

思涛之路

活在“全部的现在”——从当下出发，联结过去与未来

博客园 首页 新闻 新随笔 联系 管理 订阅

随笔- 34 文章- 0 评论- 38

一日有梦，憩于池畔。身后有笑声，惊日何故？闻曰：此潭甚浅。忽狂风起浪滔天，一龙跃入云端。留下半句诗云——莫笑池中无一物，岂非金鳞只待风！梦醒，告友人皆笑尔。

昵称：冷豪
园龄：1年8个月
粉丝：47
关注：7
+加关注

< 2017年9月 >

日	一	二	三	四	五	六
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
1	2	3	4	5	6	7

搜索

- 常用链接
- 我的随笔

我的评论

我的参与

最新评论

我的标签

更多链接
- 我的标签
- Java(10)

C++(6)

Spring(5)

数据结构与算法(4)

Hibernate(3)

MySQL(2)

设计模式(2)

数据结构(1)

Shiro(1)

Struts2(1)

更多
- 随笔分类
- 30分钟入门系列(3)

设计模式(3)

数据结构与算法(4)

- 随笔档案
- 2017年8月 (2)

2017年7月 (3)

2017年6月 (1)

2017年5月 (1)

2017年4月 (2)

2017年1月 (1)

2016年12月 (2)

2016年11月 (2)

2016年10月 (1)

2016年7月 (3)

2016年6月 (1)

2016年5月 (1)

2016年3月 (3)

2016年2月 (3)

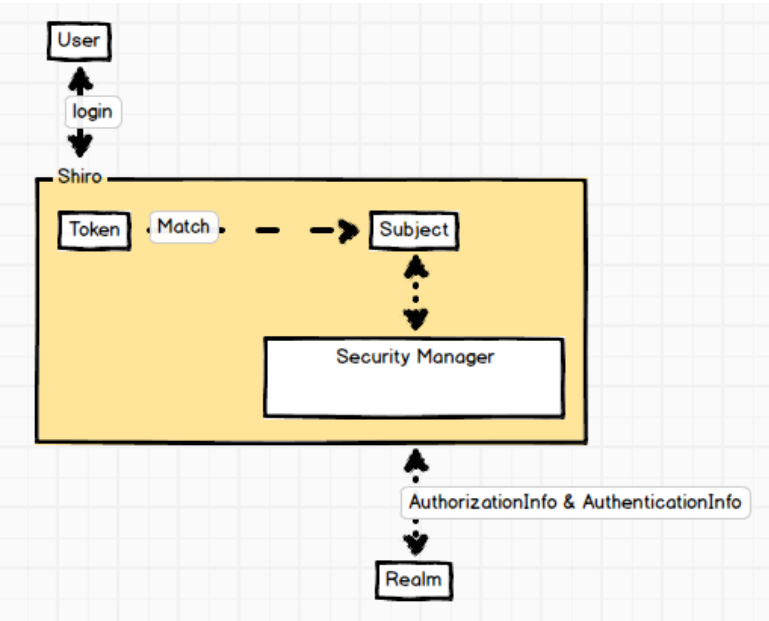
30分钟学会如何使用Shiro

本篇内容大多总结自张开涛的《跟我学Shiro》原文地址：
<http://jinnianshilongnian.iteye.com/blog/2018936>

我并没有全部看完，只是选择了一部分对我来说急需在项目使用的知识加以学习。并且对于大多数第一次接触Shiro的同学来说，掌握这些也应该足够了。

一、架构

要学习如何使用Shiro必须先从它的架构谈起，作为一款安全框架Shiro的设计相当精妙。Shiro的应用不依赖任何容器，它也可以在JavaSE下使用。但是最常用的环境还是JavaEE。下面以用户登录为例：



