Java EE 第五次实习 Spring

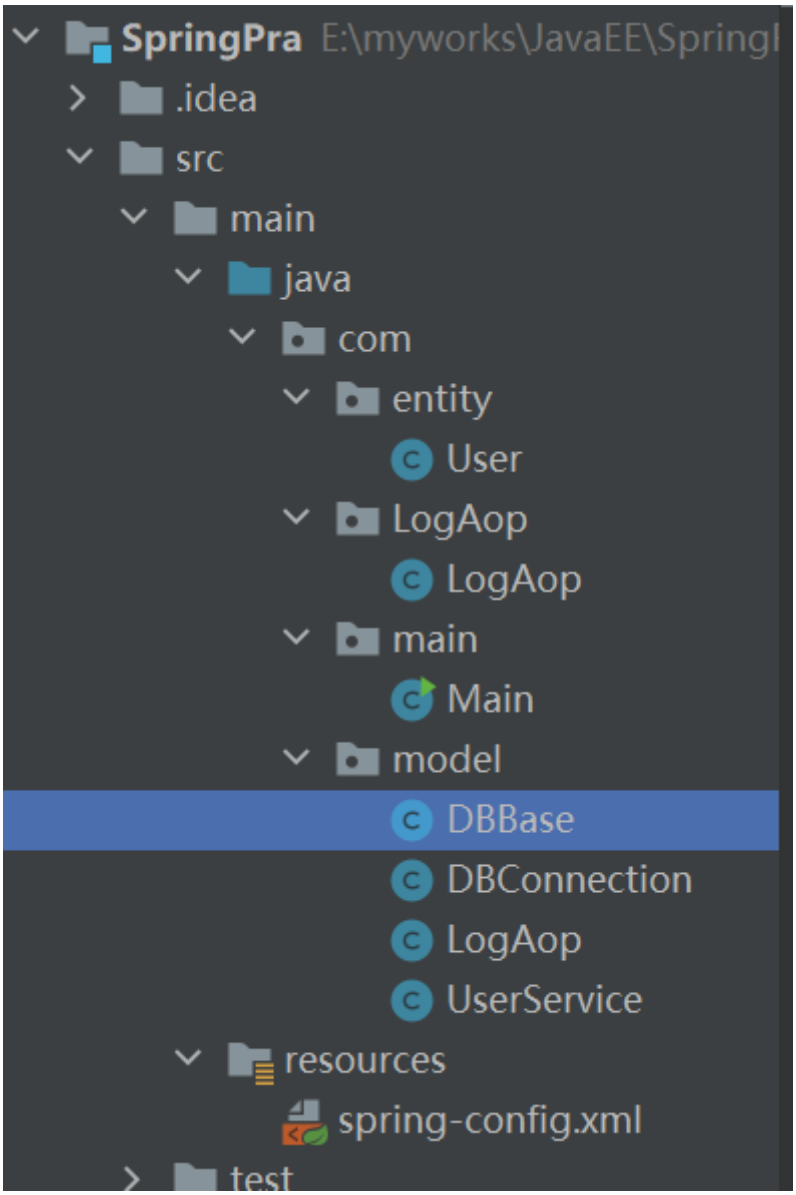
课程	姓名	学号	班级
Java EE	张宇晨	2020012249	软工2001班

开发环境与工程结构

Intellij IDEA 2020.2


IJ
IntelliJ IDEA
Ultimate 2020.2
IntelliJ IDEA 2020.2.4 (Ultimate Edition)
Build #IU-202.8194.7, built on November 24, 2020
Licensed to 宇晨 张
Subscription is active until October 14, 2023
For educational use only.
Runtime version: 11.0.9+11-b944.49 amd64
VM: OpenJDK 64-Bit Server VM by JetBrains s.r.o.
Powered by open-source software
JET BRAINS
Copyright © 2000–2022 JetBrains s.r.o.

