**一．Sql数据库连接，不涉及事务**

**Jdbc数据库连接，使用C3P0，会自动回收空闲的数据库连接，在大于最大连接数时，会断开最后一个企图连接的conn。**

**配置文件properties:**

**c3p0.driverClass = com.mysql.cj.jdbc.Driver**

**c3p0.jdbcUrl = jdbc:mysql://localhost:3306/db1?useSSL=false&serverTimezone=UTC**

**c3p0.user = root**

**c3p0.password = 123456**

**c3p0.initialPoolSize = 5**

**c3p0.maxPoolSize = 10**

**c3p0.checkoutTimeout = 3000**

**需要的jar包：**

1. **c3p0-0.9.5.2.jar**
2. **mchange-commons-java-0.2.12.jar**
3. **mysql-connector-java-8.0.16.jar**
4. **spring-jdbc-4.3.0.RELEASE.jar**

**其中：**

1. **Class AccountDao extends JdbcDaoSupport**
2. **Class AccountService includes AccountDao**
3. **不带控制层，直接使用业务层。**
4. **JdbcDaoSupport要配置dataSource(**ComboPooledDataSource**)**
5. **需要声明配置文件的位置，便于使用配置文件的文本内容：**

**<context:property-placeholder location="c3p0.properties"></context:property-placeholder>**

**<property name="jdbcUrl" value="${c3p0.jdbcUrl}"></property>**

**xml配置：**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="
6. http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context.xsd">
10. <context:component-scan base-**package**="cn.cxd"></context:component-scan>
12. <context:property-placeholder location="c3p0.properties"></context:property-placeholder>
14. <bean **class**="com.mchange.v2.c3p0.ComboPooledDataSource" id="dataSource">
15. <property name="driverClass" value="${c3p0.driverClass}"></property>
16. <property name="jdbcUrl" value="${c3p0.jdbcUrl}"></property>
17. <property name="user" value="${c3p0.user}"></property>
18. <property name="password" value="${c3p0.password}"></property>
19. <property name="initialPoolSize" value="${c3p0.initialPoolSize}"></property>
20. <property name="maxPoolSize" value="${c3p0.maxPoolSize}"></property>
21. <property name="checkoutTimeout" value="${c3p0.checkoutTimeout}"></property>
22. </bean>
24. <bean **class**="cn.cxd.daos.AccountDao" id="accountDao">
25. <property name="dataSource">
26. <ref bean="dataSource"></ref>
27. </property>
28. </bean>
30. <bean **class**="cn.cxd.services.AccountService">
31. <property name="accountDao">
32. <ref bean="accountDao"></ref>
33. </property>
34. </bean>
35. </beans>

**客户端：**

1. **package** test;
3. **import** cn.cxd.services.AccountService;
4. **import** org.springframework.context.ApplicationContext;
5. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
7. **public** **class** TestC3P0Jdbc {
8. **public** **static** **void** main(String[] args) {
9. ApplicationContext context = **new** ClassPathXmlApplicationContext("beans.xml");
10. AccountService service = context.getBean(AccountService.**class**);
11. **double** accountZ3 = service.checkAccount("张三");
12. **double** accountL4 = service.checkAccount("李四");
14. System.out.println("张三金额： "+accountZ3);
15. System.out.println("李四金额： "+accountL4);
16. service.transferAccount("张三","李四",100);
18. accountZ3 = service.checkAccount("张三");
19. accountL4 = service.checkAccount("李四");
20. System.out.println("张三金额： "+accountZ3);
21. System.out.println("李四金额： "+accountL4);
23. }
24. }

**二．手动事务管理：**

**需要的jar包：**

1. **aopalliance-1.0.0.jar**
2. **aspectjweaver-1.8.10.jar**
3. **c3p0-0.9.5.2.jar**
4. **commons-logging-1.2.jar**
5. **javax.annotation-api-1.3.2.jar**
6. **mchange-commons-java-0.2.12.jar**
7. **mysql-connector-java-8.0.16.jar**
8. **spring-aop-4.3.0.RELEASE.jar**
9. **spring-aspects-4.3.0.RELEASE.jar**
10. **spring-beans-4.3.0.RELEASE.jar**
11. **spring-context-4.3.0.RELEASE.jar**
12. **spring-core-4.3.0.RELEASE.jar**
13. **spring-expression-4.3.0.RELEASE.jar**
14. **spring-jdbc-4.3.0.RELEASE.jar**
15. **spring-tx-4.3.0.RELEASE.jar**

