**作业一 自己总结的SSM知识点**，github仓库地址：https://github.com/gitwn/javawork.git

Spring

SSM框架集由Spring、SpringMVC、MyBatis三个开源框架整合而成，常作为数据源较简单的web项目的框架。其中spring是一个轻量级的控制反转（IoC）和面向切面（AOP）的容器框架。

SpringMVC分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。

MyBatis是一个支持普通SQL查询，存储过程和高级映射的优秀持久层框架。

环境的搭建

springmvc 环境 jar 包依赖

<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/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.xxx</groupId>

<artifactId>springmvc01</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>springmvc01 Maven Webapp</name>

<url>http://maven.apache.org</url>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

<!-- spring web -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- spring mvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- web servlet -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.0.1</version>

</dependency>

</dependencies>

<!-- jetty 插件 -->

<build>

<finalName>springmvc01</finalName>

<resources>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

<plugins>

<!-- 编译环境插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>1.7</source>

<target>1.7</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<plugin>

<groupId>org.mortbay.jetty</groupId>

<artifactId>maven-jetty-plugin</artifactId>

<version>6.1.25</version>

<configuration>

<scanIntervalSeconds>10</scanIntervalSeconds>

<contextPath>/springmvc01</contextPath>

</configuration>

</plugin>

</plugins>

</build>

</project>

配置 web.xml (前端控制器配置)

<?xml version="1.0" encoding="UTF-8"?>

<web-app id="WebApp\_ID" version="3.0"

xmlns="http://java.sun.com/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd">

<!-- 表示容器启动时 加载上下文配置 这里指定 spring 相关配置 -->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:\*.xml</param-value>

</context-param>

<!-- 启用 spring 容器环境上下文监听 -->

<listener>

<listener class>org.springframework.web.context.ContextLoaderListener</listenerclass>

</listener>

<!-- 编码过滤 utf-8 -->

<filter>

<description>char encoding filter</description>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!-- servlet 请求分发器 -->

<servlet>

<servlet-name>springMvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:servlet-context.xml</param-value>

</init-param>

<!-- 表示启动容器时初始化该 Servlet -->

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springMvc</servlet-name>

<!-- 这是拦截请求, /代表拦截所有请求,拦截所有.do 请求 -->

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

servlet-context.xml 配置

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx.xsd">

<!-- 扫描 com.xxx.controller 下包 -->

<context:component-scan base-package="com.xxx.controller" />

<!-- mvc 请求映射 处理器与适配器配置-->

<mvc:annotation-driven/>

<!--配置视图解析器 默认的视图解析器- -->

<bean id="defaultViewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass"

value="org.springframework.web.servlet.view.JstlView" />

<property name="contentType" value="text/html" />

<property name="prefix" value="/WEB-INF/jsp/" />

<property name="suffix" value=".jsp" />

</bean>

</beans>

\*\*如果返回乱码：配置消息转换器 \*\*

<!-- 消息转换器 -->

<mvc:message-converters register-defaults="true">

<bean class="org.springframework.http.converter.StringHttpMessageConverter">

<property name="supportedMediaTypes" value="text/html;charset=UTF-8"/>

</bean>

</mvc:message-converters>

三个框架的整合

jar 包依赖添加 (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.mage</groupId>

<artifactId>ssm-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>war</packaging>

<name>ssm-demo Maven Webapp</name>

<!-- FIXME change it to the project's website -->

<url>http://www.example.com</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.7</maven.compiler.source>

<maven.compiler.target>1.7</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.11</version>

<scope>test</scope>

</dependency>

<!-- spring 核心 jar -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- spring 测试 jar -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-test</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- spring jdbc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-jdbc</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- spring 事物 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-tx</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- aspectj 切面编程的 jar -->

<dependency>

<groupId>org.aspectj</groupId>

<artifactId>aspectjweaver</artifactId>

<version>1.8.9</version>

</dependency>

<!-- c3p0 连接池 -->

<dependency>

<groupId>c3p0</groupId>

<artifactId>c3p0</artifactId>

<version>0.9.1.2</version>

</dependency>

<!-- mybatis -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.1</version>

</dependency>

<!-- 添加 mybatis 与 Spring 整合的核心包 -->

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis-spring</artifactId>

<version>1.3.0</version>

</dependency>

<!-- mysql 驱动包 -->

<dependency>

<groupId>mysql</groupId>

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

<version>5.1.39</version>

</dependency>

<!-- 日志打印相关的 jar -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<version>1.7.2</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.2</version>

</dependency>

<!-- 分页插件配置 -->

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>4.1.0</version>

</dependency>

<!-- spring web -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-web</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- spring mvc -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>4.3.2.RELEASE</version>

</dependency>

<!-- web servlet -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.0.1</version>

</dependency>

<!-- 添加 json 依赖 jar 包 -->

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-core</artifactId>

<version>2.7.0</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.7.0</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-annotations</artifactId>

