## 工作日志

学号：1830090115

姓名：李响

* 学习内容：

|  |  |
| --- | --- |
| 序号 | 内容 |
| 1 | 单点登录系统SSO实现。 |
| 2 | 拦截器配置和取token登录的方法等。 |

* 心得体会：

传统登陆：

已登录

订单列表

未登录

用户是否登录

访问订单系统

失败

成功

把用户信息放入session

提交用户名密码

校验用户名密码

用户登录界面

2000~3000并发

一个tomcat 不能满足业务需求，需要做集群。

Tomcat2

Webapp

Session：空

Tomcat1

Webapp

Session：user1

用户请求

反向代理服务器nginx

webapp

Tomcat3

Session共享的问题：

1. tomcat做集群配置session复制。如果集群中节点很多，会形成网络风暴。推荐节点数量不要超过5个。
2. 分布式架构。拆分成多个子系统。



注册controller代码：

package top.lixiang007.taotao.sso.controller**;**import org.apache.commons.lang3.StringUtils**;**import org.springframework.beans.factory.annotation.Autowired**;**import org.springframework.http.converter.json.MappingJacksonValue**;**import org.springframework.stereotype.Controller**;**import org.springframework.web.bind.annotation.PathVariable**;**import org.springframework.web.bind.annotation.RequestMapping**;**import org.springframework.web.bind.annotation.RequestMethod**;**import org.springframework.web.bind.annotation.ResponseBody**;**import top.lixiang007.taotao.common.pojo.TaotaoResult**;**import top.lixiang007.taotao.common.utils.ExceptionUtil**;**import top.lixiang007.taotao.pojo.TbUser**;**import top.lixiang007.taotao.sso.service.RegisterService115**;***/\*\*  
 \* 用户注册  
 \*/  
/\*\*  
 \** ***@Author*** *Xiang Li  
 \** ***@Date*** *2021/9/3 上午9:43  
 \** ***@Since*** *version-1.0  
 \** ***@青软实训*** *\*/*@Controller  
@RequestMapping("/user")  
public class RegisterController {  
  
 @Autowired(required = false)  
 private RegisterService115 registerService**;** */\*\*  
 \* 数据校验接口  
 \** ***@param*** *param 参数  
 \** ***@param*** *type 校验参数类型  
 \** ***@param*** *callback jsonp的回调函数名称  
 \** ***@return*** *\*/* @RequestMapping("/check/{param}/{type}")  
 @ResponseBody  
 public Object checkData(@PathVariable String param**,** @PathVariable Integer type**,** String callback) {  
 try {  
 TaotaoResult result = registerService.checkData(param**,** type)**;** if (StringUtils.*isNotBlank*(callback)) {  
 MappingJacksonValue mappingJacksonValue = new MappingJacksonValue(result)**;** mappingJacksonValue.setJsonpFunction(callback)**;** return mappingJacksonValue**;** }  
 return result**;** } catch (Exception e) {  
 e.printStackTrace()**;** return TaotaoResult.*build*(**500,** ExceptionUtil.*getStackTrace*(e))**;** }  
 }  
  
  
 @RequestMapping(value = "/register"**,** method = RequestMethod.*POST*)  
 @ResponseBody  
 public TaotaoResult register(TbUser user) {  
 try {  
 TaotaoResult result = registerService.register(user)**;** return result**;** } catch (Exception e) {  
 e.printStackTrace()**;** return TaotaoResult.*build*(**500,** ExceptionUtil.*getStackTrace*(e))**;** }  
 }  
}

1. 接收用户名和密码
2. 校验用户名密码
3. 生成token，可以使用UUID
4. 把用户信息写入redis，key就是token
5. 把token写入cookie。
6. 返回登录成功需要把token返回给客户端。