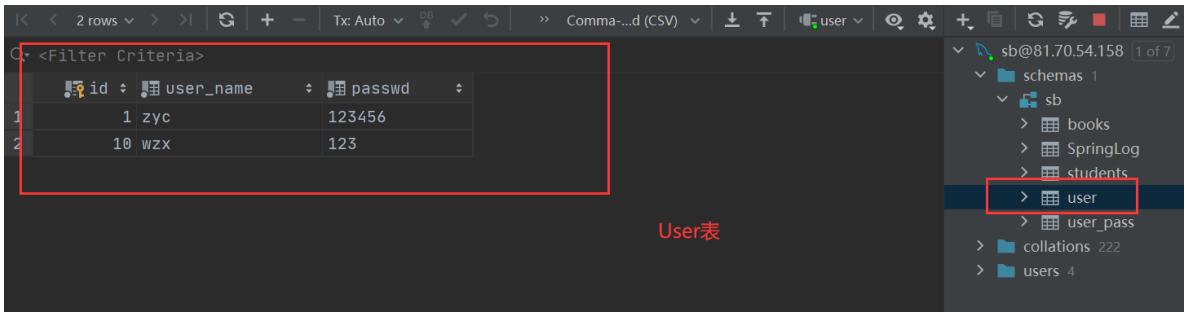
工程结构


SpringPra E:\myworks\JavaEE\SpringI
.idea
src
main
java
com
entity
User
LogAop
LogAop
main
Main
model
DBBase
DBConnection
LogAop
UserService
resources
spring-config.xml
test

