# guns-rest技术文档

## 文档简介

guns-rest 为专门提供restful api的模块，该模块中主要实现了jwt鉴权、传输数据签名的机制及提供互啊佑相关业务接口。

## 2. rest jwt 签名机制

### 2.1 REST

**核心：面向资源，HTTP规范协议**

REST：

**概述： （1）用URL定位资源，**

**（2）用HTTP动词（GET,POST,DELETE,DETC）描述操作，**

**（3）用http status code了解结果如何。**

REST是一种软件架构风格，而**不是**标准，提供一组设计原则和约束条件。满足这些约束条件和原则的应用程序或设计就是 RESTFUL。它主要用于客户端和服务器交互类的软件。

REST的约束（Client-Server、Stateless、Cache、Uniform Interface、Layered System、Code-on-Demand）

（1）客户-服务器（Client-Server）客户端服务器分离；

（2）无状态（Stateless），从客户端的每个请求要包含服务器所需要的所有信息

（3）缓存（Cachable），服务器返回信息必须被标记是否可以缓存，如果缓存，客户端可能 会重用之前的信息发送请求。

（4）分层系统（Layered System），系统组件不需要知道与他交流组件之外的事情。封装服 务，引入中间层。

（5）统一接口（Uniform Interface）

（6）支持按需代码（Code-On-Demand 可选）

1. REST描述的是在网络中client和server的一种交互形式；REST本身不实用，实用的是如何设计 RESTful API（REST风格的网络接口）；
2. Server提供的RESTful API中，**URL中只使用名词来指定资源，原则上不使用动词，且推荐用复数。**“资源”是REST架构或者说整个网络处理的核心。

比如：

左边是错误的设计，而右边是正确的：

GET /rest/api/getDogs --> GET /rest/api/dogs 获取所有小狗狗

GET /rest/api/addDogs --> POST /rest/api/dogs 添加一个小狗狗

GET /rest/api/editDogs/:dog\_id --> PUT /rest/api/dogs/:dog\_id 修改一个小狗狗

GET /rest/api/deleteDogs/:dog\_id --> DELETE /rest/api/dogs/:dog\_id 删除一个小狗狗

左边的这种设计，很明显不符合REST风格，URL只负责准确无误的暴露资源，而 getDogs/addDogs...已经包含了对资源的操作，这是不对的。相反右边却满足了，它的操作是使用标准的HTTP动词来体现。

1. 用HTTP协议里的动词来实现资源的添加，修改，删除等操作。即通过HTTP动词来实现资源的状态扭转：  
    GET 用来获取资源，  
    POST 用来添加资源（也可以用于更新资源），  
    PUT 用来更新资源，  
    DELETE 用来删除资源。

比如：  
 DELETE [http://api.qc.com/v1/](https://link.zhihu.com/?target=http%3A//api.qc.com/v1/friends)friends: 删除某人的好友 （在http parameter指定好友id）  
 POST [http://api.qc.com/v1/](https://link.zhihu.com/?target=http%3A//api.qc.com/v1/friends)friends: 添加好友  
 UPDATE [http://api.qc.com/v1/profile](https://link.zhihu.com/?target=http%3A//api.qc.com/v1/profile): 更新个人资料

禁止使用： GET [http://api.qc.com/v1/deleteFriend](https://link.zhihu.com/?target=http%3A//api.qc.com/v1/deleteFriend)

4. Server和Client之间传递某资源的一个表现形式，比如用JSON，XML传输文本，或者用JPG，WebP传输图片等。当然还可以压缩HTTP传输时的数据（on-wire data compression）。  
5. 用 HTTP Status Code传递Server的状态信息。比如最常用的 200 表示成功，500 表示Server内部错误等。

### 2.2 JWT Token

2.2.1． jwt token：

Json web token (JWT), 是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准（(RFC 7519).该token被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可直接被用于认证，也可被加密。

2.2.2． jwt token鉴权机制：

jwt token鉴权机制是指若需要请求服务器接口,必须通过AuthController获取一个请求令牌(jwt token),持有jwt token的用户才可以访问服务器的其他资源,如果没有此令牌,则访问接口会直接忽略,请求获取jwt token时,需要携带credenceName和credenceCode(可以是账号密码,可以是手机号验证码等等),校验credenceName和credenceCode成功后,会颁发给客户端一个jwt token还有一个随机字符串,用于传输过程中对数据进行签名用.基于token的鉴权机制类似于http协议也是无状态的,它不需要在服务端去保留用户的认证信息或者会话信息.这就意味着基于token认证机制的应用不需要去考虑用户在哪一台服务器登录了,这就为应用的扩展提供了便利.

