# 京淘框架搭建

## 框架介绍

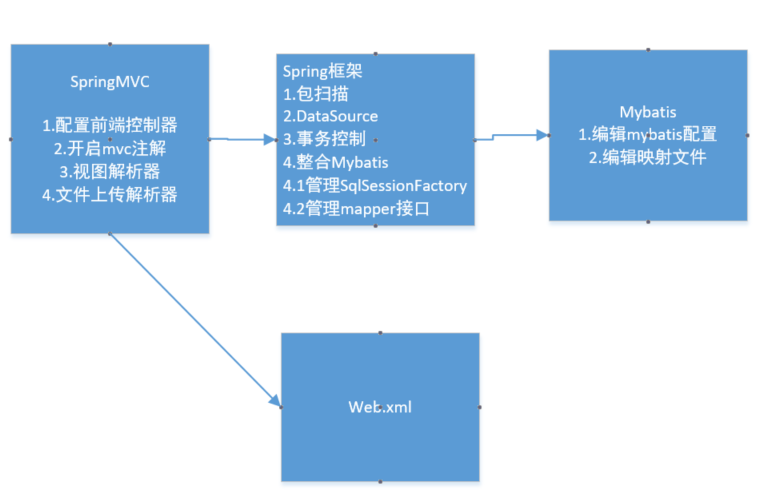
1. Spring:整合其他的主流框架
   1. IOC 将对象创建交给第三方,需要时,注入即可
   2. AOP 代理(JDK/CGlib) 在原有的方法基础上,在不修改源码前提下,对方法进行扩展.
2. SpringMVC:主要作用接收用户请求,实现数据的响应
3. Mybatis:负责数据持久层,与数据库交互(ORM)

### 框架搭建具体过程

Controller: SpringMVC

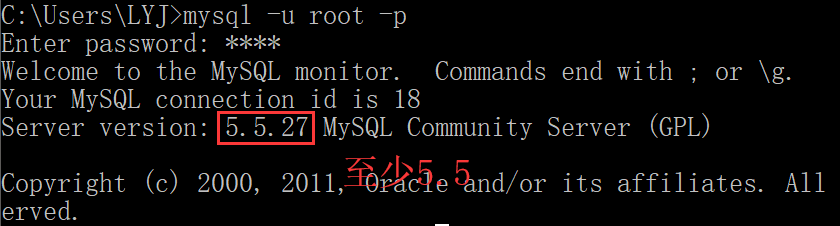
Service: Spring

Mapper: Mybatis

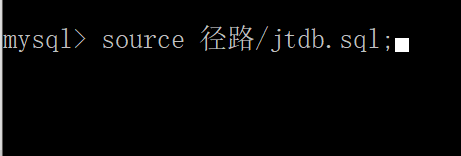


## 框架实现

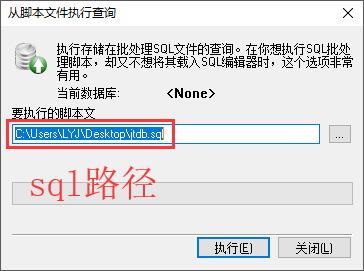
### 导入数据库



1. 通过dos命令通过执行导入命令完成.



1. 通过sql工具实现



### 实现用户信息展现

需求:用户通过localhost:8091/findAll实现用户信息的查询,要求跳转到userList.jsp页面中,通过el/jstl表达式动态获取数据并且实现展现.

### 编辑Controller

@Controller

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

@Autowired

**private** UserService userService;

/\*\*

\* ModelAndView andView = new ModelAndView();

andView.setViewName("userList");

andView.addObject("userList", userList);

return andView;

\* **@return**

\*/

//实现测试数据查询

@RequestMapping("/findAll")

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

List<User> userList = userService.findAll();

//默认使用request域

model.addAttribute("userList", userList);

**return** "userList";//返回页面逻辑名称

}

}

### 编辑Service

@Service

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

@Autowired

**private** UserMapper userMapper;

@Override

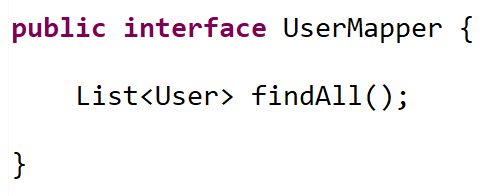
**public** List<User> findAll() {

**return** userMapper.findAll();

}

}

### 编辑Mapper



<?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">

<!-- namespace:

