Contents

[建立MAVEN项目 4](#_Toc530556279)

[建立公共子模块 6](#_Toc530556280)

[搭建用户认证中心 7](#_Toc530556281)

[</project> 7](#_Toc530556282)

[前端项目 8](#_Toc530556283)

[将前端项目注册到服务中心 9](#_Toc530556284)

[搭建服务注册中心 9](#_Toc530556285)

[搭建网关 13](#_Toc530556286)

[给前端添加断路器 14](#_Toc530556287)

[添加配置中心 15](#_Toc530556288)

[将front配置放到配置中心 16](#_Toc530556289)

[改进：将配置中心注册到服务中心 17](#_Toc530556290)

[服务追踪 18](#_Toc530556291)

[断路器监控 19](#_Toc530556292)

[断路器聚合监控 21](#_Toc530556293)

[添加email 22](#_Toc530556294)

[添加redis 25](#_Toc530556295)

[打包 27](#_Toc530556296)

[附录 28](#_Toc530556297)

[公用模块zzyboot-api 28](#_Toc530556298)

[ZzyColumn 28](#_Toc530556299)

[ZzyColumns 28](#_Toc530556300)

[ZzyEntityParent 30](#_Toc530556301)

[ZzyUtil 36](#_Toc530556302)

[ZzyRepository 39](#_Toc530556303)

[ZzyRepositoryCustom 39](#_Toc530556304)

[ZzyRepositoryImpl 40](#_Toc530556305)

[ZzyService 51](#_Toc530556306)

[ZzyServiceImpl 51](#_Toc530556307)

[用户认证中心zzyboot-usercenter 54](#_Toc530556308)

[Application 54](#_Toc530556309)

[ZzyUtilUserCenter 54](#_Toc530556310)

[UserController 54](#_Toc530556311)

[前端项目 58](#_Toc530556312)

[Application 58](#_Toc530556313)

[DES 58](#_Toc530556314)

[Jiami 65](#_Toc530556315)

[ZzyUtil 66](#_Toc530556316)

[ZzyController 70](#_Toc530556317)

[application.properties 73](#_Toc530556318)

[static 73](#_Toc530556319)

[index.html 73](#_Toc530556320)

[zzyLogin.html 74](#_Toc530556321)

[nativetoascii.html 75](#_Toc530556322)

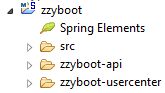
[Bootstrap 76](#_Toc530556323)

[Jquery 76](#_Toc530556324)

[Css 77](#_Toc530556325)

[js 77](#_Toc530556326)

[business 77](#_Toc530556327)



1 实现语言全球化

2 实现客户端数据绑定，如数据改变，自动将相关页面刷新，例，选“中文”，页面内容自动变成中文

3 实现数据安全传输

1 DH算法 生成共有密钥

2 sha256（客户端，服务器）将密钥变为256位

客户端： sha256.js

服务端： Jiami.java

3 DES算法TRIPLE（客户端，服务器）密钥在服务端存在session中，客户端放在本地，所有网上数据均加密

客户端: tripledes.js, mode-ecb.js

服务器：DES.java

客户端数据要将+替换：result = result.replace(/\+/g,'%2B');

服务端在解密客户端数据之前要做数据转换：

param = URLDecoder.decode(param,"utf8");

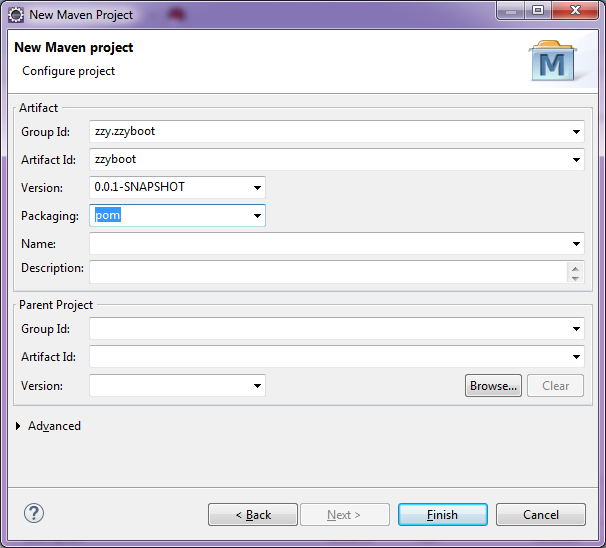
4 实现用户登录

1 建立用户身份认证服务

根据用户邮箱，电话，密码找到用户ID，生成令牌

根据用户ID，令牌验证用户合法性

# 建立MAVEN项目



POM:

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>zzy.zzyboot</groupId>

<artifactId>zzyboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>pom</packaging>

<name>zzyboot</name>

<description>zzy spring boot project</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.0.6.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

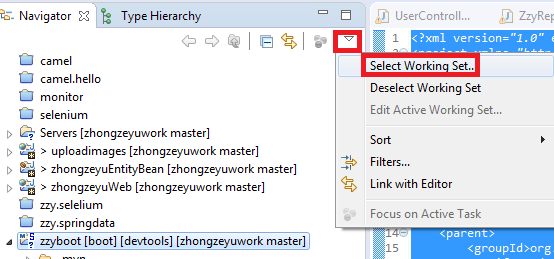
</plugin>

</plugins>

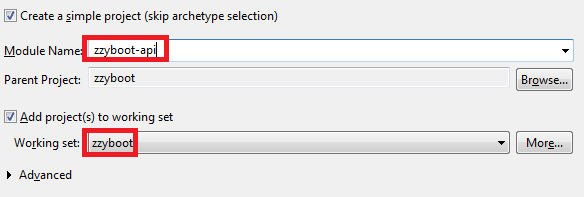
</build>

</project>

将zzyboot添加到working set



# 建立公共子模块



Pom:

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>zzy.zzyboot</groupId>

<artifactId>zzyboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

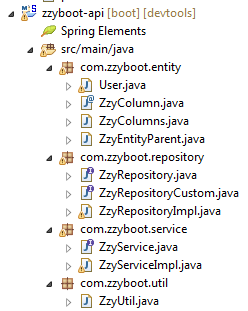
</parent>

<artifactId>zzyboot-api</artifactId>

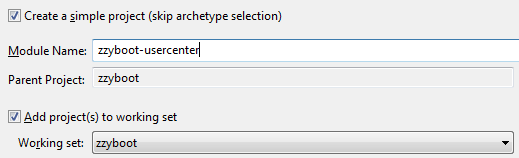
</project>

父模块zzyboot自动添加子模块

公用方法，明细见附录



# 搭建用户认证中心



添加公共模块

POM:

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>zzy.zzyboot</groupId>

<artifactId>zzyboot</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>zzyboot-usercenter</artifactId>

<dependencies>

<dependency><!-- 引入自定义的api通用包,可以使用Det部门Entity -->

<groupId>zzy.zzyboot</groupId>

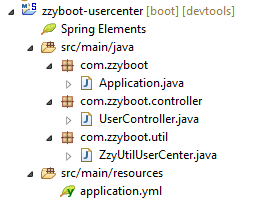
<artifactId>zzyboot-api</artifactId>

<version>${project.version}</version>

</dependency>

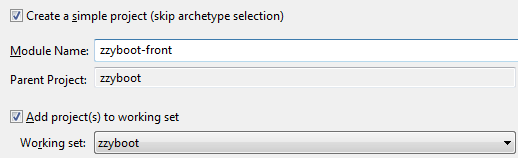
</dependencies>

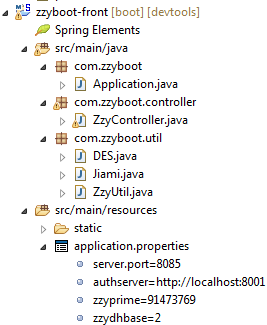
## </project>



明细见附录

# 前端项目

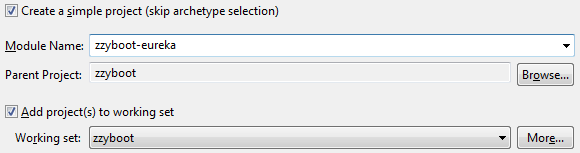




明细见附录

# 将前端项目注册到服务中心

## 搭建服务注册中心



添加Eureka Server依赖

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>

spring-cloud-starter-netflix-eureka-server

</artifactId>

</dependency>

**主类**

**package** com.zzyboot;

**import** org.springframework.boot.SpringApplication;

**import** org.springframework.boot.autoconfigure.SpringBootApplication;

**import** org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

}

**application.yml**

server:

port: 7001

spring:

application:

name: servicecenter

eureka:

instance:

hostname: localhost #可以设置DNS名

appname: servicecenter

client:

register-with-eureka: **false** #单机版 false 表示不向服务中心注册自己

fetch-registry: **false** #单机版 false 表示自己就是注册中心，不需要去检索服务

service-url:

defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/

#defaultZone: http://192.168.0.69: 7001/eureka/, http://192.168.0.9:7001/eureka/

搭建集群例：

application-registry1.yml

server:

port: 9001

spring:

application:

name: service-registry

server.

eureka:

instance:

hostname: service-registry1 #主机名称

appname: service-registry

client:

service-url:

defaultZone: http://service-registry2:9002/eureka,http://service-registry3:9003/eureka

