简单实现SSO单点登录之登录接口

笔记本：	WEB项目开发		
创建时间：	2018/10/30/周二 18:54	**更新时间：**	2018/10/30/周二 19:05
作者：	1634896520@qq.com		
URL：	about:blank		

简单实现SSO单点登录之登录接口

一、技术点

1、mybatis逆向工程（generator）自动创建mapper、model。

2、redis缓存存储sessionId、用户信息，主要用到redis设置过期时间特性以及redis高效内存读写（更多配置未实现，，后续优化），Jedis客户端控制

3、cookie与session以及区别，应用场景。

4、spring AOP（这个是用环绕通知实现的）

5、spring MVC

二、类

1、mybatis逆向工程自动生成（代码根据数据库表生成）

配置文件：**generatorConfig.xml**（修改了其中的要生成的表的配置以及数据库）

GH_model...model：**User.java、UserCriteria.java**

GH_model...mapper.generator：**UserMapper.java**

GH_model...mapping.generator: **UserMapper.xml**

2、Jedis客户端控制

GH_model...util: **jedisUtil.java**

```
import redis.clients.jedis.Jedis;

/**
* @Author : WJ
* @Date : 2018/10/24/024 19:51
*
* 注释：Jedis客户端
* 程序功能可扩展
*/
public class JedisUtil {

//连接host的 Redis 服务
public static final Jedis jedis = new Jedis("127.0.0.1");
//连接redis
public static Jedis redisConect(){
// jedis.auth("***********"); //redis登录密码，我的redis没有密码
return jedis;
}
}
```

功能：对redis进行客户端操作，主机host可修改，以及更多优化待改进，如主从复制

3、业务层

GH_model...service:**UserService.java**

```
import com.alibaba.fastjson.JSONObject;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
* @Author : WJ
* @Date : 2018/10/23/023 21:29
*/
public interface UserService {

/**用户登录检查**/
public JSONObject findUser(String userName, String userPassword, HttpServletResponse response, HttpServletRequest
request);

}
```

GH_model...service.Impl:**UserServiceImpl.java**

```
import com.alibaba.fastjson.JSONObject;
import com.jk.gh.enums.ResultEnum;
import com.jk.gh.mapper.generator.UserMapper;
import com.jk.gh.model.User;
import com.jk.gh.model.UserCriteria;
import com.jk.gh.service.UserService;
import com.jk.gh.util.*;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import redis.clients.jedis.Jedis;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.util.List;
import java.util.UUID;

/**
* @Author : WJ
* @Date : 2018/10/23/023 21:39
*/
@Service
public class UserServiceImpl implements UserService {

@Autowired
private UserMapper userMapper;

@Override
public JSONObject findUser(String userName, String userPassword, HttpServletResponse response, HttpServletRequest
request) {
//判空
if(StringUtil.isEmpty(userName)||StringUtil.isEmpty(userPassword)){
return ResultUtil.fail(ResultEnum.NULL_DATA);
}
//核对用户身份信息（查询数据库中是否有此用户,以及校验密码）
UserCriteria userCriteria = new UserCriteria();
userCriteria.createCriteria().andUserNameEqualTo(userName).andUserPasswordEqualTo(userPassword);
List<User> user = userMapper.selectByExample(userCriteria);
if(user==null){
```

```
return ResultUtil.fail(ResultEnum.LOGIN_ERROR);
}
/******************登录成功后生成token,创建 session会话，加载到redis缓存************/
//生成token,tokenKey,tokenValue
String token = user.get(0).getUserId()+UUID.randomUUID().toString();
String userToken=user.get(0).getUserName();

//服务器每创建一个session都会有一个相对应的sessionId
HttpSession session = request.getSession(true);
session.setAttribute(token,userToken);
String sessionId = session.getId();

try(//连接redis
Jedis jedis=JedisUtil.redisConect();
){
//保存在redis中
jedis.set(sessionId,token);
jedis.set(userToken,user.get(0).toString());
//设置redis中sessionId,userNameInRedis 过期时间
jedis.expire(sessionId,1800);
jedis.expire(userToken,1800);
}catch (Exception e){
return ResultUtil.fail(ResultEnum.REDIS_ERROR);
}
//登录成功
return ResultUtil.success(user.get(0));
}
}
```

功能：为用户中心的服务，进行业务逻辑处理，数据增删改查（本次实现了登录findUser的业务）

findUser具体实现：

对用户登录提交的用户名密码进行判空，数据库校验，以及添加sessionId和用户数据到redis缓存，创建session全局会话，返回用户登录成功状态码以及数据。

4、控制层

GH_web_page...control : **UserController.java**

```
import com.alibaba.fastjson.JSONObject;
import com.jk.gh.aop.SSOTokenCheckAspect;
import com.jk.gh.enums.ResultEnum;
import com.jk.gh.model.User;
import com.jk.gh.service.UserService;
import com.jk.gh.util.ResultUtil;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


/**
* @Author : WJ
* @Date : 2018/10/25/025 11:33
* <p>
* 注释: 用户中心
*/
```

```
@RestController
@RequestMapping("/user")
public class UserController {

@Autowired
private UserService userService;
private SSOTokenCheckAspect ssoTokenCheckAspect;

//用户登录，查询用户
@RequestMapping("/findUser")
@ResponseBody
public JSONObject findUser(User user, HttpServletResponse response, HttpServletRequest request) throws Exception{
JSONObject json = userService.findUser(user.getUserName(),user.getUserPassword(),response,request);
return json;
}

//获取当前用户所有信息
//用于测试单点登录
@RequestMapping("/getUser")
public JSONObject getUser() throws Exception{
return ResultUtil.fail(ResultEnum.UNKNOW_ERROR);
}
}
```

