# springboot整合shiro-登录失败次数限制(八)



这次讲讲如何限制用户登录尝试次数，防止坏人多次尝试，恶意暴力破解密码的情况出现，要限制用户登录尝试次数，必然要对用户名密码验证失败做记录， Shiro 中用户名密码的验证交给了CredentialsMatcher 所以在CredentialsMatcher里面检查，记录登录次数是最简单的做法。当登录失败次数达到限制,修改数据库中的状态字段，并返回前台错误信息。
因为之前的博客都是用的明文,这里就不对密码进行加密了,如果有需要加密,将自定义密码比较器从SimpleCredentialsMatcher改为HashedCredentialsMatcher 然后将对应的配置项打开就可以。

## 说在前面

非常抱歉，因为我之前整合的时候,只是注意功能,而没有注意细节,导致在登录失败之后,再次转发到 post方法/login 也就是真正的登录方法,导致 再次登录,然后导致下面密码错误3次之后 就 锁定 我设置的是5次.
所以将shiroConfig中的值改为 `shiroFilterFactoryBean.setLoginUrl("/");` 具体参考源代码。
另外 还需要将 自定义ShiroRealm 中 密码对比注销掉, 将密码对比 交给 底层的 密码比较器才可以 锁定用户,否则将 永远报密码错误。, 具体代码 如下:

```
        //获取用户名 密码 第二种方式
        UsernamePasswordToken usernamePasswordToken = (UsernamePasswordToken) authenticationToken;
        String username = usernamePasswordToken.getUsername();
        String password = new String(usernamePasswordToken.getPassword());

        //从数据库查询用户信息
        User user = this.userMapper.findByUserName(username);

        //可以在这里直接对用户名校验,或者调用 CredentialsMatcher 校验
        if (user == null) {
            throw new UnknownAccountException("用户名或密码错误！");
        }
        //这里将 密码对比 注销掉,否则 无法锁定  要将密码对比 交给 密码比较器
        //if (!password.equals(user.getPassword())) {
        //    throw new IncorrectCredentialsException("用户名或密码错误！");
        //}
        if ("1".equals(user.getState())) {
            throw new LockedAccountException("账号已被锁定,请联系管理员！");
        }

        //调用 CredentialsMatcher 校验 还需要创建一个类 继承CredentialsMatcher  如果在上面校验了,这个就不需要了
        //配置自定义权限登录器 参考博客:

        SimpleAuthenticationInfo info = new SimpleAuthenticationInfo(user, user.getPassword(), getName());
        return info;
```



修改登录方法改为登录之后,重定向到/index

```
@RequestMapping(value = "/login",method = RequestMethod.POST)
public String loginUser(HttpServletRequest request, String username, String password,boolean rememberMe, Model model, HttpSession session) {

    //对密码进行加密
    //password=new SimpleHash("md5", password, ByteSource.Util.bytes(username.toLowerCase() + "shiro"),2).toHex();
    //如果有点击  记住我
    UsernamePasswordToken usernamePasswordToken=new UsernamePasswordToken(username,password,rememberMe);
    //UsernamePasswordToken usernamePasswordToken = new UsernamePasswordToken(username,password);
    Subject subject = SecurityUtils.getSubject();
    try {
        //登录操作
        subject.login(usernamePasswordToken);
        return "redirect:index";
    } catch(Exception e) {
        //登录失败从request中获取shiro处理的异常信息 shiroLoginFailure:就是shiro异常类的全类名
        String exception = (String) request.getAttribute( name: "shiroLoginFailure");

        if(e instanceof UnknownAccountException){
            model.addAttribute( attributeName: "msg", attributeValue: "用户名或密码错误！");
        }

        if(e instanceof IncorrectCredentialsException){
            model.addAttribute( attributeName: "msg", attributeValue: "用户名或密码错误！");
        }

        if(e instanceof LockedAccountException){
            model.addAttribute( attributeName: "msg", attributeValue: "账号已被锁定,请联系管理员！");
        }

        //返回登录页面
        return "login";
    }
}
```



