Atitit 通用接口的设计与实现attilax 总结

[1.1. 现存的情况 1](#_Toc6780)

[1.2. 接口返回类型，与返回序列化格式 1](#_Toc21210)

[1.3. 异常传递 代替返回值模式 1](#_Toc19793)

[1.4. 通用接口原理 1](#_Toc9828)

[1.5. Autoit Autohotkey的模式 2](#_Toc26678)

[1.6. 通用版全功能接口http param模式 范例 （ 可以运用于任何场合） 2](#_Toc2125)

[1.7. 通用版数据sql接口 （快速开发接口，适用与内部人员操作模块） 3](#_Toc20815)

[1.8. 通用版数据sp存储过程调用接口 （快速开发接口，适用于任何场合） 3](#_Toc11430)

[1.9. 核心代码 4](#_Toc8444)

[1.10. 未来的展望 dsl 模式 5](#_Toc29843)

## 现存的情况

接口不够通用，每增加一个接口，就需要增加一个配置或者注解。

需要一个通用接口，，每增加一个子接口，无需做任何的配置和注解

其次对于数据库操作，3gl接口比较繁琐 ,增加了4gl dsl sql sp接口

其次，接口的返回值模型。。可以使用异常模型代替

## 接口返回类型，与返回序列化格式

返回类型类型就是ｓｔｒ　ｉｎｔ等，支持复杂格式map list,以及对象。

返回序列化格式,即是结果使用什么样的序列化返回结果。。支持json ，预计还要支持xml yaml

## 异常传递 代替返回值模式

异常拥有比返回值更好的处理模式。

异常序列化为指定的序列化格式返回，传递给调用端。。

## 通用接口原理

直接指明要调用的类与方法名。后台通过反射的模式调用。。

类似 pinvoke ， jna 模式

## Autoit Autohotkey的模式

DllCall ( "dll", "返回值类型", "函数名称" [, 类型1, 参数1[, 类型n, 参数n]] )  
  
AutoIt中的参数类型与Win32 API中的参数类型不完全相同，这点要注意。

Autohotkey

Result := DllCall("[DllFile/]Function" [, Type1, Arg1, Type2, Arg2, "Cdecl ReturnType"])

## 通用版全功能接口http param模式 范例 （ 可以运用于任何场合）

[http://localhost:8080/AjaxJsbridge\_HttpparamMode\_servlet?m=com.attilax.rest.Class4test.m1&p1=123](http://localhost:8080/AjaxJsbridge_HttpparamMode_servlet?iocFac=com.attilax.rest.JavaNewCreatorFac&createmode=new&class=com.attilax.rest.Class4test&method=m1&p1=123)

createmode参数： 类的创建模式 默认为动态new创建模式

。静态类的方法调用 为static

动态类，默认值，或者使用new

m：或者method ，指明要调用的方法,全类名加方法名，比如com.attilax.rest.Class4test.m1

retType:返回数据类型int str map list obj等

retFmt：返回数据序列化格式，一般为json，也可以为none,xml，默认为json

P1\_type ：第一个参数类型 有str int 等，默认为str

P1：第一个参数

P2\_type：第二个参数类型

P2:第二个参数

iocFac：ioc工厂：支持spring guice new 工厂模式，默认为com.attilax.rest.JavaNewCreatorFac

morennew 工厂模式为 com.attilax.rest.JavaNewCreatorFac

特点：：

全功能接口。

开发效率不是最高

## 通用版数据sql接口 （快速开发接口，适用与内部人员操作模块）

[http://localhost:8080/AjaxJsbridge\_HttpparamMode\_servlet?m=com.attilax.db.DbServiceV4qb9.executeQuery&p1=select+\*+from+ecs\_users+limit+10](http://localhost:8080/AjaxJsbridge_HttpparamMode_servlet?iocFac=com.attilax.rest.JavaNewCreatorFac&createmode=new&class=com.attilax.rest.Class4test&method=m1&p1=123)&iocFac=com.attilax.ioc.Ioc4other

注意：此数据接口是为快速开发而设置的，直接使用sql dsl存取数据，方便快捷。适用于后端管理，以及内部管理系统模块，适用于用户特定以及内部用户的模块。不适用于面向广大不特定用户的模块。。

面向广大不特定用户的模块需要隐藏sql，传递sql语句id即可，具体的sql语句应该存储在存储过程，代码或者配置文件里面 。。需要使用存储过程接口或其他接口即可

特点：：

开发效率貌似最高

非全功能接口，只针对数据操作接口。其次，不适用于面向公众人员使用的模块。

## 通用版数据sp存储过程调用接口 （快速开发接口，适用于任何场合）

调用存储过程mysql，即是使用特定sql语句，call 调用存储过程即可，比如call `sp\_查询用户`('mer')。。

[http://localhost:8080/AjaxJsbridge\_HttpparamMode\_servlet?m=com.attilax.db.DbServiceV4qb9.executeQuery&p1=call+%60sp\_%E6%9F%A5%E8%AF%A2%E7%94%A8%E6%88%B7%60%28%27mer%27%29](http://localhost:8080/AjaxJsbridge_HttpparamMode_servlet?iocFac=com.attilax.rest.JavaNewCreatorFac&createmode=new&class=com.attilax.rest.Class4test&method=m1&p1=123)&iocFac=com.attilax.ioc.Ioc4other