登录controller：

package top.lixiang007.taotao.sso.controller**;**import org.apache.commons.lang3.StringUtils**;**import org.springframework.beans.factory.annotation.Autowired**;**import org.springframework.http.converter.json.MappingJacksonValue**;**import org.springframework.stereotype.Controller**;**import org.springframework.web.bind.annotation.PathVariable**;**import org.springframework.web.bind.annotation.RequestMapping**;**import org.springframework.web.bind.annotation.RequestMethod**;**import org.springframework.web.bind.annotation.ResponseBody**;**import top.lixiang007.taotao.common.pojo.TaotaoResult**;**import top.lixiang007.taotao.common.utils.ExceptionUtil**;**import top.lixiang007.taotao.sso.service.LoginService115**;**import javax.servlet.http.HttpServletRequest**;**import javax.servlet.http.HttpServletResponse**;***/\*\*  
 \* 用户登录  
 \*/  
/\*\*  
 \** ***@Author*** *Xiang Li  
 \** ***@Date*** *2021/9/3 上午9:54  
 \** ***@Since*** *version-1.0  
 \** ***@青软实训*** *\*/*@Controller  
@RequestMapping("/user")  
public class LoginController115 {  
  
 @Autowired(required = false)  
 private LoginService115 loginService**;** @RequestMapping(value = "/login"**,** method = RequestMethod.*POST*)  
 @ResponseBody  
 public TaotaoResult login(String username**,** String password**,** HttpServletRequest request**,** HttpServletResponse response) {  
 try {  
 TaotaoResult result = loginService.login(username**,** password**,** request**,** response)**;** return result**;** } catch (Exception e) {  
 e.printStackTrace()**;** return TaotaoResult.*build*(**500,** ExceptionUtil.*getStackTrace*(e))**;** }  
 }  
  
 @RequestMapping("/token/{token}")  
 @ResponseBody  
 public Object getUserByToken(@PathVariable String token**,** String callback) {  
 try {  
 TaotaoResult result = loginService.getUserByToken(token)**;** if (StringUtils.*isNotBlank*(callback)) {  
 MappingJacksonValue mappingJacksonValue = new MappingJacksonValue(result)**;** mappingJacksonValue.setJsonpFunction(callback)**;** return mappingJacksonValue**;** }  
 return result**;** } catch (Exception e) {  
 e.printStackTrace()**;** return TaotaoResult.*build*(**500,** ExceptionUtil.*getStackTrace*(e))**;** }  
 }  
  
 @RequestMapping("/logout/{token}")  
 @ResponseBody  
 public Object logout(@PathVariable String token**,** String callback**,** HttpServletRequest request**,** HttpServletResponse response) {  
 try {  
 TaotaoResult result = loginService.logout(token**,** request**,** response)**;** if (StringUtils.*isNotBlank*(callback)) {  
 MappingJacksonValue mappingJacksonValue = new MappingJacksonValue(result)**;** mappingJacksonValue.setJsonpFunction(callback)**;** return mappingJacksonValue**;** }  
 return result**;** } catch (Exception e) {  
 e.printStackTrace()**;** return TaotaoResult.*build*(**500,** ExceptionUtil.*getStackTrace*(e))**;** }  
 }  
}

当用户提交订单时此时必须要求用户登录，可以使用拦截器来实现。

拦截器的处理流程：

1. 拦截请求url
2. 从cookie中取token
3. 如果没有toke跳转到登录页面。
4. 取到token，需要调用sso系统的服务查询用户信息。
5. 如果用户session已经过期，跳转到登录页面
6. 如果没有过期，放行。

* 遇到问题：

|  |  |
| --- | --- |
| 序号 | 问题 |
| 1 | redis.clients.jedis.exceptions.JedisDataException: ERR Client sent AUTH, but no password is set |

* 解决方案：

|  |  |
| --- | --- |
| 序号 | 方案 |
| 1 | Redis服务器没有设置密码，故客户端不发动密码，注释掉该行代码。  //jedis.auth(REDIS\_PASSWORD); |

图形用户界面, 应用程序, 网站

描述已自动生成