实现内容

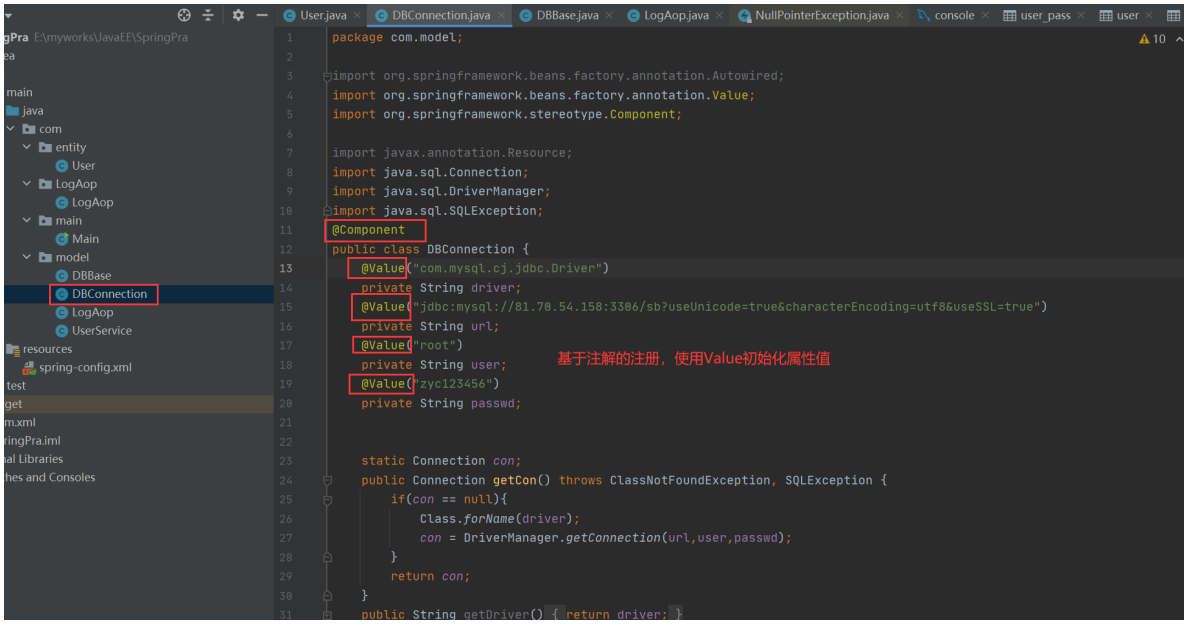
1. IOC

1）user表

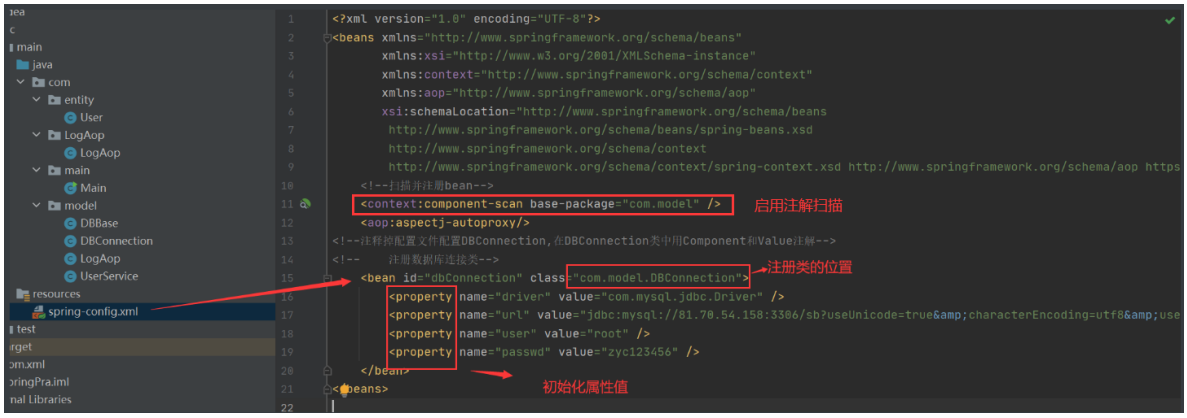


2）注册数据库连接类

基于注解注册



基于配置文件注册



3）注册DBBase和UserService

```java
UserService.java
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

@Service
public class UserService {
    @Resource
    DBBase dbBase;
    static String SELECTSQL = "select * from user";

    public List<User> list() throws SQLException, C
    {
        ResultSet rs = dbBase.selectAll(SELECTSQL);
        List<User> users = new ArrayList<~>();
        while (rs.next()){
            User user = new User();
            user.setId(rs.getInt( columnLabel: "id"))
            user.setUserName(rs.getString( columnLab
            user.setPasswd(rs.getString( columnLabel:
            users.add(user);
        }
        return users;
    }
    public void deletebyid(int id) throws SQLExcept
    {
        String SQL="delete from user where id="+id+
        dbBase.exe(SQL);
    }
```