(1) 使用用户的登录信息创建令牌

```
UsernamePasswordToken token = new UsernamePasswordToken(username, password);
```

token可以理解为用户令牌，登录的过程被抽象为Shiro验证令牌是否具有合法身份以及相关权限。

(2) 执行登陆动作

```
SecurityUtils.setSecurityManager(securityManager); // 注入SecurityManager
Subject subject = SecurityUtils.getSubject(); // 获取Subject单例对象
subject.login(token); // 登陆
```

Shiro的核心部分是SecurityManager，它负责安全认证与授权。Shiro本身已经实现了所有的细节，用户完全可以把它当做一个黑盒来使用。SecurityUtils对象，本质上就是一个工厂类似Spring中的ApplicationContext。Subject是初学者比较难于理解的对象，很多人以为它可以等同于User，其实不然。Subject中文翻译：项目，而正确的理解也恰恰如此。它是你目前所设计的需要通过Shiro保护的项的一个抽象概念。通过令牌（ token ）与项目（ subject ）的登陆（ login ）关系，Shiro保证了项目整体的安全。

(3) 判断用户

Shiro本身无法知道所持有令牌的用户是否合法，因为除了项目的设计人员恐怕谁都无法得知。因此Realm是整个框架中为数不多的必须由设计者自行实现的模块，当然Shiro提供了多种实现的途径，本文只介绍最常见也最重要的一种实现方式——数据库查询。

(4) 两条重要的英文

我在学习Shiro的过程中遇到的第一个障碍就是这两个对象的英文名称：AuthorizationInfo，AuthenticationInfo。不用怀疑自己的眼睛，它们确实长的很像，不但长的像，就连意思都十分近似。

2016年1月 (8)

最新评论

1. Re:你所不了解的五条面试忠告
嗯，不错不错
--真的见证
2. Re:你所不了解的五条面试忠告
第二点很有感触，这个星期面试时，一家公司的技术面很顺利，后续和我聊的HR高高在上，我还没同意进他公司呢，他就一副我是你大爷的态度，果断拒了，当然拒掉原因不止这一个，和HR聊的时候大概知道这个公司是卸磨……
--testplusplus
3. Re:你所不了解的五条面试忠告
说的很对，和我自己的体会非常类似。我觉得关键点是掌握主动权，要设法引导面试官到自己熟悉的领域和项目，不清楚的地方就是回答没有接触过，或者说不是自己做的。
--爱编程的小兵
4. Re:你所不了解的五条面试忠告
赞一个，说的很棒
--云中摆渡的老船长
5. Re:30分钟学会如何使用Shiro
博主 有源代码可以私发一下吗？1056596401@qq.com
--余小西

阅读排行榜

1. 30分钟学会如何使用Shiro(72874)
2. 30分钟学会反向Ajax(8797)
3. 通过正则表达式实现简单xml文件解析(4457)
4. 简单使用Apache POI(1811)
5. 你所不了解的五条面试忠告(1245)

评论排行榜

1. 30分钟学会如何使用Shiro(14)
2. 30分钟学会反向Ajax(13)
3. 你所不了解的五条面试忠告(4)
4. SSH框架整合项目（一）——搭建平台和引入依赖(4)
5. 从立项到实施——记一位三流码农的挫败经历(2)

推荐排行榜

1. 30分钟学会如何使用Shiro(20)
2. 30分钟学会反向Ajax(12)
3. 你所不了解的五条面试忠告(8)
4. SSH框架整合项目（一）——搭建平台和引入依赖(3)
5. 事务隔离级别（二）(2)

在解释它们前首先必须要描述一下Shiro对于安全用户的界定：和大多数操作系统一样。用户具有角色和权限两种最基本的属性。例如，我的Windows登陆名称是learnhow，它的角色是administrator，而administrator具有所有系统权限。这样learnhow自然就拥有了所有系统权限。那么其他人需要登录我的电脑怎么办，我可以开放一个guest角色，任何无法提供正确用户名与密码的未知用户都可以通过guest来登录，而系统对于guest角色开放的权限极其有限。

