公众号: 后端元宇宙

目录导航

扫码关注,持续输出优质好文

昵称: 雨点的名字
园龄: 7年7个月
粉丝: 1528
关注: 2
+加关注

推荐一个分布式单点登录框架XXL-SSO!

有关单点登录(SSO)之前有写过两篇文章

- 一文读懂 JWT!
- 看完这篇不能再说不懂SSO原理了!

如果说XXL-JOB你可能并不陌生,它是非常火爆的一个分布式任务调度平台。但其实在该作者还有一个非常优秀的开源项目叫XXL-SSO,这两个个项目都是1000+Star。

就可以访问所有相互信任的应用系统。 拥有"轻量级、分

这里主要是通过对XXL-SSO源码的分析，将理论和实践结合!

一、快速接入sso

1、xxl-sso特性

1. 简洁：API直观简洁，可快速上手
2. 轻量级：环境依赖小，部署与接入成本较低
3. 单点登录：只需要登录一次就可以访问所有相互信任的应用系统
4. 分布式：接入SSO认证中心的应用，支持分布式部署
5. HA：Server端与Client端，均支持集群部署，提高系统可用性

2024年3月(6)
2023年9月(4)
2023年6月(5)
2023年4月(3)
2023年2月(4)
2023年1月(3)
2022年11月(4)
2022年9月(4)
2022年6月(4)
2022年5月(5)
2022年3月(3)
2022年2月(1)
2022年1月(3)
2021年12月(6)
2021年11月(7)
更多

评论排行榜

1. 来博客园已过3年半了(57)
2. 高并发下秒杀商品，必须知道的9个细节(25)
3. Redisson实现分布式锁(1)---原理(24)
4. 算法(3)---布隆过滤器原理(23)
5. RocketMQ(1)-架构原理(19)
6. RocketMQ(2)---Docker部署RocketMQ集群(17)
7. Redisson实现分布式锁(3)—项目落地实现(16)
8. 分布式事务(4)---RocketMQ实现分布式事务项目(14)
9. 分布式事务(3)---RocketMQ实现分布式事务原理(14)
10. RocketMQ(5)---RocketMQ重试机制(14)
11. MySQL（12）---纪录一次left join一对多关系而引起的BUG(14)
12. 微信扫码登陆（2）---本地调试工具ngrok、微信回调ngrok域名(14)
13. SpringBoot(17）---SpringBoot整合RocketMQ(13)
14. Redisson实现分布式锁(2)—RedissonLock(12)

6. 跨域：支持跨域应用接入SSO认证中心
7. Cookie+Token均支持：支持基于Cookie和基于Token两种接入方式，并均提供Sample项目
8. Web+APP均支持：支持Web和APP接入
9. 实时性：系统登陆、注销状态，全部Server与Client端实时共享
10. CS结构：基于CS结构，包括Server"认证中心"与Client"受保护应用"
11. 记住密码：未记住密码时，关闭浏览器则登录态失效；记住密码时，支持登录态自动延期，在自定义延期时间的基础上，原则上可以无限延期
12. 路径排除：支持自定义多个排除路径，支持Ant表达式,用于排除SSO客户端不需要过滤的路径

2、环境

- JDK：1.7+
- Redis：4.0+

3、 源码地址

- github：https://github.com/xuxueli/xxl-sso
- gitee：https://gitee.com/xuxueli0323/xxl-sso

4、 项目结构说明

```
- xxl-sso-server：中央认证服务，支持集群
- xxl-sso-core：Client端依赖
- xxl-sso-samples：单点登陆Client端接入示例项目
  - xxl-sso-web-sample-springboot：基于Cooki
  - xxl-sso-token-sample-springboot：基于Toke
```