@Resource基于名称注入

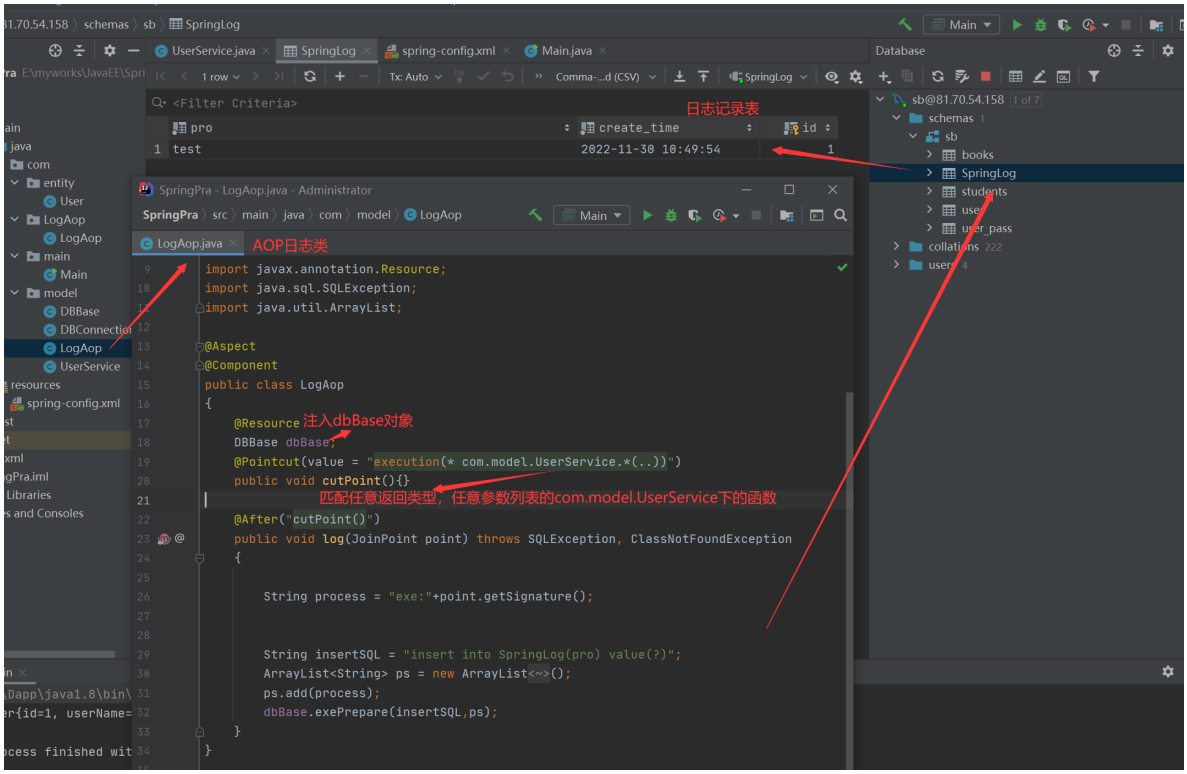
```java
DBBase.java
package com.model;

import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.sql.*;
import java.util.ArrayList;

@Component
public class DBBase {
    @Resource
    DBConnection connection;

    public ResultSet selectAll(String sql) throws SQLException, ClassNotFoundExc
    {
        Connection con = connection.getCon();
        Statement stat = con.createStatement();
        ResultSet res = stat.executeQuery(sql);
        return res;
    }
    public void exe(String sql) throws SQLException, ClassNotFoundException
    {
        Connection con = connection.getCon();
        Statement stat = con.createStatement();
        stat.execute(sql);
    }
    public void exePrepare(String sql,  ArrayList<String> params) throws SQLExce
        Connection con = connection.getCon();
        PreparedStatement pstat = con.prepareStatement(sql);
        int i=1;
        for(String p:params)
```