同理，Shiro对用户的约束也采用了这样的方式。AuthenticationInfo代表了用户的角色信息集合，AuthorizationInfo代表了角色的权限信息集合。如此一来，当设计人员对项目中的某一个url路径设置了只允许某个角色或具有某种权限才可以访问的控制约束的时候，Shiro就可以通过以上两个对象来判断。说到这里，大家可能还比较困惑。先不要着急，继续往后看就自然会明白了。

二、实现Realm

如何实现Realm是本文的重头戏，也是比较费事的部分。这里大家会接触到几个新鲜的概念：缓存机制、散列算法、加密算法。由于本文不会专门介绍这些概念，所以这里仅仅抛砖引玉的谈几点，能帮助大家更好的理解Shiro即可。

（1）缓存机制

Ehcache是很多Java项目中使用的缓存框架，Hibernate就是其中之一。它的本质就是将原本只能存储在内存中的数据通过算法保存到硬盘上，再根据需求依次取出。你可以把Ehcache理解为一个Map<String,Object>对象，通过put保存对象，再通过get取回对象。



```
<?xml version="1.0" encoding="UTF-8"?>
<ehcache name="shirocache">
    <diskStore path="java.io.tmpdir" />

    <cache name="passwordRetryCache"
        maxEntriesLocalHeap="2000"
        eternal="false"
        timeToIdleSeconds="1800"
        timeToLiveSeconds="0"
        overflowToDisk="false"
        statistics="true">
    </cache>
</ehcache>
```



以上是ehcache.xml文件的基础配置，timeToLiveSeconds为缓存的最大生存时间，timeToIdleSeconds为缓存的最大空闲时间，当eternal为false时ttl和tti才可以生效。更多配置的含义大家可以去网上查询。

（2）散列算法与加密算法

md5是本文会使用的散列算法，加密算法本文不会涉及。散列和加密本质上都是将一个Object变成一串无意义的字符串，不同点是经过散列的对象无法复原，是一个单向的过程。例如，对密码的加密通常就是使用散列算法，因此用户如果忘记密码只能通过修改而无法获取原始密码。但是对于信息的加密则是正规的加密算法，经过加密的信息是可以通过秘钥解密和还原。

（3）用户注册

请注意，虽然我们一直在谈论用户登录的安全性问题，但是说到用户登录首先就是用户注册。如何保证用户注册的信息不丢失，不泄密也是项目设计的重点。



```
public class PasswordHelper {
    private RandomNumberGenerator randomNumberGenerator = new
SecureRandomNumberGenerator();
    private String algorithmName = "md5";
    private final int hashIterations = 2;

    public void encryptPassword(User user) {
        // User对象包含最基本的字段Username和Password
        user.setSalt(randomNumberGenerator.nextBytes().toHex());
        // 将用户的注册密码经过散列算法替换成一个不可逆的新密码保存进数据，散列过程使用了盐
        String newPassword = new SimpleHash(algorithmName, user.getPassword(),
            ByteSource.Util.bytes(user.getCredentialsSalt()), hashIterations).toHex();
        user.setPassword(newPassword);
    }
}
```



如果你不清楚什么叫加盐可以忽略散列的过程，只要明白存储在数据库中的密码是根据户注册时填写的密码所产生的一个新字符串就可以了。经过散列后的密码替换用户注册时的密码，然后将User保存进数据库。剩下的工作就丢给UserService来处理。

那么这样就带来了一个新问题，既然散列算法是无法复原的，当用户登录的时候使用当初注册时的密码，我们又应该如何判断？答案就是需要对用户密码再次以相同的算法散列运算一次，再同数据库中保存的字符串比较。