2.2.3． 流程：

（1）用户使用用户名密码来请求服务器

（2）服务器进行验证用户的信息

（3）服务器通过验证发送给用户一个token

（4）客户端存储token，并在每次请求时附送上这个token值

（5）服务端验证token值，并返回数据

### 2.3 签名机制

签名机制是指客户端向服务端传输数据中,对传输数据进行md5加密,并且加密过程中利用Auth接口返回的随机字符串进行混淆加密,并把md5值同时附带给服务端,服务端通获取数据之后对数据再进行一次md5加密,若加密结果和客户端传来的数据一致,则认定客户端请求的数据是没有被篡改的,若不一致,则认为被加密的数据是被篡改的。

### 2.4 签名校验机制

**@RequestBody注解 :**

（1）、 @requestBody注解常用来处理content-type不是默认的application/x-www-form-urlcoded编码的内容，比如说：application/json或者是application/xml等。一般情况下来说常用其来处理application/json类型。

（2）、 @requestBody注解用于读取Request请求的Body部分数据，使用系统默认配置的HttpMessageConverter进行解析，然后把相应的数据绑定到Controller中的方法参数上。

通过@RequestBody可以将将请求体中的JSON字符串绑定到相应的bean上，当然，也可以将其分别绑定到对应的字符串上。

**guns-rest项目中**通过继承FastJsonHttpMessageConverter(其顶层父接口依然是HttpMessageConverter)类，实现类为WithSignMessageConverter，来实现自定义的http传输过程中的body中的参数要求及签名校验。

既-使用@RequestBody进行http参数接收的接口，参数都要进行WithSignMessageConverter类中的签名校验

## 3．guns-rest提供的相关接口

（1）用户直达应用管理接口： wayapi/v1/accessory/app

应用直达绑定修改： editBindInfo POST

应用直达删除 ： delBindInfo POST

（2）用户指纹管理接口 ： wayapi/v1/accessory/fp

指纹注册接口 ： register POST

指纹删除接口 ： delete POST

获取用户指纹及绑定的直达数据：getFpAndBindInfo POST

（3）登录相关接口 ： wayapi/v1/accessory/auth

获取验证码 ： getCode POST

用户信息初始化 ： userInit POST

用户注册或登录 ： loginByPhone POST

指纹验证登录 ： loginByFp POST

（4）上报接口 ： wayapi/v1/accessory/report

用户行为上报接口： userAction

操作系统上报接口： windowsVersion

## 问题1: guns-rest打成war包mapper文件丢失

日志信息：

2018-06-22 16:25:00.648 ERROR 4814---[Thread-37] c.s.g.r.c.aop.GlobalExceptionHandler : 运行时异常:

org.apache.ibatis.binding.BindingException: Invalid bound statement (not found): com.stylefeng.guns.rest.common.persistence.dao.WayAccessoryLogMobileCheckMapper.

selectWayAccessoryLogMobileCheckByCodeAndPhone

at org.apache.ibatis.binding.MapperMethod$SqlCommand.<init>(MapperMethod.java:225) ~[mybatis-3.4.5.jar:3.4.5]

at org.apache.ibatis.binding.MapperMethod.<init>(MapperMethod.java:48) ~[mybatis-3.4.5.jar:3.4.5]

at org.apache.ibatis.binding.MapperProxy.cachedMapperMethod(MapperProxy.java:65) ~[mybatis-3.4.5.jar:3.4.5]

at org.apache.ibatis.binding.MapperProxy.invoke(MapperProxy.java:58) ~[mybatis-3.4.5.jar:3.4.5]

at com.sun.proxy.$Proxy278.selectWayAccessoryLogMobileCheckByCodeAndPhone(Unknown Source) ~[na:na]

at com.stylefeng.guns.rest.modular.auth.validator.impl.DbValidator.validate(DbValidator.java:28) ~[classes/:0.0.1-SNAPSHOT]

at com.stylefeng.guns.rest.modular.auth.controller.AuthController.createAuthenticationToken(AuthController.java:40) ~[classes/:0.0.1-SNAPSHOT]

不满足一下规范的，都可能出现上面这个异常

* mapper.xml的namespace要写所映射接口的全称类名。
* mapper.xml中的每个statement的id要和接口方法的方法名相同
* mapper.xml中定义的每个sql的parameterType要和接口方法的形参类型相同
* mapper.xml中定义的每个sql的resultType要和接口方法的返回值的类型相同
* mapper.xml要和对应的mapper接口在同一个包下
* mapper.xml的命名规范遵守: 接口名+Mapper.xml

如果以上检查完毕，项目都遵守了上述的规范，而且你的项目是Maven项目。但是运行程序还是会出现Mybatis invalid bound statement(not found)的问题，这个时候就需要修改pom.xml文件了。在pom.xml中添加如下的代码：

<resources>

<--<resource>

<directory>src/main/webapp</directory>

</resource>-->

<resource>

<directory>src/main/resources</directory>

</resource>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.xml</include>

</includes>

</resource>

</resources>