#服务注册中心相互注册一定要显示的设置register-with-eureka 和fetch-registry的值为true,否则会服务不可用

register-with-eureka: **true**

fetch-registry: **true**

application-registry2.yml

与1几同，主机名称用2，defaultZone将2改成1

application-registry3.yml

与1几同，主机名称用3，defaultZone将3改成1

修改操作系统的host

windows电脑，在c:/windows/systems/drivers/etc/hosts 修改。

linux系统通过vim /etc/hosts

#配置服务注册中心

127.0.0.1 service-registry1

127.0.0.1 service-registry2

127.0.0.1 service-registry3

打包运行：

java -jar service-registry-0.0.1-SNAPSHOT.jar --spring.profiles.active=registry1

java -jar service-registry-0.0.1-SNAPSHOT.jar --spring.profiles.active=registry2

java -jar service-registry-0.0.1-SNAPSHOT.jar --spring.profiles.active=registry3

注意：集群时，

1.服务注册中心集群相互注册一定要开启  
register-with-eureka: true  
fetch-registry: true  
2.服务注册中心集群的spring.application.name一定要一样  
3.eureka.client.service-url.defaultZone:不能出现 localhost,一定要使用host指定主机名

前端zzyboot-usercenter添加 Eureka Discovery依赖

application.yml

eureka:

client:

service-url:

defaultZone: http://localhost:7001/eureka/

instance:

instance-id: zzyusercenter

prefer-ip-address: **true** #访问路径显示为IP



前端zzyboot-front 添加 Eureka Discovery依赖

eureka:

client:

service-url:

defaultZone: http://localhost:7001/eureka/

instance:

instance-id: front

prefer-ip-address: **true** #访问路径显示为IP



1 使用ribbon + restTemplete远程调用zzyusercenter

添加 Ribbon依赖

application.yml

修改，将IP地址用服务名称代替：authserver: http://zzyusercenter

Ribbon会自动将服务名称转成IP地址

添加RIBBON注解@LoadBalanced, 这样，一旦zzyusercenter是个集群，ribbon实现负载均衡

@Configuration

**public** **class** ConfigBean {

@Bean

@LoadBalanced

**public** RestTemplate getRestTemplate(){

**return** **new** RestTemplate();

}

}

2 使用Feign远程调用zzyusercenter

Feign自动集成Ribbon, 同时整合Hystrix,具有熔断能力

添加 Feign依赖

主类添加注解@EnableFeignClients

**添加远程zzyusercenter调用服务**

**package** com.zzyboot.service;

**import** org.springframework.cloud.openfeign.FeignClient;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

@FeignClient(value = "zzyusercenter")

**public** **interface** ZzyUserCenterService {

@PostMapping(value = "/usercheck")

String usercheck(@RequestBody String param);

}

Controller 添加该服务

@Autowired

ZzyUserCenterService zzyUserCenterService;

//@Autowired

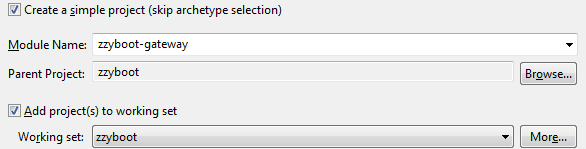
//RestTemplate restTemplate;

String userinfo = zzyUserCenterService.usercheck(param);

//restTemplate.postForObject(authserver + "/usercheck", param, String.class);

去除所有ribbon 和 restTemplate痕迹

## 搭建网关



路由功能： 将外部请求转发到具体的微服务实例上

过滤功能：

代理：Zuul与Eureka整合，将Zuul注册为Eureka服务治理下的应用，同时从Eureka中获得其他微服务的消息，以后的访问微服务通过Zuul跳 转。

添加Zuul，Eureka Discovery依赖

主类添加@EnableZuulProxy

Application.yml, 将前端路由到erp表示所有erp请求会转到front

server:

port: 8888

spring:

application:

name: gateway

eureka:

client:

service-url:

defaultZone: http://localhost:7001/eureka/

instance:

instance-id: gateway

prefer-ip-address: **true** #访问路径显示为IP

zuul:

routes:

user:

path: /erp/\*\*

serviceId: front

有可能的话，在C:\windows\system32\drivers\etc\hosts 添加以下内容127.0.0.1 zzy.com

测试：<http://localhost:8888/erp/> 自动跳转到 http://localhost:8085/

# 给前端添加断路器

这里只给Feign实现断路器功能代码

Feign自带断路器功能，在配置文件中打开断路器功能

feign:

hystrix:

enabled: **true**

ZzyUserCenterService:

@FeignClient(value = "zzyusercenter", fallback = ZzyUserCenterServiceHystric.**class**)

**public** **interface** ZzyUserCenterService {

@PostMapping(value = "/usercheck")

String usercheck(@RequestBody String param);

}

ZzyUserCenterServiceHystric:

**package** com.zzyboot.service;

**import** org.springframework.stereotype.Component;

**import** com.zzyboot.util.ZzyCommon;

@Component

**public** **class** ZzyUserCenterServiceHystric **implements** ZzyUserCenterService {

@Override

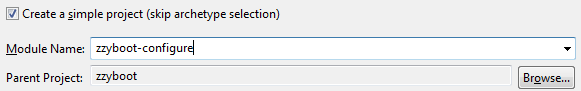
**public** String usercheck(String param) {

**return** ZzyCommon.***ZZYFAIL***+"can not access to ZzyUserCenter";

}

}

# 添加配置中心



添加依赖Config Server

主类添加注解：@EnableConfigServer

Application.yml

server:

port: 3344

spring:

application:

name: configurecenter

cloud:

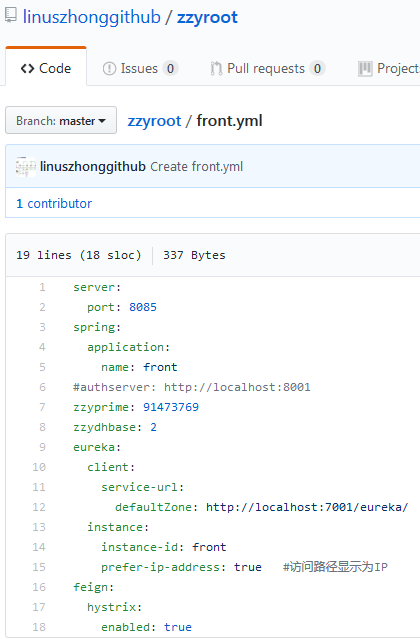
config:

server:

git:

uri: https://github.com/linuszhonggithub/zzyroot.git

## 将front配置放到配置中心



添加依赖Config Client

添加bootstrap.yml

spring:

application:

name: front

cloud:

config:

uri: http://localhost:3344

label: master

profile: default

通过配置中心<http://localhost:3344找到front.yml>

## 改进：将配置中心注册到服务中心

这样通过服务名而非IP地址访问之

Zzyboot-config: 注入依赖Eureka Discovery

Application.yml: 添加认证中心

server:

port: 3344

spring:

application:

name: configcenter

cloud:

config:

server:

git:

uri: https://github.com/linuszhonggithub/zzyroot.git

eureka:

client:

service-url:

defaultZone: http://localhost:7001/eureka/

instance:

instance-id: configcenter

prefer-ip-address: **true** #访问路径显示为IP



客户端修改 zzyboot-front: bootstrap.jml, 添加eureka, 更改config配置

spring:

application:

name: front

cloud:

config:

#uri: http://localhost:3344

label: master

profile: default

discovery:

enabled: **true**

serviceId: configcenter

eureka:

client:

service-url:

defaultZone: http://localhost:7001/eureka/

instance:

instance-id: front

prefer-ip-address: **true** #访问路径显示为IP

# 服务追踪

Zipkin下载地址<https://dl.bintray.com/openzipkin/maven/io/zipkin/java/zipkin-server/>

启动zipkin服务器： java –jar zipkin-server-2.11.7-exec.jar

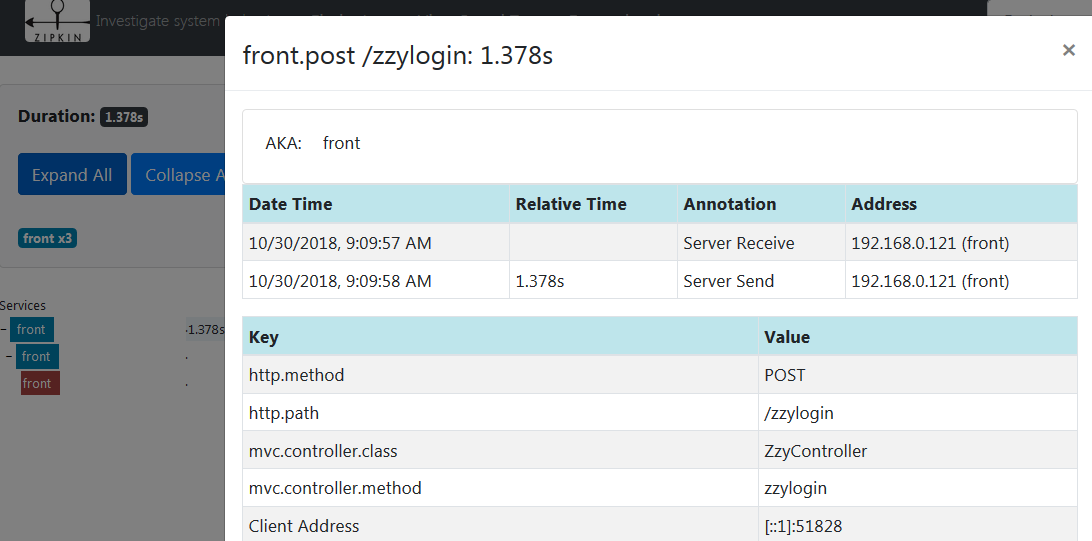
追踪zzyboot-front和项目zzyboot-usercenter:添加依赖 Zipkin Client

spring

zipkin:

base-url: <http://localhost:9411>

<http://localhost:9411>进入管理界面,可以查看服务运行时间



# 断路器监控

Zzyboot-front添加依赖 Actuator, Hystrix, Hystrix Dashboard

主类打开注解：@EnableCircuitBreaker //启动断路器

@EnableHystrixDashboard // 开启dashboard，通过图形化的方式监控: 查看 http://127.0.0.1:8085/hystrix.stream

