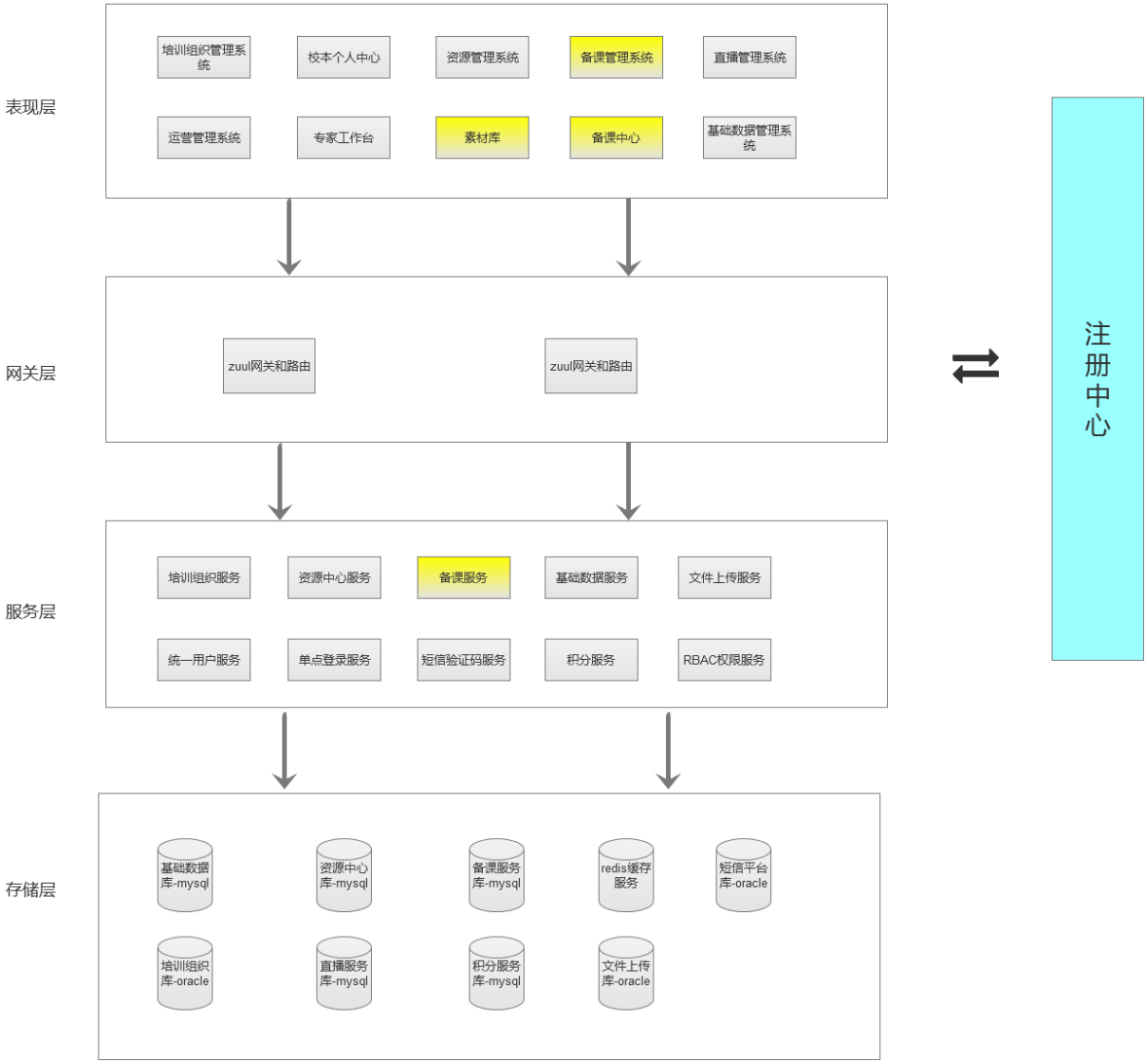
火龙果 · 整理
uml.org.cn

校本研修系统架构

--微服务架构之spring cloud

火龙果 · 整理
uml.org.cn

一、系统总体设计



火龙果 · 整理
uml.org.cn

二、为什么要微服务

单应用缺点

- 逻辑复杂
- 隔离性差、可用性差
- 模块耦合严重
- 代码臃肿

服务化优点

- 高内聚，复杂性低
- 易于拆分和管理
- 独立部署和伸缩性强
- 迭代周期短

火龙果 · 整理
uml.org.cn

三、服务化带来的问题

火龙果 · 整理
uml.org.cn

四、如何实施微服务

- 服务组件化
- 做产品的态度
- 业务拆分
- 服务调用追踪
- 去中心化处理
- 轻量化通信机制
- 基础设施自动化
- 容错设计
- 演进式设计

火龙果 · 整理
uml.org.cn

五、服务化治理框架

1、Dubbo(使用RPC通讯协议，序列化方式支持RMI、Hessian2、 dubbo2、json)