**其中：**

1. **Class AccountService01 implements IAccountService {**

**private TransactionTemplate transactionTemplate;**

**private IAccountDao accountDao;**

**……**

**}**

1. **Class AccountDao extends JdbcDaoSupport implements IAccountDao**
2. **JdbcDaoSupport要配置dataSource（**ComboPooledDataSource**）**
3. **transactionTemplate 要配置 DataSourceTransactionManager**
4. **DataSourceTransactionManager 也要配置 dataSource（**ComboPooledDataSource**）**

**Xml配置：**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="
6. http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context.xsd">
10. <context:component-scan base-**package**="cn.cxd"></context:component-scan>
12. <context:property-placeholder location="c3p0.properties"></context:property-placeholder>
14. <bean **class**="com.mchange.v2.c3p0.ComboPooledDataSource" id="dataSource">
15. <property name="driverClass" value="${c3p0.driverClass}"></property>
16. <property name="jdbcUrl" value="${c3p0.jdbcUrl}"></property>
17. <property name="user" value="${c3p0.user}"></property>
18. <property name="password" value="${c3p0.password}"></property>
19. <property name="initialPoolSize" value="${c3p0.initialPoolSize}"></property>
20. <property name="maxPoolSize" value="${c3p0.maxPoolSize}"></property>
21. <property name="checkoutTimeout" value="${c3p0.checkoutTimeout}"></property>
22. </bean>
24. <bean **class**="cn.cxd.daos.AccountDao" id="accountDao">
25. <property name="dataSource" ref="dataSource"></property>
26. </bean>
28. <bean **class**="org.springframework.jdbc.datasource.DataSourceTransactionManager" id="manager">
29. <property name="dataSource" ref="dataSource"></property>
30. </bean>
32. <bean **class**="org.springframework.transaction.support.TransactionTemplate" id="transactionTemplate">
33. <property name="transactionManager" ref="manager"></property>
34. </bean>
36. <bean **class**="cn.cxd.services.AccountService01">
37. <property name="transactionTemplate" ref="transactionTemplate"></property>
38. <property name="accountDao" ref="accountDao"></property>
39. </bean>
40. </beans>

**客户端：**

1. **package** test;
3. **import** cn.cxd.services.AccountService01;
4. **import** org.springframework.context.ApplicationContext;
5. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
7. **public** **class** TestHandTx {
8. **public** **static** **void** main(String[] args) {
9. ApplicationContext con = **new** ClassPathXmlApplicationContext("beans01.xml");
10. AccountService01 service = con.getBean(AccountService01.**class**);
12. **double** accountZ3 = service.checkAccount("张三");
13. **double** accountL4 = service.checkAccount("李四");
15. System.out.println("张三金额： "+accountZ3);
16. System.out.println("李四金额： "+accountL4);
17. service.transferAccount("张三","李四",100);
19. accountZ3 = service.checkAccount("张三");
20. accountL4 = service.checkAccount("李四");
21. System.out.println("张三金额： "+accountZ3);
22. System.out.println("李四金额： "+accountL4);
23. }
24. }

