# Spring 简介

Spring 是控制反转IOC和面向切面AOP的框架.

IoC:inversion of Control 控制反转 创建对象的权利由程序转到容器

DI:Dependency Injection 依赖注入

AOP:Aspect Orientied Programming 面向切面的编程

# Spring 开发环境搭建

## Pom.xml

|  |
| --- |
| <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.1.3.RELEASE</version>  </dependency> |

## applicationContext.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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  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/tx*  *http://www.springframework.org/schema/tx/spring-tx.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd"*  >  <bean id=*"userService"* class=*"com.tjetc.service.UserService"*></bean>  </beans> |

## UserService

|  |
| --- |
| **public** **class** UserService {    **public** **void** add() {  System.***out***.println("UserService.add()...");  } |

## 测试

|  |
| --- |
| @Test  **public** **void** testAdd(){  ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext("applicationContext.xml");  UserService userService = (UserService) context.getBean("userService");  userService.add();  } |

# Spring IOC与AOP

# IoC概述及作用

## 什么是IoC

Inversion of Control:控制权的转移,创建对象的权利由应用程序转移到容器称为控制反转

## IoC的作用

削减计算机程序的耦合(解除我们代码中的依赖关系）

# Spring基于Annotation的IoC配置

Annotation:注解

|  |
| --- |
| *<!--配置组件的扫描com.tjetc本包及其子孙包下的所有的在类上标注有@Controller,@Service,@Repository,@Component注解的类,*  *spring会把标注了这些注解的类当做你配置bean节点一样纳入spring容器管理-->*  <**context:component-scan base-package="com.tjetc"**></**context:component-scan**> |
| **package** com.tjetc.service;  **import** com.tjetc.dao.UserDao;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.stereotype.Service;  @Service  **public class** UserService {  @Autowired  **private** UserDao **userDao**;  **public void** setUserDao(UserDao userDao) {  **this**.**userDao** = userDao;  }  **public** UserService() {  System.***out***.println(**"UserService()"**);  }  **public void** add(){  System.***out***.println(**"UserService.add()"**);  **userDao**.add();  }  } |
| **package** com.tjetc.dao;  **import** org.springframework.stereotype.Repository;  @Repository  **public class** UserDao {  **public void** add(){  System.***out***.println(**"userDao.add()"**);  }  } |
| **public static void** main(String[] args) {  ClassPathXmlApplicationContext context = **new** ClassPathXmlApplicationContext(**"applicationContext.xml"**);  UserService userService = context.getBean(UserService.**class**);  userService.add();  } |
| UserService()  UserService.add()  userDao.add() |

# AOP

* AOP全名Aspect-oriented programming面向切面编程 AOP实现的关键在于AOP框架自动创建的AOP代理

# Spring事务管理

## 事务的概述

**概念:事务首先是一系列操作组成的工作单元，该工作单元内的操作是不可分割的，即要么所有操作都做，要么所有操作都不做，这就是事务。**

### ACID

事务必需满足ACID（原子性、一致性、隔离性和持久性）特性，缺一不可：

原子性（Atomicity）：即事务是不可分割的最小工作单元，事务内的操作要么全做，要么全不做；

一致性（Consistency）：在事务执行前数据库的数据处于正确的状态，而事务执行完成后数据库的数据还是处于正确的状态，即数据完整性约束没有被破坏；如银行转帐，A转帐给B，必须保证A的钱一定转给B，一定不会出现A的钱转了但B没收到，否则数据库的数据就处于不一致（不正确）的状态。

隔离性（Isolation）：并发事务执行之间无影响，在一个事务内部的操作对其他事务是不产生影响，这需要事务隔离级别来指定隔离性；

持久性（Durability）：事务一旦执行成功，它对数据库的数据的改变必须是永久的，不会因比如遇到系统故障或断电造成数据不一致或丢失。

## 事务管理

### 编程式事务管理

1、 编程式事务，需要代码侵入 ，使用TransactionTemplate 模板类 （企业不用）

transactionTemplate.execute(new TransactionCallbackWithoutResult(){

T doInTransaction(){

// 数据库操作代码

}

});

doInTransaction方法只要发生异常，事务回滚，没有异常，事务提交

### 声明式事务管理

声明式事务管理 ，无需要修改原来代码，只需要配置，为目标代码添加事务管理 ， AOP底层实现 --- 企业推荐

* **基于tx/aop命名空间的配置:**

事务相关配置：

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

<tx:attributes>

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

<tx:method name="\*" propagation="REQUIRED" isolation="READ\_COMMITTED" read-only="true"/>