2、Spring Cloud(REST API)

六、RPC VS REST

火龙果 · 整理
uml.org.cn

1、RPC--远程过程调用，目前框架有thrift、gRPC、RMI、 Hessian、Protobuf

2、Spring Cloud(REST API)

两者比较：

1. 从使用方面看，Http接口只关注服务提供方，对于客户端怎么调用，
   调用方式怎样并不关心，而RPC服务则需要客户端接口与服务端保持一致

2.从性能角度看，由于Http携带的信息过多，导致传输速度比RPC低

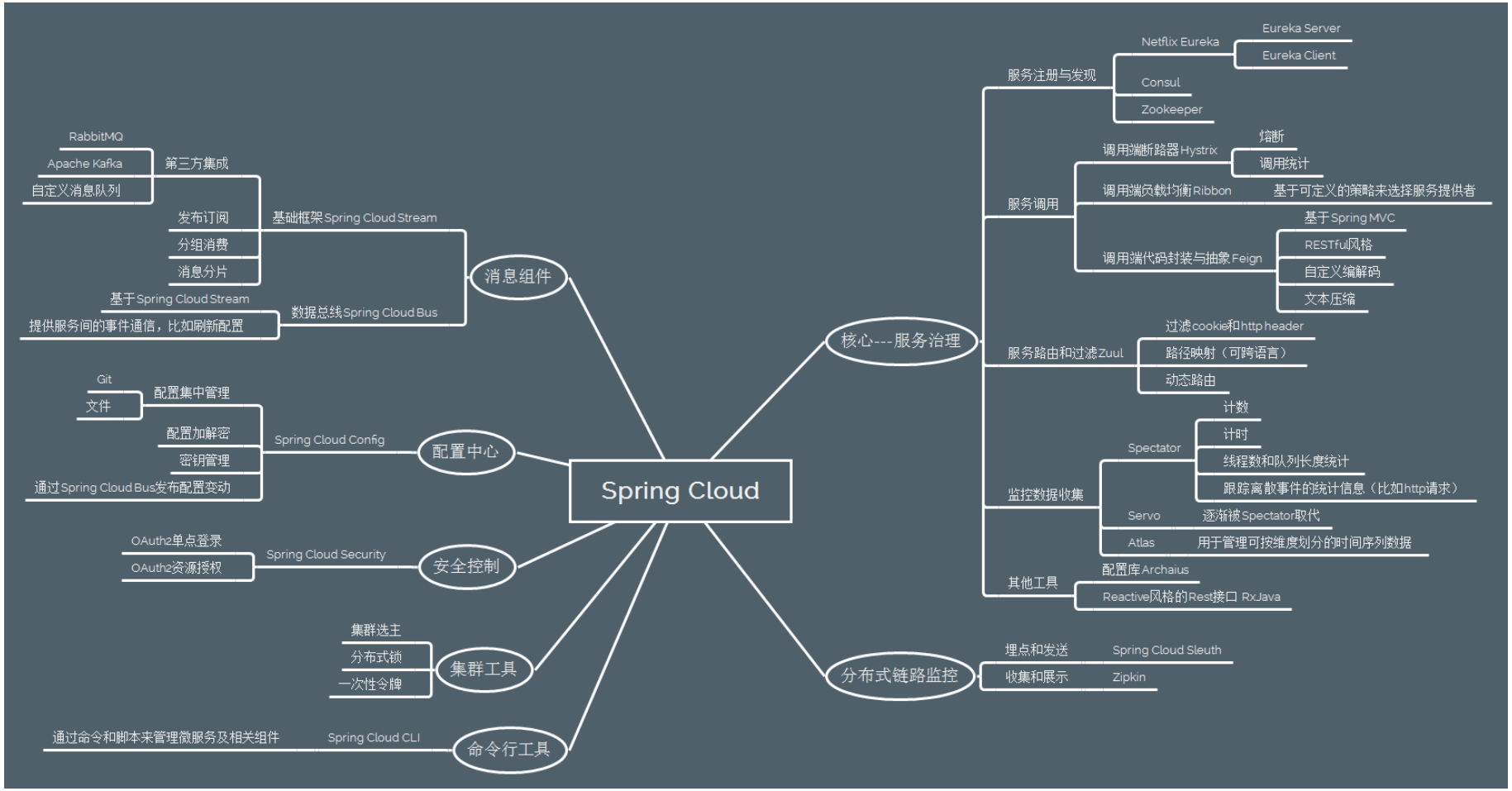
七、为什么选择Spring cloud

火龙果 · 整理
uml.org.cn

1、Dubbo 只是实现了服务治理，其它组件如配置管理和服务跟踪等组件需要依赖其它框架，使用门槛较高

2、Spring Cloud，是一个解决微服务架构实施的综合性解决框架，整合了诸多被广泛实践和证明过的框架，包括了服务治理的方方面面，如服务追踪、断路器、回退机制、消息总线、服务调用等组件。使用门槛较低