5、 架构图

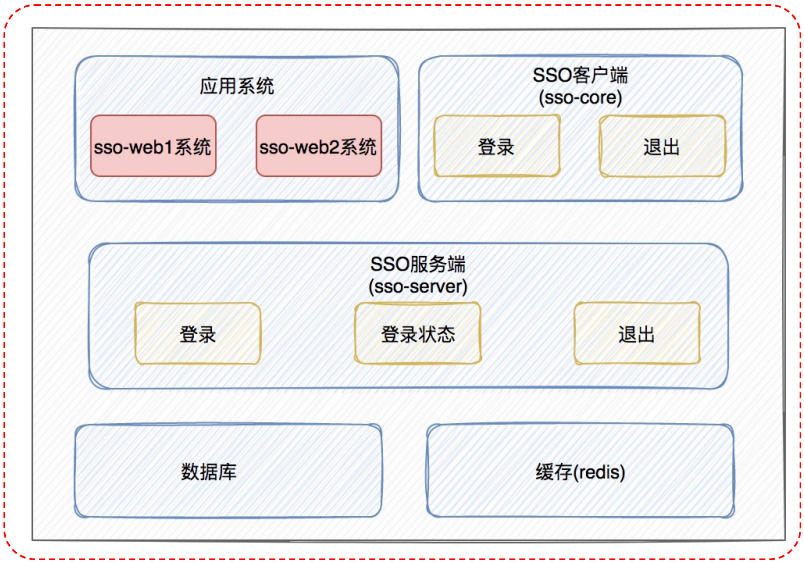
15. SpringBoot(16)—@ConditionalOnBean与@ConditionalOnClass(11)

推荐排行榜

1. 【HTTP协议】---HTTP协议详解(81)
2. Redisson实现分布式锁(1)---原理(78)
3. RocketMQ(1)-架构原理(54)
4. 高并发下秒杀商品，必须知道的9个细节(53)
5. 分布式事务(1)---2PC和3PC原理(49)
6. 【TCP协议】(2)---TCP三次握手和四次挥手(47)
7. java代码之美（1）---Java8 Lambda(43)
8. 分库分表(1) --- 理论(37)
9. 【Git】(1)---工作区、暂存区、版本库、远程仓库(37)
10. SpringBoot(16)—@ConditionalOnBean与@ConditionalOnClass(35)

最新评论

1. Re:给你的 SpringBoot 工程部署的 jar 包瘦瘦身吧!
@愚夫c jdk必须一致，不然也容易出问题...
--景伟·郭
2. Re:数据库界的Swagger：一键生成数据库文档!
😃
--你会很厉害的
3. Re:Netty+WebSocket 获取火币交易所数据项目
现在这个火币的API连不上了
--sunny_HH
4. Re:给你的 SpringBoot 工程部署的 jar 包瘦瘦身吧!
写的详细，不错，但现在稍微规模一定的公司都是通过jekins这种持续集成工具来部署的，



目录导航

应用系统：sso-web系统(8081端口)、sso-web系统(8082端口)（需要登录的系统）

SSO客户端：登录、退出（独立jar包给应用系统引用）

SSO服务端：登录（登录服务）、登录状态（提供登录状态校验/登录信息查询的服务）、退出（用户注销服务）

数据库：存储用户账户信息(一般使用Mysql,在当前项目中为了简便并没有查询数据库)

缓存：存储用户的登录信息(使用Redis)

二、快速接入XXL-SSO框架

1、 部署认证中心(sso-server)

只需要修改配置文件即可,配置文件位置：application.properties

```
## 配置redis
xxl.sso.redis.address=redis://118.31.224.65:6379
##  登录态有效期窗口，默认24H，当登录态有效期
xxl.sso.redis.expire.minute=1440
```

2、 部署'单点登陆Client端接入示例项目'

这里指需要接入SSO的系统，在当前项目有xxl-sso-web-sample-springboot和 xxl-sso-token-sample-springboot 两个示例项目，这里暂且以sso-web为示例。

你说的这种问题一般都不会太关注了

--kungge

5. Re:一文详解脏读、不可重复读、幻读

请问后面的文章发了吗?

--下沙grefus

6. Re:看一遍就懂：MVCC原理详解

MVCC能否解决了幻读问题那里说错了吧。 我查阅得知的是： 不可重复读是读取了其他事务更改的数据，针对update操作 幻读是读取了其他事务新增的数据，针对insert和delete操作 不可重复读（...