1.必须与接口的路径匹配

2.名称必须唯一,否则程序启动报错

3.sql语句必须和接口方法保持一致 -->

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

<!--必须配置别名,否则报错

ctrl + shift + y 小写

ctrl + shift + x 大写

-->

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

select \* from user

</select>

</mapper>

## 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>

<!--1.配置前端控制器

如果没有自定义配置文件,则前端控制器会加载一个默认的

xml进行项目的启动

默认配置文件写法 springmvc-servlet.xml

-->

<servlet>

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

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

<!--加载自定义的配置文件 -->

<init-param>

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

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

</init-param>

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

</servlet>

<!--

1. \*.do 表示请求路径以.do结尾,进行拦截

2. / 拦截所有请求和静态资源文件

-->

<servlet-mapping>

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

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

</servlet-mapping>

<!--2.配置POST乱码过滤器 -->

<filter>

<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>

<welcome-file-list>

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

</welcome-file-list>

</web-app>

### 编辑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:context=*"http://www.springframework.org/schema/context"*

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

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

xsi:schemaLocation=*"*

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

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

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

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

<!--1.开启MVC注解 -->

<mvc:annotation-driven/>

<!--2.放行静态资源配置 -->

<mvc:default-servlet-handler/>

<!--3.配置视图解析器 内部资源视图解析器 -->

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

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

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

</bean>

</beans>

### 编辑Spring.xml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.配置包扫描 -->

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

<!--2-1 加载pro配置文件 -->

<context:property-placeholder

location="classpath:/properties/\*.properties"/>

<!--2.2 配置druid数据源

#{集合Id.key} 先从容器中获取集合,再从集合取值

${key} 从容器中直接取值

-->

<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>

<!--3.1 事务注解写法 只要是更新操作,添加注解-->

<tx:annotation-driven/>

<!--3.2 声明式事务处理的写法 需要提前定义事务控制策略

以后写方法不需要添加任何注解.

1.配置事务管理器 真正的实现事务提交/回滚

2.定义事务通知 定义事务策略 哪些方法需要事务

3.定义事务切面 通知 + 切入点表达式

-->

<!--3.2.1 定时事务管理器 -->

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

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

</bean>

<!--3.2.2 定义事务通知

propagation="REQUIRED" 表示必须添加事务 默认值

propagation="SUPPORTS" 表示事务支持

方法默认不添加事务.

如果该方法执行时之前已经有事务,则合并事务执行.

propagation="REQUIRES\_NEW"

每次都会新创建事务

-->

<tx:advice id="txAdvice">

<tx:attributes>

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

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

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

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

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

</tx:attributes>

</tx:advice>

<!--3.2.3配置切面

AOP中默认的代理策略

如果目标对象有接口 默认采用JDK方式创建代理对象

要求:必须有接口

如果目标对象没有接口 则采用CGLIB形式创建代理对象

特点:创建出来的代理对象,都是目标对象的子类

proxy-target-class="true" 强制使用cglib

粗粒度控制: within(包名.类名) 按类匹配

细粒度控制: 方法参数级别

execution(返回值类型 包名.类名.方法名(参数列表))

-->

<aop:config>

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

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

</aop:config>

</beans>

## 编辑Spring整合mybatis

### 编辑mybatis配置文件

<?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>

<!--ORM对象关系映射

表: tb\_user 对象 user

user\_id userId

user\_name userName

如果启动驼峰映射规则,则可以实现自动的映射,

具体过程:

结果集user\_id去除中间的"\_"并且后边首字母大小

变为userId,之后为user对象指定的属性赋值

注意事项:

如果采用驼峰映射,则必须满足要求.

字段或属性不能随意修改

-->

<settings>

<!-- 开启驼峰自动映射 -->

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

</settings>

</configuration>

### 编辑spring整合mybatis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.管理SqlSessionFactoryBean -->

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

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

<property name="configLocation"

value="classpath:/mybatis/mybatis-config.xml"/>

<property name="mapperLocations"

value="classpath:/mybatis/mapper/\*.xml"/>

<!--定义别名包 -->

<property name="typeAliasesPackage"

value="com.jt.manage.pojo"/>

</bean>

<!--2.管理Mapper接口为其创建代理对象 -->

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

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

</bean>

</beans>