八、Spring cloud技术体系



火龙果 · 整理
uml.org.cn

SpringBoot

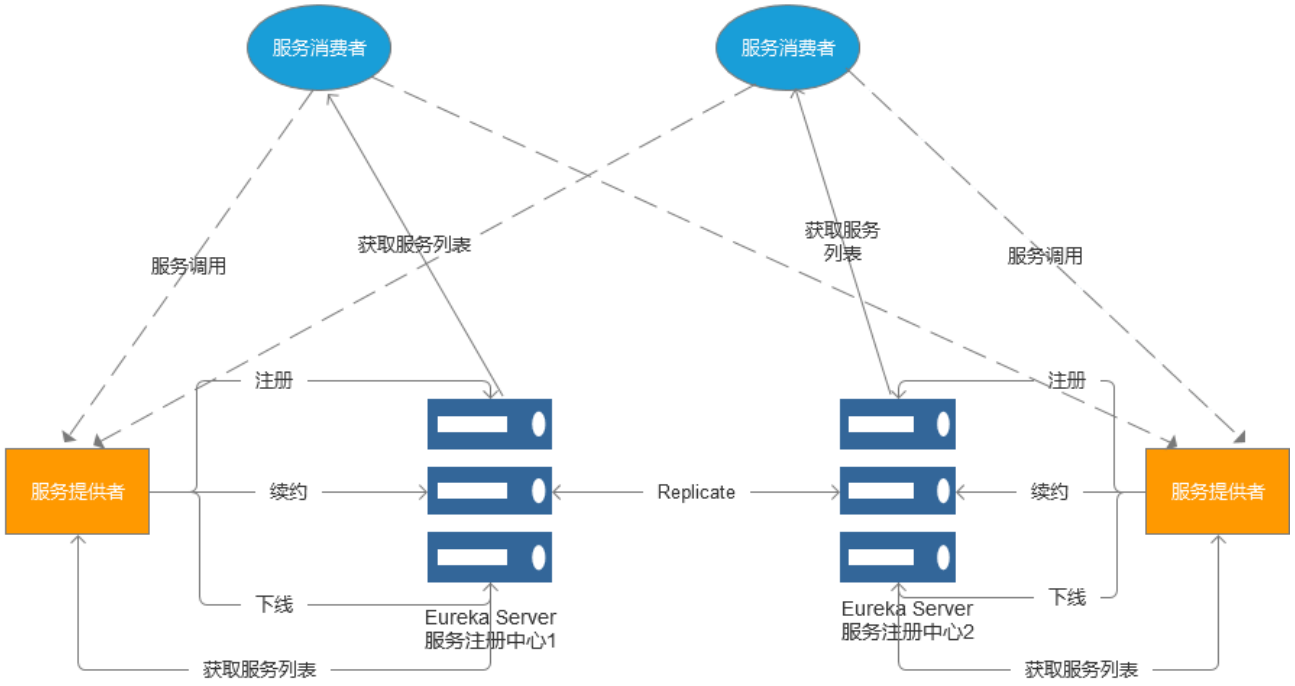
传统spring框架	Spring boot
1、配置web.xml,加载spring和spring MVC	1、大量自动化的配置，简化了原有spring的配置
2、配置数据库连接，配置spring事务	2、类似模块化的starter poms的定义，简化maven配置
3、配置加载配置文件的读取，开启注解	3、根据不同环境加载配置文件
4、部署tomcat	4、自嵌tomcat、Jetty容器，可以融入DOCKER

火龙果 · 整理
uml.org.cn

服务化治理

- 注册中心
- Spring Cloud Eureka

服务治理机制


服务消费者
服务消费者
服务调用
获取服务列表
获取服务
列表
服务调用
注册
注册
服务提供者
续约
Replicate
续约
服务提供者
下线
下线
Eureka Server
服务注册中心1
Eureka Server
服务注册中心2
获取服务列表
获取服务列表

火龙果 · 整理
uml.org.cn

Eureka Server配置

- 声明一个Eureka Server注册中心

```
package com.gzedu.ecloud.discovery.eureka;

import ...
/**
 * 使用Eureka做服务发现.
 *
 * @author xushuqin
 */
@SpringBootApplication
@EnableEurekaServer
public class EurekaApplication {
    public static void main(String[] args) {

        new SpringApplicationBuilder(EurekaApplication.class).web(true).run(args);
    }
}
```