<version>2.7.0</version>

</dependency>

<!-- 文件上传包依赖 -->

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.2</version>

</dependency>

<!--jstl包-->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

<!--是程序可以读取到mapper下面的xml配置文件-->

<build>

<finalName>spring-mybatis</finalName>

<resources>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.xml</include>

</includes>

</resource>

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.properties</include>

</includes>

</resource>

</resources>

</build>

</project>

web.xml 文件配置

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee"

xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"

id="WebApp\_ID" version="3.0">

<!--spring和mybatis的配置文件-->

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring.xml</param-value>

</context-param>

<!--spring监听器-->

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<!--编码过滤器-->

<filter>

<description>char encoding filter</description>

<filter-name>encodingFilter</filter-name>

<filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>

<init-param>

<param-name>encoding</param-name>

<param-value>UTF-8</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<!--springmvc servlet-->

<servlet>

<servlet-name>springmvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:springmvc.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<!--springmvc需要拦截的内容-->

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

Springmvc.xml 配置文件

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xmlns:context="http://www.springframework.org/schema/context"

xsi:schemaLocation="

http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd

http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-3.0.xsd">

<!-- 扫描 com.xxx.controller 下包 -->

<context:component-scan base-package="com.mage.controller" />

<!-- mvc 请求映射处理器与适配器 -->

<mvc:annotation-driven />

<!--配置视图解析器 默认的视图解析器- -->

<bean id="defaultViewResolver"

class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="viewClass" value="org.springframework.web.servlet.view.JstlView" />

<property name="contentType" value="text/html" />

<property name="prefix" value="/WEB-INF/jsp/" />

<property name="suffix" value=".jsp" />

</bean>

<!-- json 支持 -->

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping">

</bean>

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">

<property name="messageConverters">

<list>

<bean class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter" />

</list>

</property>

</bean>

<!-- 文件上传配置 -->

<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<property name="maxUploadSize">

<value>104857600</value>

</property>

<property name="maxInMemorySize">

<value>4096</value>

</property>

</bean>

<!-- &lt;!&ndash;全局异常的处理1&ndash;&gt;

<bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">

&lt;!&ndash;默认的异常视图&ndash;&gt;

<property name="defaultErrorView" value="error"></property>

&lt;!&ndash;前台获取数据&ndash;&gt;

<property name="exceptionAttribute" value="ex"></property>

<property name="exceptionMappings">

<props>

<prop key="com.mage.exception.TestException">error1</prop>

</props>

</property>

</bean>-->

</beans>

spring.xml配置

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/aop

http://www.springframework.org/schema/aop/spring-aop.xsd

http://www.springframework.org/schema/tx

http://www.springframework.org/schema/tx/spring-tx-3.0.xsd">

<!-- 扫描基本包 过滤 controller 层 -->

<context:component-scan base-package="com.mage" >

<context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller" />

</context:component-scan>

<!-- 加载 properties 配置文件 -->

<context:property-placeholder location="classpath:db.properties" />

<aop:aspectj-autoproxy /><!-- aop -->

<!-- 配置 c3p0 数据源 -->

<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">

<property name="driverClass" value="${jdbc.driver}"></property>

<property name="jdbcUrl" value="${jdbc.url}"></property>

<property name="user" value="${jdbc.username}"></property>

<property name="password" value="${jdbc.password}"></property>

</bean>

<!-- 配置事务管理器 -->

<bean id="txManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource"></property>

</bean>

<!-- 设置事物增强 -->

<tx:advice id="txAdvice" transaction-manager="txManager">

<tx:attributes>

<tx:method name="get\*" read-only="true" />

<tx:method name="find\*" read-only="true" />

<tx:method name="query\*" read-only="true" />

<tx:method name="load\*" read-only="true" />

<tx:method name="add\*" propagation="REQUIRED" />

<tx:method name="insert\*" propagation="REQUIRED" />

<tx:method name="update\*" propagation="REQUIRED" />

<tx:method name="delete\*" propagation="REQUIRED" />

</tx:attributes>

</tx:advice>

<!-- aop 切面配置 -->

<aop:config>

<aop:pointcut id="servicePointcut" expression="execution(\* com.mage.service..\*.\*(..))" />

<aop:advisor advice-ref="txAdvice" pointcut-ref="servicePointcut" />

</aop:config>

<!-- 配置 sqlSessionFactory -->

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"></property>

<property name="configLocation" value="classpath:mybatis.xml" />

<property name="mapperLocations" value="classpath:com/mage/mapper/\*.xml" />

</bean>

<!-- 配置扫描器 -->

<bean id="mapperScanner"

class="org.mybatis.spring.mapper.MapperScannerConfigurer">

<!-- 扫描 com.xxx.dao 这个包以及它的子包下的所有映射接口类 -->

<property name="basePackage" value="com.mage.dao" />

<property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />

</bean>

</beans>

spring IOC创建对象的两种方式

1.创建容器对象

2.在当前包的所在类下找有关bean的配置文件 测试方便

spring主配置文件创建对象的方式

1.调用无参构造器

2.调用带参构造器

3.用工厂模式创建 工厂里面分为静态方法和非静态方法

spring给对象属性赋值依赖注入的方式

1.构造函数赋值

2.set方法注入值

分为 普通字段赋值和集合属性赋值(list map property)