功能：对用户中心的请求进行处理，返回视图（本次实现登录请求 findUser 以及获取当前用户数据的请求getUser）

5、SSO拦截认证

GH_web_page...aop:**SSOTokenCheckAspect.java**

```
import com.jk.gh.enums.ResultEnum;
import com.jk.gh.util.JedisUtil;
import com.jk.gh.util.ResultUtil;
import com.jk.gh.util.StringUtil;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;
import org.springframework.stereotype.Component;
import org.springframework.web.context.request.RequestContextHolder;
import org.springframework.web.context.request.ServletRequestAttributes;
import redis.clients.jedis.Jedis;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

/**
* @Author : WJ
* @Date : 2018/10/24/024 21:24
* <p>
* 注释: 单点登录实现之拦截用户需要登录认证的所有接口
*/
@Aspect
@Component
public class SSOTokenCheckAspect {

private Object jsonObject= ResultUtil.fail(200001,"用户未登录或登录超时！"); //jsonObject默认返回值
private HttpServletRequest request =null;
private HttpSession session=null;

/**
```

```
* 匹配所有controller 包下 UserController 的方法
*/
@Pointcut(value = "execution(* com.jk.gh.controller..*(..))")
public void Point() { }



/**
* 在进入UserController下（除了登录action）所有需要用户认证的action之前执行
* 环绕通知
* @param point 切面对象
*/
@Around(value = "Point()")
public Object around(ProceedingJoinPoint point) throws Throwable {
// 拦截登录action
String methodName = point.getSignature().getName();
Boolean boo = StringUtil.startsWithAny(methodName, new String[]{"findUser"});
if(!boo){
request = ((ServletRequestAttributes)
RequestContextHolder.getRequestAttributes()).getRequest();
session = request.getSession();
//连接redis,采用jdk 7 新特性try with resource 语法，自动close jedis
try(Jedis jedis=JedisUtil.redisConect();){

//校验token
String sessionId = session.getId();
String token = jedis.get(sessionId);
if(token!=null){

//根据token获取session中存储的redis 用户信息的 Key
String userToken = (String) session.getAttribute(token);

if(userToken!=null){
//根据tokenInRedis拿到用户数据
String userToString = jedis.get(userToken);
//更新redis缓存时间
jedis.expire(sessionId,1800);
jedis.expire(userToken,1800);

System.out.println("token and userToken======"+token+"----------"+userToken);

return ResultUtil.success(userToString);
}else{
//用户登录超时
return jsonObject;
}
}else{
//浏览器未存储用户sessionId，跳转到登录页面
//或者返回未登录状态码
return jsonObject;
}
}catch (Exception e) {
e.printStackTrace();
return ResultUtil.fail(ResultEnum.REDIS_ERROR);
}
}
//继续执行目标方法
jsonObject = point.proceed();
//返回目标方法执行结果
return jsonObject;
}
```

```
}
```

功能：拦截UserController下的除了登录请求以外所有关于用户的请求（可修改添加拦截注册请求以及其他请求）

（单点登录的具体认证过程在此AOP中）

pom.xml需添加依赖：

```
<!--Jedis客户端-->
<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-redis</artifactId>

<version>1.6.2.RELEASE</version>

</dependency>

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.9.0</version>

</dependency>
```

spring-mvc.xml需配置开启aop注解：

```
<!--开启基于注解的aop功能 -->
<aop:aspectj-autoproxy></aop:aspectj-autoproxy>
```

三、单点登录认证过程以及原理

1、原理

①一次登录，同一浏览器内，无需重复登录

②单session原理：同一浏览器，同一域名下，只存储一个session，session存储在服务器上，默认关闭浏览器失效，客户端利用cookie存储sessionId，默认关闭浏览器失效。

③利用redis的过期时间设置特性，设置sessionId过期时间

④利用redis高效内存读写，在一次登录时，将用户数据写入redis，后续客户端请求性能更优。

2、认证过程

登录请求（findUser）：

①得到请求

②aop拦截，判断到无需认证处理，放行

③service层进行业务逻辑处理（**创建会话，写入sessionId与用户数据到redis，设置过期时间，cookie默认关闭浏览器过期**）

④返回登录成功

其他请求（如getUser）：

①得到请求

②aop拦截，**判断到需要认证处理，获取客户端存储的cookie值，得到sessionId，根据sessionId到redis缓存中查找token，如果没有，则返回未登录状态或者登录超时或者重定向到登录页面，反之根据token获取session中存储的用户数据令牌tokenUser，再根据tokenUser读取redis缓存中的用户数据，进行redis数据过期时间更新，将认证成功状态码以及用户数据返回给前端，认证成功。**

③其他无需认证请求，按照正常请求顺序到service层获取数据。

四、注册接口暂未开发

注销操作，主要是要删除redis缓存中的sessionId（jedis.del(sessionId)）