Eureka Server配置

- 配置Eureka Server

```yaml
server:
  port: 6688
# 指定该Eureka实例的主机名
eureka:
  server:
    enableSelfPreservation: false
    evictionIntervalTimerInMs: 5000
  instance:
    hostname: 172.16.146.183
    leaseRenewalIntervalInSeconds: 10 #心跳时间
    leaseExpirationDurationInSeconds: 30 #等待下一次心跳的超时时间
    preferIpAddress: true
    instanceId: ${spring.cloud.client.ipAddress}:${spring.application.name}:${server.port}:@project.version@
  client:
    registerWithEureka: true
    fetchRegistry: false
    serviceUrl:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
    healthcheck:
      enabled: true
```

Eureka Server配置

- 如何接入Eureka Server

方式一

```
@EnableZuulProxy
@EnableHystrix
@SpringCloudApplication
@RibbonClients(defaultConfiguration = {RibbonConfig.class})
public class GatewayApplication {

    public static void main(String[] args) {
        new SpringApplicationBuilder(GatewayApplication.class).web(WebApplicationType.SERVLET).run(args);
    }

    @Bean
    public FilterRegistrationBean corsFilter() {
        final UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        final CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.addAllowedHeader("*");
        corsConfiguration.addAllowedOrigin("*");
        corsConfiguration.addAllowedMethod("*");
        source.registerCorsConfiguration( path: "/**", corsConfiguration);
        FilterRegistrationBean bean = new FilterRegistrationBean(new org.springframework.web.filter.CorsFilter(s
        bean.setOrder(0);
        return bean;
    }
```

Eureka Server配置

- 如何接入Eureka Server
  方式二

```
 */
@EnableCaching
@EnableDiscoveryClient
@EnableFeignClients(basePackages = {"com.gzedu.ecloud.frontend", "com.gzedu.ecloud.train.*", "com.gzedu.ecloud.rbac.*"
@EnableCircuitBreaker
@SpringBootApplication
@ComponentScan(basePackages = {"com.gzedu.ecloud.frontend", "com.gzedu.ecloud.train.*", "com.gzedu.ecloud.rbac.*", "co
public class TrainManageApplication {

    public static void main(String[] args) { SpringApplication.run(TrainManageApplication.class, args); }

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() { return new RestTemplate(); }
}
```

火龙果 · 整理
uml.org.cn

服务化治理

- 客户端负载均衡
- Spring Cloud Ribbon

火龙果 · 整理
uml.org.cn

什么是客户端负载均衡

- spring cloud中的ribbon，客户端会有一个服务器地址列表，在发送请求前通过负载均衡算法选择一个服务器，然后进行访问，这是客户端负载均衡；即在客户端就进行负载均衡算法分配。

火龙果 · 整理
uml.org.cn

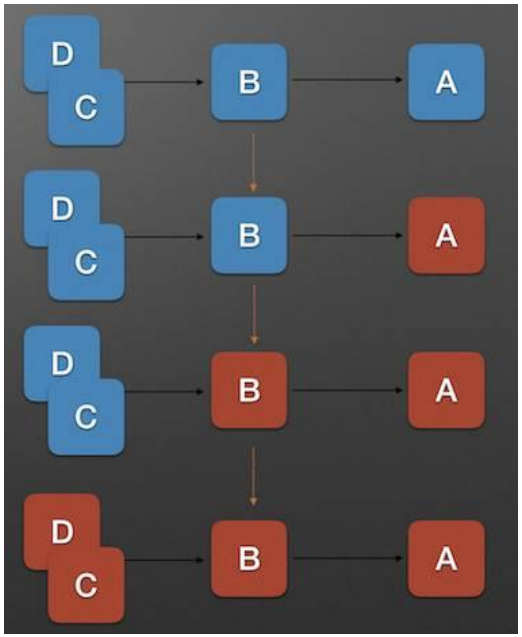
服务化治理

- 服务容错机制
- Spring Cloud Hystrix

火龙果 · 整理
uml.org.cn

服务雪崩效应

- 服务雪崩效应是一种因 服务提供者 的不可用导致 服务调用者 的不可用,并将不可用 逐渐放大 的过程.



火龙果 · 整理
uml.org.cn

服务雪崩效应的原因

- 硬件故障
- 程序Bug
- 缓存击穿
- 用户大量请求
- 同步等待造成的资源耗尽（线程资源耗尽，服务调用者提供的服务也将不可用）

火龙果 · 整理
uml.org.cn

服务雪崩效应应对措施

- 流量控制（网关限流、关闭重试）
- 改进缓存模式（同步改为异步刷新）
- 服务自动扩容
- 服务调用者降级服务（资源隔离、不可用服务调用快速失败）
- 同步等待造成的资源耗尽（线程资源耗尽，服务调用者提供的服务也将不可用）