4)UserService的增删改功能函数

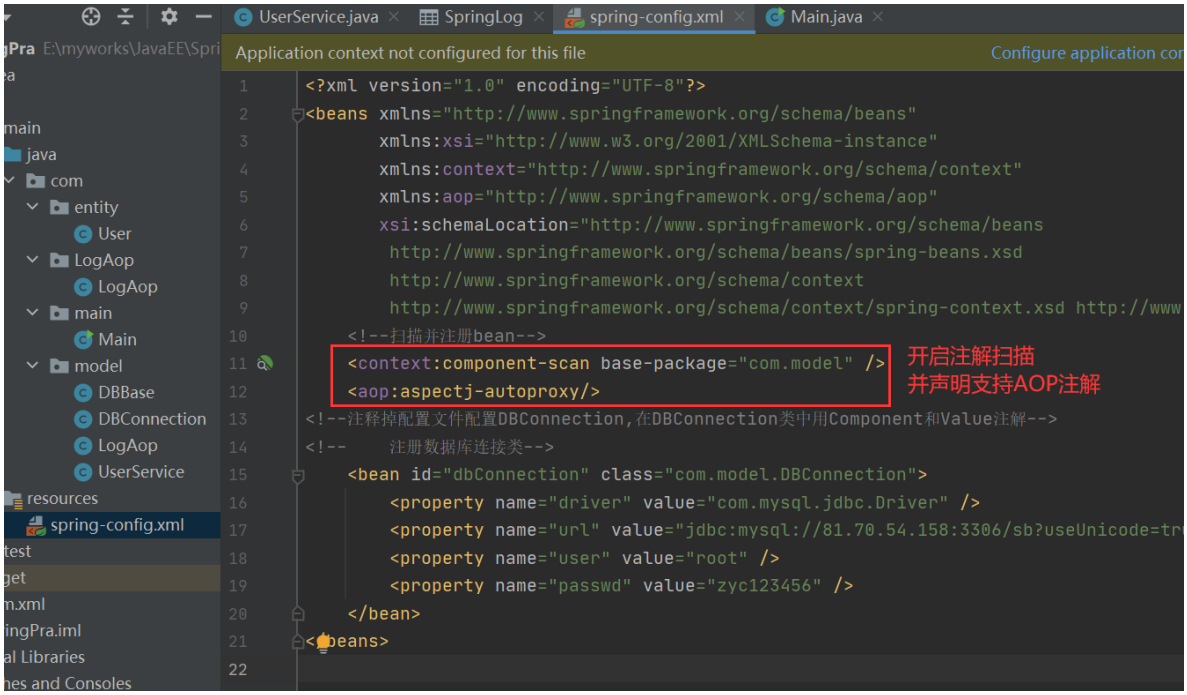
```java
UserService.java   spring-config.xml   Main.java
    public void deletebyid(int id) throws SQLException, ClassNotFoundException
    {
        String SQL="delete from user where id="+id+";";
        dbBase.exe(SQL);
    }                                                    删

    public void AddNewUser(String name, String pwd) throws Exception{
        String insertSQL = "insert into user(user_name,passwd) value(?,?)";
        ArrayList<String> ps = new ArrayList<~>();


            // ps.add("'"+request.getParameter("name")+"'");
            ps.add(name);
            ps.add(pwd);                                 增

            dbBase.exePrepare(insertSQL,ps);
        }
    public boolean AlterUser(int id,String name, String pwd) throws Exception
    {
        String insertSQL = "update usser set user_name=?,passwd=? where id=?";
        ArrayList<String> ps = new ArrayList<~>();


        // ps.add("'"+request.getParameter("name")+"'");
        ps.add(name);
        ps.add(pwd);                                         改
        ps.add(String.valueOf(id));
        dbBase.exePrepare(insertSQL,ps);
        return true;
    }
}
```

