# 单点登录介绍

## 什么是单点登录

SSO英文全称Single Sign On，单点登录。SSO是在多个应用系统中，用户只需要登录一次就可以访问所有相互信任的应用系统，它包括可以将这次主要的登录映射到其他应用中用于同一个用户的登录的机制。它是目前比较流行的企业业务整合的解决方案之一。

## 为什么要有单点登录系统

集群环境下会出现要求用户多次登录的情况。

解决方案：

1、配置tomcat集群(500)。配置tomcat Session复制。节点数不要超过5个。

2、可以使用Session服务器（sso系统），保存Session信息。需要模拟Session。

**session和redis的共同特点：**

1、kv形式存储

2、过期时间

单点登录系统是使用redis模拟Session，实现Session的统一管理。

## 需求分析

登录:

把用户信息装到redis(token，user)，再把token装到cookie(token\_key,token)

查询

先从cookie中取出token，再通过token从redis中查询用户信息

# 工程搭建

usian\_sso\_service

Pom.xml:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <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">  <parent>  <artifactId>usian\_parent</artifactId>  <groupId>com.usian</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>usian\_sso\_service</artifactId>  <dependencies>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_mapper</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_utils</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_redis</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

Application.yml

|  |
| --- |
| spring:  application:  name: usian-sso-service  datasource:  driver-class-name: com.mysql.jdbc.Driver  url: jdbc:mysql://127.0.0.1:3306/usian?characterEncoding=UTF-8  username: root  password: 123  type: com.alibaba.druid.pool.DruidDataSource  redis:  cluster:  nodes:  - 192.168.245.133:7001  - 192.168.245.133:7002  - 192.168.245.133:7003  - 192.168.245.133:7004  - 192.168.245.133:7005  - 192.168.245.133:7006  jedis:  pool:  max-active: 20 #连接池最大连接数  max-idle: 10 #连接池中的最大空闲连接  min-idle: 5 # 连接池中的最小空闲连接  server:  port: 8098  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:8761/eureka/ |

Logback.xml:

启动类：

usian\_sso\_feign

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <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">  <parent>  <artifactId>usian\_parent</artifactId>  <groupId>com.usian</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>usian\_sso\_feign</artifactId>  <dependencies>  <!--Spring Cloud OpenFeign Starter -->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-openfeign</artifactId>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_pojo</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

usian\_sso\_web

Pom.xml:

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <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">  <parent>  <artifactId>usian\_parent</artifactId>  <groupId>com.usian</groupId>  <version>1.0-SNAPSHOT</version>  </parent>  <modelVersion>4.0.0</modelVersion>  <artifactId>usian\_sso\_web</artifactId>  <dependencies>  <dependency>  <groupId>com.usian</groupId>  <artifactId>usian\_sso\_feign</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId>  </dependency>  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  </dependency>  <dependency>  <groupId>com.usian</groupId>  <artifactId>common\_utils</artifactId>  <version>1.0-SNAPSHOT</version>  </dependency>  </dependencies>  </project> |

Application.yml:

|  |
| --- |
| spring:  application:  name: usian-sso-web  server:  port: 8099  eureka:  client:  service-url:  defaultZone: http://127.0.0.1:8761/eureka/ |

Logback.xml

启动类：

# 注册信息校验

Usian\_sso\_service

Com.usian.service.SSOSservice:

|  |
| --- |
| /\*\*  \* 用户注册校验  \* @param checkValue  \* @param checkFlag  \* @return  \*/  @Override  public Boolean checkUserInfo(String checkValue, Integer checkFlag) {  // 1、查询条件根据参数动态生成：1、2分别代表username、phone  TbUserExample tbUserExample = new TbUserExample();  TbUserExample.Criteria criteria = tbUserExample.createCriteria();  if(checkFlag==1){  criteria.andUsernameEqualTo(checkValue);  }else if(checkFlag==2){  criteria.andPhoneEqualTo(checkValue);  }  // 2、从tb\_user表中查询数据  List<TbUser> tbUsers = tbUserMapper.selectByExample(tbUserExample);  // 3、判断查询结果，如果查询到数据返回false。  if(tbUsers!=null && tbUsers.size()>0){  return false;  }  return true;  } |

# 用户注册

Usian\_sso\_service

Com.usian.service.SSOService:

|  |
| --- |
| /\*\*  \*用户注册  \* @param tbUser  \* @return  \*/  @Override  public Integer userRegister(TbUser tbUser) {  //将密码做加密处理。  tbUser.setPassword(MD5Utils.digest(tbUser.getPassword()));  //补齐数据  Date date = new Date();  tbUser.setCreated(date);  tbUser.setUpdated(date);  return tbUserMapper.insertSelective(tbUser);  } |

# 用户登录

Usian\_sso\_service

Com.usian.SSOService:

|  |
| --- |
| /\*\*  \* 用户登录  \* @param username  \* @param password  \* @return  \*/  @Override  public Map userLogin(String username, String password) {  // 1、判断用户名密码是否正确。  String pwd = MD5Utils.digest(password);  TbUserExample tbUserExample = new TbUserExample();  TbUserExample.Criteria criteria = tbUserExample.createCriteria();  criteria.andUsernameEqualTo(username);  criteria.andPasswordEqualTo(pwd);  List<TbUser> tbUsers = tbUserMapper.selectByExample(tbUserExample);  if(tbUsers!=null && tbUsers.size()>0){  TbUser tbUser = tbUsers.get(0);  // 2、登录成功后生成token。Token相当于原来的jsessionid，字符串，可以使用uuid。  String token = UUID.randomUUID().toString();  // 3、把用户信息保存到redis。Key就是token，value就是TbUser对象转换成json。  tbUser.setPassword(null);  redisClient.set(USER\_INFO+":"+token,tbUser);  // 5、设置key的过期时间。模拟Session的过期时间。  redisClient.expire(USER\_INFO+":"+token,SESSION\_EXPIRE);  Map<String, Object> map = new HashMap<>();  map.put("token",token);  map.put("userid",tbUser.getId());  map.put("username",tbUser.getUsername());  return map;  }  return null;  } |

## 通过token查询用户信息

业务逻辑

1、从url中取参数token

2、根据token查询redis

3、如果查询不到数据，前台删除cookie中的用户信息

1. 如果查询到数据，说明用户已经登录需要重置key的过期时间

Usian\_sso\_service

Application.yml:

|  |
| --- |
| #单点登录  USER\_INFO: USER\_INFO  SESSION\_EXPIRE: 86400 |

Com.usian.service.SSOService:

|  |
| --- |
| /\*\*  \* 查询用户登录是否过期  \* @param token  \* @return  \*/  @Override  public TbUser getUserByToken(String token) {  TbUser tbUser = (TbUser) redisClient.get(USER\_INFO + ":" + token);  if(tbUser!=null){  //需要重置key的过期时间。  redisClient.expire(USER\_INFO+":"+token,SESSION\_EXPIRE);  return tbUser;  }  return null;  } |

# 退出登录

Usian\_sso\_service

Com.usian.service.SSOService:

|  |
| --- |
| /\*\*  \* 用户退出登录  \* @param token  \*/  @Override  public Boolean logOut(String token) {  return redisClient.del(USER\_INFO + ":" + token);  } |