(4) 匹配

CredentialsMatcher是一个接口，功能就是用来匹配用户登录使用的令牌和数据库中保存的用户信息是否匹配。当然它的功能不仅如此。本文要介绍的是这个接口的一个实现类：HashedCredentialsMatcher

```
public class RetryLimitHashedCredentialsMatcher extends HashedCredentialsMatcher {
    // 声明一个缓存接口，这个接口是Shiro缓存管理的一部分，它的具体实现可以通过外部容器注入
    private Cache<String, AtomicInteger> passwordRetryCache;

    public RetryLimitHashedCredentialsMatcher(CacheManager cacheManager) {
        passwordRetryCache = cacheManager.getCache("passwordRetryCache");
    }

    @Override
    public boolean doCredentialsMatch(AuthenticationToken token, AuthenticationInfo info) {
        String username = (String) token.getPrincipal();
        AtomicInteger retryCount = passwordRetryCache.get(username);
        if (retryCount == null) {
            retryCount = new AtomicInteger(0);
            passwordRetryCache.put(username, retryCount);
        }
        // 自定义一个验证过程：当用户连续输入密码错误5次以上禁止用户登录一段时间
        if (retryCount.incrementAndGet() > 5) {
            throw new ExcessiveAttemptsException();
        }
        boolean match = super.doCredentialsMatch(token, info);
        if (match) {
            passwordRetryCache.remove(username);
        }
        return match;
    }
}
```

可以看到，这个实现里设计人员仅仅是增加了一个不允许连续错误登录的判断。真正匹配的过程还是交给它的直接父类去完成。连续登录错误的判断依靠Ehcache缓存来实现。显然match返回true为匹配成功。

(5) 获取用户的角色和权限信息

说了这么多才到我们的重点Realm，如果你已经理解了Shiro对于用户匹配和注册加密的全过程，真正理解Realm的实现反而比较简单。我们还得回到上文提及的两个非常类似的对象AuthorizationInfo和AuthenticationInfo。因为Realm就是提供这两个对象的地方。

```
public class UserRealm extends AuthorizingRealm {
    // 用户对应的角色信息与权限信息都保存在数据库中，通过UserService获取数据
    private UserService userService = new UserServiceImpl();

    /**
     * 提供用户信息返回权限信息
     */
    @Override
    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principals) {
        String username = (String) principals.getPrimaryPrincipal();
        SimpleAuthorizationInfo authorizationInfo = new SimpleAuthorizationInfo();
        // 根据用户名查询当前用户拥有的角色
        Set<Role> roles = userService.findRoles(username);
        Set<String> roleNames = new HashSet<String>();
        for (Role role : roles) {
            roleNames.add(role.getRole());
        }
        // 将角色名称提供给info
        authorizationInfo.setRoles(roleNames);
        // 根据用户名查询当前用户权限
        Set<Permission> permissions = userService.findPermissions(username);
        Set<String> permissionNames = new HashSet<String>();
    }
}
```

```

for (Permission permission : permissions) {
    permissionNames.add(permission.getPermission());
}
// 将权限名称提供给info
authorizationInfo.setStringPermissions(permissionNames);

return authorizationInfo;
}

/**
 * 提供账户信息返回认证信息
 */
@Override
protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken token) throws
AuthenticationException {
    String username = (String) token.getPrincipal();
    User user = userService.findByUsername(username);
    if (user == null) {
        // 用户名不存在抛出异常
        throw new UnknownAccountException();
    }
    if (user.getLocked() == 0) {
        // 用户被管理员锁定抛出异常
        throw new LockedAccountException();
    }
    SimpleAuthenticationInfo authenticationInfo = new
SimpleAuthenticationInfo(user.getUsername(),
        user.getPassword(), ByteSource.Util.bytes(user.getCredentialsSalt()),
getName());
    return authenticationInfo;
}
}

```

根据Shiro的设计思路，用户与角色之前的关系为多对多，角色与权限之间的关系也是多对多。在数据库中需要因此建立5张表，分别是用户表（存储用户名，密码，盐等）、角色表（角色名称，相关描述等）、权限表（权限名称，相关描述等）、用户-角色对应中间表（以用户ID和角色ID作为联合主键）、角色-权限对应中间表（以角色ID和权限ID作为联合主键）。具体dao与service的实现本文不提供。总之结论就是，Shiro需要根据用户名和密码首先判断登录的用户是否合法，然后再对合法用户授权。而这个过程就是Realm的实现过程。