2. AOP

1）日志表与日志类



2）配置文件



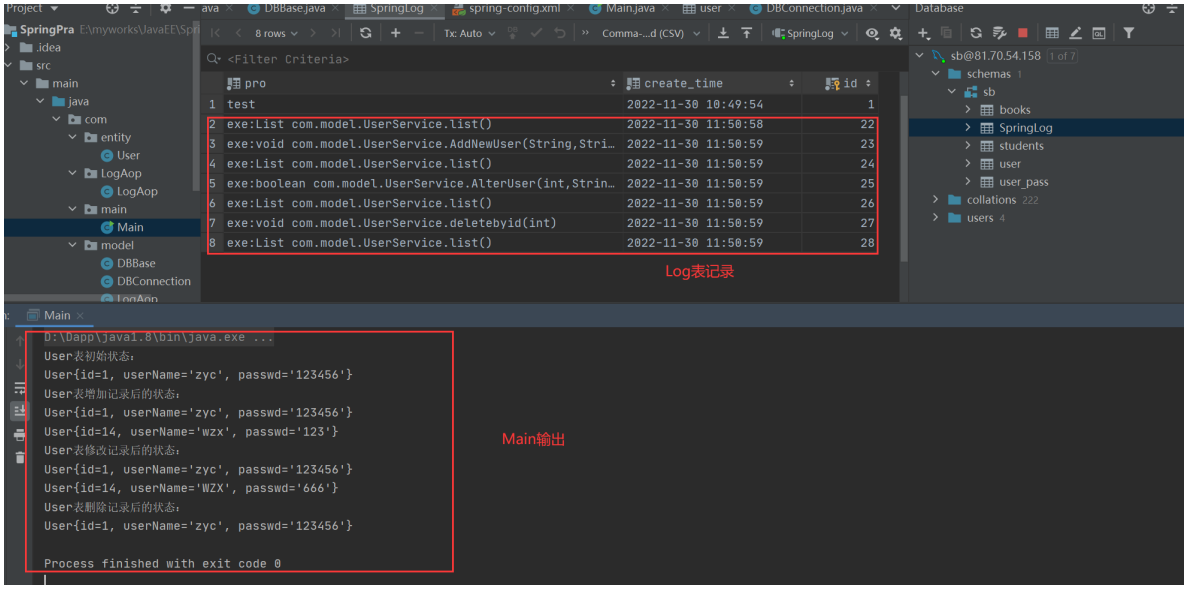
3. 测试

1）测试类Main

```
package com.main;

import com.entity.User;
import com.model.DBBase;
import com.model.DBConnection;
import com.model.UserService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```java
import java.sql.SQLException;
import java.util.List;

public class Main {
    public static void main(String[] argv) throws Exception {
        ApplicationContext context = new ClassPathXmlApplicationContext("spring-
config.xml");
        UserService userService = context.getBean(UserService.class);

//        DBConnection dc = context.getBean(DBConnection.class);
        System.out.println("User表初始状态：");
        List<User> test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
        userService.AddNewUser("wzx","123");
        System.out.println("User表增加记录后的状态：");
        test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
        userService.AlterUser(14,"WZX","666");
        System.out.println("User表修改记录后的状态：");
        test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
        userService.deletebyid(14);
        System.out.println("User表删除记录后的状态：");
        test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
    }
}
```

2) 测试结果



4. 分析——注解与配置文件优缺点

1)XML

优点：

1. xml是集中式的元数据，不需要和代码绑定
2. 使用xml配置可以让软件更具有扩展性
3. 基于xml配置的时候，只需要修改xml即可，不需要对现有的程序进行修改。
4. 容易与其他系统进行数据交互，数据共享方便

缺点：

1. 应用程序中如果使用了xml配置，需要解析xml的工具或者是是第三方类库的支持
2. 在程序编译期间无法对其配置项的正确性进行验证，只能在运行期发现
3. 开发的时候，既要维护代码又要维护配置文件，使得开发的效率降低

2)注解

优点：

1. 注解的解析可以不依赖于第三方库，可以之间使用Java自带的反射
2. 注解和代码在一起的，之间在类上，降低了维护两个地方的成本
3. 注解如果有问题，在编译期间，就可以验证正确性，如果出错更容易找

缺点：

1. 修改的话比较麻烦。如果需要对注解进行修改的话，就需要对整个项目重新编译
2. 注解功能没有xml配置齐全
3. 处理业务类之间的复杂关系，不如xml容易修改，也不及xml明了
4. 在程序中注解太多会导致代码不够简洁，影响代码质量

总之，在开发中一般是注解与XML配置结合使用，对经常修改属性值的类用XML注册更便于维护，不经常修改的类用注解注册提高开发效率。

代码附录

com.entity.user

```
package com.entity;
```

```java
public class User {
    private int id;
    private String userName;
    private String passwd;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public void setPasswd(String passwd) {
        this.passwd = passwd;
    }

    public String getUserName() {
        return userName;
    }

    public String getPasswd() {
        return passwd;
    }
    @Override
    public String toString() {
        return "User{" +
                "id=" + id +
                ", userName='" + userName + '\'' +
                ", passwd='" + passwd + '\'' +
                '}';
    }
}
```

com.LogAop.LogAop

```java
package com.LogAop;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;

@Aspect
@Component
public class LogAop {
    @Pointcut(value = "execution(* com.model.*Service.*(..))")
    public void cutPoint(){}

    @After(value = "cutPoint()")
    public void log(JoinPoint point){
```

```
        System.out.println(point.getSignature() + "方法被执行.....");
    }
}
```

