服务器端搭建

SSM框架：Spring+SpringMVC+MyBatis SpringBoot工具

创建所有项目都是基于maven创建。 maven项目管理(清理-编译-测试-打包-部署)和依赖(jar)管理的工具。

> 解压maven

```
D:\tools\apache-maven-3.8.8
```

> 配置maven

打开D:\tools\apache-maven-3.8.8\conf\settings.xml

```
-- 本地仓库    首先创建一个本地仓库文件夹 D:\tools\repository2
<!-- 配置maven的本地仓库的地址 -->
<localRepository>D:\tools\repository2</localRepository>

-- 下载jar服务器镜像改为国内
<mirror>
    <id>aliyunmaven</id>
    <mirrorOf>*</mirrorOf>
    <name>阿里云公共仓库</name>
    <url>https://maven.aliyun.com/repository/public</url>
</mirror>
```

> maven集成Idea

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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <!-- 当前项目坐标 -->
    <groupId>com.haiyang</groupId>
    <artifactId>mall-server-021</artifactId>
    <version>1.0.0</version>

    <packaging>jar</packaging>

    <!-- 运行参数 -->
    <properties>
        <java.version>1.8</java.version>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <!-- springboot版本 -->
        <spring-boot.version>2.3.7.RELEASE</spring-boot.version>
    </properties>

    <!-- 依赖 -->
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

```
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!-- MySQL数据库驱动-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <!-- lombok插件-->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
            </exclusions>
        </dependency>
    </dependencies>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-dependencies</artifactId>
                <version>${spring-boot.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                    <encoding>UTF-8</encoding>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <version>2.3.7.RELEASE</version>
            </plugin>
        </plugins>
    </build>
</project>
```

application.yaml

```
server:
  port: 10001
  servlet:
    context-path: /
```

```
注意：
    yaml文件配置属性节点层次必须对其。
    属性值前面必须有一个空格

项目启动，浏览器访问：
http://localhost:10001/index

如果设置 context-path: /api
http://localhost:10001/api/index
```

> 启动类

包：包名全部小写。包是分层级。

```
例如：com.haiyang
对应物理磁盘上就是文件夹。

包名--域名倒置   cctv.com   baidu.com
```

分层开发：

```
Vue界面
    |
控制层：  处理前端发来的请求. Vue请求( 登录 )。    com.haiyang.controller
    |
业务逻辑层   项目具体功能和流程   登录方法    com.haiyang.service
    |
数据访问层：  对数据库CRUD操作        com.haiyang.mapper
    |
MySQL
```

```
package com.haiyang;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MallServer021Application {
    public static void main(String[] args) {
        SpringApplication.run(MallServer021Application.class, args);
    }
}
```

> 创建IndexController控制器

```
package com.haiyang.controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController   //当前类是一个控制器
@RequestMapping("/index")
public class IndexController {
```

```
    //处理请求方法
    //方法返回数据  就是响应给前端的数据
    @RequestMapping("/test")
    public String test(){
        return "Hello SpringBoot";
    }
}
```

实体类

封装数据:

```
MySQL sys_account表 -----查询account_id是19988999988----->实体类对象
      sys_account表-------------所有用户--------------->List<实体类>    泛型

Account       sys_account
实体类<--映射-->表
变量  --------- 字段
```

```
---实体类代码结构---
私有变量 -- 映射表中字段
无参构造方法
变量set和get方法
```

```
package com.haiyang.entity;
import lombok.Data;

@Data
public class Account {
    private String accountId;
    private String password;
    private String accountName;
}
```

代码逆向生成

```
<dependency>
   <groupId>mysql</groupId>
   <artifactId>mysql-connector-java</artifactId>
   <version>5.1.49</version>
</dependency>
```

编写控制器的父类

```
package com.haiyang.common;
//控制器的父类
public class BaseController {

}
```

编写实体类父类

```
package com.haiyang.common;
import com.baomidou.mybatisplus.annotation.TableField;
import java.time.LocalDateTime;

public class BaseEntity {
    /**
     * 创建时间
     */
    @TableField("created")
    private LocalDateTime created;

    /**
     * 修改时间
     */
    @TableField("updated")
    private LocalDateTime updated;

    @TableField("statu")
    private Integer statu;
}
```

> 配置mybatis-plus框架

导入mybatisPlus逆向代码生成的相关jar包，修改pom.xml

```
<!--  mybatis-plus -->
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-boot-starter</artifactId>
    <version>3.4.2</version>
</dependency>
<!-- 代码生成器 -->
<dependency>
    <groupId>com.baomidou</groupId>
    <artifactId>mybatis-plus-generator</artifactId>
    <version>3.4.1</version>
</dependency>
<!-- 代码自动生成器模板依赖-->
<dependency>
    <groupId>org.apache.velocity</groupId>
    <artifactId>velocity-engine-core</artifactId>
    <version>2.2</version>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-freemarker</artifactId>
</dependency>
```

配置springboot项目，整合myBatisPlus，修改application.yaml

```yaml
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/system?useUnicode=true&characterEncoding=utf-8&useSSL=false&serverTimezone=GMT%2B8
    username: root
    password: root
```

修改CategroyMapper接口

```
加上@Mapper
package com.haiyang.mapper;
import com.haiyang.entity.Category;
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import org.apache.ibatis.annotations.Mapper;

@Mapper
public interface CategoryMapper extends BaseMapper<Category> {

}
```

编写CategoryController中测试方法

```
package com.haiyang.controller;
import com.haiyang.entity.Category;
import com.haiyang.service.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;
import com.haiyang.common.BaseController;

import java.util.List;

@RestController
@RequestMapping("/category")
public class CategoryController extends BaseController {
    //需要创建CategoryService对象
    //@Autowired Spring框架自动创建好该对象
    @Autowired
    private CategoryService cService;

    //处理 前端请求所有商家分类的请求
    @RequestMapping("/list")
    public List<Category> list(){
        //获得所有的商家分类数据  sys_category表中数据

        //返回 List<Category>
        return cService.list();
    }
}
```

```
访问地址:http://localhost:10001/category/list
```