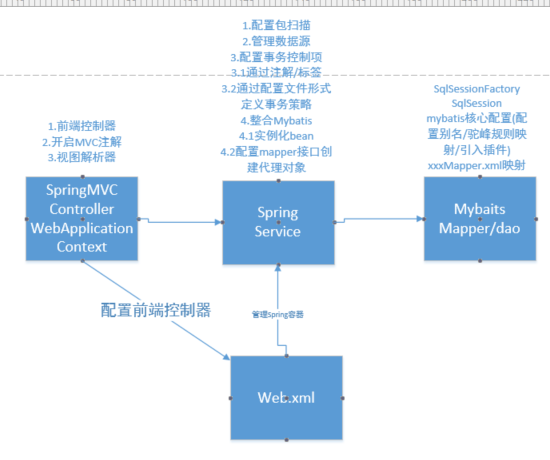
# 京淘后台环境搭建(二)

## 框架搭建

### 框架配置图

说明:

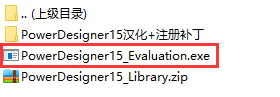
1. Mybatis:负载数据持久化的 dao/mapper
2. Spring:可以整合现阶段所有主流的框架(IOC/AOP) service
3. SpringMVC:主要负责与用户进行交互.负责封装数据/回显数据 controller



## 代码实现业务

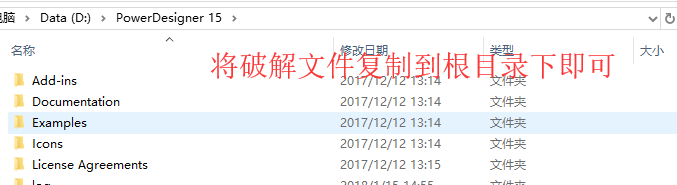
### 安装PD

1. 执行exe文件

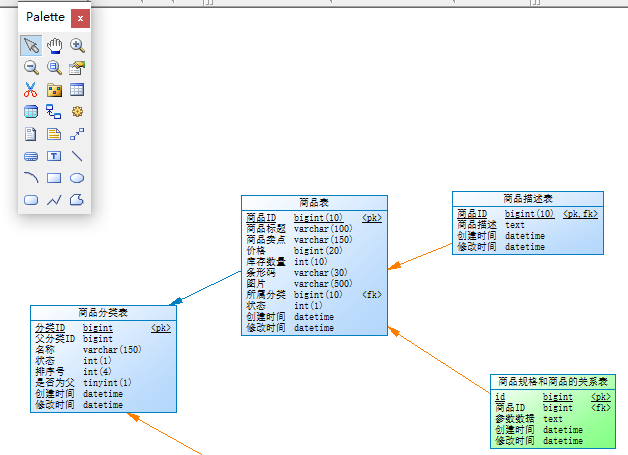


1. 汉化破解

将破解文件复制,之后保存到PD安装的根目录下即可.破解成功后重启PD即可.



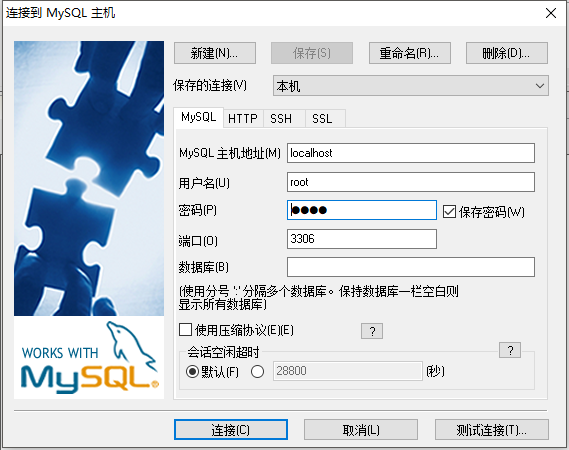
1. 使用PD工具打开PDM文件



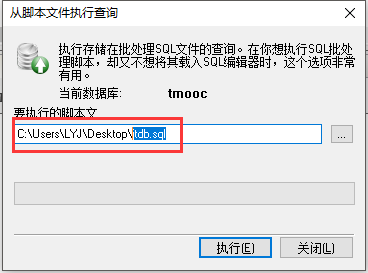
数据建模工具.

### 导入数据库

1. 安装sqlYog工具



1. 导入数据库



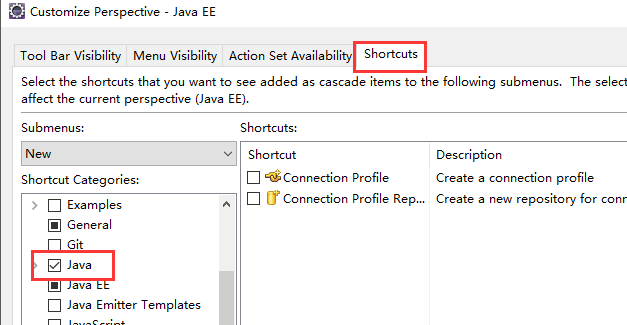
### 需求

要求:

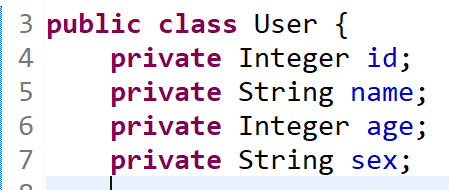
根据localhost:8091/findAll查询用户全部信息.要求使用jsp实现用户数据的回显.