## 限制登录次数

自定义RetryLimitHashedCredentialsMatcher继承SimpleCredentialsMatcher

```java
package com.springboot.test.shiro.config.shiro;

import java.util.concurrent.atomic.AtomicInteger;

import com.springboot.test.shiro.modules.user.dao.UserMapper;
import com.springboot.test.shiro.modules.user.dao.entity.User;
import org.apache.log4j.Logger;
import org.apache.shiro.authc.AuthenticationInfo;
import org.apache.shiro.authc.AuthenticationToken;
import org.apache.shiro.authc.LockedAccountException;
import org.apache.shiro.authc.credential.SimpleCredentialsMatcher;
import org.apache.shiro.cache.Cache;
import org.apache.shiro.cache.CacheManager;
import org.springframework.beans.factory.annotation.Autowired;


/**
 * @author: WangSaiChao
 * @date: 2018/5/25
 * @description: 登陆次数限制
 */
public class RetryLimitHashedCredentialsMatcher extends SimpleCredentialsMatcher {

    private static final Logger logger = Logger.getLogger(RetryLimitHashedCredentialsMatcher.class);
    @Autowired
    private UserMapper userMapper;
    private Cache<String, AtomicInteger> passwordRetryCache;

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

    @Override
    public boolean doCredentialsMatch(AuthenticationToken token, AuthenticationInfo info) {

        //获取用户名
        String username = (String)token.getPrincipal();
        //获取用户登录次数
        AtomicInteger retryCount = passwordRetryCache.get(username);
        if (retryCount == null) {
            //如果用户没有登陆过,登陆次数加1 并放入缓存
            retryCount = new AtomicInteger(0);
            passwordRetryCache.put(username, retryCount);
        }
        if (retryCount.incrementAndGet() > 5) {
            //如果用户登陆失败次数大于5次 抛出锁定用户异常  并修改数据库字段
            User user = userMapper.findByUserName(username);
            if (user != null && "0".equals(user.getState())){
                //数据库字段 默认为 0  就是正常状态 所以 要改为1
                //修改数据库的状态字段为锁定
                user.setState("1");
                userMapper.update(user);
            }
            logger.info("锁定用户" + user.getUsername());
            //抛出用户锁定异常
            throw new LockedAccountException();
        }
        //判断用户账号和密码是否正确
        boolean matches = super.doCredentialsMatch(token, info);
        if (matches) {
            //如果正确,从缓存中将用户登录计数 清除
            passwordRetryCache.remove(username);
        }
        return matches;
    }

    /**
     * 根据用户名 解锁用户
     * @param username
     * @return
     */
    public void unlockAccount(String username){
        User user = userMapper.findByUserName(username);
        if (user != null){
            //修改数据库的状态字段为锁定
            user.setState("0");
            userMapper.update(user);
```

```
            passwordRetryCache.remove(username);
        }
    }

}
```

在shiroConfig中配置该bean

```
/**
 * 配置密码比较器
 * @return
 */
@Bean("credentialsMatcher")
public RetryLimitHashedCredentialsMatcher retryLimitHashedCredentialsMatcher(){
    RetryLimitHashedCredentialsMatcher retryLimitHashedCredentialsMatcher = new RetryLimitHashedCredentialsMatcher(ehCacheManager());

    //如果密码加密,可以打开下面配置
    //加密算法的名称
    //retryLimitHashedCredentialsMatcher.setHashAlgorithmName("MD5");
    //配置加密的次数
    //retryLimitHashedCredentialsMatcher.setHashIterations(1024);
    //是否存储为16进制
    //retryLimitHashedCredentialsMatcher.setStoredCredentialsHexEncoded(true);

    return retryLimitHashedCredentialsMatcher;
}
```

在shiroRealm中配置密码比较器

```
/**
 *  身份认证realm; (这个需要自己写，账号密码校验；权限等)
 * @return
 */
@Bean
public ShiroRealm shiroRealm(){
    ShiroRealm shiroRealm = new ShiroRealm();

    ......

    //配置自定义密码比较器
    shiroRealm.setCredentialsMatcher(retryLimitHashedCredentialsMatcher());
    return shiroRealm;
}
```

在ehcache-shiro.xml添加缓存项

```
<!-- 登录失败次数缓存
     注意 timeToLiveSeconds 设置为300秒 也就是5分钟
     可以根据自己的需求更改
 -->
<cache name="passwordRetryCache"
       maxEntriesLocalHeap="2000"
       eternal="false"
       timeToIdleSeconds="0"
       timeToLiveSeconds="300"
       overflowToDisk="false"
       statistics="true">
</cache>
```

在LoginController中添加解除admin用户限制方法

```
/**
 * 解除admin 用户的限制登录
 * 写死的 方便测试
 * @return
 */
@RequestMapping("/unlockAccount")
public String unlockAccount(Model model){
    model.addAttribute("msg","用户解锁成功");

    retryLimitHashedCredentialsMatcher.unlockAccount("admin");

    return "login";
}
```

注意： 为了方便测试,记得将 unlockAccount 权限改为 任何人可访问。

在login.html页面 添加 解锁admin用户的按钮

```
1  <a href="/unlockAccount">解锁admin用户</a></button>
```

测试结果