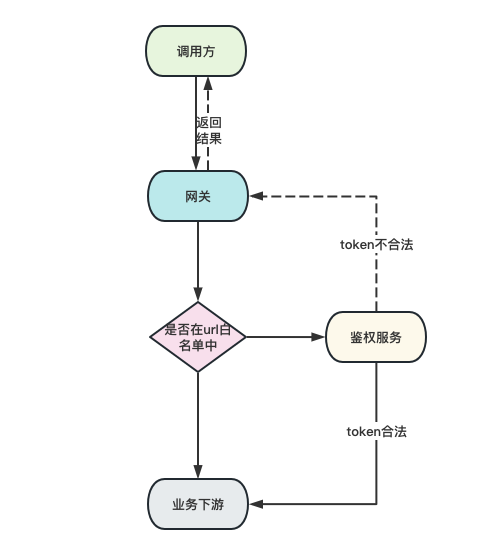
**6-22 | 网关服务引入鉴权+白名单功能**

前边我们已经将网关组件给引入了，同时也讲解了账户服务的鉴权功能的实现。那么本章节我们将会讲解如何在网关的过滤器里面去引入鉴权功能。

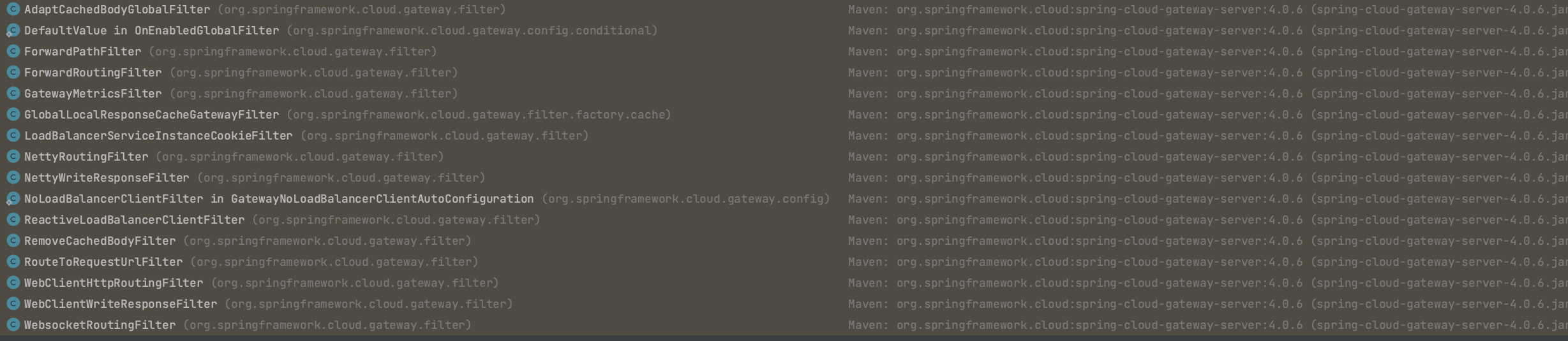
这里是我们请求的链路图：



**如何在网关层加入校验**

org.springframework.cloud.gateway.filter.GlobalFilter

Gateway网关默认有一套过滤链的设计思路，所有的请求过滤器都需要实现这个GlobalFilter接口，因此我们可以自定义一个GlobalFilter接口，然后在里面进行鉴权判断的逻辑



|  |
| --- |
| Java public interface GlobalFilter {   /\*\*  \* Process the Web request and (optionally) delegate to the next {@code GatewayFilter}  \* through the given {@link GatewayFilterChain}.  \* @param exchange the current server exchange  \* @param chain provides a way to delegate to the next filter  \* @return {@code Mono<Void>} to indicate when request processing is complete  \*/  Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain);  } |