添加HystrixMetricsStreamServlet

**public** **class** Application {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(Application.**class**, args);

}

@Bean

**public** ServletRegistrationBean getServlet() {

HystrixMetricsStreamServlet streamServlet = **new** HystrixMetricsStreamServlet();

ServletRegistrationBean registrationBean = **new** ServletRegistrationBean(streamServlet);

registrationBean.setLoadOnStartup(1);

registrationBean.addUrlMappings("/hystrix.stream");

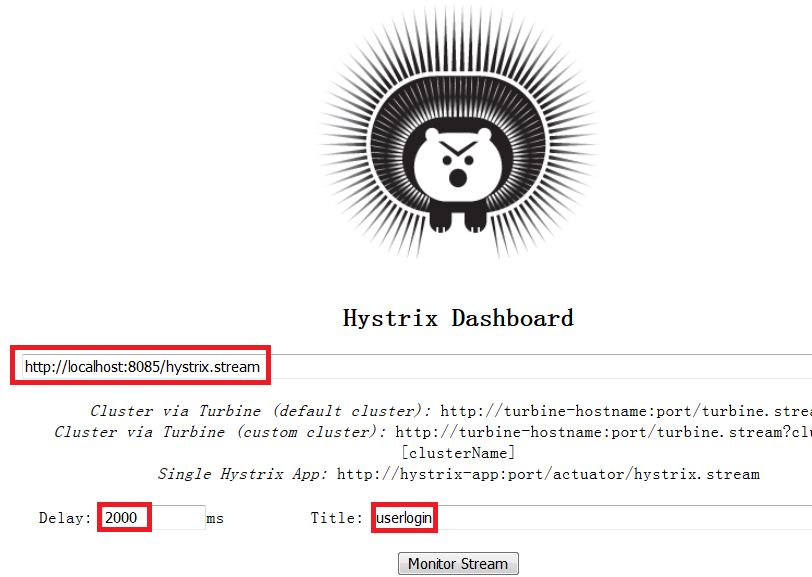
registrationBean.setName("HystrixMetricsStreamServlet");

**return** registrationBean;

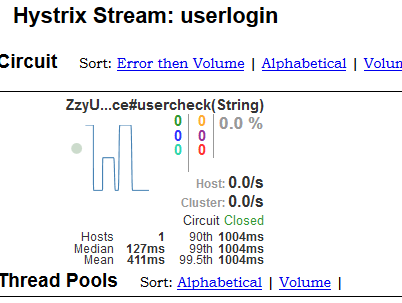
}

}

<http://localhost:8085/hystrix>



点Monitor Stream



# 断路器聚合监控

Hystrix Dashboard只能监控某个服务，Hystrix Turbine整合每个服务的Hystrix Dashboard数据

创建项目zzyboot-turbine,引入依赖：Actuator, Hystrix, Hystrix Dashboard, Eureka Discovery,Turbine

主类添加注解：

@EnableHystrix

@EnableHystrixDashboard

@EnableCircuitBreaker

@EnableTurbine

…未成功

# 添加email

Zzyemail依赖： Mail, Eureka Discovery, ZzyCommon

配置：

server:

port: 8002

spring:

application:

name: zzymail

mail:

host: smtp.gmail.com

port: 587

username: kuzcosys@gmail.com

password: gwhwdkhth

default-encoding: UTF-8

properties:

mail.smtp.auth: **true**

mail.smtp.starttls.enable: **true**

redis:

host: localhost

database: 0

port: 6379

jedis:

pool:

max-active: 3

max-wait: -1

max-idle: 8

min-idle: 0

eureka:

client:

service-url:

defaultZone: http://192.168.0.69:7001/eureka/

instance:

instance-id: zzymail

prefer-ip-address: **true** #访问路径显示为IP

mailfrom: linuszhong@hotmail.com

MainService

@Component

**public** **class** MainService{

@Autowired

**private** JavaMailSender mailSender;

@Value("${mailfrom}")

**private** String mailfrom;

**public** String sendAttachmentsMail(String from, String to,String cc, String bcc, String subject, String content, String filePath){

**if**(from == **null** || from.length() < 1){

from = mailfrom;

}

**if**(to == **null** || to.length() < 1){

**return** ZzyCommon.***ZZYFAIL\_EMAIL\_NOTO***;

}

**if**(!ZzyCommon.*isEmail*(from)){

**return** ZzyCommon.***ZZYFAIL\_EMAIL\_WRONGFROM***;

}

String[] toA = to.split(ZzyCommon.***STREMAIL***);

**boolean** hasTo = **false**;

**for**(**int** i = 0; i < toA.length; i++){

**if**(toA[i].length() < 1){

**continue**;

}

**if**(!ZzyCommon.*isEmail*(toA[i])){

**return** ZzyCommon.***ZZYFAIL\_EMAIL\_WRONGTO***+" " + toA[i];

}

hasTo = **true**;

}

**if**(!hasTo){

**return** ZzyCommon.***ZZYFAIL\_EMAIL\_NOTO***;

}

MimeMessage message = mailSender.createMimeMessage();

MimeMessageHelper helper;

**try** {

helper = **new** MimeMessageHelper(message,**true**);

helper.setFrom(from);

helper.setTo(toA);

**if**(cc != **null** && cc.length() > 0){

helper.setCc(cc.split(ZzyCommon.***STREMAIL***));

}

String[] bccA;

**if**(bcc == **null** || bcc.length() < 1){

bccA = **new** String[]{mailfrom};

}**else**{

bcc+=ZzyCommon.***STREMAIL***+mailfrom;

bccA = bcc.split(ZzyCommon.***STREMAIL***);

}helper.setBcc(bccA);

helper.setSubject(subject);

System.***out***.println("mail send 3");

helper.setText(content,**true**);

**if**(filePath!=**null** && filePath.length() > 0){

FileSystemResource file = **new** FileSystemResource(**new** File(filePath));

String fileName = filePath.substring(filePath.lastIndexOf(File.***separator***));

helper.addAttachment(fileName, file);

}

mailSender.send(message);

} **catch** (MessagingException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** ZzyCommon.***ZZYSUCCESS***;

}

}

Controller:

@RestController

**public** **class** MainController {

@Autowired

MainService mainService;

@Autowired

**private** StringRedisTemplate redisTemplate;

@PostMapping("/mail")

**public** String mail(@RequestBody String param){

**try** {

param = URLDecoder.*decode*(param,"UTF-8");

} **catch** (UnsupportedEncodingException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

System.***out***.println("mail param is " + param);

String[] paramA = param.split(ZzyCommon.***STRSEPLINE***);

//check if session is valid

String sessioninfo = paramA[paramA.length - 1];

System.***out***.println("sessioninfo is " + sessioninfo);

String[] sessioninfoA = sessioninfo.split(ZzyCommon.***STRSEPITEM***);

System.***out***.println("sessioninfoA length is " + sessioninfoA.length);

String username = sessioninfoA[sessioninfoA.length - 2];

String token = sessioninfoA[sessioninfoA.length - 1];

**if**(!ZzyCommon.*tokenvalid*(username, token,redisTemplate)){

System.***out***.println("token not exists");

**return** ZzyCommon.***ZZYFAIL\_USERINVALID***;

}

**int** index = 0;

String from = paramA[index++];

String to = paramA[index++];

String cc = paramA[index++];

String bcc = paramA[index++];

String subject = paramA[index++];

String content = paramA[index++];

String filename = paramA[index++];

**return** mainService.sendAttachmentsMail(from, to, cc, bcc, subject, content, filename);

}

}

# 添加消息

Zzymail:将所有的email请求放到消息总线，再从总线中获取发送邮件，这样可以达到异步效果

## 配置exchange , queue

打开RabbitMQ: localhost:15672

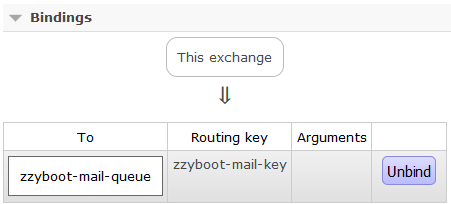
Login: linus

Password: linusmq

添加queue: zzyboot-mail-queue

添加exchanger: zzyboot-mail-exchange

绑定zzyboot-mail-queue 到 zzyboot-mail-exchange, routing key: zzyboot-mail-key



也可以用代码实现

**package** com.zzyboot.config;

**import** org.springframework.amqp.core.Binding;

**import** org.springframework.amqp.core.BindingBuilder;

**import** org.springframework.amqp.core.DirectExchange;

**import** org.springframework.amqp.core.ExchangeBuilder;

**import** org.springframework.amqp.core.Queue;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

@Configuration

**public** **class** MessageRabbitMqConfiguration {

@Bean

DirectExchange messageDirect(){

**return**(DirectExchange) ExchangeBuilder.*directExchange*("zzyboot-mail-exchange").durable(**true**).build();

}

@Bean

**public** Queue messageQueue(){

**return** **new** Queue("zzyboot-mail-queue");

}

@Bean

Binding messageBinding(DirectExchange messageDirect, Queue messageQueue){

**return** BindingBuilder.*bind*(messageQueue).to(messageDirect).with("zzyboot-mail-key");

}

}

添加依赖RabbitMQ

配置：

spring

rabbitmq:

host: localhost

port: 5672

username: guest

password: guest

## 发送msg

@Autowired

**private** RabbitTemplate rabbitTemplate;

@PostMapping("/mail")