火龙果 · 整理
uml.org.cn

Hystrix预防雪崩措施

- 资源隔离

	线程池隔离	信号量隔离
线程	与调用线程非相同线程	与调用线程相同
开销	排队、调度、上下文开销	无线程切换，开销第
异步	支持	不支持
并发	支持（最大线程池大小）	支持（最大信号量）

- 熔断器
- 命令模式

火龙果 · 整理
uml.org.cn

Hystrix熔断器使用

```
/**
 * @author liaoyujian
 */
@EnableCaching
@EnableDiscoveryClient
@EnableFeignClients(basePackages = {"com.gzedu.ecloud.frontend", "com.gzedu.ecloud.train.*", "com.gzedu.ecloud.rbac.*"})
@EnableCircuitBreaker
@SpringBootApplication
@ComponentScan(basePackages = {"com.gzedu.ecloud.frontend", "com.gzedu.ecloud.train.*", "com.gzedu.ecloud.rbac.*", "com.gzedu.e
public class TrainManageApplication {

    public static void main(String[] args) { SpringApplication.run(TrainManageApplication.class, args); }

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() { return new RestTemplate(); }
}
```

Hystrix熔断器使用

火龙果 · 整理
uml.org.cn

```java
/**
 * @author wangguopeng
 * @since 2018-01-26
 */
//TODO feign调用地址修改
@FeignClient( value = "ecloud-base-data", fallback = ProfileServiceImpl.class)
public interface ProfileService {/*http://172.16.169.192:8083  http://172.16.146.184:8083*/
    /**
     * 根据用户id获取用户的当前信息
     * @param userId
     * @return
     */
    @PostMapping("/user/manager/platformMgrDetail/{userId}")
    Result getProfile(@PathVariable("userId") String userId);

    /**
     * 修改当前用户的基本信息
     * @param params
     * @return
     */
    @PostMapping("/user/manager/editPlatformMgr")
    Result editProfile(@RequestBody Map<String, Object> params);
}
```

Hystrix熔断器使用

火龙果 · 整理
uml.org.cn

```java
import ...

/**
 * @author wangguopeng
 * @since 2018-01-26
 */
@Service
public class ProfileServiceImpl implements ProfileService {
    @Override
    public Result getProfile(String userId) {
        return ResultData.error( msg: "获取失败！");
    }

    @Override
    public Result editProfile(Map<String, Object> params) {
        return ResultData.error( msg: "修改失败！");
    }
}
```

火龙果 · 整理
uml.org.cn

Hystrix Dashboard

localhost:2001/hystrix/monitor?stream=http%3A%2F%2Flocalhost%3A9000%2Fhystrix.stream

Hystrix Stream: http://localhost:9000/hystrix.stream

Circuit

Sort: Error then Volume | Alphabetical | Volume | Error | Mean | Median | 90 | 99 | 99.5

helloKey

0 | 0 | 0.0 % → 最近10秒的错误比例
0 | 0
0

Host: 0.0/s → 请求频率
Cluster: 0.0/s

Circuit Closed → 短路状态

Hosts	1	90th	123ms
Median	74ms	99th	143ms
Mean	86ms	99.5th	143ms

→ 百分位延迟统计

Thread Pools

Sort: Alphabetical | Volume |

HelloService

Host: 0.0/s
Cluster: 0.0/s

Active	0	Max Active	0
Queued	0	Executions	0
Pool Size	10	Queue Size	5

火龙果 · 整理
uml.org.cn

服务化治理

声明式服务调用

Spring Cloud Feign

Spring Cloud Feign

火龙果 · 整理
uml.org.cn

- 只需创建一接口，并用注解的方式配置它，即可完成对服务提供的接口绑定
- 整合了Spring cloud Ribbon与Spring cloud Hystrix

Spring Cloud Feign使用

火龙果 · 整理
uml.org.cn

```java
/**
 * @author liaoyujian
 */
@EnableCaching
@EnableDiscoveryClient
@EnableFeignClients(basePackages = {"com.gzedu.ecloud.frontend", "com.gzedu.ecloud.train.*", "com.gzedu.ecloud.rbac.*"})
@EnableCircuitBreaker
@SpringBootApplication
@ComponentScan(basePackages = {"com.gzedu.ecloud.frontend", "com.gzedu.ecloud.train.*", "com.gzedu.ecloud.rbac.*", "com.gzedu
public class TrainManageApplication {

    public static void main(String[] args) { SpringApplication.run(TrainManageApplication.class, args); }

    @Bean
    @LoadBalanced
    public RestTemplate restTemplate() { return new RestTemplate(); }
}
```

