## Spring Boot开发手册

Macbook笔记本，位置在：~/jwork/demo

集成开发环境：IDEA

手动开发环境：VS Code、maven命令行

《Spring Boot+Vue 3大型前后端分离项目实战》(2023)

掘金小册：<https://juejin.cn/book/6844733826191589390>

+ 前端开发

+ docker发布

1. 建立项目

使用IDEA创建demo项目，基于Spring Boot 2.6.0，JDK 1.8

1. Lombok数据实体

导入依赖：pom.xml

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</dependency>

创建data/User.java

@Data注解，可以避免写getter/setter，简化代码

1. Mybatis对接数据库

导入依赖：pom.xml

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus-boot-starter</artifactId>

<version>3.5.2</version>

</dependency>

实体类：data/User.java，要继承Serializable

用@TableName注解表名，用@TableId注解主键，用@TableField注解字段

注意字段中的下划线要转换为大小写驼峰写法

@TableName("tb\_user")

@Data

public class User implements Serializable

映射类：dao/UserMapper.java，继承BaseMapper<User>

public interface UserMapper extends BaseMapper<User>

映射声明：resources/mapper/UserMapper.xml，指向映射类

<mapper namespace="com.hifiax.demo.dao.UserMapper">

配置参数：resources/application.properties，指向映射声明

spring.datasource.name=testdb-datasource

spring.datasource.driverClassName=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/test\_db?useUnicode=true&serverTimezone=Asia/Shanghai&characterEncoding=utf8&autoReconnect=true&useSSL=false&allowMultiQueries=true

spring.datasource.username=hifi

spring.datasource.password=hifi

mybatis-plus.mapper-locations=classpath:mapper/\*Mapper.xml

初始化配置：DemoApplication.java

@MapperScan("com.hifiax.demo.dao")

使用前要声明，然后才能用：DemoApplicationTests.java

@Autowired

private UserMapper userMapper;

User user=userMapper.selectById(1);

1. Swagger接口定义

导入依赖：pom.xml

<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>

注册类：config/Swagger2Config.java

@Configuration

@EnableSwagger2

public class Swagger2Config {

@Bean

public Docket api() {

return new Docket(DocumentationType.SWAGGER\_2)

.apiInfo(apiInfo())

.select()

.apis(RequestHandlerSelectors.basePackage("ltd.newbee.mall.controller"))

.paths(PathSelectors.any())

.build();

}

private ApiInfo apiInfo() {

return new ApiInfoBuilder()

.title("swagger-api文档")

.description("swagger文档 by 13")

.version("1.0")

.build();

}

}

用@ApiOperation注解函数信息

可访问：<http://localhost:8080/swagger-ui.html>

1. REST接口

样例：~/gwork/fault-detect-fe

前后端分离，采用RestController，仅返回数据而非页面。样例如下：

@RestController  
@RequestMapping("/api/v1")  
public class NotifyController {

@PutMapping("/notify")  
public String notifyMessage() {

return "ok";  
 }  
}

访问命令：

curl -X PUT hz1.hifiax.com:8080/api/v1/notify

1. 前端开发

可参考：~/feixiong/law-llm-fe，~/vwork/test

npm create vite@latest

npm install

npm run dev

远程访问配置：

vite.config.js文件中增加：

import { fileURLToPath, URL } from 'node:url'

resolve: {

alias: {

'@': fileURLToPath(new URL('./src', import.meta.url))

}

},

server: {

allowedHosts: ['localhost', 'hz1.hifiax.com'],

host: true,

port: 8000,

open: false,

cors: true,

strictPort: false,

proxy: {

"/api": {

target: "http://localhost:8080/api/v1",

changeOrigin: true,

rewrite:(path) => path.replace(/^\/api/, "")

},

}

}

设置路由：

npm install vue-router@next -S

修改main.js：

import router from './router'

const app=createApp(App)

app.use(router)

app.mount('#app')

更新App.vue

以及创建router/index.js，views/Home.vue

导入ElementPlus组件：

npm install element-plus -S

修改main.js：

import ElementPlus from 'element-plus'

import 'element-plus/dist/index.css'

app.use(ElementPlus)

导入axios：

npm install axios

1. Docker准备

HZ1主机，hifi用户

安装docker

sudo apt update

sudo apt-get -y install apt-transport-https ca-certificates curl software-properties-common

curl -fsSL https://mirrors.aliyun.com/docker-ce/linux/ubuntu/gpg | sudo apt-key add -

sudo apt-key fingerprint 0EBFCD88

sudo add-apt-repository "deb [arch=amd64] https://mirrors.aliyun.com/docker-ce/linux/ubuntu $(lsb\_release -cs) stable"

sudo apt update

sudo apt-get install docker-ce docker-ce-cli containerd.io

docker --version ；28.1.1

免sudo使用

sudo usermod -aG docker hifi

newgrp - docker

编写Dockerfile，如下：

FROM openjdk:8

COPY target/fault\_detect-0.0.1-SNAPSHOT.jar app.jar

ENTRYPOINT ["java","-jar","/app.jar"]

制作镜像

docker pull openjdk:8

docker build -t fault-detect:1.0 .

docker run -d -p 8080:8080 fault-detect:1.0

安装kubectl

sudo snap install kubectl --classic

1. 容器云准备

Azure AKS

新建资源组hifi，集群trading，节点池hifipool，订阅:Azure subscription 1

容器仓库：hifiwork.azurecr.io

对AKS集群trading，附加容器注册表hifiwork

安装az工具

curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash

容器云配置

az login ；选择订阅

az account list --output table ；查看订阅号

az account set --subscription "e14bfcdd-9618-4665-89d2-e420360aa107"

az aks get-credentials --resource-group hifi --name trading ；获得证书

更新.kube/config，并设trading集群为默认

镜像仓库配置

az acr update -n hifiwork --admin-enabled true

az acr credential show --resource-group hifi --name hifiwork ；获得密码

az acr login --name hifiwork

1. 云原生应用

上传镜像

docker tag fault-detect:1.0 hifiwork.azurecr.io/fault-detect:1.0

docker push hifiwork.azurecr.io/fault-detect:1.0

编写：deployment.yml

kubectl apply -f deployment.yml

kubectl get pods

编写：service.yml

kubectl apply -f service.yml

kubectl get service

外部访问

curl 20.3.27.26:8080/api/v1/alert

1. 云原生特性

POD故障重启

kubectl delete pod fd-deployment-5664484d9c-d8lxv

kubectl get pods

可以看到瞬间又启动一个新的POD

节点故障重启

在管理界面删除一个节点

kubectl get nodes

kubectl get pods

kubectl get deployments

可以看到POD被分配到其它节点并重启

动态扩缩容

修改deployment.yml

replicas: 2 => 3

kubectl apply -f deployment.yml

可以看到动态扩容

kubectl scale --replicas 2 Deployment/fd-deployment

用命令行作动态缩容

滚动更新

docker tag fault-detect:1.0 hifiwork.azurecr.io/fault-detect:1.1

docker push hifiwork.azurecr.io/fault-detect:1.1

kubectl apply -f deployment.yml

kubectl rollout status deployment fd-deployment ；查看更新状态

kubectl describe deployment fd-deployment ；查看应用版本