HTTP SERVICE 框架使用文档

1.httpService-zaq介绍

httpService-zaq是一个基本http协议开发的分布式事务控制业务处理框架，提供了一套快速开发分布事务接口和客户端调用的的解决方案（见下图），事务控制属于【业务补偿型事务】，它强烈的拥抱了Spring,现在持久层暂时只以Hibernate处理，以原始的Servlet做控制，开发出的接口正则格式为：^http://.\*?/httpService/.\*?/.\*?/.\*?$解析后servlet-mapping为/httpService/{模块}/{Action}/{方法}

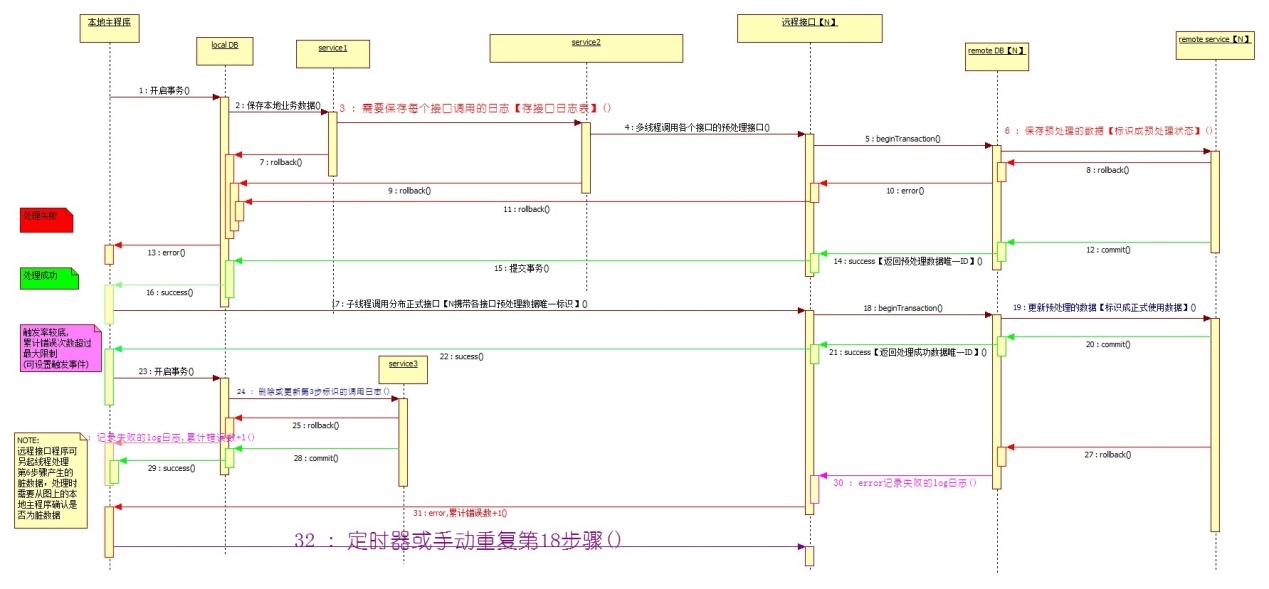
{模块}：eg ：struts2 的nameSpace

{Action}：注入到Spring容器中的Action类的前缀

{方法}：实现了com.zaq.ihttp.web.server.HttpServiceBaseAction的方法名或继承了 OaBaseAction需要实现，具体为： saveOrUpdatePrepare,saveOrUpdate,delPrepare,del,query

客户调用 ：见下面接口调用的demo例子

2.原理图



3.部署配置

a.在web.xml中配置：

<filter-mapping>

<filter-name>encodingFilter</filter-name>

<url-pattern>/httpService/\*</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>NoCache</filter-name>

<url-pattern>/httpService/\*</url-pattern>

</filter-mapping>

<context-param>

<param-name>httpServiceFilePath</param-name>

<!-- 实现开发时调用客户端的配置信息，详情见http-service.properties 的注释内容--> <param-value>com/zaq/conf/http-service.properties</param-value>

</context-param>

<listener>

<listener-class>com.zaq.ihttp.web.client.HttpServiceClientListener</listener-class>

</listener>

<servlet>

<servlet-name>httpServiceServlet</servlet-name>

<servlet-class>

com.zaq.ihttp.web.server.HttpServiceSevrlet

</servlet-class>

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

</servlet>

<servlet-mapping>

<servlet-name>httpServiceServlet</servlet-name>

<url-pattern>/httpService/\*</url-pattern>

</servlet-mapping>

b.在app-context.xml配置：

<import resource="classpath:spring-httpService.xml"/>

c. app-resources.xml的sessionFactory-》mappingLocations-》list配置：

<value>classpath:HttpServiceFirewall.hbm.xml</value>

<value>classpath:HttpServiceCommit.hbm.xml</value>

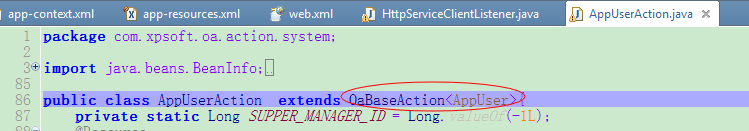
d. http-service.properties参数详细说明请见

http-service.properties文件的注释

4.编程说明（基于oa-core.jar）

a.开发一个httpService服务

只需要将Action类的父类BaseAction 换成OaBaseAction<T>

eg: 

可以开发的服务接口有下面5个（接口说明详见【接口说明5】）：

就可以为AppUser对象开发出其对应的操作接口

b.请求httpService包含分布式事务的接口

需要将Service的父类BaseService换成OaBaseService



并将ServiceImpl的父类BaseServiceImpl换成OaBaseServiceImpl



action对分布式事务处理的接口调用demo如下

**public** String **callTest**(){

**final** AppUser au = **new** AppUser();

au.setTitle(Short.*valueOf*((**short**) 1));

au.setUsername("user" );

au.setPassword("1");

au.setFullname("李海" );

au.setAddress("testAddress");

au.setEducation("test");

au.setEmail("user" + "@xpsoft.com");

au.setAccessionTime(**new** Date());

au.setPhoto("photo");

au.setZip("00003");

au.setStatus(Short.*valueOf*((**short**) 1));

au.setFax("020-003034034");

au.setPosition("UserManager");

au.setDelFlag(Constants.FLAG\_UNDELETED);

**final** String host="ljt";

**final** String packagez="system";

**final** String action="appUser";

TransactionCommand command0=**new** TransactionCommand() {

@Override

**public** HttpServiceCommit **execute**() {

appUserService.save(au);

**return** **null**;

}

};

TransactionCommand command1=**new** TransactionCommand() {

@Override

**public** HttpServiceCommit **execute**() {

**return** appUserService.prepareSaveOrUpdate(host, packagez, action, au);

}

};

// HttpServiceCommit[] commits= appUserService.prepareTransaction(command0,command1);

// HttpServiceCommit[] commits= appUserService.saveWithLocal(host, packagez, action, au);

**boolean** retBoo=appUserService.callCommon(command0,command1);

// HttpServiceCommit[] commits=appUserService.saveWithLocal(host, packagez, action, au);

// boolean retBoo=appUserService.saveReCall(commits,new SimpleCallBack());

System.out.println(retBoo);

setJsonString(retBoo+"");

RetObj<AppUser> retAu=appUserService.query(host, packagez, action, **new** BasicNameValuePair("Q\_username\_S\_EQ","admin"));

System.out.println("userId======"+retAu.getObjs().get(0).getUserId());

**return** SUCCESS;

}

5.接口说明：

详细见接口说明文档javadoc