# 单点登录

## **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

## **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。

缺点：多个tomcat节点会导致session复制工作量大，影响效果。

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

优点：复合分布式系统的各司其职的原则，每个模块完成自己的任务，而且登录及注册也是个比较复杂的过程。

**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

## **2.1.创建usian\_sso\_service**

**pom.xml**

<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>

### **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: 1111

type: com.alibaba.druid.pool.DruidDataSource

redis:

cluster:

nodes:

- 192.168.233.131:7001

- 192.168.233.131:7002

- 192.168.233.131:7003

- 192.168.233.131:7004

- 192.168.233.131:7005

- 192.168.233.131:7006

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/>

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

### **logback.xml**

### **创建启动类**

@SpringBootApplication

@EnableDiscoveryClient

@MapperScan("com.usian.mapper")

public class SSOServiceApp {

public static void main(String[] args) {

SpringApplication.run(SSOServiceApp.class, args);

}

}

## **2.2.创建usian\_sso\_feign**

### **pom.xml**

<!--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>

## **2.3.创建usian\_sso\_web**

### **pom.xml**

<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>

### **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**

### **创建启动类**

@SpringBootApplication

@EnableDiscoveryClient

@EnableFeignClients

public class SSOWebApp {

public static void main(String[] args) {

SpringApplication.run(SSOWebApp.class, args);

}

}

# **注册信息校验**

## **usian\_sso\_web**

### **controller**

/\*\*

\* 用户注册与登录

\*/

@RestController

@RequestMapping("/frontend/sso")

public class SSOController {

@Autowired

private SSOSeriviceFeign ssoSeriviceFeign;

@RequestMapping("checkUserInfo/{checkValue}/{checkFlag}")

public Result checkUserInfo(@PathVariable String checkValue,

@PathVariable Integer checkFlag){

Boolean checkUserInfo= ssoSeriviceFeign.checkUserInfo(checkValue,checkFlag);

if(checkUserInfo){

return Result.ok(checkUserInfo);

}

return Result.error("校验失败");

}

}

## **usian\_sso\_feign**

@FeignClient("usian-sso-service")

public interface SSOServiceFeign {

@RequestMapping("/service/checkUserInfo/{checkValue}/{checkFlag}")

public boolean checkUserInfo(@PathVariable String checkValue,

@PathVariable Integer checkFlag);

}

## **usian\_sso\_service**

### **service**

/\*\*

\* 用户注册与登录业务层

\*/

@Service

@Transactional

public class SSOServiceImpl implements SSOService {

@Autowired

private TbUserMapper tbUserMapper;

/\*\*

\* 对用户的注册信息(用户名与电话号码)做数据校验

\*/

@Override

public boolean checkUserInfo(String checkValue, Integer checkFlag) {

TbUserExample example = new TbUserExample();

TbUserExample.Criteria criteria = example.createCriteria();

// 1、查询条件根据参数动态生成：1、2分别代表username、phone

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> list = tbUserMapper.selectByExample(example);

// 3、判断查询结果，如果查询到数据返回false。

if (list == null || list.size() == 0) {

// 4、如果没有返回true。

return true;

}

// 5、如果有返回false。

return false;

}

}

### **controller**

/\*\*

\* 用户注册与登录

\*/

@RestController

@RequestMapping("/service/sso")

public class SSOController {

@Autowired

private SSOService ssoService;

/\*\*

\* 对用户的注册信息(用户名与电话号码)做数据校验

\*/

@RequestMapping("/checkUserInfo/{checkValue}/{checkFlag}")

public boolean checkUserInfo(String checkValue, Integer checkFlag) {

return this.ssoService.checkUserInfo(checkValue, checkFlag);

}

}

# **用户注册**

## **usian\_sso\_web**

### **controller**

/\*\*

\* 用户注册

\*/

@RequestMapping("/userRegister")

public Result userRegister(TbUser user){

Integer userRegister = ssoServiceFeign.userRegister(user);

if(userRegister==1){

return Result.ok();

}

return Result.error("注册失败");

}

## **usian\_sso\_feign**

@RequestMapping("/service/sso/userRegister")

public Integer userRegister(@RequestBody TbUser user);

## **usian\_sso\_service**

### **controller**

/\*\*

\* 用户注册

\*/

@RequestMapping("/userRegister")

public Integer userRegister(@RequestBody TbUser user) {

return this.ssoService.userRegister(user);

}

### **service**

/\*\*

\* 用户注册

\*/

@Override

public Integer userRegister(TbUser user) {

//将密码做加密处理。

String pwd = MD5Utils.digest(user.getPassword());

user.setPassword(pwd);

//补齐数据

user.setCreated(new Date());

user.setUpdated(new Date());

return this.tbUserMapper.insert(user);

}

# **用户登录**

## **usian\_sso\_web**

### **controller**

/\*\*

\* 用户登录

\*/

@RequestMapping("/userLogin")

public Result userLogin(String username,String password){

Map map = ssoServiceFeign.userLogin(username, password);

if(map!=null){

return Result.ok(map);

}

return Result.error("登录失败");

}

## **usian\_sso\_feign**

@RequestMapping("/service/sso/userLogin")

public Map userLogin(@RequestParam String username, @RequestParam String password);

## **usian\_sso\_service**

### **controller**

/\*\*

\* 用户登录

\*/

@RequestMapping("/userLogin")

public Map userLogin(String username, String password) {

return this.ssoService.userLogin(username, password);

}

### **service**

@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 example = new TbUserExample();

TbUserExample.Criteria criteria = example.createCriteria();

criteria.andUsernameEqualTo(username);

criteria.andPasswordEqualTo(pwd);

List<TbUser> userList = this.tbUserMapper.selectByExample(example);

if(userList == null || userList.size() <= 0){

return null;

}

TbUser tbUser = userList.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,String> map = new HashMap<String,String>();

map.put("token",token);

map.put("userid",tbUser.getId().toString());

map.put("username",tbUser.getUsername());

return map;

# **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

4、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\_web**

### **controller**

/\*\*

\* 查询用户登录是否过期

\* @param token

\* @return

\*/

@RequestMapping("/getUserByToken/{token}")

@ResponseBody

public Result getUserByToken(@PathVariable String token) {

TbUser tbUser = ssoServiceFeign.getUserByToken(token);

if(tbUser!=null){

return Result.ok();

}

return Result.error("登录过期");

}

## **usian\_sso\_feign**

@PostMapping("/service/sso/getUserByToken/{token}")

public TbUser getUserByToken(@PathVariable String token);

## **usian\_sso\_service**

### **controller**

/\*\*

\* 查询用户登录是否过期

\* @param token

\* @return

\*/

@RequestMapping("/getUserByToken/{token}")

@ResponseBody

public TbUser getUserByToken(@PathVariable String token) {

return ssoService.getUserByToken(token);

}

### **service**

/\*\*

\* 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\_web**

/\*\*

\* 用户退出登录

\*/

@RequestMapping("/logOut")

public Result logOut(String token){

Boolean logOut = ssoServiceFeign.logOut(token);

if(logOut){

return Result.ok();

}

return Result.error("退出失败");

}

## **usian\_sso\_feign**

@PostMapping("/service/sso/logOut")

public Boolean logOut(@RequestParam String token);

## **usian\_sso\_service**

### **controller**

/\*\*

\* 用户退出登录

\*/

@RequestMapping("/logOut")

public Boolean logOut(String token){

return this.ssoService.logOut(token);

}

### **service**

/\*\*

\* 用户退出登录

\* @param token

\*/

@Override

public Boolean logOut(String token) {

return redisClient.del(USER\_INFO + ":" + token);

}