（6）会话

用户的一次登录即为一次会话，Shiro也可以代替Tomcat等容器管理会话。目的是当用户停留在某个页面长时间无动作的时候，再次对任何链接的访问都会被重定向到登录页面要求重新输入用户名和密码而不需要程序员在Servlet中不停的判断Session中是否包含User对象。启用Shiro会话管理的另一个用途是可以针对不同的模块采取不同的会话处理。以淘宝为例，用户注册淘宝以后可以选择记住用户名和密码。之后再次访问就无需登陆。但是如果你要访问支付宝或购物车等链接依然需要用户确认身份。当然，Shiro也可以创建使用容器提供的Session最为实现。

三、与SpringMVC集成

有了注册模块和Realm模块的支持，下面就是如何与SpringMVC集成开发。有过框架集成经验的同学一定知道，所谓的集成基本都是一堆xml文件的配置，Shiro也不例外。

（1）配置前端过滤器

先说一个题外话，Filter是过滤器，interceptor是拦截器。前者基于回调函数实现，必须依靠容器支持。因为需要容器装配好整条FilterChain并逐个调用。后者基于代理实现，属于AOP的范畴。

如果希望在WEB环境中使用Shiro必须首先在web.xml文件中配置

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://java.sun.com/xml/ns/javaee"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
    id="WebApp_ID" version="3.0">
    <display-name>Shiro_Project</display-name>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
    <servlet>
        <servlet-name>SpringMVC</servlet-name>

```

```

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
<init-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>classpath:springmvc.xml</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
<async-supported>true</async-supported>
</servlet>
<servlet-mapping>
  <servlet-name>SpringMVC</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-
class>
</listener>
<listener>
  <listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>
</listener>
<context-param>
  <param-name>contextConfigLocation</param-name>
  <!-- 将Shiro的配置文件交给Spring监听器初始化 -->
  <param-value>classpath:spring.xml,classpath:spring-shiro-web.xml</param-value>
</context-param>
<context-param>
  <param-name>log4jConfigLoaction</param-name>
  <param-value>classpath:log4j.properties</param-value>
</context-param>
<!-- shiro配置 开始 -->
<filter>
  <filter-name>shiroFilter</filter-name>
  <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
  <async-supported>true</async-supported>
  <init-param>
    <param-name>targetFilterLifecycle</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>shiroFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
<!-- shiro配置 结束 -->
</web-app>

```

熟悉Spring配置的同学可以重点看有绿字注释的部分，这里是使Shiro生效的关键。由于项目通过Spring管理，因此所有的配置原则上都是交给Spring。DelegatingFilterProxy的功能是通知Spring将所有的Filter交给ShiroFilter管理。

接着在classpath路径下配置spring-shiro-web.xml文件

```

<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.1.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-
3.1.xsd
    http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd">

  <!-- 缓存管理器 使用Ehcache实现 -->
  <bean id="cacheManager" class="org.apache.shiro.cache.ehcache.EhCacheManager">
    <property name="cacheManagerConfigFile" value="classpath:ehcache.xml" />
  </bean>

  <!-- 凭证匹配器 -->
  <bean id="credentialsMatcher" class="utils.RetryLimitHashedCredentialsMatcher">
    <constructor-arg ref="cacheManager" />
    <property name="hashAlgorithmName" value="md5" />
  </bean>

```

```

<property name="hashIterations" value="2" />
<property name="storedCredentialsHexEncoded" value="true" />
</bean>

<!-- Realm实现 -->
<bean id="userRealm" class="utils.UserRealm">
    <property name="credentialsMatcher" ref="credentialsMatcher" />
</bean>

<!-- 安全管理器 -->
<bean id="securityManager" class="org.apache.shiro.web.mgt.DefaultWebSecurityManager">
    <property name="realm" ref="userRealm" />
</bean>