即是p1参数为 call `sp\_查询用户`('mer')。。 ,注意url参数的urlencode编码

特点：：

开发效率较高

非全功能接口，只针对数据操作接口。

## 核心代码

|  |
| --- |
| @Override  **public** **void** **service**(ServletRequest req, ServletResponse response) {  **try** {  HttpServletRequest **req2** = (HttpServletRequest) req;  Global.*req*.set(req2);  String **iocx** =get\_iocFac( req.getParameter("iocFac"));  String **class\_meth\_full** = (getMeth(req));  String **cls** =refx.*getClassName*(class\_meth\_full);  String **meth\_name**=refx.*getMethodName*(class\_meth\_full);    Class **c** = Class.*forName*(cls);      List<Object> **params** = Lists.*newArrayList*();  **for** (**int** **i** = 1; i < 7; i++) {  String **pname** = "p" + i;  **if** (req.getParameter(pname) == **null**)  **continue**;  String **p** = req.getParameter(pname);  String **ptype** = getPtype(req, pname);  Object **p\_obj**=getP(p,ptype);  params.add(p\_obj);    }    Object[] **p\_a** = params.toArray();  String **createmode** =req.getParameter("createmode");  Object **rzt**;  **if**(isStaticMode(createmode))  {  rzt = MethodUtils.*invokeStaticMethod*(c, meth\_name, p\_a);    }  **else**//def is new  {  Object **cls\_obj** = com.attilax.reflect.MethodUtils.*invokeStaticMethod*(iocx, "getBean", c);  rzt = MethodUtils.*invokeMethod*(cls\_obj, meth\_name, p\_a);  }        System.***out***.println(rzt);  // Object r = new DslParser().parseV3(req2.getParameter("dsl"));  **if** (req.getParameter("retFmt") != **null**) {  **if** (req.getParameter("retFmt").equals("json"))  response.getWriter().println(AtiJson.*toJson*(rzt)); // �벻Ҫ�޸Ļ�ɾ��  } **else**  response.getWriter().println(rzt.toString()); // �벻Ҫ�޸Ļ�ɾ��  } **catch** (Exception **e**) {  ***logger***.error("--err", e);  ***logger***.error("--err2", AtiJson.*toJson*(e));  filex.*save\_safe*(~~core~~.*getTrace*(e), "c:\\0rechglog\\" + filex.*getUUidName*() + ".log");  ExUtil.*throwExV2*(e);  }  } |

## 未来的展望 dsl 模式

直接支持java表达式，<http://xxxxxx/api?dsl=new> com.xxx.userservice().reg()

第二发展，支持方法链

作者:: 绰号:老哇的爪子claw of Eagle 偶像破坏者Iconoclast image-smasher

捕鸟王"Bird Catcher kok 虔诚者Pious 宗教信仰捍卫者 Defender Of the Faith. 卡拉卡拉红斗篷 Caracalla red cloak 万兽之王

简称：： Emir Attilax Akbar 埃米尔 阿提拉克斯 阿克巴

全名：：Emir Attilax Akbar bin Mahmud bin attila bin Solomon bin adam Al Rapanui 埃米尔 阿提拉克斯 阿克巴 本 马哈茂德 本 阿提拉 本 所罗门 本亚当 阿尔 拉帕努伊

常用名：艾提拉（艾龙）， EMAIL:1466519819@qq.com

头衔：uke总部o2o负责人，全球网格化项目创始人，

uke交友协会会长 uke捕猎协会会长 Emir Uke部落首席大酋长，

uke宗教与文化融合事务部部长， uke宗教改革委员会副主席

uke制度与重大会议委员会委员长，uke保安部首席大队长,uke制度检查委员会副会长，

uke 首席cto 软件部门总监 技术部副总监 研发部门总监主管 产品部副经理 项目部副经理 uke科技研究院院长 uke软件培训大师

uke波利尼西亚区大区连锁负责人 汤加王国区域负责人 uke克尔格伦群岛区连锁负责人，莱恩群岛区连锁负责人，uke布维岛和南乔治亚和南桑威奇群岛大区连锁负责人

Uke软件标准化协会理事长理事长 Uke 数据库与存储标准化协会副会长

uke终身教育学校副校长 Uke医院 与医学院方面的创始人

uec学院校长， uecip图像处理机器视觉专业系主任 uke文档检索专业系主任

Uke图像处理与机器视觉学院首席院长

Uke 户外运动协会理事长 度假村首席大村长 uke出版社编辑总编

转载请注明来源：attilax的专栏 ?http://blog.csdn.net/attilax

--Atiend v8