**public** String mail(@RequestBody String param){

String param=”a test message”;

String exchange = "zzyboot-mail-exchange";

String routingKey = "zzyboot-mail-key";

rabbitTemplate.convertAndSend(exchange, routingKey, param);

}

## 接收msg

监听消息

@Component

**public** **class** MessageConsumer {

@Autowired

MainService mainService;

@RabbitListener(queues = "zzyboot-mail-queue")

**public** **void** handler(String param){

String[] paramA = param.split(ZzyCommon.***STRSEPLINE***);

**int** index = 0;

String from = paramA[index++];

String to = paramA[index++];

String cc = paramA[index++];

String bcc = paramA[index++];

String subject = paramA[index++];

String content = paramA[index++];

String filename = paramA[index++];

mainService.sendAttachmentsMail(from, to, cc, bcc, subject, content, filename);

}

}

# 添加redis

Zzycommon添加redis依赖

ZzyCommon.java

**public** **static** Map<String, String> *usertoken* = **new** ConcurrentHashMap<String, String>();

**public** **static** **boolean** tokenvalid(String username, String token,StringRedisTemplate redisTemplate){

String param = username + ***STRSEPITEM*** + token;

**if**(*usertoken*.containsKey(param)){

**return** **true**;

}

String result = *redisGet*(redisTemplate,param);

**if**(result == **null** || result.length() < 1){

**return** **false**;

}

**return** **true**;

}

**public** **static** String tokenvalidstr(String username, String token,StringRedisTemplate redisTemplate){

String param = username + ***STRSEPITEM*** + token;

**if**(*usertoken*.containsKey(param)){

**return** ***ZZYSUCCESS*** + *usertoken*.get(param);

}

String result = *redisGet*(redisTemplate,param);

**if**(result == **null** || result.length() < 1){

**return** ***ZZYFAIL\_USERINVALID***;

}

**return** ***ZZYSUCCESS*** + result;

}

**public** **static** String tokenkeep(String realUserName, **int** token, String url,StringRedisTemplate redisTemplate){

String param = realUserName + ***STRSEPITEM*** + token;

*usertoken*.put(param, url + ***STRSEPITEM*** + param);

*redisSet*(redisTemplate,param, url + ***STRSEPITEM*** + param);

**return** ***ZZYSUCCESS*** + url + ***STRSEPITEM*** + param;

}

**public** **static** String redisGet(StringRedisTemplate redisTemplate, String key){

**return** redisTemplate.opsForValue().get(key);

}

**public** **static** **void** redisSet(StringRedisTemplate redisTemplate, String key, String value){

redisTemplate.opsForValue().set(key, value);

}

Zzyusercenter, Zzyemail配置

spring:

redis:

host: localhost

database: 0

port: 6379

jedis:

pool:

max-active: 3

max-wait: -1

max-idle: 8

min-idle: 0

Zzyusercenter, Zzyemail控制台中添加

@Autowired

**private** StringRedisTemplate redisTemplate;

。。。

**if**(!ZzyCommon.*tokenvalid*(username, token,redisTemplate)){

System.***out***.println("token not exists");

**return** ZzyCommon.***ZZYFAIL\_USERINVALID***;

}

//generate token

Random random = **new** Random();

**int** itoken = random.nextInt(100000);

String url = u.getUrl();

**if**(realUserName == **null**){

realUserName = "";

}

**return** ZzyCommon.*tokenkeep*(realUserName, itoken, url,redisTemplate);

# 打包

在zzyboot目录下进入cmd依次执行

mvn clean

mvn compile

mvn package

zzyboot-common, zzyboot-api无main异常，为这两个包添加带main的类

同时在调用它的zzyboot-usercenter, zzyboot-front添加以下内容，以防止无法定位到正确主类异常

<properties>

<start-class>com.zzyboot.Application</start-class>

</properties>

异常：zzyboot-usercenter, zzyboot-front运行异常，原因，无法找到依赖包的类

在被依赖的类zzyboot-api和zzpboot-common的pom中加入以下内容

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<classifier>exec</classifier>

</configuration>

</plugin>

</plugins>

</build>

# 附录

# 公用模块zzyboot-api

### ZzyColumn

package com.zzyboot.entity;

import static java.lang.annotation.ElementType.FIELD;

import static java.lang.annotation.RetentionPolicy.RUNTIME;

import java.lang.annotation.Retention;

import java.lang.annotation.Target;

@Retention(RUNTIME)

@Target(FIELD)

public @interface ZzyColumn {

boolean isprimkey() default false;

boolean iskeyword() default false;

String name() default "";

String label() default "";

String fieldtype() default "String";

int minlength() default 0;

boolean ispassword() default false;

}

### ZzyColumns

package com.zzyboot.entity;

import java.lang.reflect.Field;

import lombok.Data;

@Data

public class ZzyColumns {

private String name;

private String label;

private Boolean isPrimary;

private Boolean isKeyword;

private Boolean isUnique;

private Boolean isRequired;

private Integer minLength;

private Field fi;

public ZzyColumns(String name, String label, Boolean isPrimary, Boolean isKeyword, Boolean isUnique,Boolean isRequired, Integer minLength, Field f) {

super();

this.name = name;

this.label = label;

this.isPrimary = isPrimary;

this.isKeyword = isKeyword;

this.isUnique = isUnique;

this.isRequired = isRequired;

this.minLength = minLength;

this.fi = f;

}

}

### ZzyEntityParent

**package** com.zzyboot.entity;

**import** java.lang.annotation.Annotation;

**import** java.lang.reflect.Field;

**import** java.sql.Date;

**import** java.util.ArrayList;

**import** java.util.Arrays;

**import** java.util.List;

**import** javax.persistence.Column;

**import** javax.persistence.GeneratedValue;

**import** javax.persistence.GenerationType;

**import** javax.persistence.Id;

**import** javax.persistence.MappedSuperclass;

**import** javax.persistence.Version;

**import** com.zzyboot.util.ZzyUtil;

**import** lombok.Data;

@Data

@MappedSuperclass

**public** **class** ZzyEntityParent **implements** Cloneable {

@Id

@GeneratedValue(strategy = GenerationType.***AUTO***)

@Column(updatable = **false**, nullable = **false**)

**protected** Long id;

@Version

@Column

**protected** Long zzyoptlock = 0L;

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = 1;

result = prime \* result + ((id == **null**) ? 0 : id.hashCode());

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

ZzyEntityParent other = (ZzyEntityParent) obj;

**if** (id == **null**) {

**if** (other.id != **null**)

**return** **false**;

} **else** **if** (!id.equals(other.id))

**return** **false**;

**return** **true**;

}

@Column

**protected** Long entryid;

@Column

**protected** Long entrytime;

**public** List<ZzyColumns> getColumnDef(){

String tablename = **this**.getClass().getName();

**if**(ZzyUtil.*tableColumns*.containsKey(tablename)){

**return** ZzyUtil.*tableColumns*.get(tablename);

}

List<ZzyColumns> thisColumn;

thisColumn = **new** ArrayList<ZzyColumns>();

System.***out***.println("getColumnDef begin");

**if**(**this**.getClass().getSuperclass()!=**null**){

Field[] fields = **this**.getClass().getSuperclass().getDeclaredFields();

//Field[] fields = this.getClass().getFields();

thisColumn = getColumnDef(fields);

}

Field[] fields = **this**.getClass().getDeclaredFields();

thisColumn.addAll(getColumnDef(fields));

ZzyUtil.*tableColumns*.put(tablename,thisColumn);

**return** thisColumn;

}

**public** List<ZzyColumns> getColumnDef(Field[] fields){

//if(thisColumn!=null)return thisColumn;

List<ZzyColumns> thisColumn = **new** ArrayList<ZzyColumns>();

//System.out.println("getColumnDef begin");

//System.out.println("fields cnt is " + fields.length);

**for**(**int** i = 0 ; i < fields.length; i++){

Field fi = fields[i];

ZzyColumns zc = **null**;

Annotation[] annotations = fi.getDeclaredAnnotations();

**for**(**int** j = 0; j < annotations.length; j++){

Annotation aj = annotations[j];

//System.out.println("annotations " + j + " is " + aj.toString() + " for " + fi.getName());

//if(aj.getClass().equals(ZzyColumn.class)){

**if**(aj.annotationType().equals(ZzyColumn.**class**)){

**if**(zc == **null**){

zc = **new** ZzyColumns(fi.getName(), "" ,**false**,**false**,**false**,**false**,0, fi);

//System.out.println("f ajj" + i + " is " + fi.getName());

}

ZzyColumn ajj = (ZzyColumn)aj;

**if**(ajj.label() != **null**)zc.setLabel(ajj.label());

**if**(ajj.iskeyword())zc.setIsKeyword(**true**);

**if**(ajj.isprimkey())zc.setIsPrimary(**true**);

**if**(ajj.minlength() != 0)zc.setMinLength(ajj.minlength());

}**else** **if**(aj.annotationType().equals(Column.**class**)){

**if**(zc == **null**){

/\*if(fi.getName().equals("zzyoptlock")){

continue;

}\*/

zc = **new** ZzyColumns(fi.getName(), "" ,**false**,**false**,**false**,**false**,0,fi);

//System.out.println("f cj" + i + " is " + fi);