<!-- Shiro的Web过滤器 -->
<bean id="shiroFilter" class="org.apache.shiro.spring.web.ShiroFilterFactoryBean">
    <property name="securityManager" ref="securityManager" />
    <property name="loginUrl" value="/" />
    <property name="unauthorizedUrl" value="/" />
    <property name="filterChainDefinitions">
        <value>
            /authc/admin = roles[admin]
            /authc/** = authc
            /** = anon
        </value>
    </property>
</bean>

<bean id="lifecycleBeanPostProcessor"
class="org.apache.shiro.spring.LifecycleBeanPostProcessor" />
</beans>

```

需要注意filterChainDefinitions过滤器中对于路径的配置是有顺序的，当找到匹配的条目之后容器不会再继续寻找。因此带有通配符的路径要放在后面。三条配置的含义是：/authc/admin需要用户有admin权限、/authc/**用户必须登录才能访问、/**其他所有路径任何人都可以访问。

说了这么多，大家一定关心在Spring中引入Shiro之后到底如何编写登录代码呢。

```

@Controller
public class LoginController {
    @Autowired
    private UserService userService;

    @RequestMapping("login")
    public ModelAndView login(@RequestParam("username") String username,
    @RequestParam("password") String password) {
        UsernamePasswordToken token = new UsernamePasswordToken(username, password);
        Subject subject = SecurityUtils.getSubject();
        try {
            subject.login(token);
        } catch (IncorrectCredentialsException ice) {
            // 捕获密码错误异常
            ModelAndView mv = new ModelAndView("error");
            mv.addObject("message", "password error!");
            return mv;
        } catch (UnknownAccountException uae) {
            // 捕获未知用户名异常
            ModelAndView mv = new ModelAndView("error");
            mv.addObject("message", "username error!");
            return mv;
        } catch (ExcessiveAttemptsException eae) {
            // 捕获错误登录过多的异常
            ModelAndView mv = new ModelAndView("error");
            mv.addObject("message", "times error!");
            return mv;
        }
        User user = userService.findByUsername(username);
        subject.getSession().setAttribute("user", user);
        return new ModelAndView("success");
    }
}

```


登录完成以后，当前用户信息被保存进Session。这个Session是通过Shiro管理的会话对象，要获取依然必须通过Shiro。传统的Session中不存在User对象。



```
@Controller
@RequestMapping("authc")
public class AuthcController {
    // /authc/** = authc 任何通过表单登录的用户都可以访问
    @RequestMapping("anyuser")
    public ModelAndView anyuser() {
        Subject subject = SecurityUtils.getSubject();
        User user = (User) subject.getSession().getAttribute("user");
        System.out.println(user);
        return new ModelAndView("inner");
    }

