简介

app的用户更多是通过用户自主注册的方式，因此对于用户的租户隔离是通过`URI`的路劲隔离方式来实现。其中租户的ID通过`ag_admin`中的租户表来获取。

原理

通过对访问网关后端路劲`URI`上增加租户ID和请求头部标志来扩展。请求格式如下：

```
Header:
x-tenant-flag 1

URI:
/api/{tenant_id}/{后端服务}/xxx

示例: app用户注册
/api/ac88ceb386aa4231b09bf472cb937c24/app/appUser/register
```

网关核心逻辑代码

同时网关会将请求转发到后端的服务作进一步加工，比如在后端请求的头部，增加租户的标志`x-tenant-auth`，具体看下方代码

```
package com.github.wxiaoqi.security.gate.filter;

import com.github.ag.core.context.BaseContextHandler;
import
com.github.wxiaoqi.security.common.constant.RequestHeaderConstants;
import com.netflix.zuul.context.RequestContext;
import org.apache.commons.lang.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.cloud.netflix.zuul.filters.ZuulProperties;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;
```

```
import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;
import java.io.IOException;

/**
 * 租户路径拦截
 * @author ace
 * @create 2018/2/9.
 */
@Component("atenantFilter")
@WebFilter(filterName = "atenantFilter", urlPatterns = {"/api"})
@Order(-2147483648)
public class TenantFilter implements Filter {
    @Autowired
    private ZuulProperties zuulProperties;

    @Override
    public void init(FilterConfig filterConfig) throws
ServletException {

    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException {
        HttpServletRequest httpServletRequest =
(HttpServletRequest) request;
        String tenantFlag = ((HttpServletRequest)
request).getHeader(RequestHeaderConstants.TENANT_FLAG);
        if(StringUtils.isNotBlank(tenantFlag)) {
            String urlPath = httpServletRequest.getRequestURI();
            urlPath =
urlPath.substring(zuulProperties.getPrefix().length() + 1,
urlPath.length());
            String realPath =
urlPath.substring(urlPath.indexOf("/"), urlPath.length());
            String tenant = urlPath.substring(0,
urlPath.indexOf("/"));
            RequestContext ctx =
RequestContext.getCurrentContext();
            // 将租户id放置请求头部传递后端

ctx.addZuulRequestHeader(RequestHeaderConstants.TENANT,tenant);
            BaseContextHandler.set("tenant", tenant);
            chain.doFilter(new
UriWrapperRequest((HttpServletRequest) request,
```

```
zuulProperties.getPrefix() + realPath), response);
        }else{
            chain.doFilter(request, response);
        }
    }

    @Override
    public void destroy() {

    }

    class UriWrapperRequest extends HttpServletRequestWrapper {
        private String uri;

        public UriWrapperRequest(HttpServletRequest request, String
uri) {
            super(request);
            this.uri = uri;
        }

        @Override
        public String getRequestURI() {
            return this.uri;
        }

        @Override
        public String getServletPath() {
            return this.uri;
        }
    }
}
```

App后端服务核心代码

后端服务通过拦截器获取网关传递的租户ID，加入到当前线程的上下文里面，从而保证mybatis拦截可以取到当前的租户ID，从而进行数据隔离。当然也可以通过传入的租户在注册的时候，给app用户打上租户的标志。

```
import com.github.ag.core.context.BaseContextHandler;
import
com.github.wxiaoqi.security.common.constant.RequestHeaderConstants;
import org.apache.commons.lang.StringUtils;
import org.springframework.beans.factory.annotation.Autowired;
```

```java
import
org.springframework.cloud.netflix.zuul.filters.ZuulProperties;
import org.springframework.core.annotation.Order;
import org.springframework.stereotype.Component;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

/**
 * 租户路径拦截
 * @author ace
 * @create 2018/2/9.
 */
@Component("atenantFilter")
@WebFilter(filterName = "atenantFilter", urlPatterns = {"/api"})
@Order(-2147483648)
public class TenantFilter implements Filter {
    @Override
    public void init(FilterConfig filterConfig) throws
ServletException {

    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse
response, FilterChain chain) throws IOException, ServletException {
        HttpServletRequest httpServletRequest =
(HttpServletRequest) request;
        String tenantId =
httpServletRequest.getHeader(RequestHeaderConstants.TENANT);
        if(StringUtils.isNotBlank(tenantId)) {
            BaseContextHandler.setTenantID(tenantId);
        }
        chain.doFilter(request, response);
    }

    @Override
    public void destroy() {

    }

}
```

app用户注册核心代码

```
@Override
public void insertSelective(AppUser entity) {
    ValidatorUtils.validateEntity(entity, AddGroup.class);
    String password = encoder.encode(entity.getPassword());
    entity.setPassword(password);
    entity.setName("xxx_"+entity.getMobile());
    entity.setTenantId(BaseContextHandler.getTenantID());
    super.insertSelective(entity);
}
```

