单点登录系统介绍

- 传统的登录

  用户名： 密码： 提交

- servlet中处理
  …

  String name = request.getParameter(“username”);
  String passwd = request.getParameter(“passwd”);
  User user = new User();
  user.setName(name);
  user.setPassword(passwd);
  if(“jack”.equals(name) && “123”.equals(passwd)){
  request.getSession().setAttribute(“user”,user);
  response.getWriter().print(“登陆成功”);
  }else{
  response.getWriter().print(“用户名或密码有误”);
  response.getWriter().flush();
  }
  index.jsp得到页面返回数据

```
<c:if test="${empty(user)}">
        <a href="login.jsp" >请登录</a>
        </c:if>
    <c:if test="${not empty(user)}">
        welcome,${user.name}
        </c:if>
```

taotao-project登录系统

```
//注册时保存密码
public Boolean saveUser(User user) {
    user.setId(null);
    user.setCreated(new Date());
    user.setUpdated(user.getCreated());
    // 密码通过MD5进行加密处理
    user.setPassword(DigestUtils.md5Hex(user.getPassword()));
    return this.userMapper.insert(user) == 1;
}
登录时doLogin
 @RequestMapping(value="doLogin",method=RequestMethod.POST)
    @ResponseBody
    public Map<String,Object> doLogin(@RequestParam("username") String usernam
e,
            @RequestParam("password") String password,HttpServletRequest reque
st,HttpServletResponse response){
        Map<String,Object> result = new HashMap<>();
        try{
            String token = this.userService.doLogin(username,password);
            if(null == token){
                //登陆失败
                result.put("status", 400);
            }else{
                //登陆成功
                result.put("status", 200);
                CookieUtils.setCookie(request, response, COOKIE_NAME, token);
            }
        }catch(Exception e){
            e.printStackTrace();
            // 登录失败
            result.put("status", 500);
        }
        return result;
    }


//userService中的doLogin()
 public String doLogin(String username, String password) throws Exception {
    User record = new User();
    record.setUsername(username);
    User user = this.userMapper.selectOne(record);
    if (null == user) {
        return null;
    }
    // 比对密码是否正确
    if (!StringUtils.equals(DigestUtils.md5Hex(password), user.getPassword()))
{
        return null;
    }

    // 登录成功
    // 生成token
    String token = DigestUtils.md5Hex(System.currentTimeMillis() + username);

    // 将用户数据保存到redis中MAPPER.writeValueAsString(user),数据会保存成json
    this.redisService.set("TOKEN_" + token, MAPPER.writeValueAsString(user), 6
0 * 30);
    /*
         * {"id":8,"username":"montser","phone":"13478245269"
         * ,"email":null,"created":1474279970000,"updated":1474279970000}
         * */
        return token;
    }

   重点：登录成功生成token=DigestUtils.md5Hex(System.currentTimeMillis() + userna
me);
   保存在redis中{
        key : TOKEN_+token
        value:{   } user对象的json格式
   }
     CookieUtils.setCookie(request, response, COOKIE_NAME, token);
     同时把用户数据保存在cookie中COOKIE_NAME = "TT_TOKEN";前台调用js中cookie插件，就
可以从cookie中直接拿到token.cookie中数据为:TT_TOKEN=token;
```

这样下次用户每次进入首页都会调用queryUsertByToken()方法去查找用户名。

```
public User queryUerByToken(String token){
String key = "TOKEN_"+token;
String jsonData = this.redisService.get(key);
if(StringUtils.isEmpty(jsonData)){
        return null;
    }
     try{
        //一旦查找到token,就表示用户再次登录，需要刷新用户的生存时间
        this.redisService.expire(key, 60*30);
        return MAPPER.readValue(jsonData, User.class);
        //把json数据封装成user对象
    }catch(Exception e){
        e.printStackTrace();
    }
    return null;
}
```

使用拦截器实现用户是否登录的校验
//使用springmvc中拦截器验证用户是否登录

```
 <!-- springmvc拦截器  -->
    <mvc:interceptors>
        <mvc:interceptor>
        //定义拦截器拦截的请求路径order指包名
            <mvc:mapping path="/order/**"/>
            <bean class="com.taotao.web.handlerInterceptor.UserLoginHandlerInt
erceptor"/>
        </mvc:interceptor>
    </mvc:interceptors>
```

//使用springmvc中拦截器验证用户是否登录

```java
public class UserLoginHandlerInterceptor implements HandlerInterceptor {
public static final String COOKIE_NAME = "TT_TOKEN";

@Autowired
private PropertieService propertieService;

@Autowired
private UserService userService;


/*
 * 返回为false 时，表示请求结束，后续的Interceptor 和Controller 都不会再执行；当返回值
为true时就会继续调用下一个Interceptor 的preHandle方法，
 * 如果已经是最后一个Interceptor 的时候就会是调用当前请求的Controller 方法。
 * */

@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse respo
nse, Object handler)
        throws Exception {

    UserThreadLocal.set(null);//清空当前线程中的User对象
    //登录页面sso.taotao.com/user/login.html
    String loginUrl = propertieService.TAOTAO_SSO_URL + "/user/login.html";
    String token = CookieUtils.getCookieValue(request, COOKIE_NAME);
    if (StringUtils.isEmpty(token)) {
        // 未登录状态,重定向到登录页面
        response.sendRedirect(loginUrl);
        return false;
    }
    //从redis中查询token对应的值
    User user = this.userService.queryUserByToken(token);
    if (null == user) {
        // 未登录状态
        response.sendRedirect(loginUrl);
        return false;
    }
    //处于登录状态

    UserThreadLocal.set(user); //将User对象放置到ThreadLocal中
    return true;
}

@Override
public void postHandle(HttpServletRequest request, HttpServletResponse respons
e, Object handler,
        ModelAndView modelAndView) throws Exception {

}

@Override
public void afterCompletion(HttpServletRequest request, HttpServletResponse re
sponse, Object handler,
        Exception ex) throws Exception {

}
}
```