### 编辑User POJO对象

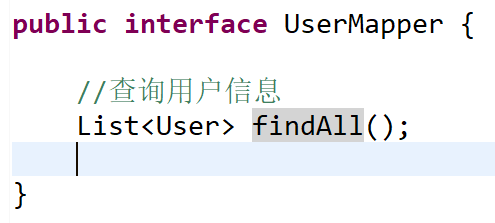
1. 修改视图



1. 编辑pojo



### 编辑Mapper接口/实现类



编辑映射文件

<?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">

<!--

1.表示映射文件 名称不能重复

2.表示与接口的映射关系.

-->

<mapper namespace=*"com.jt.manage.mapper.UserMapper"*>

<!--包名通过配置文件的形式定义.

编辑sql时,切记不要加";"号

-->

<select id=*"findAll"* resultType=*"User"*>

select \* from user

</select>

</mapper>

### 编辑Service

@Service

**public** **class** UserServiceImpl **implements** UserService {

@Autowired

**private** UserMapper userMapper;

@Override

**public** List<User> findAll() {

**return** userMapper.findAll();

}

}

### 编辑Controller

@Controller

**public** **class** UserController {

@Autowired

**private** UserService userService;

@RequestMapping("/findAll")

**public** String findAll(Model model){

List<User> userList =

userService.findAll();

//将数据传递到页面中 将数据保存到request域中

model.addAttribute("userList", userList);

//返回的页面的名称

**return** "userList";

}

}

### 编辑页面

通过jsp中的jstl表达式遍历数据

<%@ page language=*"java"* import=*"java.util.\*"* contentType=*"text/html; charset=utf-8"* pageEncoding=*"utf-8"*%>

<%@ taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"* %>

<html>

<head>

<meta http-equiv=*"content-type"* content=*"text/html; charset=UTF-8"*>

<title>用户列表页面</title>

</head>

<body>

<table width=*"60%"* algin=*"center"* border=*"1"*>

<tr align=*"center"*>

<td colspan=*"5"*><h2>用户信息</h2></td>

</tr>

<tr align=*"center"*>

<td>ID</td>

<td>姓名</td>

<td>年龄</td>

<td>性别</td>

</tr>

<!--通过jstl表达式遍历数据, -->

<c:forEach items=*"*${userList}*"* var=*"u"*>

<tr align=*"center"*>

<td>${u.id}</td>

<td>${u.name}</td>

<td>${u.age}</td>

<td>${u.sex}</td>

</tr>

</c:forEach>

</table>

</body>

</html>

## 编辑配置文件

### 编辑web.xml

<?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\_2\_5.xsd"*

id=*"jt-manage"* version=*"2.5"*>

<display-name>jt-manage</display-name>

<!--配置前端控制器 -->

<servlet>

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

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

<!--加载容器SpringMVC/配置文件 -->

<init-param>

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

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

</init-param>

</servlet>

<!--表示springMVC拦截请求的路径

\*.do/action 只要请求中有.do结尾的请求 就会拦截

/service/\* 拦截以service开头的全部请求.

/ 放行动态资源请求(.jsp),拦截静态资源和请求,不需要添加后缀

RESTFul参数风格??? 需要后期放行静态资源

/\* 拦截全部的资源请求 禁止使用

-->

<servlet-mapping>

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

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

</servlet-mapping>

<!--解决post乱码问题 配置过滤器 -->

<filter>

<filter-name>characterEncoding</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>characterEncoding</filter-name>

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

</filter-mapping>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

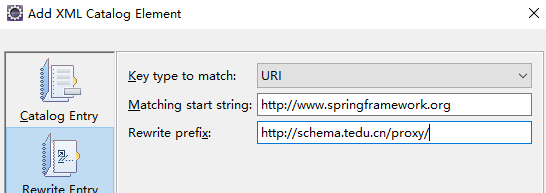
</welcome-file-list>

</web-app>

### 配置springMVC.xml

1. 配置代理服务器

代理路径: <http://schema.tedu.cn/proxy/>



1. 编辑配置文件

<?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:context=*"http://www.springframework.org/schema/context"*