com.main.Main

```
package com.main;

import com.entity.User;
import com.model.DBBase;
import com.model.DBConnection;
import com.model.UserService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import java.sql.SQLException;
import java.util.List;

public class Main {
    public static void main(String[] argv) throws Exception {
        ApplicationContext context = new ClassPathXmlApplicationContext("spring-
config.xml");
        UserService userService = context.getBean(UserService.class);

//        DBConnection dc = context.getBean(DBConnection.class);
        System.out.println("User表初始状态：");
        List<User> test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
        userService.AddNewUser("wzx","123");
        System.out.println("User表增加记录后的状态：");
        test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
        userService.AlterUser(14,"WZX","666");
        System.out.println("User表修改记录后的状态：");
        test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
        userService.deletebyid(14);
        System.out.println("User表删除记录后的状态：");
        test = userService.list();
        for (int i = 0;i<test.size();++i)
        {
            System.out.println(test.get(i));
        }
    }
}
```

com.model.DBDase

```java
package com.model;

import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.sql.*;
import java.util.ArrayList;

@Component
public class DBBase {
    @Resource
    DBConnection connection;

    public ResultSet selectAll(String sql) throws SQLException,
ClassNotFoundException
    {
        Connection con = connection.getCon();
        Statement stat = con.createStatement();
        ResultSet res = stat.executeQuery(sql);
        return res;
    }
    public void exe(String sql) throws SQLException, ClassNotFoundException
    {
        Connection con = connection.getCon();
        Statement stat = con.createStatement();
        stat.execute(sql);
    }
    public void exePrepare(String sql,  ArrayList<String> params) throws
SQLException, ClassNotFoundException {
        Connection con = connection.getCon();
        PreparedStatement pstat = con.prepareStatement(sql);
        int i=1;
        for(String p:params)
        {
            pstat.setString(i++,p);
        }
        pstat.execute();
        pstat.close();
    }
}
```

com.model.DBConnection

```java
package com.model;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
@Component
public class DBConnection {
    @Value("com.mysql.cj.jdbc.Driver")
    private String driver;
```

```
    @Value("jdbc:mysql://81.70.54.158:3306/sb?
useUnicode=true&characterEncoding=utf8&useSSL=true")
    private String url;
    @Value("root")
    private String user;
    @Value("zyc123456")
    private String passwd;


    static Connection con;
    public Connection getCon() throws ClassNotFoundException, SQLException {
        if(con == null){
            Class.forName(driver);
            con = DriverManager.getConnection(url,user,passwd);
        }
        return con;
    }
    public String getDriver() {
        return driver;
    }
    public void setDriver(String driver) {
        this.driver = driver;
    }
    public String getUrl() {
        return url;
    }
    public void setUrl(String url) {
        this.url = url;
    }
    public String getUser() {
        return user;
    }

    public void setUser(String user) {
        this.user = user;
    }

    public String getPasswd() {
        return passwd;
    }

    public void setPasswd(String passwd) {
        this.passwd = passwd;
    }
}
```

com.model.LogAop

```
package com.model;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;

import javax.annotation.Resource;
```

```
import java.sql.SQLException;
import java.util.ArrayList;

@Aspect
@Component
public class LogAop
{
    @Resource
    DBBase dbBase;
    @Pointcut(value = "execution(* com.model.UserService.*(..))")
    public void cutPoint(){}

    @After("cutPoint()")
    public void log(JoinPoint point) throws SQLException, ClassNotFoundException
    {

        String process = "exe:"+point.getSignature();


        String insertSQL = "insert into SpringLog(pro) value(?)";
        ArrayList<String> ps = new ArrayList<String>();
        ps.add(process);
        dbBase.exePrepare(insertSQL,ps);
    }
}
```