--zhu666

7. Re:order by 语句怎么优化?

如果select的字段比较多，这时候不能全加索引，所以避免不了走内部排序了吗，怎么优化呢

--pengfqa

8. Re:看完这篇你不能再说不懂SSO原理了!

全网把单点登录流程讲的最清楚的👍

--noodleOnce

9. Re:RocketMQ(5)---RocketMQ重试机制

那个timeout的例子我也没看懂，大佬能解释一遍吗?

--wsep

10. Re:docker-compose 搭建Prometheus+Grafana监控系统

还不错

--Edwin05

11. Re:OAuth 2.0详解

微信是服务商，A网站是第三方 第三方在登录时向服务商请求数据

--catcatcarrot

12. Re:看一遍就懂：MVCC原理详解

"不论是快照读和当前读都不能解决"这个说法有歧义，当前读其实就是加锁，也就是说当前

1）、 maven依赖

```
<dependency>
    <groupId>com.xuxueli</groupId>
    <artifactId>xxl-sso-core</artifactId>
    <version>${最新稳定版}</version>
</dependency>
```

2）配置 XxlSsoFilter

参考代码：com.xxl.sso.sample.config.XxlSsoConfig

```
@Bean
public FilterRegistrationBean xxlSsoFilterRegistra
    // xxl-sso, redis init
    JedisUtil.init(xxlSsoRedisAddress);
    // xxl-sso, filter init
    FilterRegistrationBean registration = new Filte
    registration.setName("XxlSsoWebFilter");
    registration.setOrder(1);
    registration.addUrlPatterns("/*");
    registration.setFilter(new XxlSsoWebFilter());
    registration.addInitParameter(Conf.SSO_SERV
    registration.addInitParameter(Conf.SSO_LOGC
    return registration;
}
```

3. application.properties 修改

```
## 中央认证服务地址
xxl.sso.server=http://ssoserver.com:8080/xxl-ssc
## 退出接口
xxl.sso.logout.path=/logout
## 排除走sso的接口
xxl-sso.excluded.paths=/excludedUrl
## redis地址
xxl.sso.redis.address=redis://118.11.214.65:6379
```

目录导航

度其实可以解决MVCC的幻读问题

--iceqing

13. Re:给你的 SpringBoot 工程部署的 jar 包瘦瘦身吧!

java.lang.NoClassDefFoundError: org/springframework/boot/SpringApplication at com.dhpay.fundflow.App...

--一叶兰舟飘

14. Re:看完这篇你不能再说不懂SSO原理了!

mark

--大漠孤阳

15. Re:Spring Event 观察者模式, 业务解耦神器

66666

--薛小谦

三、快速验证

1、修改host文件

修改Host文件：域名方式访问认证中心，模拟跨域与线上真实环境

```
127.0.0.1       ssoserver.com
127.0.0.1       webb.com
127.0.0.1       weba.com
```

2、启动项目

分别运行 “xxl-sso-server” 与 “xxl-sso-web-sample-springboot”,为了验证单点登录，这里sso-web需求启动两次，只是一次是8081端口，一次是8082端口。

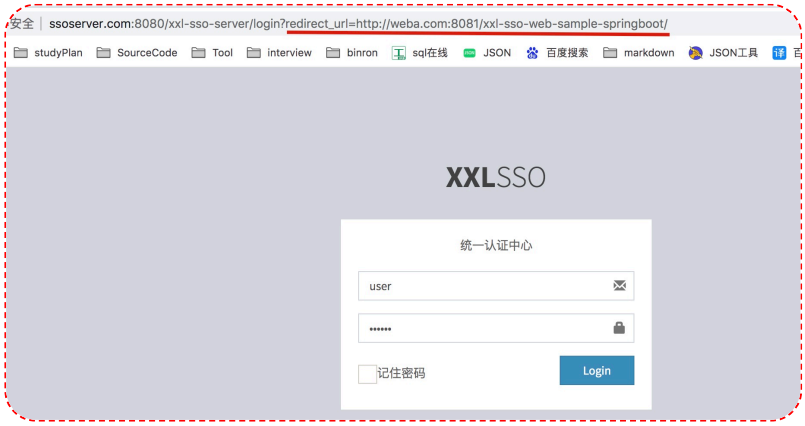
```
## 1、SSO认证中心地址：
http://ssoserver.com:8080/xxl-sso-server
## 2、Client01应用地址：
http://weba.com:8081/xxl-sso-web-sample-spri
## 3、Client02应用地址：
http://webb.com:8082/xxl-sso-web-sample-spri
```