IOC

依赖注入原理，控制反转思想，是一种设计思想而不是技术；就是把设计好的对象交给容器bean（xml或注解）控制，不再是对象里使用new关键字，避免了硬编码所造成的过度程序耦合，实现解耦。

AOP

实现原理就是Java动态代理,但是动态代理必须实现接口,所以spring的aop是用cglib实现的

<aop:aspectj-autoproxy/>

代码实现

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.After;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.annotation.Pointcut;

import org.springframework.stereotype.Component;

/\*\*

\* 声明切面组件

\*/

@Component

@Aspect

public class LogCut {

/\*\*

\* 定义切入点 匹配方法规则定义

\* 匹配规则表达式含义 拦截 com.xxx.service 包下 以及子包下 所有类的所有

方法

\*/

@Pointcut("execution (\* com.xxx.service..\*.\*(..))")

public void cut(){}

/\*\*

\* 声明前置通知 并将通知应用到定义的切入点上

\* 目标泪方法执行前 执行该通知

\*/

@Before(value="cut()")

public void before(){

System.out.println("前置通知.....");

}

/\*\*

\* 声明返回通知 并将通知应用到切入点上

\* 目标类方法执行完毕执行该通知

\*/

@AfterReturning(value="cut()")

public void afterReturning(){

System.out.println("返回通知....");

}

/\*\*

\* 声明最终通知 并将通知应用到切入点上

\* 目标类方法执行过程中是否发生异常 均会执行该通知 相当于异常中的 finally

\*/

@After(value="cut()")

public void after(){

System.out.println("最终通知....");

}

/\*\*

\* 声明异常通知 并将通知应用到切入点上

\* 目标类方法执行时发生异常 执行该通知

\*/

@AfterThrowing(value="cut()",throwing="e")

public void afterThrowing(Exception e){

System.out.println("异常通知....方法执行异常时执行:"+e);

}

/

\*\*

\* 声明环绕通知 并将通知应用到切入点上

\* 方法执行前后 通过环绕通知定义相应处理

\*/

@Around(value="cut()")

public Object around(ProceedingJoinPoint pjp) throws Throwable{

System.out.println("环绕前置...");

System.out.println("环绕通知");

System.out.println(pjp.getTarget()+"--"+pjp.getSignature());

Object result=pjp.proceed();//执行目标对象方法

System.out.println("环绕后置...");

return result;

}

}

配置方式实现

配置切面，切入点，通知

<!-- aop 相关配置 -->

<aop:config>

<!-- aop 切面配置 -->

<aop:aspect ref="logCut">

<!-- 定义 aop 切入点 -->

<aop:pointcut expression="execution (\* com.mage.service..\*.\*(..))"

id="cut"/>

<!-- 配置前置通知 指定前置通知方法名 并引用切入点定义 -->

<aop:before method="before" pointcut-ref="cut"/>

<!-- 配置返回通知 指定返回通知方法名 并引用切入点定义 -->

<aop:after-returning method="afterReturning" pointcut-ref="cut"/>

<!-- 配置异常通知 指定异常通知方法名 并引用切入点定义 -->

<aop:after-throwing method="afterThrowing" throwing="e" pointcutref="cut"/>

<!-- 配置最终通知 指定最终通知方法名 并引用切入点定义 -->

<aop:after method="after" pointcut-ref="cut"/>

<!-- 配置环绕通知 指定环绕通知方法名 并引用切入点定义 -->

<aop:around method="around" pointcut-ref="cut"/>

</aop:aspect>

</aop:config>

简述Bean的作用域有几种？

1、ingleton，单例模式，bean的实例只有一个

2、protopyte，每次容器获取bean时，都会创建一个新的实例

3、request，用于web应用环境，针对每次的http请求都会创建一个实例

4、session，同一个会话共享一个实例，不同的会话使用不同的实例

5、 global，仅在Porplet的web环境中使用，同一个全局会话共享一个实例。对于非Porplet环境等同于session

简述自动装配都有几种类型？

ByName byType constructor

SpringMVC主要解决接收请求与处理响应的问题。

关于SpringMVC框架，必须掌握和理解的：

1. 创建SpringMVC项目，并完成配置；

2. 掌握常用注解：`@RequestMapping`、`@ResponseBody`、`@RequestParam`、`@RestController`、`@GetMapping`、`@PostMapping`、`@ExceptionHandler`；

3. 掌握转发与重定向，及转发时封装转发的数据；

4. 掌握响应JSON正文的方式；

掌握拦截器的使用；

MyBatis  
MyBatis解决了传统模式下持久层开发比较繁琐的问题。

关于MyBatis框架，必须掌握和理解的：  
1. 基本的增删改查功能开发；  
2. 查询时，什么时候需要自定义别名；  
3. 什么时候需要使用VO类；  
4. 掌握`<resultMap>`的使用；  
5. 掌握动态SQL中的`<foreach>`的使用；  
6. 理解`#{}`和`${}`的区别。