}

Column cj = (Column)aj;

**if**(cj.unique())zc.setIsUnique(**true**);

**if**(!cj.nullable())zc.setIsRequired(**true**);

}

}

**if**(zc != **null**)thisColumn.add(zc);

}

**return** thisColumn;

}

**public** String getDBList(List<ZzyEntityParent> list){

StringBuilder sb = **new** StringBuilder();

**for**(ZzyEntityParent z: list){

sb.append(z.toString() + ZzyUtil.***STRSEPLINE***);

}

**return** sb.toString();

}

**public** **void** setValues(String param){

String[] s = param.split(ZzyUtil.***STRSEPITEM***);

List<ZzyColumns> fields = getColumnDef();

setValues(fields,s);

}

**private** **void** setField(Field fi, ZzyEntityParent z, String si){

**try** {

**if**(si.length() < 1){

fi.set(z, **null**);

**return**;

}

fi.setAccessible(**true**);

**if**(fi.getType().equals(Boolean.**class**)){

fi.set(z, Boolean.*valueOf*(si));

}**else** **if**(fi.getType().equals(Integer.**class**)){

fi.set(z,**new** Integer(si));

}**else** **if**(fi.getType().equals(Long.**class**)){

fi.set(z, **new** Long(si));

}**else** **if**(fi.getType().equals(Double.**class**)){

fi.set(z, **new** Double(si));

}**else** **if**(fi.getType().equals(Date.**class**)){

fi.set(z, ZzyUtil.*getDate*(si));

}**else**{

fi.set(z, si);

}

} **catch** (IllegalArgumentException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IllegalAccessException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**public** **void** setValues(List<ZzyColumns> fields,Object[] s){

**int** index = 0;

**for**(ZzyColumns z: fields){

Field fi = z.getFi();

//System.out.println("fi is " +fi);

String si = s[index++]+"";

//System.out.println(" setValues " + fi.getName()+",values is " + si);

setField(fi,**this**,si);

}

}

**public** String toStringZzy(){

StringBuilder sb = **new** StringBuilder();

List<ZzyColumns> fields = getColumnDef();

**for**(ZzyColumns z: fields){

Field fi = z.getFi();

**try** {

fi.setAccessible(**true**);

sb.append(fi.get(**this**)+ZzyUtil.***STRSEPITEM***);

} **catch** (IllegalArgumentException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IllegalAccessException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

/\*Field[] fs = this.getClass().getDeclaredFields();

for(Field fi: fs){

String si = null;

Annotation[] annotations = fi.getDeclaredAnnotations();

for(int j = 0; j < annotations.length; j++){

Annotation aj = annotations[j];

if(aj.getClass().equals(ZzyColumn.class)){

if(si == null){

try {

fi.setAccessible(true);

sb.append(fi.get(this)+ZzyUtil.STRSEPITEM);

} catch (IllegalArgumentException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} catch (IllegalAccessException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

break;

}

}else if(aj.getClass().equals(Column.class)){

if(si == null){

try {

fi.setAccessible(true);

sb.append(fi.get(this)+ZzyUtil.STRSEPITEM);

} catch (IllegalArgumentException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} catch (IllegalAccessException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

break;

}

}

}

}\*/

**return** sb.toString();

}

**public** ZzyEntityParent newObj(ZzyEntityParent zOld, Object[] obj,List<ZzyColumns> listColumn){

ZzyEntityParent zNew = **null**;

**try** {

zNew = (ZzyEntityParent)zOld.clone();

zNew.setValues(listColumn, obj);

/\* int index = 0;

for(ZzyColumns zcol: listColumn){

Field fi = zcol.getFi();

System.out.println(fi.getName() +", obj is " + obj[index++]);

//fi.set(zNew, obj[index++]);

}

\*/ } **catch** (Exception e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**return** zNew;

}

**public** String getTableName(){

String table=**this**.getClass().getName();

String[] tableA = table.split("\\.");

table = tableA[tableA.length - 1];

**return** table;

}

**public** String getWhereUniq(){

List<ZzyColumns> listColumn = **this**.getColumnDef();

List<String> whereList = **new** ArrayList<String>();

**for**(ZzyColumns zcol: listColumn){

**if**(zcol.getIsUnique()){

whereList.add(zcol.getName());

}

}

**return** getWhere(whereList);

}

**public** String getWhere(String[] wheres){

List<String> whereList = Arrays.*asList*(wheres);

**return** getWhere(whereList);

}

**public** String getWhere(List<String> whereList){

String where="";

String whereSeg = "";

**int** index = 1;

**for**(Object s: whereList){

where += whereSeg + s + "=?" + (index++);

whereSeg=" and ";

}

**return** where;

}

**public** Object[] getWhereUniqParam(){

List<Object> listParam = **new** ArrayList<Object>();

List<ZzyColumns> listColumn = **this**.getColumnDef();

**for**(ZzyColumns zcol: listColumn){

**if**(zcol.getIsUnique()){

Field fi = zcol.getFi();

**try** {

Object zObj = fi.get(**this**);

listParam.add(zObj);

} **catch** (IllegalArgumentException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IllegalAccessException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

Object[] whereparam=listParam.toArray();

**return** whereparam;

}

**public** String getCols(){

List<ZzyColumns> listColumn = **this**.getColumnDef();

StringBuilder sb = **new** StringBuilder();

String strColsSeg = "";

**for**(ZzyColumns zcol: listColumn){

sb.append(strColsSeg + zcol.getName());

strColsSeg = ",";

}

**return** sb.toString();

}

**public** **void** update(String[] updateinfos){

List<ZzyColumns> listColumn = **this**.getColumnDef();

**for**(**int** i = 1; i < updateinfos.length; i++){

String ui = updateinfos[i];

String[] uiA = ui.split(ZzyUtil.***STRSEPITEM***);

**for**(ZzyColumns z: listColumn){

**if**(z.getName().equals(uiA[0])){

setField(z.getFi(),**this**,uiA[1]);

**break**;

}

}

}

}

}

### ZzyUtil

package com.zzyboot.util;

import java.sql.Date;

import java.sql.Timestamp;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import com.zzyboot.entity.ZzyColumns;

public class ZzyUtil {

public final static String STRSEPITEM="zy~";

public final static String STRSEPLINE="zy!";

public final static String ZZYSUCCESS = "Success:";

public final static String ZZYSUCCESSDELETE = ZZYSUCCESS + "deleted successfully";

public final static String ZZYFAIL = "Fail:";

public final static Integer ENTRYID = -1;

public final static String ZZYFAILADD = ZZYFAIL+"This is not new record";

public final static String ZZYFAILADDDUP = ZZYFAIL+"data duplicate conflict";

public final static String ZZYFAILADDINCOMPLETE = ZZYFAIL+"data is not complete";

public final static String ZZYFAILDELETENOTEXISTS = ZZYFAIL+"data is not exists";

public final static String ZZYFAILUPDATENODATA = ZZYFAIL+"data is not exists";

public static Map<String, List<ZzyColumns>> tableColumns = new HashMap<String, List<ZzyColumns>>();

public static String ENTITYPATH = "com.zzycommon.entity.";

public static long getNow(){

return System.currentTimeMillis();

}

public static Date getDate(String aDate){

String[] adateA=aDate.split("\r\n");

aDate=adateA[0].trim();

if(aDate==null || aDate.length()<1)return null;

if(aDate.toLowerCase().equals("null"))return null;

aDate=aDate.trim();

if(aDate.indexOf(" ")>0){

String[] dA=aDate.split(" ");

if(dA.length==3){

String month="01";

if(dA[0].toLowerCase().equals("jan"))month="01";

else if(dA[0].toLowerCase().equals("feb"))month="02";

else if(dA[0].toLowerCase().equals("mar"))month="03";

else if(dA[0].toLowerCase().equals("apr"))month="04";

else if(dA[0].toLowerCase().equals("may"))month="05";

else if(dA[0].toLowerCase().equals("jun"))month="06";

else if(dA[0].toLowerCase().equals("jul"))month="07";

else if(dA[0].toLowerCase().equals("aug"))month="08";

else if(dA[0].toLowerCase().equals("sep"))month="09";

else if(dA[0].toLowerCase().equals("oct"))month="10";

else if(dA[0].toLowerCase().equals("nov"))month="11";

else if(dA[0].toLowerCase().equals("dec"))month="12";

String year=dA[2];

if(year.length()<3)year="20"+year;

String day=dA[1];

aDate=year+"-"+month+"-"+day;

}

}

if(aDate.length()<1)return null;

Timestamp dRtn=getDatetime(aDate);

if(dRtn==null)return null;

return new Date(dRtn.getTime());

}

public static Timestamp getDatetime(String aDate){

if(aDate==null || aDate.length()<1)return null;

if(aDate.toLowerCase().equals("null"))return null;

if(aDate.indexOf(" ")<1){

//systemout("aDate is "+aDate);

if(aDate.length()==8)aDate="20"+aDate;

Date aD=Date.valueOf(aDate);

return new Timestamp(aD.getTime());

}

if(aDate.indexOf("-")<1)return null;

return Timestamp.valueOf(aDate);

}

}

### ZzyRepository

package com.zzyboot.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.zzyboot.entity.ZzyEntityParent;

@Repository

public interface ZzyRepository extends JpaRepository<ZzyEntityParent, Long>, ZzyRepositoryCustom {

}

### ZzyRepositoryCustom

package com.zzyboot.repository;

import java.util.List;

import com.zzyboot.entity.ZzyEntityParent;

public interface ZzyRepositoryCustom {

public String add(ZzyEntityParent z);

public String delete(ZzyEntityParent z,String uniqCol);

public List<ZzyEntityParent> findAll(ZzyEntityParent z);

public String update(ZzyEntityParent z, String updateinfo);

public ZzyEntityParent findOne(ZzyEntityParent z, String cols, String wheres);