3、验证

SSO登录流程

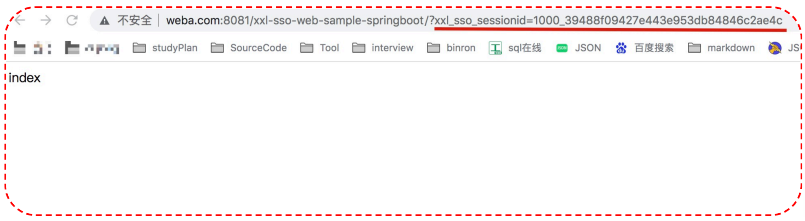
正常情况下，登录流程如下：

1、访问 "Client01应用地址" ，将会自动 redirect 到 "SSO认证中心地址" 登录界面

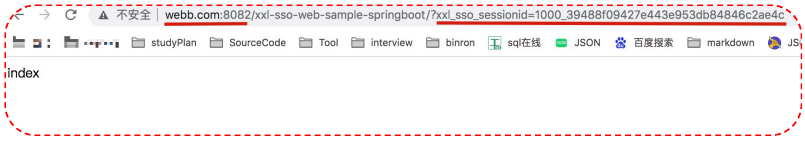


目录导航

2、成功登录后，将会自动 redirect 返回到 "Client01应用地址"，并切换为已登录状态



3、此时，访问 "Client02应用地址"，不需登陆将会自动切换为已登录状态



很明显 Client01 登录成功后,Client02无需再重新登录就可以访问了。

SSO注销流程

正常情况下，注销流程如下：

1、访问 "Client01应用地址" 配置的 "注销登陆path
2、此时，访问 "Client02应用地址"，也将会自动注

四、核心代码分析

1、SSO客户端(sso-core)拦截器

主要看sso拦截器流程就可以了

```
            // 3、校验用户是否存在，是否过期 这个
            XxlSsoUser xxlUser = SsoWebLoginHelp
            // 4、令牌校验失败
            if (xxlUser == null) {
                //获取当前请求地址
                String link = req.getRequestURL().toS
                // 重定向到sso认证服务的 登录接口
                String loginPageUrl = ssoServer.conca
                        + "?" + Conf.REDIRECT_URL + "=
                res.sendRedirect(loginPageUrl);
                return;
            }
            // ser sso user
            request.setAttribute(Conf.SSO_USER, xxl
            // 已经登录 放行
            chain.doFilter(request, response);
            return;
        }
```

在看下上面的loginCheck方法

JAVA 折叠 复制 全屏

```
    /**
     * 令牌校验
     *
     * @return 用户信息
     */
    public static XxlSsoUser loginCheck(HttpServl

        //去cookie去获取xxl_sso_sessionid 其实就是
        String cookieSessionId = CookieUtil.getVal

        // 这里去redis中获取用户信息 有可能获取不到
        //1、cookieSessionId为空 那么直接返回null
        //2、cookieSessionId不为空，但在redis获取
        //3、redis获取到了用户信息，但超过有效期了
        XxlSsoUser xxlUser = SsoTokenLoginHelper
        if (xxlUser != null) {
            return xxlUser;
        }
        // 如果获取不到 所以已经在其它系统退出登录
        SsoWebLoginHelper.removeSessionIdByCo
        //如果是 sso登录成功后 回调过来的 这个时候
        String paramSessionId = request.getParam
```

```
        xxlUser = SsoTokenLoginHelper.loginCheck
        if (xxlUser != null) {
            CookieUtil.set(response, Conf.SSO_SESSI
            return xxlUser;
        }
        return null;
    }
```