**代码实现**

按照上边的思路，我们可以这么来设计和实现：

|  |
| --- |
| Java package org.qiyu.live.gateway.filter;  import jakarta.annotation.Resource; import org.apache.dubbo.config.annotation.DubboReference; import org.qiyu.live.account.interfaces.IAccountTokenRPC; import org.qiyu.live.common.interfaces.enums.GatewayHeaderEnum; import org.qiyu.live.gateway.properties.GatewayApplicationProperties; import org.springframework.cloud.gateway.filter.GatewayFilterChain; import org.springframework.cloud.gateway.filter.GlobalFilter; import org.springframework.core.Ordered; import org.springframework.http.HttpCookie; import org.springframework.http.server.ServletServerHttpRequest; import org.springframework.http.server.reactive.ServerHttpRequest; import org.springframework.stereotype.Component; import org.springframework.util.CollectionUtils; import org.springframework.util.StringUtils; import org.springframework.web.server.ServerWebExchange; import reactor.core.publisher.Mono;  import java.net.URI; import java.util.List;  /\*\*  \* @Author idea  \* @Date: Created in 10:57 2023/6/20  \* @Description  \*/ @Component public class AccountCheckFilter implements GlobalFilter, Ordered {   @DubboReference  private IAccountTokenRPC accountTokenRPC;  @Resource  private GatewayApplicationProperties gatewayApplicationProperties;   @Override  public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  ServerHttpRequest request = exchange.getRequest();  URI uri = request.getURI();  String urlPath = uri.getPath();  if (StringUtils.isEmpty(urlPath)) {  return Mono.empty();  }  List<String> filterUrlList = gatewayApplicationProperties.getNotCheckUrlList();  for (String urlItem : filterUrlList) {  if (urlPath.startsWith(urlItem)) {  System.out.println("满足匹配规则");  return chain.filter(exchange);  }  }  if (!CollectionUtils.isEmpty(filterUrlList) && filterUrlList.contains(urlPath)) {  System.out.println("可以不用进行token校验");  return chain.filter(exchange);  }  System.out.println("网关请求处理");  List<HttpCookie> httpCookieList = request.getCookies().get("qytk");  if (CollectionUtils.isEmpty(httpCookieList)) {  //禁止访问  System.err.println("error visit");  return Mono.empty();  }  HttpCookie httpCookie = httpCookieList.get(0);  if (httpCookie == null) {  System.err.println("error visit");  return Mono.empty();  }  Long userId = accountTokenRPC.getUserIdByToken(httpCookie.getValue());  if (userId == null) {  System.out.println("执行不通过，无法请求");  return Mono.empty();  }  ServerHttpRequest.Builder builder = request.mutate();  //把userId传递给到下游去  builder.header(GatewayHeaderEnum.USER\_LOGIN\_ID.getName(), String.valueOf(userId));  return chain.filter(exchange.mutate().request(builder.build()).build());  }   @Override  public int getOrder() {  return 0;  } } |

同时，我们可以设置一个白名单url的配置类；

|  |
| --- |
| Java package org.qiyu.live.gateway.properties;  import org.springframework.boot.context.properties.ConfigurationProperties; import org.springframework.cloud.context.config.annotation.RefreshScope; import org.springframework.context.annotation.Configuration;  import java.util.List;  /\*\*  \* @Author idea  \* @Date: Created in 11:33 2023/6/20  \* @Description  \*/ @ConfigurationProperties(prefix = "qiyu.gateway") @Configuration @RefreshScope public class GatewayApplicationProperties {   private List<String> notCheckUrlList;   public List<String> getNotCheckUrlList() {  return notCheckUrlList;  }   public void setNotCheckUrlList(List<String> notCheckUrlList) {  this.notCheckUrlList = notCheckUrlList;  }   @Override  public String toString() {  return "GatewayApplicationProperties{" +  "notCheckUrlList=" + notCheckUrlList +  '}';  } } |

白名单配置可以在nacos中进行实现：

|  |
| --- |
| Java spring:  cloud:  gateway:  discovery:  locator:  enabled: true  routes:  - id: qiyu-live-api  uri: lb://qiyu-live-api  predicates:  - Path=/live/api/\*\*  dubbo:  application:  name: qiyu-live-gateway  qos-enable: false  registry:  address: nacos://qiyu.nacos.com:8848?namespace=qiyu-live-test&&username=qiyu\_rw&&password=qiyu1867122  logging:  level:  org.springframework.cloud.gateway: INFO  reactor.netty.http.client: INFO  qiyu:  gateway:  notCheckUrlList:  - /live/api/userLogin/test |