}

### ZzyRepositoryImpl

package com.zzyboot.repository;

import java.lang.reflect.Field;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

import javax.persistence.Query;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.transaction.annotation.Transactional;

import com.zzyboot.entity.ZzyColumns;

import com.zzyboot.entity.ZzyEntityParent;

import com.zzyboot.util.ZzyUtil;

@Transactional

public class ZzyRepositoryImpl implements ZzyRepositoryCustom {

@Autowired

@PersistenceContext

private EntityManager em;

@Override

public String add(ZzyEntityParent z) {

// TODO Auto-generated method stub

if(z.getId() != null){

return ZzyUtil.ZZYFAILADD;

}

//check if uniq col exists

boolean uniqvalid = getUniqValid(z);

if(!uniqvalid){

return ZzyUtil.ZZYFAILADDINCOMPLETE;

}

//check if it exists

boolean exists = getUniqExists(z);

if(exists){

return ZzyUtil.ZZYFAILADDDUP;

}

z = save(z);

return z.toStringZzy();

}

@Transactional

public ZzyEntityParent save(ZzyEntityParent z){

em.persist(z);

return z;

}

private boolean getUniqValid(ZzyEntityParent z) {

String table=z.getClass().getName();

String[] tableA = table.split("\\.");

table = tableA[tableA.length - 1];

List<ZzyColumns> listColumn = z.getColumnDef();

for(ZzyColumns zcol: listColumn){

if(zcol.getIsUnique()){

Field fi = zcol.getFi();

try {

Object zObj = fi.get(z);

if(zObj == null){

return false;

}

if((zObj+"").trim().length()<1){

return false;

}

} catch (IllegalArgumentException e) {

// TODO Auto-generated catch block

e.printStackTrace();

return false;

} catch (IllegalAccessException e) {

// TODO Auto-generated catch block

e.printStackTrace();

return false;

}

}

}

return true;

}

private ZzyEntityParent getUniq(ZzyEntityParent z) {

String table=z.getTableName();

/\*.getClass().getName();

String[] tableA = table.split("\\.");

table = tableA[tableA.length - 1];\*/

String where=z.getWhereUniq();

String strCols = z.getCols();

Object[] whereparam=z.getWhereUniqParam();

Object resultOriginal = getSingle(strCols,table, where, whereparam);

if(resultOriginal == null){

return null;

}

Object[] result = (Object[]) resultOriginal;

if(result!=null && result.length>0){

List<ZzyColumns> listColumn = z.getColumnDef();

ZzyEntityParent oNew= z.newObj(z,result,listColumn);

System.out.print(oNew.toString());

return oNew;

}

return null;

}

private boolean getUniqExists(ZzyEntityParent z) {

ZzyEntityParent result = getUniq(z);

if(result!=null){

System.out.println("result is " + result);

return true;

}

return false;

}

public Long getSingleID(String table, String where, Object[] whereparam) {

String strSqlCol = "id";

Object resultOriginal = getSingle(strSqlCol, table, where, whereparam);

if(resultOriginal == null){

return -1l;

}

Object[] result=(Object[])resultOriginal;

System.out.println("result.length is " + result.length);

if(result !=null && result.length > 0){

return (Long)result[0];

}

return -1l;

}

public Object getSingle(String strSqlCol,String table, String where, Object[] whereparam) {

String strSql = "select " + strSqlCol + " from " + table;

if(where.length()>0){

strSql+=" where " + where;

}

Query query = em.createNativeQuery(strSql);

for(int i = 0; i < whereparam.length; i++){

query.setParameter(i+1, whereparam[i]);

}

Object result = null;

try{

List listRtn = query.getResultList();

if(listRtn.size() > 0){

result=listRtn.get(0);

}

}catch(Exception e){

e.printStackTrace();

return null;

}

return result;

}

public List<ZzyEntityParent> findAll(ZzyEntityParent z) {

// TODO Auto-generated method stub

return getMore(z);

}

public List<ZzyEntityParent> getMore(ZzyEntityParent z) {

return getMore(z,null);

}

public List<ZzyEntityParent> getMore(ZzyEntityParent z,String strSqlCol) {

return getMore(z,strSqlCol,null,null);

}

@SuppressWarnings("unchecked")

public List getMore(ZzyEntityParent z,String strSqlCol, String where, Object[] whereparam) {

if(strSqlCol == null || strSqlCol.length() < 1){

strSqlCol = z.getCols();

}

String strSql = "select " + strSqlCol + " from " + z.getTableName();

if(where!=null && where.length()>0){

strSql+=" where " + where;

}

//System.out.println(strSql);

Query query = em.createNativeQuery(strSql);

if(whereparam!=null){

for(int i = 0; i < whereparam.length; i++){

query.setParameter(i+1, whereparam[i]);

}

}

List<ZzyEntityParent> result = new ArrayList<ZzyEntityParent>();

try{

List<Object> resultDB = query.getResultList();

List<ZzyColumns> listColumn = z.getColumnDef();

System.out.println("resultDB length is " + resultDB.size());

for(Object o: resultDB){

Object[] oA = (Object[]) o;

ZzyEntityParent znew = z.newObj(z, oA,listColumn);

result.add(znew);

}

}catch(Exception e){

//e.printStackTrace();

return null;

}

return result;

}

@Override

public String delete(ZzyEntityParent z, String uniqCol) {

String[] uniqCols = uniqCol.split(ZzyUtil.STRSEPITEM);

String where=z.getWhereUniq();

String strCols = z.getCols();

Object resultOriginal = getSingle(strCols,z.getTableName(), where, uniqCols);

if(resultOriginal == null){

return ZzyUtil.ZZYFAILDELETENOTEXISTS;

}

Object[] result = (Object[]) resultOriginal;

if(result!=null && result.length>0){

List<ZzyColumns> listColumn = z.getColumnDef();

ZzyEntityParent oNew= z.newObj(z,result,listColumn);

System.out.print(oNew.toString());

return delete(oNew);

}

return null;

}

@Transactional

private String delete(ZzyEntityParent z){

Long id = z.getId();

em.remove(em.getReference(z.getClass(),id));

return ZzyUtil.ZZYSUCCESSDELETE + "(" +id+ ")";

}

@Override

public String update(ZzyEntityParent z, String updateinfo) {

String[] updateinfos = updateinfo.split(ZzyUtil.STRSEPLINE);

String strSql = "select " + z.getCols() + " from " + z.getTableName() + " where id=?1";

//System.out.println(strSql);

Query query = em.createNativeQuery(strSql);

query.setParameter(1, updateinfos[0]);

Object resultOriginal = null;

try{

resultOriginal = query.getSingleResult();

if(resultOriginal == null){

return ZzyUtil.ZZYFAILUPDATENODATA;

}

Object[] result = (Object[]) resultOriginal;

if(result!=null && result.length>0){

List<ZzyColumns> listColumn = z.getColumnDef();

ZzyEntityParent oNew= z.newObj(z,result,listColumn);

oNew.update(updateinfos);

ZzyEntityParent zNew = update(oNew);

return zNew.toStringZzy();

}

}catch(Exception e){

e.printStackTrace();

return ZzyUtil.ZZYFAILUPDATENODATA;

}

return ZzyUtil.ZZYFAILUPDATENODATA;

}

@Transactional

private ZzyEntityParent update(ZzyEntityParent z){

em.merge(z);

return z;

}

@Override

public ZzyEntityParent findOne(ZzyEntityParent z, String cols, String wheres) {

String[] colA = cols.split(ZzyUtil.STRSEPITEM);

String[] whereA = wheres.split(ZzyUtil.STRSEPITEM);

Object resultOriginal = getSingle(z.getCols(),z.getTableName(), z.getWhere(colA), whereA);

if(resultOriginal == null){

return null;

}

Object[] result = (Object[]) resultOriginal;

if(result!=null && result.length>0){

List<ZzyColumns> listColumn = z.getColumnDef();

ZzyEntityParent oNew= z.newObj(z,result,listColumn);

//System.out.print(oNew.toString());

return oNew;

}

return null;

}

}

### ZzyService

package com.zzyboot.service;

import java.util.List;

import com.zzyboot.entity.ZzyEntityParent;

public interface ZzyService {

public String add(ZzyEntityParent z);

public String delete(ZzyEntityParent z, String uniqCol);

public List<ZzyEntityParent> findAll(ZzyEntityParent z);

public String update(ZzyEntityParent z, String updateinfo);

public ZzyEntityParent findOne(ZzyEntityParent z, String cols, String wheres);

}

### ZzyServiceImpl

package com.zzyboot.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.zzyboot.entity.User;

import com.zzyboot.entity.ZzyEntityParent;

import com.zzyboot.repository.ZzyRepository;

import com.zzyboot.util.ZzyUtil;

@Service

public class ZzyServiceImpl implements ZzyService {

@Autowired

ZzyRepository repository;

public String add(ZzyEntityParent z){

return repository.add(z);

}

@Override

public List<ZzyEntityParent> findAll(ZzyEntityParent z) {

// TODO Auto-generated method stub

return repository.findAll(z);

}

@Override

public String delete(ZzyEntityParent z, String uniqCol) {

return repository.delete(z,uniqCol);

}

@Override

public String update(ZzyEntityParent z, String updateinfo) {

// TODO Auto-generated method stub

return repository.update(z,updateinfo);