2、认证服务器(sso-server)登录接口

```
    /**
     * sso认证中心 登录接口
     */
    @RequestMapping(Conf.SSO_LOGIN)
    public String login(Model model, HttpServlet

        // 同样的 该判断sso上有没有全局会话
        XxlSsoUser xxlUser = SsoWebLoginHelper.l
        //如果 其它系统登录成功过 这个就不回为null
        //也不用在登录了
        if (xxlUser != null) {
            // success redirect
            String redirectUrl = request.getParamete
            if (redirectUrl!=null && redirectUrl.trim()

                String sessionId = SsoWebLoginHelpe
                String redirectUrlFinal = redirectUrl +
                return "redirect:" + redirectUrlFinal;
            } else {
                return "redirect:/";
            }
        }
        //只有全局会话不存在 才会跳转登录页面
        model.addAttribute("errorMsg", request.ge
        model.addAttribute(Conf.REDIRECT_URL, re
        return "login";
    }
```

3、认证服务器(sso-server)退出接口

```
@RequestMapping(Conf.SSO_LOGOUT)
public String logout(HttpServletRequest requ
    // 退出操作
    SsoWebLoginHelper.logout(request, respor
    // 跳转到登录页
    redirectAttributes.addAttribute(Conf.REDIR
    return "redirect:/login";
}
```

再来看下 logout 方法做了哪些事情

```
public static void logout(HttpServletRequest re
                          HttpServletResponse respons

    String cookieSessionId = CookieUtil.getVal
    if (cookieSessionId==null) {
        return;
    }
    //1、删除全局缓存 redis中 清除cookieSessio
    String storeKey = SsoSessionIdHelper.parse
    if (storeKey != null) {
        SsoLoginStore.remove(storeKey);
    }
    //2、清除全局会话
    CookieUtil.remove(request, response, Conf.
}
```

整个核心代码的逻辑都在这里了，其实结合上一篇的理论篇，理解起来就一点也不复杂了。

和原图片。其它情况一律禁止转载!

目录导航

分类: 【框架】-- SpringSecurity , 【Study】--优化经验

好文要顶 关注我 收藏该文 微信分享

雨点的名字
粉丝 - 1528 关注 - 2
会员号: 799
+加关注

« 上一篇: 看完这篇你不能再说不懂SSO原理了!
» 下一篇: IDEA插件Apifox,一键自动生成接口文档!

posted on 2023-02-22 08:40 雨点的名字 阅读(1511) 评论(1) 编辑 收藏 举报

会员力量,点亮园子希望 刷新页面 返回顶部

登录后才能查看或发表评论,立即 登录 或者 逛逛 博客园首页

【推荐】轻量又高性能的 SSH 工具 IShell: AI 加持,快人一步
【推荐】100%开源!大型工业跨平台软件C++源码提供,建模,组态!
【推荐】2024阿里云超值优品季,精心为您准备的上云首选必备产品
【推荐】「废话少说,放码过来」: 博客园2024夏季短袖T恤上架啦
【推荐】会员力量,点亮园子希望,期待您升级成为博客园VIP会员

目录导航

编辑推荐:

- ［架构师视角系列］风控场景下配置中心的设计实战
- 旧物利用 - 将机顶盒改造为一台 Linux 开发机!
- 这是DDD建模最难的部分（其实很简单）
- 为了落地DDD，我是这样“PUA”大家的
- 神秘 Arco 样式出现，祭出 Webpack 解决预期外的引用问题

阅读排行:

- 实习第一天，不小心透露了，我是拆迁户
- 记录兼职运维的一天
- 我们常用的地铁卡/银行卡，竟然运行着一个 Java 虚拟机
- 这就是为什么你学不会DDD
- 开源的 P2P 跨平台传文件应用「GitHub 热点速览」