

Search HacPai



xjtushilei

一名学生，喜欢钻研，热爱技术

1 回帖 • 2.4K 浏览 • 10 个月前



spring-security-jwt Restful API Token 安全认证



介绍

自己尝试在基于 spring-boot 的 restful API 的安全认证上做的调研与测试。

微服务架构是未来的趋势，但是在 API 安全认证方面的文章不多，大多数是基于 MVC 和 session 的，但是微服务时代的 API 都是无状态的，遵循 Restful(附一篇比较好的博客介绍 restful [理解 RESTful 架构](#)) 约束。不能满足需求，所以我就推动一下，将文章弄新一点。

适合读者

- Spring-boot 的 Restful API 安全认证，无状态，无 session
- spring-data-jpa 进行用户权限存储管理，基于 mysql，不是网上一大堆教程基于内存的用户认真
- jwt 协议的 token

为什么不采用 httpBasic

由于基于 httpBasic 认证的很简单，网上教程一大堆，不做过多介绍。这里我采用的是基于 jwt 协议的 API 安全认证。

用户每次都在传输过程中传输用户名和密码，太不安全了

用户每次请求一个 url 都会访问数据库，会导致数据库的访问压力！太大了。

为什么不采用 OAuth2

OAuth2 协议也是十分优秀的，但是我还是个菜鸟，虽然理解了 OAuth2，实现感觉也不难，但是还米有看懂 spring-boot 的框架源码，所以在集成方面不是很顺手。

其实还有一点就是 OAuth2 相对 jwt 等来说还是相对复杂的。如果我们不对外提供提供授权服务，name 使用 OAuth 个人感觉还是相当费劲的，传输效率和模式上，都没有 jwt 等 token 来的方便。

其实，基于 token 的认证，大多数自己实现也是十分方便的，加密解密，存在 redis 里，然后自动过期，一切都很简单，大家自己设计适合自己的认证才是最关键的。

实现

基于 mysql 的用户认证

网上代码一大堆，其实就是继承 UserDetailsService 就行了，需要自己设计 jpa 的表，我的代码里有例子，可以随意扩展。

```
@Service
public class UserService implements UserDetailsService {
    @Autowired
    SysUserRepository sysUserRepository;
```

然后在 WebSecurityConfigurerAdapter 中进行配置自己的 UserDetailsService，很简单。

```
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    UserDetailsService customUserService() {
        return new UserService();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(customUserService());
    }
}
```

jwt 的 token 生成

就是 jwt 协议的 加密和解密。需要设置加密算法，加密密钥，过期时间等。

加密的时候，把 username 加进去，讲道理应该是 userid。这样我们前端就可以使用该 id 来进行请求其他服务。

解密的时候，需要进行过期校验，密钥校验等

```
class TokenAuthenticationService {
    static final long EXPIRATIONTIME = 1000*60*60*24*1; // 1 days
    static final String SECRET = "ThisIsASecret";
    static final String TOKEN_PREFIX = "Bearer";
    static final String HEADER_STRING = "Authorization";
}
```

```

static void addAuthentication(HttpServletResponse res, String username) {
    String JWT = Jwts.builder()
        .setSubject(username)
        .setExpiration(new Date(System.currentTimeMillis() + EXPIRATIONTIME))
        .signWith(SignatureAlgorithm.HS512, SECRET)
        .compact();
    res.addHeader(HEADER_STRING, TOKEN_PREFIX + " " + JWT);
}

static Authentication getAuthentication(HttpServletRequest request) {
    String token = request.getHeader(HEADER_STRING);
    if (token != null) {
        // parse the token.
        String user = Jwts.parser()
            .setSigningKey(SECRET)
            .parseClaimsJws(token.replace(TOKEN_PREFIX, ""))
            .getBody()
            .getSubject();

        return user != null ?
            new UsernamePasswordAuthenticationToken(user, null, emptyList()) :
            null;
    }
    return null;
}