Spring Cloud Feign使用

火龙果 · 整理
uml.org.cn

```java
/**
 * @author wangguopeng
 * @since 2018-01-27
 */
//TODO feign调用地址修改
@FeignClient( value = "ecloud-base-data", fallback = PasswordServiceImpl.class)
public interface PasswordService {
    /**
     * 修改用户的密码
     *
     * @param userId
     * @param oldPwd
     * @param newPwd
     * @return
     */
    @PostMapping("/user/user/changePwd/{userId}")
    Result editPassword(@PathVariable("userId") String userId,
                        @RequestParam("oldPwd") String oldPwd,
                        @RequestParam("newPwd") String newPwd);
}
```

火龙果 · 整理
uml.org.cn

服务化治理

API网关

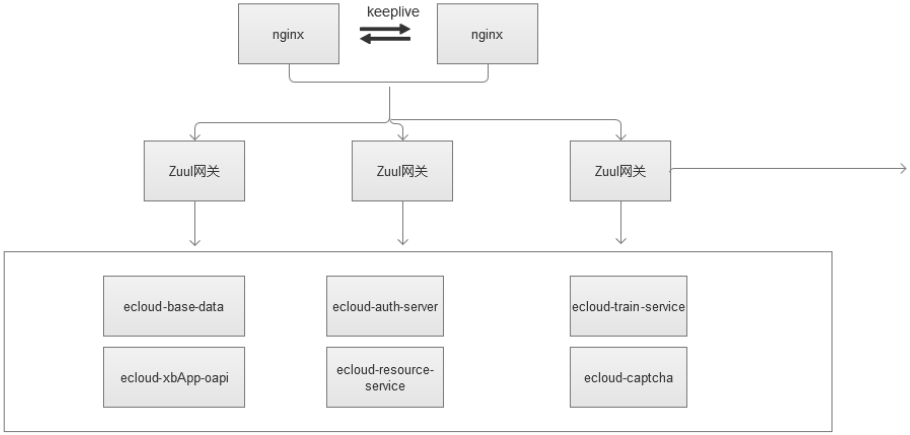
Spring Cloud Zuul

Spring Cloud Zuul

火龙果 · 整理
uml.org.cn

- Spring Cloud Zuul路由是微服务架构的不可或缺的一部分，提供动态路由，监控，弹性，安全等的边缘服务。
- 目前ecloud-gateway网关服务已支持路由、白名单、安全和缓存等服务

校本云网关架构图



路由规则表

path	服务名	说明
/xbapp/**	ecloud-xbApp-oapi	研修社区APP接口
/captcha/**	ecloud-captcha	短信验证码接口口
/baseData/**	ecloud-base-data	基础数据服务接口
/trainService/**	ecloud-train-service	培训组织服务
/oauthService/**	ecloud-oauth-server	授权登录服务
/ecloudResource/**	ecloud-resource-service	资源中心服务

签名机制

1、签名机制，为防止接口数据篡改和重复调用，增加接口参数校验机制，算法为MD5(appKey+appSecret+timestamp)，appKey是分配给客户端的ID，appSecret是分配给客户端的密钥，timestamp为unix时间戳，请求的URL有效时间为15分钟
2、增加白名单，白名单内的host或者URL不需要签名验证

cache机制(尚未实现)

1、添加指定的URL配置缓存机制
2、自定义http head头部的协议来决定是否需要缓存

防重复提交表单功能(未实现)

1、支持添加指定的URL

火龙果 · 整理
uml.org.cn

Zuul安全机制

- 签名机制，为防止接口数据篡改和重复调用，增加接口参数校验机制，sig签名算法为MD5(appKey+appSecret+timestamp)，appKey是分配给客户端的ID，appSecret是分配给客户端的密钥，timestamp为unix时间戳，请求的URL有效时间为15分钟
- Token机制，用户在登录之后会返回一个access_token，客户端在访问需要登录之后才能访问的资源，需要在在Authorization头部使用Bearer模式新增token，如head(“Authorization”,” Bearer token”)