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

xsi:schemaLocation=*"*

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

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

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

*http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*

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

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

<!--开启mvc注解 -->

<mvc:annotation-driven/>

<!--放行静态资源 -->

<mvc:default-servlet-handler/>

<!--配置视图解析器 内部资源视图解析器

主要作用:拼接前缀和后缀

路径问题:要求使用视图解析器时,必须添加绝对路径

如果采用相对路径可能会造成请求页面偏差

localhost:8091/addUser/WEB-INF/views/updateUser

-->

<bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

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

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

</bean>

</beans>

### 编辑pro配置文件

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/jtdb?useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

jdbc.username=root

jdbc.password=root

### 编辑Spring配置文件

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

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

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

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

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

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd

http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd">

<!--1.开启包扫描 -->

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

<!--2.1spring加载pro

如果需要加载多个配置文件则使用通配符

-->

<!-- <context:property-placeholder location="classpath:/properties/\*.properties"/> -->

<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<value>classpath:/properties/jdbc.properties</value>

</list>

</property>

</bean>

<!--2.2管理数据源 -->

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">

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

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

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

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

</bean>

<!--配置Spring事务策略

注解形式: 配置简单 但是使用时必须为方法或者/类添加注解

配置文件形式:

配置方式较为繁琐,

但是以后写法方法时不需要再关注事务控制

-->

<!--3.1配置事务管理器 -->

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

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

</bean>

<!--3.2定义事务管理策略

事务传播属性

REQUIRED 表示必须添加事务.

NEVER 表示从不添加事务

SUPPORTS 如果之前的操作有事务,则合并为一个事务,如果之前

没有事务,则不添加事务.

"REQUIRES\_NEW" 每次添加新的事务

-->

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

<tx:attributes>

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

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

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

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

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

</tx:attributes>

</tx:advice>

</beans>

### 配置spring配置文件

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

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

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

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

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

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd

http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd">

<!--1.开启包扫描 -->

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

<!--2.1spring加载pro

如果需要加载多个配置文件则使用通配符

-->

<!-- <context:property-placeholder location="classpath:/properties/\*.properties"/> -->

<bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">

<property name="locations">

<list>

<value>classpath:/properties/jdbc.properties</value>

</list>

</property>

</bean>

<!--2.2管理数据源 -->

<bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">

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

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

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

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

</bean>

<!--配置Spring事务策略

注解形式: 配置简单 但是使用时必须为方法或者/类添加注解

配置文件形式:

配置方式较为繁琐,

但是以后写法方法时不需要再关注事务控制

-->

<!--3.1配置事务管理器 -->

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

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

</bean>

<!--3.2定义事务管理策略

事务传播属性

REQUIRED 表示必须添加事务.

NEVER 表示从不添加事务

SUPPORTS 如果之前的操作有事务,则合并为一个事务,如果之前

没有事务,则不添加事务.

"REQUIRES\_NEW" 每次添加新的事务

-->

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

<tx:attributes>

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

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

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

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

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

</tx:attributes>

</tx:advice>

<!--3.3配置事务切面

Content Model : (pointcut\*, advisor\*, aspect\*)

按类匹配:(粗粒度)

within(包名.类名)

按方法和参数匹配:(细粒度)

execution(返回值类型 包名.类名.方法名(参数列表))

AOP:

在不影响源码的条件下,对方法做了扩展.

通知类型:

1.前置通知

2.后置通知

3.异常通知

4.最终通知

5.环绕通知 真正关于目标方法执行

-->

<aop:config>

<aop:pointcut expression="execution(\* com.jt.manage.service..\*.\*(..))" id="pc"/>

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

</aop:config>

</beans>

### 配置mybatis

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

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<!--配置顺序

Content Model : (properties?, settings?, typeAliases?, typeHandlers?, objectFactory?,

objectWrapperFactory?, reflectorFactory?, plugins?, environments?, databaseIdProvider?, mappers?)

-->

<settings>

<!-- 开启驼峰自动映射

字段:

user\_id

user\_name

对象:

userId

userName

优点:

可以将数据库中的字段,按照驼峰规则自动

映射到POJO对象的属性中.自动的将"\_"去掉

之后首字母大写.

注意事项:

引入驼峰映射规则后,必须按照规则使用,否则

程序运行有问题.

-->

<setting name=*"mapUnderscoreToCamelCase"* value=*"true"* />

</settings>

</configuration>

### Spring整合mybatis

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

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

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

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

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

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

xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd

http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd">

<!--实例化SqlSessionFactoryBean -->

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

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

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

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

<property name="typeAliasesPackage" value="com.jt.manage.pojo"/>

</bean>

<!--为mapper接口创建代理对象

1.JDK动态代理

要求:要求被代理者必须实现接口

2.CGlib动态代理

特点:有无接口都可以创建代理对象.但是创建出来的代理

对象是目标对象的子类.

3.spring创建代理对象的原则

如果被代理者实现了接口,则采用JDK代理.如果

没有实现接口,则采用CGLIB

特点:只有cgLib可以强制使用,强制使用JDK代理有风险.

-->

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

<property name="basePackage" value="com.jt.manage.mapper"/>

</bean>

</beans>

### Web容器加载spring配置

<init-param>

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

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

</init-param>