**三．使用TransactionProxyFactoryBean实现事务半自动配置**

**其中：**

**1.Class AccountDao extends JdbcDaoSupport implements IAccountDao**

**2.Class AccountService02 implements IAccountService {**

**private IAccountDao accountDao;…}**

**service不需要事务模板了..**

**3.xml配置TransactionProxyFactoryBean需要id，不能无id使用class参数直接强转，**

**是业务类的代理类..jdkProxy是默认的代理方式,客户端直接getBean这个id，向上转型为接口，当然这个事务模板需要target。**

**4.xml配置Manager需要dataSource,而TransactionProxyFactoryBean需要Manager**

**Xml配置：**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xsi:schemaLocation="
6. http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context.xsd">
10. <context:component-scan base-**package**="cn.cxd"></context:component-scan>
12. <context:property-placeholder location="c3p0.properties"></context:property-placeholder>
14. <bean **class**="com.mchange.v2.c3p0.ComboPooledDataSource" id="dataSource">
15. <property name="driverClass" value="${c3p0.driverClass}"></property>
16. <property name="jdbcUrl" value="${c3p0.jdbcUrl}"></property>
17. <property name="user" value="${c3p0.user}"></property>
18. <property name="password" value="${c3p0.password}"></property>
19. <property name="initialPoolSize" value="${c3p0.initialPoolSize}"></property>
20. <property name="maxPoolSize" value="${c3p0.maxPoolSize}"></property>
21. <property name="checkoutTimeout" value="${c3p0.checkoutTimeout}"></property>
22. </bean>
24. <bean **class**="cn.cxd.daos.AccountDao" id="accountDao">
25. <property name="dataSource" ref="dataSource"></property>
26. </bean>
28. <bean **class**="org.springframework.jdbc.datasource.DataSourceTransactionManager" id="manager">
29. <property name="dataSource" ref="dataSource"></property>
30. </bean>

33. <bean **class**="cn.cxd.services.AccountService02" id="accountService02">
34. <property name="accountDao" ref="accountDao"></property>
35. </bean>
37. <bean id="proxyService" **class**="org.springframework.transaction.interceptor.TransactionProxyFactoryBean">
38. <property name="proxyInterfaces">
39. <list>
40. <value>cn.cxd.Impls.IAccountService</value>
41. </list>
42. </property>
43. <property name="target" ref="accountService02"></property>
44. <property name="transactionManager" ref="manager"></property>
45. <property name="transactionAttributes">
46. <props>
47. <prop key="\*">PROPAGATION\_REQUIRED </prop>
48. </props>
49. </property>
50. </bean>
51. </beans>

**客户端使用：**

1. **package** test;
3. **import** cn.cxd.Impls.IAccountService;
4. **import** cn.cxd.services.AccountService02;
5. **import** org.springframework.context.ApplicationContext;
6. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
7. **import** org.springframework.transaction.interceptor.TransactionProxyFactoryBean;
9. **public** **class** TestHalfTx {
10. **public** **static** **void** main(String[] args) {
11. ApplicationContext context = **new** ClassPathXmlApplicationContext("beans02.xml");
12. IAccountService service = (IAccountService)context.getBean("proxyService");
13. **double** accountZ3 = service.checkAccount("张三");
14. **double** accountL4 = service.checkAccount("李四");
16. System.out.println("张三金额： "+accountZ3);
17. System.out.println("李四金额： "+accountL4);
18. service.transferAccount("张三","李四",100);
20. accountZ3 = service.checkAccount("张三");
21. accountL4 = service.checkAccount("李四");
22. System.out.println("张三金额： "+accountZ3);
23. System.out.println("李四金额： "+accountL4);
24. }
25. }