火龙果 · 整理
uml.org.cn

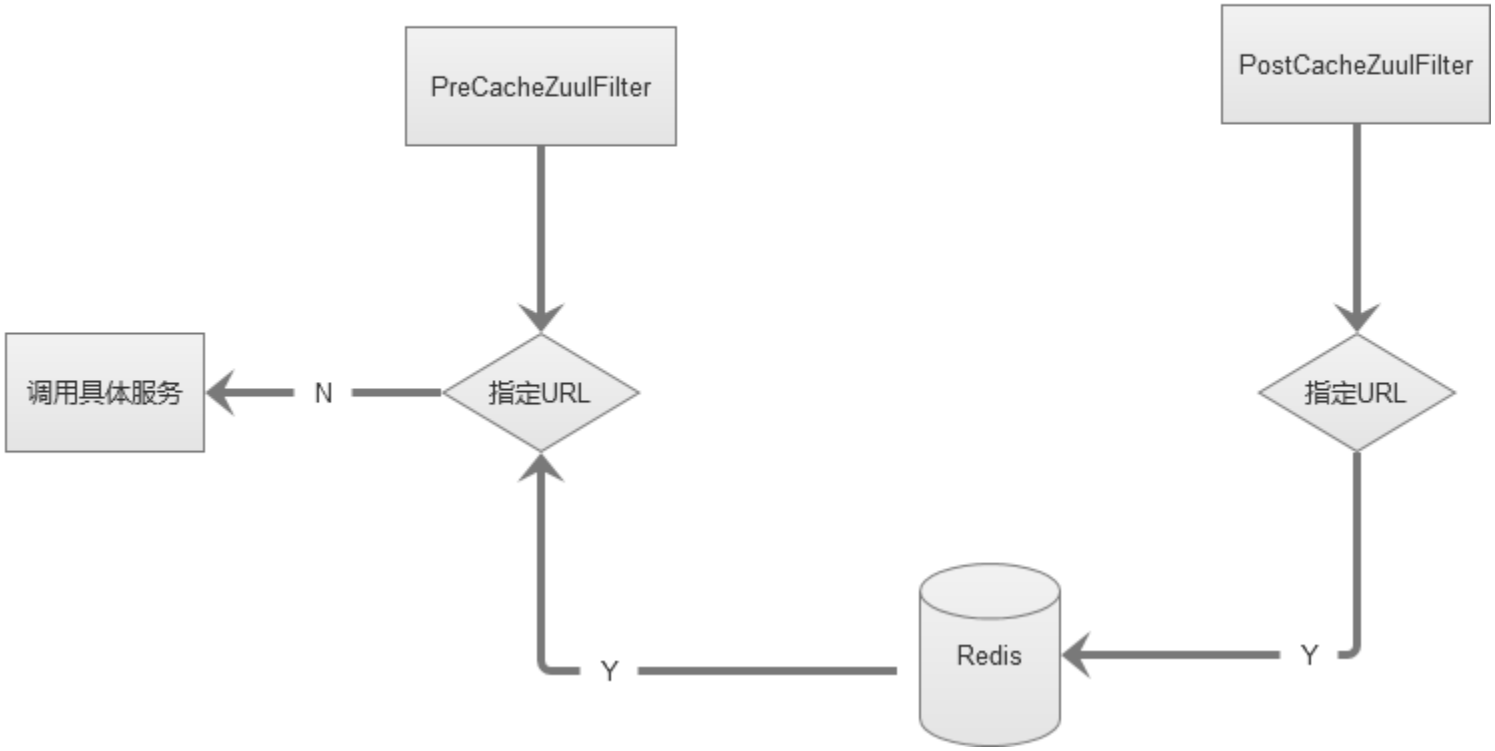
ZuulFilter类型

- PRE
- Route
- Post
- Error

Zuul servlet执行过滤类型顺序

火龙果 · 整理
uml.org.cn

Zuul缓存实现



火龙果 · 整理
uml.org.cn

Zuul缓存实现

```java
public class PreCacheZuulFilter extends ZuulFilter {
    private Logger logger = LoggerFactory.getLogger(PreCacheZuulFilter.class);
    @Autowired
    private ZuulCacheService zuulCacheService;

    @Override
    public String filterType() { return PRE_TYPE; }

    @Override
    public int filterOrder() { return PRE_DECORATION_FILTER_ORDER + 1; }

    @Override
    public boolean shouldFilter() {
        String uri = RequestContext.getCurrentContext().getRequest().getRequestURI();
        GatewayCache gc = zuulCacheService.getGatewayCacheByUrl(uri);
        if (gc == null) {
            return false;
        }
        return true;
    }

    @Override
    public Object run() throws ZuulException {
        HttpServletRequest request = RequestContext.getCurrentContext().getRequest();
        String uri = request.getRequestURI();
        Map<String, String[]> parameterMap = request.getParameterMap();
        CacheKey ck = zuulCacheService.createKey(uri, parameterMap, request.getMethod());
        try {
            String realKey = Lang.md5(ck.getKey().getBytes(), salt: null, hashIterations: 1);
            String jsonData = zuulCacheService.get(ck.getPath() + realKey);
            if (Lang.isEmpty(jsonData)) {
                return null;
            }
            logger.info("get data from cache,[key={},realKey={},path={}]", ck.getKey(), realKey,ck.getPath());
            setResponse(jsonData);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    private void setResponse(String jsonData) {
        RequestContext ctx = RequestContext.getCurrentContext();
        ctx.setSendZuulResponse(false);
        ctx.setResponseBody(jsonData);
        ctx.getResponse().setContentType("application/json;charset=UTF-8");
        ctx.getResponse().addHeader( s: "CACHE", s1: "GATEWAY CACHE");
```