```

jwt 的 filter 进行 rul 认证

这里设置 登录 只能从 post 进行，并设置了两个 filter 分别对 login 和其他 url 进行拦截。

```
@Override
```

```

protected void configure(HttpSecurity http) throws Exception {
    http.csrf().disable().authorizeRequests()

```

```

        .antMatchers("/").permitAll()
        .antMatchers(HttpMethod.POST, "/login").permitAll()
        .anyRequest().authenticated()
        .and()
        // We filter the api/login requests
        .addFilterBefore(new JWTLoginFilter("/login", authenticationManager()),
            UsernamePasswordAuthenticationFilter.class)
        // And filter other requests to check the presence of JWT in header
        .addFilterBefore(new JWTAuthenticationFilter(),
            UsernamePasswordAuthenticationFilter.class);
    }

```

如何使用

使用之前

由于我集成了 spring-data-jpa，所以在使用之前需要配置数据库，并插入一些数据。

还好，我在初始化工程的时候替你们做了。我在 init 目录中进行了初始化设置，所以会自动插入两个用户名。

```

@Override
public void run(String... args) throws Exception {
    logger.info(">>>>>>>>>>>>>>>服务启动检查1, 开始检查用户系统<<<<<<<<<<<<<<<");

    if (sysUserRepository.count() != 0) {
        logger.info(">>>>>>>>>>>>>>>用户已经初始化<<<<<<<<<<<<<<<");
    } else {
        logger.info(">>>>>>>>>>>>>>>不存在初始用户, 开始创建<<<<<<<<<<<<<<<");
        SysRole sysRole1 = new SysRole(1L, "ROLE_ADMIN");
        SysRole sysRole2 = new SysRole(2L, "ROLE_USER");
        sysRoleRepository.save(Arrays.asList(sysRole1, sysRole2));
    }
}

```

```

List<SysRole> roles1 = Arrays.asList(sysRole1);
SysUser sysUser1 = new SysUser(1L, "root", "nibudong");
sysUser1.setRoles(roles1);

List<SysRole> roles2 = Arrays.asList(sysRole2);
SysUser sysUser2 = new SysUser(2L, "shilei", "nicai");
sysUser2.setRoles(roles2);

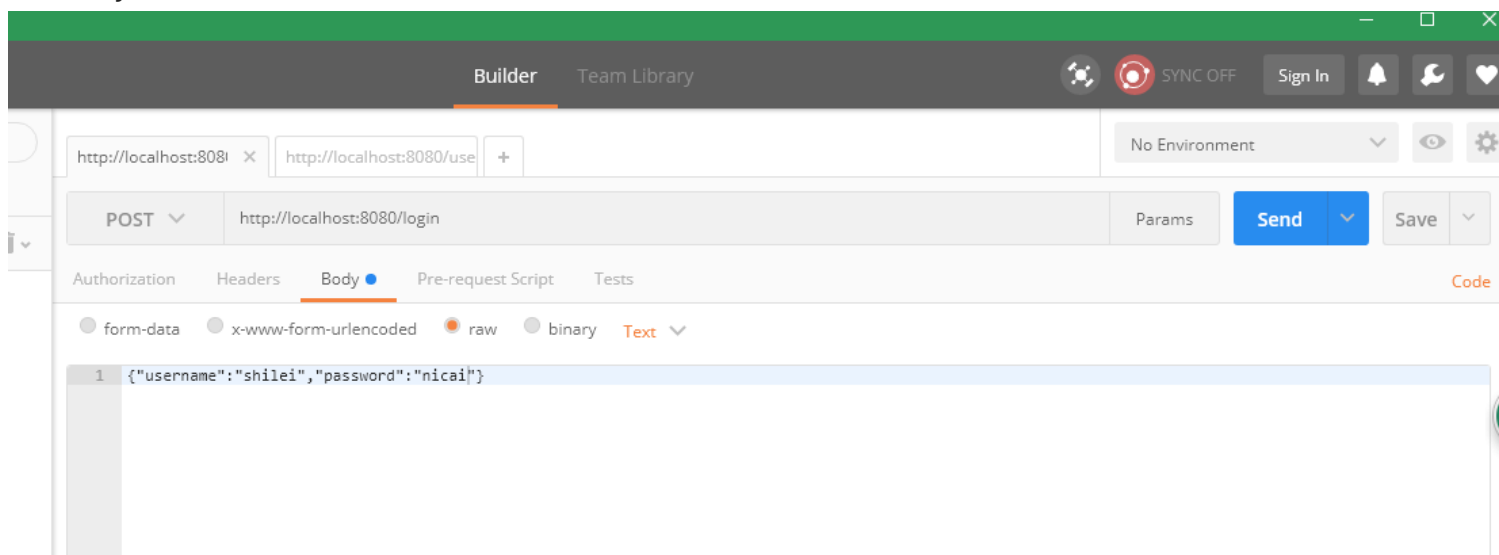
sysUserRepository.save(Arrays.asList(sysUser1, sysUser2));
logger.info(">>>>>>>>>>>>>>>初始用户创建结束<<<<<<<<<<<<<<<");
}

logger.info(">>>>>>>>>>>>>>>检查用户系统结束<<<<<<<<<<<<<<<");
}