**四.aop标签配置事务**

**其中：**

**1.Class AccountDao extends JdbcDaoSupport implements IAccountDao**

**2.Class AccountService02 implements IAccountService {**

**private IAccountDao accountDao;…}**

**service不需要事务模板了..**

**3.xml配置** **<tx:advice>需要manager,而manage需要dataSource，**

**4.** **<tx:advice>内部有** <tx:attributes>**，配置**<tx:method name="\*" propagation="REQUIRED"/>

**Method代表将有Spring完成事务配置，**

**5.在<aop:config>中配置pointcut和advise，就是上面的tx:advice，可以是拦截了所有的方法，但会被配置成事务的仅仅是在tx:advise中配置的method.**

**Xml配置：**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:tx="http://www.springframework.org/schema/tx" xmlns:aop="http://www.springframework.org/schema/aop"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context.xsd
10. http://www.springframework.org/schema/tx
11. http://www.springframework.org/schema/tx/spring-tx.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">
13. <context:component-scan base-**package**="cn.cxd"></context:component-scan>
15. <context:property-placeholder location="c3p0.properties"></context:property-placeholder>
17. <bean **class**="com.mchange.v2.c3p0.ComboPooledDataSource" id="dataSource">
18. <property name="driverClass" value="${c3p0.driverClass}"></property>
19. <property name="jdbcUrl" value="${c3p0.jdbcUrl}"></property>
20. <property name="user" value="${c3p0.user}"></property>
21. <property name="password" value="${c3p0.password}"></property>
22. <property name="initialPoolSize" value="${c3p0.initialPoolSize}"></property>
23. <property name="maxPoolSize" value="${c3p0.maxPoolSize}"></property>
24. <property name="checkoutTimeout" value="${c3p0.checkoutTimeout}"></property>
25. </bean>
27. <bean **class**="cn.cxd.daos.AccountDao" id="accountDao">
28. <property name="dataSource" ref="dataSource"></property>
29. </bean>
31. <bean **class**="org.springframework.jdbc.datasource.DataSourceTransactionManager" id="manager">
32. <property name="dataSource" ref="dataSource"></property>
33. </bean>
35. <bean id="service" **class**="cn.cxd.services.AccountService02">
36. <property name="accountDao" ref="accountDao"></property>
37. </bean>
39. <tx:advice id="tranAdvise" transaction-manager="manager">
40. <tx:attributes>
41. <tx:method name="\*" propagation="REQUIRED"/>
42. </tx:attributes>
43. </tx:advice>
45. <aop:config>
46. <aop:pointcut id="pointCut" expression="execution(\* cn.cxd.services.AccountService02.\*(..))"/>
47. <aop:advisor pointcut-ref="pointCut" advice-ref="tranAdvise"></aop:advisor>
48. </aop:config>

51. </beans>

**客户端使用：**

1. **package** test;
3. **import** cn.cxd.Impls.IAccountService;
4. **import** org.springframework.context.ApplicationContext;
5. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
7. **public** **class** TestWholeTx {
8. **public** **static** **void** main(String[] args) {
9. ApplicationContext context = **new** ClassPathXmlApplicationContext("beans03.xml");
10. IAccountService service = (IAccountService)context.getBean("service");
11. System.out.println(service.getClass().getSuperclass());
12. **double** accountZ3 = service.checkAccount("张三");
13. **double** accountL4 = service.checkAccount("李四");
15. System.out.println("张三金额： "+accountZ3);
16. System.out.println("李四金额： "+accountL4);
17. service.transferAccount("张三","李四",100);
19. accountZ3 = service.checkAccount("张三");
20. accountL4 = service.checkAccount("李四");
21. System.out.println("张三金额： "+accountZ3);
22. System.out.println("李四金额： "+accountL4);
23. }
24. }

**五.注解配置**

**其中：**

**1.Class AccountDao extends JdbcDaoSupport implements IAccountDao**

**2.Class AccountService02 implements IAccountService {**

**private IAccountDao accountDao;…}**

**service不需要事务模板了..**

**3.xml配置** <tx:annotation-driven transaction-manager="manager" ></tx:annotation-driven>  **需要manager,而manage需要dataSource，**

**4.service可以仅仅使用@Transactional，如果要一步到位可以加个@Component,且成员对象@autowired**

**Xml配置：**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xmlns:context="http://www.springframework.org/schema/context"
5. xmlns:tx="http://www.springframework.org/schema/tx" xmlns:aop="http://www.springframework.org/schema/aop"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans.xsd
8. http://www.springframework.org/schema/context
9. http://www.springframework.org/schema/context/spring-context.xsd
10. http://www.springframework.org/schema/tx
11. http://www.springframework.org/schema/tx/spring-tx.xsd http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd">
13. <context:component-scan base-**package**="cn.cxd"></context:component-scan>
15. <context:property-placeholder location="c3p0.properties"></context:property-placeholder>
17. <bean **class**="com.mchange.v2.c3p0.ComboPooledDataSource" id="dataSource">
18. <property name="driverClass" value="${c3p0.driverClass}"></property>
19. <property name="jdbcUrl" value="${c3p0.jdbcUrl}"></property>
20. <property name="user" value="${c3p0.user}"></property>
21. <property name="password" value="${c3p0.password}"></property>
22. <property name="initialPoolSize" value="${c3p0.initialPoolSize}"></property>
23. <property name="maxPoolSize" value="${c3p0.maxPoolSize}"></property>
24. <property name="checkoutTimeout" value="${c3p0.checkoutTimeout}"></property>
25. </bean>
27. <bean **class**="cn.cxd.daos.AccountDao" id="accountDao">
28. <property name="dataSource" ref="dataSource"></property>
29. </bean>
31. <bean **class**="org.springframework.jdbc.datasource.DataSourceTransactionManager" id="manager">
32. <property name="dataSource" ref="dataSource"></property>
33. </bean>
35. <!--    可以不用,直接@conponent@autowired,就不必声明service了，但是accountDao还是得声明，
36. 因为暂时不知道怎么使用注解获取配置文件。
37. <bean id="service" **class**="cn.cxd.services.AccountService03">
38. <property name="accountDao" ref="accountDao"></property>
39. </bean>
40. -->
42. <tx:annotation-driven transaction-manager="manager" ></tx:annotation-driven>