火龙果 · 整理
uml.org.cn

Zuul缓存实现

```java
public class PostCacheZuulFilter extends ZuulFilter {
    private Logger logger = LoggerFactory.getLogger(PostCacheZuulFilter.class);
    @Autowired
    private ZuulCacheService zuulCacheService;

    @Override
    public String filterType() { return POST_TYPE; }

    @Override
    public int filterOrder() { return SEND_RESPONSE_FILTER_ORDER - 2; }

    @Override
    public boolean shouldFilter() {
        HttpServletRequest request = RequestContext.getCurrentContext().getRequest();
        String uri = RequestContext.getCurrentContext().getRequest().getRequestURI();
        GatewayCache gc = zuulCacheService.getGatewayCacheByUrl(uri);
        if (gc == null || !Strings.equalsIgnoreCase(request.getMethod(), gc.getMethod())) {
            return false;
        }
        return true;
    }

    @Override
    public Object run() throws ZuulException {
        HttpServletRequest request = RequestContext.getCurrentContext().getRequest();
        String uri = RequestContext.getCurrentContext().getRequest().getRequestURI();
        try {
            ObjectMapper mapper = new ObjectMapper();
            InputStream stream = RequestContext.getCurrentContext().getResponseDataStream();
            if (stream == null) {
                return null;
            }
            String body = StreamUtils.copyToString(stream, Charset.forName("UTF-8"));
            ResultData resultData = mapper.readValue(body, ResultData.class);
            if (resultData.getStatus() == BaseConstant.STATUS_NORMAL) {
                Map<String, String[]> parameterMap = request.getParameterMap();
                CacheKey ck = zuulCacheService.createKey(uri, parameterMap, request.getMethod());
                String realKey = Lang.md5(ck.getKey().getBytes(), salt: null, hashIterations: 1);
                GatewayCache gc = zuulCacheService.getGatewayCacheByUrl(uri);
                logger.info("write data to redis cache,[key={},realKey={},path={}]", ck.getKey(), realKey,ck.getPath());
                zuulCacheService.put( key: ck.getPath() + realKey, body, gc.getExpired());
                setResponse(body);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
```

火龙果 · 整理
uml.org.cn

服务化治理

- Auth认证服务
- Spring Cloud Security

OAUTH2.0协议

- OAuth 2 是一个授权框架，或称授权标准，它可以使第三方应用程序或客户端获得对HTTP服务上（例如 Google，GitHub ）用户帐户信息的有限访问权限。OAuth 2 通过将用户身份验证委派给托管用户帐户的服务以及授权客户端访问用户帐户进行工作。综上，OAuth 2 可以为 Web 应用 和桌面应用以及移动应用提供授权流程。

OAUTH2.0协议

火龙果 · 整理
uml.org.cn

用户认证

用户授权

OAUTH2.0 角色

- 资源所有者（Resource Owner）用户
- 资源服务器（Resource Server）ecloud-oauth-server
- 授权服务器（Authorization Server）ecloud-oauth-server
- 客户端（Client）第三方应用

火龙果 · 整理
uml.org.cn

OAUTH2.0使用场景

- 授权码模式（authorization code）
- 简化模式（implicit）
- 密码模式（resource owner password credentials）
- 客户端模式（client credentials）

OAUTH2.0 授权码模式

火龙果 · 整理
uml.org.cn

请求授权

http://localhost:8080/oauth/authorize?response_type=code&client_id=demoApp&redirect_uri=http://xxx.com

携带code获取token

curl -i –d "grant_type=authorization_code&code=p1ancF&client_id=demoApp&client_secret=demoAppSecret“ -X POST http://localhost:8080/oauth/token

携带token请求资源

curl -i http://localhost:8080/api/blog/1?access_token=c80408d4-5afb-4f87-9538-9fb45b149941

JWT(Json web token)

- JWT(json web token)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。 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

- 基于session的登录认证 ,在传统的用户登录认证中，因为http是无状态的，所以都是采用session方式。用户登录成功，服务端会保证一个session，当然会给客户端一个sessionId，客户端会把sessionId保存在cookie中，每次请求都会携带这个sessionId

JWT 编码格式

- yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiend6IiwiYWdlIjoiMTgifQ.UQmqAUhUrpDVV2ST7mZKyLTomVfg7sYkEjmdDI5XF8Q

火龙果 · 整理
uml.org.cn

JWT 编码格式

Head playload

- yJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJuYW1lIjoiend6IiwiYWdlIjoiMTgifQ
  .UQmqAUhUrpDVV2ST7mZKyLTomVfg7sYkEjmdDI5XF8Q

signature

Signature=HMACSHA256((base64 (header) + '.' + base64 (payload)),’secret’);

火龙果 · 整理
uml.org.cn

持续集成与灰度发布

- 整合JENKINS
- 整合DOCKER

火龙果 · 整理
uml.org.cn

Thank You!