```

如何使用

1. 先获得 jwt 的 token



Body Cookies **Headers (9)** Tests

Status: 200 OK Time: 19 ms

Authorization → Bearer
eyJhbGciOiJIUzUxMi9.eyJzdWIiOiJzaGlzZWkiLCJleHAiOiJ0OTcwNjMxNDI5LZ0ciLCk9apJ1DCMbVMLuGeAwq6dtMc5sN2OMjSCTf41OgraxZvNOJ99iGZWsEqKbGDWKHSTJApC-_RC7XTLzw

Cache-Control → no-cache, no-store, max-age=0, must-revalidate

Content-Length → 0

Date → Fri, 09 Jun 2017 02:52:29 GMT

2. 再请求 url

Builder Team Library SYNC OFF Sign In

http://localhost:8080/login http://localhost:8080/ X + No Environment

GET http://localhost:8080/users Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Code

Key	Value	Description	...	Bulk Edit	Presets
<input checked="" type="checkbox"/> Authorization	eyJhbGciOiJIUzUxMi9.eyJzdWIiOiJzaGlzZWkiLCJleHAiOiJ0OTcwNjMxNDI5LZ0ciLCk9apJ1DCMbVMLuGeAwq6dtMc5sN2OMjSCTf41OgraxZvNOJ99iGZWsEqKbGDWKHSTJApC-_RC7XTLzw				
New key	value	description			

Body Cookies Headers (9) Tests Status: 200 OK Time: 17 ms

Pretty Raw Preview Text

```
1 [{"users":[{"firstname":"Richard", "lastname":"Feynman"}, {"firstname":"Marie", "lastname":"Curie"}]}
```

代码

代码在的 github 里，<https://github.com/xjtushilei/spring-boot-security>

最后

登录请求一定要使用 HTTPS，否则无论 Token 做的安全性多好密码泄露了也是白搭

Spring RESTful jwt Java



1 回帖




请输入回帖内容...



xjtushilei • 10 个月前




相关帖子

 8-16 English code

 使用 Swagger 文档化和定义 RESTful API< 转 >


 请问如何写 java 后台接口?

 Java 开源博客 Solo 2.8.0 发布

 spring 配置文件中 util:properties 和 context:property-placeholder


 spring data jpa 动态表名的问题

随便看看


 小程序从注册到审核通过遇到的那些坑

 哪里有日本私活?

 Go 开源博客平台 Pipe 1.2.0 发布!

 【转载】Netty 学习 (三) -Netty 重要接口讲解

 【高效工具】使用 MyBatis Generator 自动生成代码

 版本控制工具小结

 test

为未来而构建的社区



微信
扫一扫

关于 API 公告 领域 标签 数据统计 广告投放
Feel easy about trust.

© 2018 hacpai.com 黑客与画家
Powered by B3log 开源 • Sym 2.6.0 • 86ms