    // /authc/admin = user[admin] 只有具备admin角色的用户才可以访问，否则请求将被重定向至登录界面
    @RequestMapping("admin")
    public ModelAndView admin() {
        Subject subject = SecurityUtils.getSubject();
        User user = (User) subject.getSession().getAttribute("user");
        System.out.println(user);
        return new ModelAndView("inner");
    }
}
```



四、总结

Shiro是一个功能很齐全的框架，使用起来也很容易，但是要想用好却有相当难度。完整项目的源码就不在这里提供了，需要交流的同学可以给我留言或直接查阅张开涛的博客。如果大家感觉我写的还可以，也希望能给我一些反馈意见。

五、推荐

这是一个学习shiro的网站，希望系统学习同学可以前往。

分类: [30分钟入门系列](#)

标签: [Shiro](#)


好文要顶

关注我

收藏该文







[冷豪](#)
[关注 - 7](#)
[粉丝 - 47](#)
[+加关注](#)

20

0

« 上一篇: [代理模式和面向切面编程](#)
» 下一篇: [30分钟学会反向Ajax](#)

posted @ 2016-07-22 22:44 冷豪 阅读(72874) 评论(14) 编辑 收藏

发表评论

- #1楼 2017-03-22 23:16 | 全是为了他
楼主说的好详细：
推荐基于springMVC + Mybatis + Redis 的Shiro整套项目Demo，
推荐一下：<http://www.sojson.com/shiro>

支持(1) 反对(0)
- #2楼 2017-03-22 23:16 | 全是为了他
楼主说的好详细：
推荐基于springMVC + Mybatis + Redis 的Shiro整套项目Demo，
推荐一下：<http://www.sojson.com/shiro>。

支持(1) 反对(0)
- #3楼 2017-04-11 14:31 | orchidlove

思路清晰，写的不错！有用

支持(0) 反对(0)

#4楼 2017-05-18 11:23 | 兜兜里有tang

受教了

支持(0) 反对(0)

#5楼 2017-06-19 14:24 | 熊猫你好

你好，你的文章写的挺好的，但是我有几个问题想问下，你可以加我QQ吗？
1227431615

支持(0) 反对(0)

#6楼 2017-06-20 17:03 | 可可_可可

你好，能方便发下demo参考下吗，我的扣扣352919618

支持(0) 反对(0)

#7楼 2017-07-19 11:16 | 仰天大笑出门去

谢谢,有收获

支持(0) 反对(0)

#8楼 2017-07-20 20:03 | 永恒星辰陨落

Unable to obtain input stream for cacheManagerConfigFile [classpath:ehcache.xml]

支持(0) 反对(0)

#9楼 2017-07-20 20:04 | 永恒星辰陨落

博主，我跟着你配置，发现报错了，Unable to obtain input stream for cacheManagerConfigFile [classpath:ehcache.xml]，博主这要怎么办呢？

支持(0) 反对(0)

#10楼[楼主] 2017-07-20 21:01 | 冷豪

@ 永恒星辰陨落
看错误提示似乎和Ehcache有关。由于我有一年没有再研究这个东西了。不少东西都忘记了。推荐你先把Ehcache从架中删除。看能否启动起来，再逐步判断错误吧。

支持(0) 反对(0)

#11楼 2017-07-20 21:18 | 永恒星辰陨落

@
解决这个问题后又出现了Cannot convert value of type [org.apache.shiro.cache.ehcache.EhCacheManager] to required type [org.springframework.cache.CacheManager] for property 'cacheManager': no matching setters or conversion strategy found

支持(0) 反对(0)

#12楼[楼主] 2017-07-20 21:23 | 冷豪

应该是你删除了cacheManager以后，在Spring的注入中依然要求注入cacheManager造成的吧。你可以加我qq2772073

支持(0) 反对(0)

#13楼 2017-07-27 17:23 | 变富喷雾

牛逼clazz 没毛病烙铁 双击666

支持(0) 反对(0)

#14楼 2017-08-02 16:28 | 余小西

博主 有源代码可以私发一下吗？1056596401@qq.com

支持(0) 反对(0)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问网站首页](#)。

- 【推荐】50万行VC++源码: 大型组态工控、电力仿真CAD与GIS源码库
- 【推荐】腾讯云上实验室 1小时搭建人工智能应用
- 【推荐】可嵌入您系统的“在线Excel”！SpreadJS 纯前端表格控件
- 【推荐】阿里云“全民云计算”优惠升级



最新IT新闻:

- 牛奶+橙汁=结石？那我肚子里能堆小山了
 - Facebook和亚马逊有望竞标英超联赛转播权
 - 永久拉黑！京东严惩两家第三方店铺：卖假手机配件
 - 掌阅成湘均：通过资本整合一堆业务形成的竞争力不可靠
 - 在世界无车日这天，ofo一口气登陆四国六城
- » [更多新闻...](#)



最新知识库文章:

- 如何阅读计算机科学类的书
 - Google 及其云智慧
 - 做到这一点，你也可以成为优秀的程序员
 - 写给立志做码农的大学生
 - 架构腐化之谜
- » [更多知识库文章...](#)