}

public ZzyEntityParent findOne(ZzyEntityParent z, String cols, String wheres) {

// TODO Auto-generated method stub

return repository.findOne(z,cols,wheres);

}

}

# 用户认证中心zzyboot-usercenter

## Application

package com.zzyboot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

public static void main(String[] args){

SpringApplication.run(Application.class, args);

}

}

## ZzyUtilUserCenter

package com.zzyboot.util;

import java.util.Map;

import java.util.concurrent.ConcurrentHashMap;

public class ZzyUtilUserCenter {

public static Map<String, String> usertoken = new ConcurrentHashMap<String, String>();

}

## UserController

**package** com.zzyboot.controller;

**import** java.io.UnsupportedEncodingException;

**import** java.net.URLDecoder;

**import** java.util.List;

**import** java.util.Random;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.zzyboot.entity.User;

**import** com.zzyboot.entity.ZzyEntityParent;

**import** com.zzyboot.service.ZzyServiceImpl;

**import** com.zzyboot.util.ZzyUtil;

**import** com.zzyboot.util.ZzyUtilUserCenter;

@RestController

**public** **class** UserController {

@Autowired

ZzyServiceImpl zzyService;

@GetMapping("/userinfo")

**public** String getUserInfo(){

StringBuilder sb = **new** StringBuilder();

List<ZzyEntityParent> users =zzyService.findAll(**new** User());

System.***out***.println("users are " + users.toString());

**for**(ZzyEntityParent z: users){

System.***out***.println(z.toString());

sb.append(z.toStringZzy() + ZzyUtil.***STRSEPLINE***);

}

**return** sb.toString();

}

@PostMapping("/add")

**public** String Add(@RequestBody String param){

User u = **new** User();

**try** {

param = URLDecoder.*decode*(param,"UTF-8");

} **catch** (UnsupportedEncodingException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

//System.out.println("add param is " + param);

u.setValues(param);

String result = zzyService.add(u);

System.***out***.println("add result is " + result);

**return** result;

}

@PostMapping("/delete")

**public** String Delete(@RequestBody String param){

User u = **new** User();

**try** {

param = URLDecoder.*decode*(param,"UTF-8");

} **catch** (UnsupportedEncodingException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

//System.out.println("add param is " + param);

String result = zzyService.delete(u, param);

System.***out***.println("delete result is " + result);

**return** result;

}

@PostMapping("/update")

**public** String Update(@RequestBody String param){

**try** {

param = URLDecoder.*decode*(param,"UTF-8");

} **catch** (UnsupportedEncodingException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

//System.out.println("add param is " + param);

User u = **new** User();

String result = zzyService.update(u, param);

System.***out***.println("update result is " + result);

**return** result;

}

@PostMapping("/usercheck")

**public** String UserCheck(@RequestBody String param){

**try** {

param = URLDecoder.*decode*(param,"UTF-8");

} **catch** (UnsupportedEncodingException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

String[] paramArray = param.split(ZzyUtil.***STRSEPITEM***);

String username = paramArray[0];

String token = paramArray[1];

**if**(ZzyUtilUserCenter.*usertoken*.containsKey(param)){

**return** ZzyUtil.***ZZYSUCCESS*** + ZzyUtilUserCenter.*usertoken*.get(param);

}

User u = **null**;

String realUserName = **null**;

u = (User)zzyService.findOne(**new** User(),"email",username);

**if**(u != **null**){

System.***out***.println(u.getPassword()+" ==? " + token);

**if**(u.getPassword().equals(token)){

realUserName = u.getUsername();

}

}

**if**(realUserName == **null**){

u = (User)zzyService.findOne(**new** User(),"cell",username);

**if**(u != **null**){

**if**(u.getPassword().equals(token)){

realUserName = u.getUsername();

}

}

}

**if**(realUserName == **null**){

u = (User)zzyService.findOne(**new** User(),"username" + ZzyUtil.***STRSEPITEM*** + "password",username + ZzyUtil.***STRSEPITEM*** +token);

**if**(u != **null**){

realUserName = username;

}

}

**if**(realUserName == **null**){

**return** ZzyUtil.***ZZYFAIL***+"login is not valid";

}

//generate token

Random random = **new** Random();

**int** itoken = random.nextInt(100000);

String url = u.getUrl();

param = realUserName + ZzyUtil.***STRSEPITEM*** + itoken;

ZzyUtilUserCenter.*usertoken*.put(param, url + ZzyUtil.***STRSEPITEM*** + param);

**return** ZzyUtil.***ZZYSUCCESS*** + ZzyUtilUserCenter.*usertoken*.get(param);

}

}

# 前端项目

## Application

package com.zzyboot;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

public static void main(String[] args) {

SpringApplication.run(Application.class, args);

}

}

## DES

package com.zzyfrontcomponent.util;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.security.spec.InvalidKeySpecException;

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESKeySpec;

public class DES {

private static final String DES\_ALGORITHM = "DES";

public static String encryption(String plainData, String secretKey) throws Exception {

Cipher cipher = null;

try {

cipher = Cipher.getInstance(DES\_ALGORITHM);

cipher.init(Cipher.ENCRYPT\_MODE, generateKey(secretKey));

}

catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

catch (NoSuchPaddingException e) {

e.printStackTrace();

} catch (InvalidKeyException e) {

} try {

// 为了防止解密时报javax.crypto.IllegalBlockSizeException: Input length must

// be multiple of 8 when decrypting with padded cipher异常,

// 不能把加密后的字节数组直接转换成字符串

byte[] buf = cipher.doFinal(plainData.getBytes());

return Base64Utils.encode(buf);

} catch (IllegalBlockSizeException e) {

e.printStackTrace();

throw new Exception("IllegalBlockSizeException", e);

} catch (BadPaddingException e) {

e.printStackTrace();

throw new Exception("BadPaddingException", e);

}

}

public static String decryption(String secretData, String secretKey) throws Exception {

System.out.println("decryption:["+secretData+"]["+secretKey+"]");

Cipher cipher = null;

try {

cipher = Cipher.getInstance(DES\_ALGORITHM);

cipher.init(Cipher.DECRYPT\_MODE, generateKey(secretKey));

}

catch (NoSuchAlgorithmException e) {

e.printStackTrace();

throw new Exception("NoSuchAlgorithmException", e);

} catch (NoSuchPaddingException e) {

e.printStackTrace();

throw new Exception("NoSuchPaddingException", e);

} catch (InvalidKeyException e) {

e.printStackTrace();

throw new Exception("InvalidKeyException", e);

} try {

byte[] buf = cipher.doFinal(Base64Utils.decode(secretData.toCharArray()));

return new String(buf);

} catch (IllegalBlockSizeException e) {

e.printStackTrace();

throw new Exception("IllegalBlockSizeException", e);

} catch (BadPaddingException e) {

e.printStackTrace();

throw new Exception("BadPaddingException", e);

}

}

private static SecretKey generateKey(String secretKey)

throws NoSuchAlgorithmException, InvalidKeySpecException, InvalidKeyException {

SecretKeyFactory keyFactory = SecretKeyFactory.getInstance(DES\_ALGORITHM);

DESKeySpec keySpec = new DESKeySpec(secretKey.getBytes());

keyFactory.generateSecret(keySpec);

return keyFactory.generateSecret(keySpec);

}

static private class Base64Utils{

static private char[] alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=" .toCharArray();

static private byte[] codes = new byte[256];

static {

for (int i = 0; i < 256; i++)

codes[i] = -1;

for (int i = 'A'; i <= 'Z'; i++)

codes[i] = (byte) (i - 'A');

for (int i = 'a'; i <= 'z'; i++)

codes[i] = (byte) (26 + i - 'a');

for (int i = '0'; i <= '9'; i++)

codes[i] = (byte) (52 + i - '0');

codes['+'] = 62; codes['/'] = 63;

}

static private String encode(byte[] data){

char[] out = new char[((data.length + 2) / 3) \* 4];

for (int i = 0, index = 0; i < data.length; i += 3, index += 4) {

boolean quad = false;

boolean trip = false;

int val = (0xFF & (int) data[i]);

val <<= 8;

if ((i + 1) < data.length) {

val |= (0xFF & (int) data[i + 1]);

trip = true;

}

val <<= 8;

if ((i + 2) < data.length) {

val |= (0xFF & (int) data[i + 2]);

quad = true;

}

out[index + 3] = alphabet[(quad ? (val & 0x3F) : 64)];

val >>= 6;

out[index + 2] = alphabet[(trip ? (val & 0x3F) : 64)];

val >>= 6;

out[index + 1] = alphabet[val & 0x3F];

val >>= 6;

out[index + 0] = alphabet[val & 0x3F];

}

return new String(out);

}

static private byte[] decode(char[] data){

int len = ((data.length + 3) / 4) \* 3;

if (data.length > 0 && data[data.length - 1] == '=') --len;

if (data.length > 1 && data[data.length - 2] == '=') --len;

byte[] out = new byte[len];

int shift = 0;

int accum = 0;

int index = 0;

for (int ix = 0; ix < data.length; ix++) {

int value = codes[data[ix] & 0xFF];

if (value >= 0) {

accum <<= 6;

shift += 6;

accum |= value;

if (shift >= 8) {

shift -= 8;

out[index++] = (byte) ((accum >> shift) & 0xff);

}

}

}

if (index != out.length) throw new Error("miscalculated data length!");

return out;

}

}

/\*public static void main(String[] args) throws Exception {

//待加密内容

String str = "zy~sdfszy~300cd61e0783c43a5a2799286a335d7e034379b4971587b552936bde2f8f18b4";

