# 分布式事务集成文档

# 项目非分布式处理

项目中应尽量避免分布式事务的存在，实现方式如下，项目A.controller调用BCD.save方法时，尽量采用A.controller调用B.save B.save方法调用C.save C.save方法调用D.save

实现代码如下

Public class A {

@Autowired

Private B b;

@RequestMapping

Public ResultMessage saveInfo (Object info){

B.save(info);

}

}

Public class B {

@Autowired

Private BMapper bmapper;

@Autowired

Private C c;

@Transactional

Public ResultMessage saveInfo (Object info) {

bmapper.save(info);

C.saveInfo(info);

}

}

Public class C {

@Autowired

Private CMapper cmapper;

@Autowired

Private D d;

@Transactional

Public ResultMessage saveInfo (Object info) {

cmapper.save(info);

d.saveInfo(info);

}

}

Public class D {

@Autowired

Private DMapper dmapper;

@Transactional

Public ResultMessage saveInfo (Object info) {

dmapper.save(info);

}

}

# 分布式处理

无法避免情况下, 项目中引入jar包

<**dependency**>  
 <**groupId**>com.zuma</**groupId**>  
 <**artifactId**>zm\_transaction\_api</**artifactId**>  
 <**version**>1.1</**version**>  
</**dependency**>

Service项目中配置文件修改如下

Application-dataSource\*.xml文件

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

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

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

<property name="plugins">

<list>

<bean class="com.zuma.transaction.interceptor.ZmbatisInterceptor"></bean>

</list>

</property>

</bean>

Application-root\*.xml文件

添加bean注入 并启用aop自动代理

<bean class="com.zuma.transaction.aop.ServiceAspect"></bean>

<aop:aspectj-autoproxy />

Web项目配置文件修改如下

Application-consumer.xml文件添加配置

<**dubbo:consumer filter="zumaDubboFilter"**></**dubbo:consumer**>

Application-root\*.xml文件添加配置

<aop:aspectj-autoproxy/>

<bean id="dubboTransactionInfo" class="com.zuma.transaction.dubbo.api.DubboTransactionInfo"/>

Controller接口中需要加入分布式事务的方法上添加注解@Compensable

@RequestMapping(**"/saveMenu"**)  
@ResponseBody  
@Compensable  
**public** ResultMessage saveMenu(SysSiteMenuBean bean, HttpServletRequest request) {

}