</tx:attributes>

</tx:advice>

<aop:config>

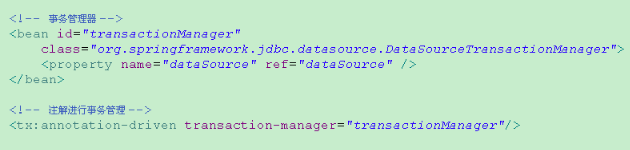
<aop:pointcut id="serviceMethod" expression="execution(\* cn..chapter9.service..\*.\*(..))"/>

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

</aop:config>

* **使用注解实现声明式事务管理:**

第一步： 在配置文件中， 激活使用注解进行事务管理



第二步： 在需要管理事务的业务类（业务方法）上添加@Transactional 注解

# Spring 整合Struts2、Hibernate实现用户注册删改查功能

基于SSH整合实现用户表的注册删改查设计实现

## 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.tjetc</groupId>  <artifactId>day0620spring</artifactId>  <version>1.0-SNAPSHOT</version>  <packaging>war</packaging>  <dependencies>  <dependency>  <groupId>org.apache.struts</groupId>  <artifactId>struts2-core</artifactId>  <version>2.5.2</version>  </dependency>  <dependency>  <groupId>org.apache.struts</groupId>  <artifactId>struts2-spring-plugin</artifactId>  <version>2.5.2</version>  </dependency>  <!-- 添加Hibernate依赖 -->  <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-core</artifactId>  <version>4.1.1.Final</version>  </dependency>  <!-- mysql数据库的驱动包 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>8.0.30</version>  </dependency>  <dependency>  <groupId>jstl</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>4.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>4.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-orm</artifactId>  <version>4.1.3.RELEASE</version>  </dependency>  <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjweaver</artifactId>  <version>1.6.12</version>  </dependency>  <dependency>  <groupId>commons-dbcp</groupId>  <artifactId>commons-dbcp</artifactId>  <version>1.2.2</version>  </dependency>  <dependency>  <groupId>c3p0</groupId>  <artifactId>c3p0</artifactId>  <version>0.9.1.2</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>4.0.1</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.13.2</version>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.tomcat.maven</groupId>  <artifactId>tomcat7-maven-plugin</artifactId>  <version>2.2</version>  <configuration>  <port>8080</port>  <path>/</path>  </configuration>  </plugin>  </plugins>  </build>  </project> |