**qiyu-live-framework-web-starter模块的开发**

定义一个web请求上下文：

|  |
| --- |
| Java package org.qiyu.live.web.starter;   import java.util.HashMap; import java.util.Map;  /\*\*  \* 旗鱼直播 用户信息拦截器  \*  \* @Author idea  \* @Date: Created in 08:58 2023/6/25  \* @Description  \*/ public class QiyuRequestContext {   private static final ThreadLocal<Map<Object, Object>> resources = new InheritableThreadLocalMap<>();   public static void put(Object key, Object value) {  if (key == null) {  throw new IllegalArgumentException("key cannot be null");  }  if (value == null) {  resources.get().remove(key);  return;  }  resources.get().put(key, value);  }   public static Long getUserId() {  Object result = get(RequestConstants.QIYU\_USER\_ID);  return result == null ? null : Long.valueOf(String.valueOf(result));  }   private static Object get(Object key) {  if (key == null) {  throw new IllegalArgumentException("key cannot be null");  }  return resources.get().get(key);  }   public static void clear() {  resources.remove();  }   private static final class InheritableThreadLocalMap<T extends Map<Object, Object>> extends InheritableThreadLocal<Map<Object, Object>> {   @Override  protected Map<Object, Object> initialValue() {  return new HashMap();  }   @Override  protected Map<Object, Object> childValue(Map<Object, Object> parentValue) {  if (parentValue != null) {  return (Map<Object, Object>) ((HashMap<Object, Object>) parentValue).clone();  } else {  return null;  }  }  }  } |

当每次有外界请求的时候，都可以将用户id往线程本地变量中进行保存：

|  |
| --- |
| Java package org.qiyu.live.web.starter;  import jakarta.servlet.http.HttpServletRequest; import jakarta.servlet.http.HttpServletResponse; import org.qiyu.live.common.interfaces.enums.GatewayHeaderEnum; import org.springframework.util.StringUtils; import org.springframework.web.servlet.HandlerInterceptor; import org.springframework.web.servlet.ModelAndView;  /\*\*  \* 旗鱼直播 用户信息拦截器  \*  \* @Author idea  \* @Date: Created in 08:48 2023/6/25  \* @Description  \*/ public class QiyuUserInfoInterceptor implements HandlerInterceptor {   //所有web请求来到这里的时候，都要被拦截  @Override  public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {  String userIdStr = request.getHeader(GatewayHeaderEnum.USER\_LOGIN\_ID.getName());  //可能是走了接口白名单，这里不做取数判断  if (StringUtils.isEmpty(userIdStr)) {  return true;  }  Long userId = Long.valueOf(userIdStr);  //把userId放入线程本地变量中  QiyuRequestContext.put(RequestConstants.QIYU\_USER\_ID, userId);  return HandlerInterceptor.super.preHandle(request, response, handler);  }   @Override  public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {  QiyuRequestContext.clear();  HandlerInterceptor.super.postHandle(request, response, handler, modelAndView);  } } |

最后要记得将相关配置装配到web上下文中：

|  |
| --- |
| Java package org.qiyu.live.web.starter;  import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.InterceptorRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  @Configuration public class WebConfig implements WebMvcConfigurer {   @Bean  public QiyuUserInfoInterceptor qiyuUserInfoInterceptor() {  return new QiyuUserInfoInterceptor();  }   @Override  public void addInterceptors(InterceptorRegistry registry) {  registry.addInterceptor(qiyuUserInfoInterceptor()).addPathPatterns("/\*\*").excludePathPatterns("/error");  }  } |

相关请求常量：

|  |
| --- |
| Java package org.qiyu.live.web.starter;  /\*\*  \* @Author linhao  \* @Date created in 8:19 上午 2021/9/6  \*/ public enum RequestConstants {   QIYU\_USER\_ID, } |

|  |
| --- |
| Java package org.qiyu.live.common.interfaces.enums;  /\*\*  \* 网关服务传递给下游的header枚举  \*  \* @Author idea  \* @Date: Created in 08:44 2023/6/25  \* @Description  \*/ public enum GatewayHeaderEnum {   USER\_LOGIN\_ID("用户id","qiyu\_gh\_user\_id");   String desc;  String name;   GatewayHeaderEnum(String desc, String name) {  this.desc = desc;  this.name = name;  }   public String getDesc() {  return desc;  }   public String getName() {  return name;  } } |

**注意点**

补全gateway网关中的header问题，否则mono.empty方法返回的时候会有跨域异常

|  |
| --- |
| Java ServerHttpResponse response = exchange.getResponse(); HttpHeaders headers = response.getHeaders(); headers.add(HttpHeaders.ACCESS\_CONTROL\_ALLOW\_ORIGIN, "http://web.qiyu.live.com:5500"); headers.add(HttpHeaders.ACCESS\_CONTROL\_ALLOW\_METHODS, "POST, GET"); headers.add(HttpHeaders.ACCESS\_CONTROL\_ALLOW\_CREDENTIALS, "true"); headers.add(HttpHeaders.ACCESS\_CONTROL\_ALLOW\_HEADERS, "\*"); headers.add(HttpHeaders.ACCESS\_CONTROL\_EXPOSE\_HEADERS, ALL); headers.add(HttpHeaders.ACCESS\_CONTROL\_MAX\_AGE, MAX\_AGE); |