com.UserService

```
package com.model;

import com.entity.User;
import org.springframework.stereotype.Service;

import javax.annotation.Resource;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;

@Service
public class UserService {
    @Resource
    DBBase dbBase;
    static String SELECTSQL = "select * from user";

    public List<User> list() throws SQLException, ClassNotFoundException
    {
        ResultSet rs = dbBase.selectAll(SELECTSQL);
        List<User> users = new ArrayList<User>();
        while (rs.next()){
            User user = new User();
            user.setId(rs.getInt("id"));
            user.setUserName(rs.getString("user_name"));
            user.setPasswd(rs.getString("passwd"));
            users.add(user);
        }
```

```
        return users;
    }
    public void deletebyid(int id) throws SQLException, ClassNotFoundException
    {
        String SQL="delete from user where id="+id+";";
        dbBase.exe(SQL);
    }

    public void AddNewUser(String name, String pwd) throws Exception{
        String insertSQL = "insert into user(user_name,passwd) value(?,?)";
        ArrayList<String> ps = new ArrayList<String>();


            // ps.add("'"+request.getParameter("name")+"'");
            ps.add(name);
            ps.add(pwd);

            dbBase.exePrepare(insertSQL,ps);
        }
    public boolean AlterUser(int id,String name, String pwd) throws Exception
    {
        String insertSQL = "update user set user_name=?,passwd=? where id=?";
        ArrayList<String> ps = new ArrayList<String>();


        // ps.add("'"+request.getParameter("name")+"'");
        ps.add(name);
        ps.add(pwd);
        ps.add(String.valueOf(id));
        dbBase.exePrepare(insertSQL,ps);
        return true;
    }
}
```

spring-config.xml

```
<?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"
       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
https://www.springframework.org/schema/aop/spring-aop.xsd">
    <!--扫描并注册bean-->
    <context:component-scan base-package="com.model" />
    <aop:aspectj-autoproxy/>
<!--注释掉配置文件配置DBConnection,在DBConnection类中用Component和Value注解-->
<!--    注册数据库连接类-->
<!--    <bean id="dbConnection" class="com.model.DBConnection">-->
<!--        <property name="driver" value="com.mysql.jdbc.Driver" />-->
<!--        <property name="url" value="jdbc:mysql://81.70.54.158:3306/sb?
useUnicode=true&amp;characterEncoding=utf8&amp;useSSL=true" />-->
<!--        <property name="user" value="root" />-->
```

```
<!--        <property name="passwd" value="zyc123456" />-->
<!--    </bean>-->
</beans>
```

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>org.example</groupId>
    <artifactId>Spring</artifactId>
    <version>1.0-SNAPSHOT</version>
    <!--工程打包类型为 war 包-->
    <packaging>jar</packaging>
    <repositories>
        <!--配置仓库地址-->
        <repository>
            <id>central</id>
            <name>aliyun maven</name><!--配置为国内仓库-阿里云-->
            <url>http://maven.aliyun.com/nexus/content/groups/public/</url>
            <layout>default</layout>
        </repository>
    </repositories>

    <dependencies>
        <!--Java web 依赖包-->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.25</version>
        </dependency>
        <dependency>
            <groupId>jstl</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
        </dependency>
        <dependency>
            <groupId>taglibs</groupId>
            <artifactId>standard</artifactId>
            <version>1.1.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.1</version>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework</groupId>
            <artifactId>spring-aspects</artifactId>
            <version>5.3.1</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <!--            <plugin>-->
            <!--                <groupId>org.eclipse.jetty</groupId>-->
            <!--                <artifactId>jetty-maven-plugin</artifactId>-->
            <!--                <version>9.3.0.v20150612</version>-->
            <!--                <configuration>-->
            <!--                    <webAppConfig>-->
            <!--                        <contextPath>/</contextPath>&lt;!&ndash;
上下文路径&ndash;&gt;-->
            <!--                    </webAppConfig>-->
            <!--                    <httpConnector>-->
            <!--                        <port>80</port>-->
            <!--                    </httpConnector>-->
            <!--                </configuration>-->
            <!--            </plugin>-->
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <version>2.2</version>
                <configuration>
                    <port>80</port>
                    <path>/StuMS</path>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```