## applicationContext.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:tx="http://www.springframework.org/schema/tx"  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/tx http://www.springframework.org/schema/tx/spring-tx.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">  <context:component-scan base-package="com" />  <!-- 配置数据连接 -->  <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">  <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />  <property name="url" value="jdbc:mysql://localhost:3306/yd?useUnicode=true&amp;characterEncoding=utf8&amp;useSSL=false&amp;allowPublicKeyRetrieval=true&amp;serverTimezone=Asia/Shanghai"></property>  <property name="username" value="root"></property>  <property name="password" value="123456"></property>  <property name="initialSize" value="1" />  <property name="maxActive" value="1" />  <property name="maxIdle" value="1" />  <property name="maxWait" value="500" />  <property name="defaultAutoCommit" value="false" />  </bean>  <bean id="transactionManager"  class="org.springframework.orm.hibernate4.HibernateTransactionManager">  <property name="sessionFactory" ref="sessionFactory" />  </bean>  <bean id="sessionFactory"  class="org.springframework.orm.hibernate4.LocalSessionFactoryBean">  <property name="dataSource" ref="dataSource" />  <property name="packagesToScan">  <list>  <!-- annotaion方式 -->  <value>com.\*\*.domain</value>  </list>  </property>  <property name="hibernateProperties">  <props>  <!-- 指定Hibernate的连接方言 -->  <prop key="hibernate.dialect">  org.hibernate.dialect.MySQLDialect  </prop>  <prop key="hibernate.show\_sql">  true  </prop>  <prop key="hibernate.hbm2ddl.auto">  update  </prop>  </props>  </property>  </bean>  <!-- 定义事务通知 -->  <tx:advice id="txAdvice" transaction-manager="transactionManager">  <!-- 定义方法的过滤规则 -->  <tx:attributes>  <!-- 定义所有以add,update,del开头的方法都使用事务 -->  <tx:method name="add\*" propagation="REQUIRED" />  <tx:method name="update\*" propagation="REQUIRED" />  <tx:method name="del\*" propagation="REQUIRED" />  <!-- 定义所有的方法是只读的,除了上面定义使用事务的除外 -->  <tx:method name="\*" read-only="true" />  </tx:attributes>  </tx:advice>  <!-- 定义AOP配置 -->  <aop:config>  <!-- 定义一个切入点 -->  <aop:pointcut id="targetMethod" expression="execution(\* com..service.\*Service.\*(..))" />  <!-- 对切入点和事务的通知,进行适配 -->  <aop:advisor advice-ref="txAdvice" pointcut-ref="targetMethod" />  </aop:config>  </beans> |

## struts.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"  "http://struts.apache.org/dtds/struts-2.5.dtd">  <struts>  <package name="my" namespace="/" extends="struts-default">  <global-allowed-methods>regex:.\*</global-allowed-methods>  <action name="user\_\*" class="userAction" method="{1}">  <result name="toList" type="redirect">user\_list</result>  <result name="list">/list.jsp</result>  <result name="add">/add.jsp</result>  <result name="update">/update.jsp</result>  </action>  </package>  </struts> |

## 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" version="2.5">  <display-name>0627ssh</display-name>  <welcome-file-list>  <welcome-file>index.html</welcome-file>  <welcome-file>index.htm</welcome-file>  <welcome-file>index.jsp</welcome-file>  <welcome-file>default.html</welcome-file>  <welcome-file>default.htm</welcome-file>  <welcome-file>default.jsp</welcome-file>  </welcome-file-list>  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <filter>  <filter-name>encoding</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>encoding</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <filter>  <filter-name>hibernateFilter</filter-name>  <filter-class>  org.springframework.orm.hibernate4.support.OpenSessionInViewFilter  </filter-class>  </filter>  <filter-mapping>  <filter-name>hibernateFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <filter>  <filter-name>struts2</filter-name>  <filter-class>  org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter  </filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  </web-app> |

## UserAction

|  |
| --- |
| @Controller  @Scope("prototype")  public class UserAction implements ModelDriven<User>{  @Autowired  private UserService userService;  private User user=new User();  public User getUser() {  return user;  }  public void setUser(User user) {  this.user = user;  }    @Override  public User getModel() {  return user;  }  public String list(){  String username=user.getUsername();  username=username==null?"":username;  List<User> list=userService.list(username);  ServletActionContext.getRequest().setAttribute("list", list);  return "list";  }  public String findById(){  user=userService.findById(user.getId());  return "update";  }  public String update(){  System.out.println("UserAction.update([])方法:"+user);  boolean b=userService.update(user);  return "toList";  }  public String add(){  System.out.println("UserAction.add([])方法:"+user);  boolean b=userService.add(user);  return "toList";  }  public String del(){  System.out.println("UserAction.del([])方法:"+user);  boolean b=userService.del(user);  return "toList";  }  } |

## UserService

|  |
| --- |
| public interface UserService {  boolean add(User user);  User findByUsername(String username);  List<User> list(String username);  User findById(int id);  boolean update(User user);  boolean del(User user);  } |

## UserServiceImpl

|  |
| --- |
| @Service  public class UserServiceImpl implements UserService {  @Autowired  private UserDao userDao;  @Override  public boolean add(User user) {  return userDao.add(user);  }  @Override  public User findByUsername(String username) {  return userDao.findByUsername(username);  }  @Override  public List<User> list(String username) {  return userDao.list(username);  }  @Override  public User findById(int id) {  return userDao.findById(id);  }  @Override  public boolean update(User user) {  return userDao.update(user);  }  @Override  public boolean del(User user) {  return userDao.del(user);  }  } |

