findBy+属性名(首字母要大写)* List***<***Comment***>*** findByArticleId***(***String articleId***)***;//这个是自定义方法  ***}***  //service层  **commentRepository**.findAll***()***;  方法二：使用模板类  *//原生的查询类 有更新器 如果需要使用mongo的原生命令，就得用mongoTemplate*  @Autowired **private** MongoTemplate **mongoTemplate**;  **public void** thumbup***(***String id***) {*** Query query = **new** Query***()***;  query.addCriteria***(***Criteria.*where****(*"\_id"*)***.is***(***id***))***;   Update update = **new** Update***()***;  update.inc***(*"thumbup"**,1***)***;  **mongoTemplate**.updateFirst***(***query,update,**"comment"*)***;  ***}*** |

# 5．拦截过滤器

你继承实现了那种类型的过滤器，便具有那种作用，通过order可以设定此过滤器的执行顺序，在拦截之后，你要设定拦截之后的动作

过滤有序则设置order的大小，order数值越小越优先

|  |
| --- |
| @Component **public class** RSARequestFilter **extends** ZuulFilter ***{***  /\*\*  \* 定义过滤器的类型  \* @return  \*/  @Override  public String filterType() {  // 前置过滤，在转发微服前执行  return "pre";  }  /\*\*  \* 过滤器执行的顺序  \* 数值越小，越优先执行  \* @return  \*/  @Override  public int filterOrder() {  return 0;  }  /\*\*  \* 是否过滤（启用）  \* 返回true时，才会调用run方法  \* @return  \*/  @Override  public boolean shouldFilter() {  return true;  }  /\*\*  \* 过滤的处理方法  \* @return  \* @throws ZuulException@Override  **public** Object run***()* throws** ZuulException ***{***  ***}***  ***}*** |

# 6.微服务-SpringCloud

## 6.1引入SpringCloud依赖

|  |
| --- |
| 一般在父工程中引入  <dependencyManagement>  <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-dependencies</artifactId>  <version>Greenwich.SR1</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies>  </dependencyManagement> |

## 6.2引入注册中心

|  |
| --- |
| <dependencies>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  </dependency>  </dependencies> |

## 6.3添加注册中心配置

|  |
| --- |
| server:  port: 6868  eureka:  client:  register-with-eureka: false #是否将自己注册到eureka中  fetch-registry: false #是否从eureka中获取信息  service-url:  defaultZone: http://127.0.0.1:${server.port}/eureka/ |

## 6.4.编写启动类

|  |
| --- |
| @SpringBootApplication  @EnableEurekaServer  public class EurekaApplication {  public static void main(String[] args) {  SpringApplication.run(EurekaApplication.class, args);  }  } |

===========以上是编写注册中心微服务

===========以下是微服务注册到注册中心

## 6.5添加Eureka依赖

|  |
| --- |
| <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency> |

## 6.6修改配置文件，使用Eureka

1.defaultZone: <http://127.0.0.1:6868/eureka/>:

未配置eureka.client.serviceUrl.defaultZone值，EurekaClientConfigBean设置的默认值http://localhost:8761/eureka/不会被覆盖

2.prefer-ip-address: true #以IP地址注册到服务中心，相互注册使用IP地址

单节点部署为分布式Eureka集群，设置当前机器hostname为多个，不同节点eureka.instance.hostname设置为不同值，同时prefer-ip-address禁止设置为true，否则DS Replicas列表中会出现不应该出现的节点。

|  |
| --- |
| eureka:  client:  service-url:  defaultZone: <http://127.0.0.1:6868/eureka/>  instance:  prefer-ip-address: true #以IP地址注册到服务中心，相互注册使用IP地址 |

## 6.6在微服务中添加@EnableEurekaClient依赖

|  |
| --- |
| @SpringBootApplication  //配置Mapper包扫描  @MapperScan("com.tensquare.article.dao")  @EnableEurekaClient  //(注册中心微服务为@EnableEurekaServer,使用的为@EnableEurekaClient)  public class ArticleApplication {  public static void main(String[] args) {  SpringApplication.run(ArticleApplication.class, args);  }  } |

# 7.Spring-Test测试类

7.1引入依赖

|  |
| --- |
| ***<*dependency*>  <*groupId*>***org.springframework.boot***</*groupId*>  <*artifactId*>***spring-boot-starter-test***</*artifactId*>  <*scope*>***test***</*scope*> </*dependency*>*** |

7.2使用

|  |
| --- |
| @RunWith***(***SpringJUnit4ClassRunner.**class*)*** @SpringBootTest***(***classes= EncryptApplication.**class*)//启动类* public class** EncryptTest ***{***  ………………………  } |