app用户管理模块

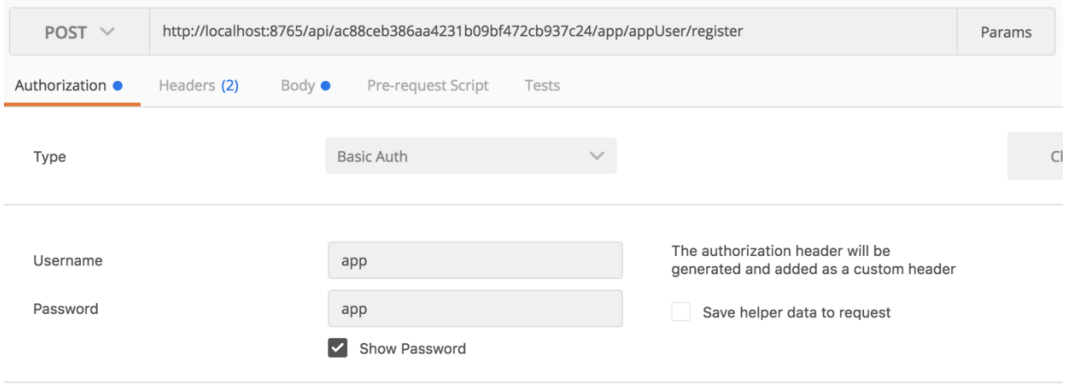
```
/**
 * 会员用户信息管理
 */
@RestController
@RequestMapping("admin/appUser")
@CheckClientToken
@CheckUserToken
public class AppUserAdminController extends
BaseController<AppUserBiz,AppUser,Integer> {

}
```

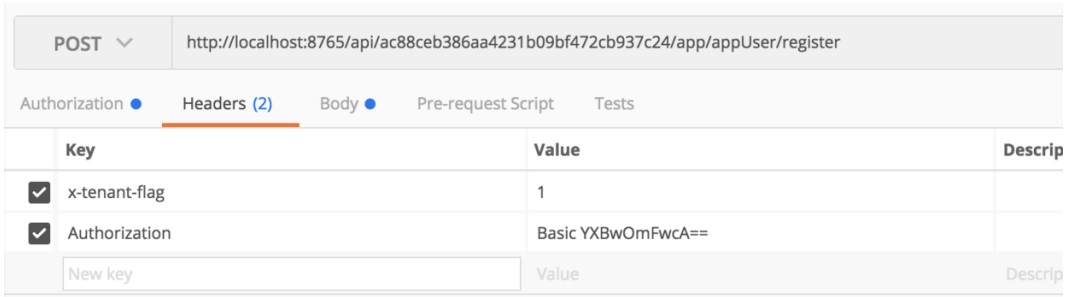
postman 调用示例

注册

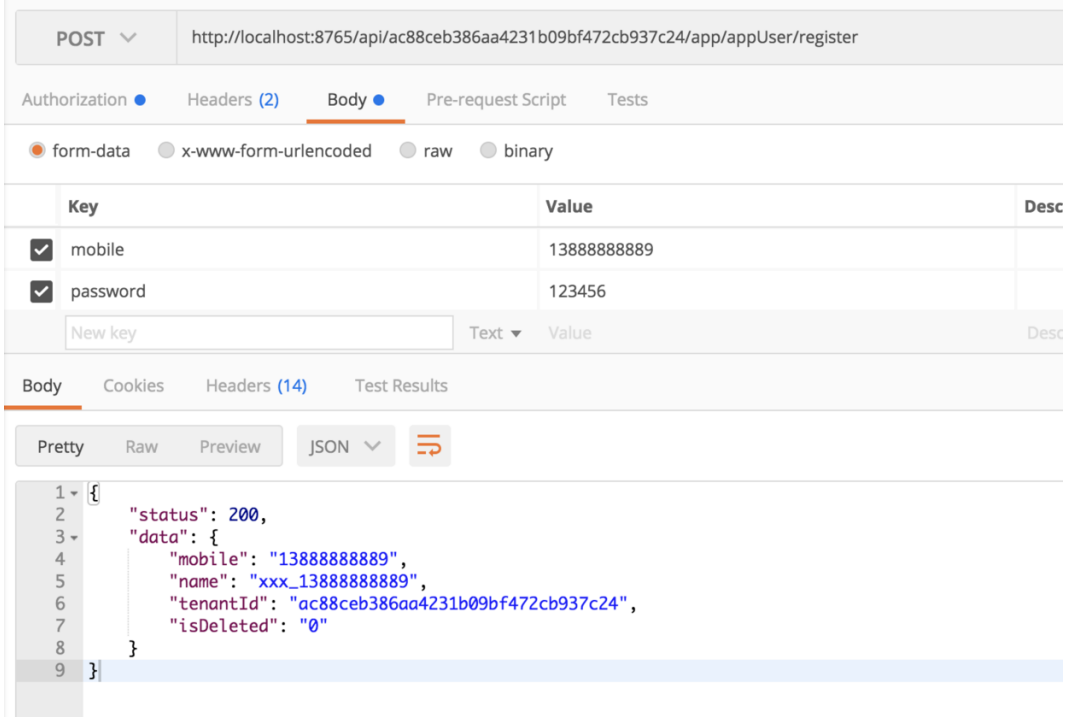
- 配置app基础认证



- 配置租户标志



- 提交相关注册信息和返回结果



登录

- 配置app基础认证（同上）
- 配置租户标志（同上）
- app用户登录