String key = "bbac949246ad7969fda8ba559f21347b681b366a8bd6de36c392f6f5be157010";

String strSe = DES.encryption(str, key);

strSe = "eKLmiNY6H1MRW6L1f4aSbaZmvaK3s5myNkAMbrUHes1ZuW1mQQNl5jdSG78dUwDradgto22NCa8URu12t2yi71a76gVin6DT/NqigWgf5Kg=";

System.out.println(strSe);

System.out.println(DES.decryption(strSe, key));

}\*/

}

## Jiami

package com.zzyfrontcomponent.util;

import java.io.UnsupportedEncodingException;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public class Jiami {

public static String sign(String str, String type){

String s = Encrypt(str, type);

return s;

}

public static String Encrypt(String strSrc, String algorithm){

MessageDigest md = null;

String strDes = "";

try {

md = MessageDigest.getInstance(algorithm);

md.update(strSrc.getBytes("UTF-8"));

strDes = bytes2Hex(md.digest());

} catch (NoSuchAlgorithmException e) {

// TODO Auto-generated catch block

e.printStackTrace();

} catch (UnsupportedEncodingException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return strDes;

}

public static String bytes2Hex(byte[] bytes){

StringBuffer stringBuffer = new StringBuffer();

String temp = null;

for (int i=0;i<bytes.length;i++){

temp = Integer.toHexString(bytes[i] & 0xFF);

if (temp.length()==1){

//1得到一位的进行补0操作

stringBuffer.append("0");

} stringBuffer.append(temp);

}

return stringBuffer.toString();

}

}

## ZzyUtil

package com.zzyfrontcomponent.util;

import java.io.UnsupportedEncodingException;

import java.net.URLDecoder;

import java.net.URLEncoder;

import java.nio.charset.StandardCharsets;

import javax.servlet.http.HttpSession;

public class ZzyUtil {

public final static String STRSEPITEM="zy~";

public final static String ZZYSUCCESS = "Success:";

public final static String ZZYFAIL = "Fail:";

public final static String ZZYFAIL\_NoSessionKey = ZZYFAIL + "no session key";

public final static String ZZYFAIL\_DES\_DECREPT = ZZYFAIL + "DES decreption failed";

public final static String ZZYFAIL\_DES\_ENCREPT = ZZYFAIL + "DES encreption failed";

public static int getMod(int source, int exponent, int divider){

long rtn = 1l;

for(int i = 0; i < exponent; i++){

rtn \*= source;

rtn = rtn % divider;

}

return (int)rtn;

}

public static String getSessionKeySecrete(HttpSession httpSession){

Object rtn = httpSession.getAttribute("sessionkeysecrete");

if(rtn == null){

return ZZYFAIL\_NoSessionKey;

}

if((rtn+"").length()<1){

return ZZYFAIL\_NoSessionKey;

}

return rtn+"";

}

public static String getDecode(String param, HttpSession httpSession){

/\*final byte[] requestContent;

requestContent = IOUtils.toByteArray(request.getReader());

return new String(requestContent, StandardCharsets.UTF\_8);\*/

//param = new String(param.getBytes(), StandardCharsets.UTF\_8);

try {

//System.out.println("before decode param is " + param);

param = URLDecoder.decode(param,"utf8");

//System.out.println("after decode param is " + param);

} catch (UnsupportedEncodingException e1) {

// TODO Auto-generated catch block

e1.printStackTrace();

}

String sessionkeysecrete = getSessionKeySecrete(httpSession);

if(sessionkeysecrete.indexOf(ZZYFAIL)==0){

return sessionkeysecrete;

}

String result = "";

try {

result = DES.decryption(param, sessionkeysecrete);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

return ZzyUtil.ZZYFAIL\_DES\_DECREPT;

}

return result;

}

public static String getEncode(String srtn, HttpSession httpSession){

String sessionkeysecrete = getSessionKeySecrete(httpSession);

if(sessionkeysecrete.indexOf(ZZYFAIL)==0){

return sessionkeysecrete;

}

String result = "";

try {

result = DES.encryption(srtn,sessionkeysecrete);

} catch (Exception e) {

// TODO Auto-generated catch block

e.printStackTrace();

return ZZYFAIL\_DES\_ENCREPT;

}

return ZZYSUCCESS + result;

}

}

## ZzyController

package com.zzyfrontcomponent.controller;

import java.util.Random;

import javax.servlet.http.HttpSession;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.client.RestTemplate;

import com.zzyfrontcomponent.util.DES;

import com.zzyfrontcomponent.util.Jiami;

import com.zzyfrontcomponent.util.ZzyUtil;

@RestController

public class ZzyController {

@Value("${authserver}")

private String authserver;

@Value("${zzyprime}")

private int zzyprime;

@Value("${zzydhbase}")

private int zzydhbase;

@Autowired

RestTemplate restTemplate;

@PostMapping("/zzylogin")

public String zzylogin(@RequestBody String param,HttpSession httpSession){

System.out.println("param is " + param);

param = ZzyUtil.getDecode(param, httpSession);

if(param.indexOf(ZzyUtil.ZZYFAIL)==0){

return param;

}

System.out.println("post result is " +param);

/\*String[] paramArray = param.split(ZzyUtil.STRSEPITEM);

String username = paramArray[1];

String token = paramArray[2];\*/

//check if the username and token is valid to authentic server

System.out.println("authserver is " + authserver);

String userinfo = restTemplate.postForObject(authserver + "/usercheck", param, String.class);

System.out.println("userinfo is " + userinfo);

return userinfo;

/\*if(userinfo)

username = "linuszhong";

token = "linuszhongtoken";

String url = "http://localhost:8085/index.html";

return ZzyUtil.getEncode(url + ZzyUtil.STRSEPITEM + username + ZzyUtil.STRSEPITEM + token,httpSession);\*/

}

@PostMapping("/isuservalid")

public String isUserValid(@RequestBody String param,HttpSession httpSession){

param = ZzyUtil.getDecode(param, httpSession);

if(param.indexOf(ZzyUtil.ZZYFAIL)==0){

return param;

}

System.out.println("post result is " +param);

String[] paramArray = param.split(ZzyUtil.STRSEPITEM);

String username = paramArray[0];

String token = paramArray[1];

//check if the username and token is valid to authentic server

System.out.println("authserver is " + authserver);

if(username.equals("linuszhong") && token.equals("linuszhongtoken")){

return ZzyUtil.ZZYSUCCESS;

}

return ZzyUtil.ZZYFAIL;

}

@PostMapping("/getsessionkey")

public String getsessionkey(@RequestParam("param") Integer param,HttpSession httpSession){

System.out.println("sessionkey param is " +param+",zzyprime is " + zzyprime);

//check if the username and token is valid to authentic server

Random random = new Random();

int r = random.nextInt(1000);

int rtn = ZzyUtil.getMod(zzydhbase,r,zzyprime);

System.out.println("rtn is " + rtn);

int sessionkey = ZzyUtil.getMod(param,r,zzyprime);

System.out.println("sessionkey is " + sessionkey);

String sessionkeySecrete = Jiami.sign(sessionkey+"","SHA-256");

System.out.println("sessionkeySecrete is " + sessionkeySecrete);

httpSession.setAttribute("sessionkeysecrete",sessionkeySecrete);

return ZzyUtil.ZZYSUCCESS+rtn;

}

}

## application.properties

server.port=8085

authserver=http://localhost:8081

zzyprime=91473769

zzydhbase=2

## static

### index.html

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="author" content="zhong ze yu">

<link rel="icon" href="favicon.ico">

<title></title>

<link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="css/zzy.css" rel="stylesheet">

</head>

<body>

</html>

<script src="js/common/zzyMain.js"></script>

<script type='text/javascript'>

let $ndashboard = $("<div></div>");

$('body').append($ndashboard);

$ndashboard.zzydashboard({});

</script>

### zzyLogin.html

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

<meta name="author" content="zhong ze yu">

<link rel="icon" href="favicon.ico">

<title></title>

<link href="bootstrap/css/bootstrap.min.css" rel="stylesheet">

<link href="css/zzy.css" rel="stylesheet">

</head>

<body>

</body>

</html>

<script src="js/common/zzyMain.js"></script>

<script type='text/javascript'>

let $nlogin = $("<div></div>");

$('body').append($nlogin);

$nlogin.zzylogin({});

</script>

### nativetoascii.html

<HTML>

<HEAD>

<TITLE>ASC←→NATIVE</TITLE>

<script language="javascript">

function native2ascii(){

let regexp=/[^\x00-\xff]/g;

let n=document.getElementById("native").value;

let a=n;

while(m=regexp.exec(n)){

a=a.split(m[0]).join(escape(m[0]).split("%").join("\\"));

}

document.getElementById("ascii").value=a;

}

function ascii2native() {

let a=document.getElementById("ascii").value;

let n=a;

n=unescape(n.split("\\").join("%"));

document.getElementById("native").value=n;

}

</script>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"></HEAD>

<BODY>

<h1>ASC←→NATIVE</h1>

ASC:<br>

<textarea id="ascii" rows="10" cols="100"></textarea><br>

<input type="button" id="back" value="ascii2native" onClick="ascii2native()"/>

&nbsp;&nbsp;&nbsp;&nbsp;

<input type="button" id="convert" value="native2ascii" onClick="native2ascii()"/><br>

汉字:

<br>

<textarea id="native" rows="10" cols="100"></textarea>

</BODY>

</HTML>

### Bootstrap

### Jquery

### Css

#### zzy.css

input.required{background-color:yellow}

body{margin: 10px;}

### js

common



Components



Service



### business

#### service