45. </beans>

**客户端使用：**

1. **package** test;
3. **import** cn.cxd.Impls.IAccountService;
4. **import** cn.cxd.services.AccountService03;
5. **import** org.springframework.context.ApplicationContext;
6. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
8. **public** **class** TestAnnoTx {
9. **public** **static** **void** main(String[] args) {
10. ApplicationContext context = **new** ClassPathXmlApplicationContext("beans04.xml");
11. IAccountService service = (IAccountService)context.getBean("service");
12. System.out.println(service.getClass().getSuperclass());
13. **double** accountZ3 = service.checkAccount("张三");
14. **double** accountL4 = service.checkAccount("李四");
16. System.out.println("张三金额： "+accountZ3);
17. System.out.println("李四金额： "+accountL4);
18. service.transferAccount("张三","李四",100);
20. accountZ3 = service.checkAccount("张三");
21. accountL4 = service.checkAccount("李四");
22. System.out.println("张三金额： "+accountZ3);
23. System.out.println("李四金额： "+accountL4);
24. }
25. }

**六.web项目和Spring中只让配置文件加载一次**

**首先创建webProject,在idea中配置tomcat:参考：**

[**https://blog.csdn.net/qq\_26344609/article/details/75577234**](https://blog.csdn.net/qq_26344609/article/details/75577234)

**其中：**

1. **配置servlet,要导入javax-servlet.jar,继承HttpServlet**
2. **如果在servlet中调用业务层，getBean则有可能每次行为都要解析Spring的xml，那么就要在web.xml中完成一些配置，使得仅加载一次xml,后续就可以一直获取，而不必一次serlvet,一次读取Spring-xml。其中这个Spring.xml的位置还是在类路径，即src下。**

**在web.xml中要添加：**

1. <context-param>
2. <param-name>contextConfigLocation</param-name>
3. <param-value>classpath:applicationBeans.xml</param-value>
4. </context-param>
6. <listener>
7. <listener-**class**>org.springframework.web.context.ContextLoaderListener</listener-**class**>
8. </listener>

**3. 而serlvet若要获取service的bean,则需要：**

**WebApplicationContext context = WebApplicationContextUtils.getWebApplicationContext(this.getServletContext());**

**LoginService service = (LoginService) context.getBean("service");**

**Web.Xml配置如下：**

1. <?xml version="1.0" encoding="UTF-8"?>
2. <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"
5. version="4.0">
7. <context-param>
8. <param-name>contextConfigLocation</param-name>
9. <param-value>classpath:applicationBeans.xml</param-value>
10. </context-param>
12. <listener>
13. <listener-**class**>org.springframework.web.context.ContextLoaderListener</listener-**class**>
14. </listener>
15. </web-app>

**Servlet配置：**

1. **package** cn.cxd.servlets;
3. **import** cn.cxd.services.LoginService;
4. **import** com.mysql.cj.log.Log;
5. **import** org.springframework.web.context.WebApplicationContext;
6. **import** org.springframework.web.context.support.WebApplicationContextUtils;
8. **import** javax.servlet.ServletException;
9. **import** javax.servlet.annotation.WebServlet;
10. **import** javax.servlet.http.HttpServlet;
11. **import** javax.servlet.http.HttpServletRequest;
12. **import** javax.servlet.http.HttpServletResponse;
13. **import** java.io.IOException;
15. @WebServlet("/login")
16. **public** **class** LoginServlet **extends** HttpServlet {
17. **protected** **void** doGet(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
18. String username = req.getParameter("username");
19. String password = req.getParameter("password");
20. resp.setContentType("text/html; charset=UTF-8");
22. WebApplicationContext context = WebApplicationContextUtils.getWebApplicationContext(**this**.getServletContext());
23. LoginService service = (LoginService) context.getBean("service");
24. resp.getWriter().write(service.login(username,password));
25. System.out.println(service.hashCode());
26. System.out.println(service.hashCode());
27. System.out.println(service.hashCode());
28. System.out.println(service.hashCode());
29. }
30. }