## UserDao

|  |
| --- |
| public interface UserDao {  boolean add(User user);  User findByUsername(String username);  List<User> list(String username);  User findById(int id);  boolean update(User user);  boolean del(User user);  } |

## UserDaoImpl

|  |
| --- |
| @Repository  public class UserDaoImpl implements com.tjetc.dao.UserDao {  @Autowired  private SessionFactory sessionFactory;  @Override  public boolean add(User user) {  return sessionFactory.getCurrentSession().save(user)!=null;  }  @Override  public User findByUsername(String username) {    return (User) sessionFactory.getCurrentSession().createQuery("from User u where u.username=?").setString(0, username).uniqueResult();  }  @Override  public List<User> list(String username) {    return sessionFactory.getCurrentSession().createQuery("from User u where u.username like ?").setString(0,"%"+username+"%").list();  }  @Override  public User findById(int id) {  return (User) sessionFactory.getCurrentSession().createQuery("from User u where u.id=?").setInteger(0, id).uniqueResult();  }  @Override  public boolean update(User user) {  try {  sessionFactory.getCurrentSession().update(user);  return true;  } catch (HibernateException e) {  e.printStackTrace();  }  return false;  }  @Override  public boolean del(User user) {  try {  sessionFactory.getCurrentSession().delete(user);  return true;  } catch (HibernateException e) {  e.printStackTrace();  }  return false;  }  } |

## User

|  |
| --- |
| @Entity  public class User {  @Id  @GeneratedValue  private int id;  private String username;  private String password;  @DateTimeFormat(pattern = "yyyy-MM-dd")  @Type(type = "date")  private Date birthday;  private String phone;  public Date getBirthday() {  return birthday;  }  public void setBirthday(Date birthday) {  this.birthday = birthday;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getUsername() {  return username;  }  public void setUsername(String username) {  this.username = username;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  } |

## list.jsp

|  |
| --- |
| <%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Title</title>  <base href="${pageContext.request.contextPath}/"/>  <script>  function query() {  location.href='user\_list?username='+document.getElementById("username").value  }  function add() {  location.href='add.jsp'  }  </script>  </head>  <body>  <div>  <input type="text" id="username" value="${username}">  <button onclick="query()">查询</button>  <button onclick="add()">注册</button>  </div>  <table border="1" cellspacing="0" width="70%">  <tr align="center">  <th>序号</th>  <th>用户名</th>  <th>密码</th>  <th>生日</th>  <th>手机</th>  <th>操作</th>  </tr>  <c:forEach items="${list}" var="user">  <tr align="center">  <td>${user.id}</td>  <td>${user.username}</td>  <td>${user.password}</td>  <td>${user.birthday}</td>  <td>${user.phone}</td>  <td>  <a href="user\_findById?id=${user.id}">修改</a>  <a href="user\_del?id=${user.id}">删除</a>  </td>  </tr>  </c:forEach>  </table>  </body>  </html> |

## add.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>用户注册</title>  <base href="${pageContext.request.contextPath}/"/>  </head>  <body>  <form action="user\_add" method="post">  用户名:<input type="text" name="username" /><br/>  密码:<input type="password" name="password" /><br/>  生日:<input type="date" name="birthday" /><br/>  手机:<input type="text" name="phone" /><br/>  <input type="submit" value="提交"/><br/>  </form>  </body>  </html> |

## update.jsp

|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Title</title>  <base href="${pageContext.request.contextPath}/"/>  </head>  <body>  <form action="user\_update" method="post">  <input type="hidden" name="id" value="${user.id}">  用户名:<input type="text" name="username" value="${user.username}"/><br/>  密码:<input type="password" name="password" value="${user.password}"/><br/>  生日:<input type="date" name="birthday" value="${user.birthday}"/><br/>  手机:<input type="text" name="phone" value="${user.birthday}" /><br/>  <input type="submit" value="提交"/><br/>  </form>  </body>  </html> |

