

微服务登录之JWT

一、简介

1.1 登录的模式

目前市场上的登录模式，比较主流的：

1、多设备登录

常见：QQ、微信、陌陌 即时通讯 各大直播（虎牙、斗鱼、企鹅电竞）

不同设备可以同时在线

2、限定次数登录

常见：各大视频网站（腾讯视频、爱奇艺视频）、协同办公系统

3、单点登录

SSO:Single Sign ON

一处登录，处处可用

一处注销，处处注销

常见：PC端内部管理系统、PC端网站系统：百度（新闻、贴吧、视频、网盘等）

4、唯一登录

一个账号只能在线一个

常见：游戏（王者荣耀）、支付

5、普通登录

只校验账号和密码是否正确

常见：小外包项目、初始项目

模仿 相互借鉴

1.2 令牌

前后端分离：前端、App、小程序和后端完全分离，各干各个

好处：

1、分工明确 降低耦合

2、多终端的支持

前后端分离怎么保证登录信息的保存呢？

令牌 (Token) : 登录成功之后的身份标记

JWT:JSON Web Tokens

- 1、解决前后端分离的令牌的问题
- 2、解决分布式或微服务的登录令牌

JSON Web Tokens是一种开放的行业标准 [RFC 7519](#)方法，用于在双方之间安全地表示登录认证令牌安全地实现身份验证

官网: <https://jwt.io/>

官网可以实现: 解码, 验证和生成JWT

1.3 JWT格式

JWT是由三部分组成:

1、Header

头部: json对象, 签名算法 (加密的方式)、令牌的类型 (jwt)

示例: {alg:"HS256",typ:"JWT"}

2、Payload

负载: json对象, 包含的内容, 传递的数据

官网给出了一些值:

iss:签发人

exp:过期时间

sub:主题

示例: {sub:"18515990152",exp:'2019-07-11 18:00:00'}

3、signature

签名: 是对前两部分的内容按照一定的规则, 进行加密

JWT格式就是将上述的三块内容, 进行Base64URL转码之后使用.拼接字符串 形成基于JWT实现的Token

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWUiOiJkZWVhbnR5ODkwaWibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTY2MjM5MDIyQyJ9.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJlV_adQssw5c


Base64URL:就是Base64升级版

url传递参数, 特殊字符: = / Base64URL会将base64编码之后的内容的 特殊字符给替换

= 替换 空

+替换 -

/替换_



DebuggerLibrariesIntroductionAskGet a T-shirt!Crafted by

EncodedPASTE A TOKEN HERE

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c

DecodedEDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
"alg": "HS256",
"typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

JWT默认的Java的加密方式 缩写形式

Java	Java	Java
<div><div><div><div>Sign</div><div>HS256</div></div><div><div>Verify</div><div>HS384</div></div><div><div>iss check</div><div>HS512</div></div><div><div>sub check</div><div>RS256</div></div><div><div>aud check</div><div>RS384</div></div><div><div>exp check</div><div>RS512</div></div><div><div>nbf check</div><div>ES256</div></div><div><div>iat check</div><div>ES384</div></div><div><div>jti check</div><div>ES512</div></div><div><div></div><div>PS256</div></div><div><div></div><div>PS384</div></div><div><div></div><div>PS512</div></div></div></div>	<div><div><div><div>Sign</div><div>HS256</div></div><div><div>Verify</div><div>HS384</div></div><div><div>iss check</div><div>HS512</div></div><div><div>sub check</div><div>RS256</div></div><div><div>aud check</div><div>RS384</div></div><div><div>exp check</div><div>RS512</div></div><div><div>nbf check</div><div>ES256</div></div><div><div>iat check</div><div>ES384</div></div><div><div>jti check</div><div>ES512</div></div><div><div></div><div>PS256</div></div><div><div></div><div>PS384</div></div><div><div></div><div>PS512</div></div></div></div>	<div><div><div><div>Sign</div><div>HS256</div></div><div><div>Verify</div><div>HS384</div></div><div><div>iss check</div><div>HS512</div></div><div><div>sub check</div><div>RS256</div></div><div><div>aud check</div><div>RS384</div></div><div><div>exp check</div><div>RS512</div></div><div><div>nbf check</div><div>ES256</div></div><div><div>iat check</div><div>ES384</div></div><div><div>jti check</div><div>ES512</div></div><div><div></div><div>PS256</div></div><div><div></div><div>PS384</div></div><div><div></div><div>PS512</div></div></div></div>

1.4 JWT工作模式

开发中基于JWT实现登录令牌的生成，用来代替简易的Token

使用步骤：

1、依赖jar

使用开源第三方技术 生成和解析JWT

2、设置合理的加密方式

3、生成并记录密钥

4、封装工具类

实现JWT的生成和解析

1.5 JWT的实现

Java实现对JWT操作

1、依赖jar

``

```
<!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->
<dependency>
    <groupId>io.jsonwebtoken</groupId>
    <artifactId>jjwt</artifactId>
    <version>0.9.1</version>
</dependency>
<!-- https://mvnrepository.com/artifact/com.auth0/java-jwt -->
<dependency>
    <groupId>com.auth0</groupId>
    <artifactId>java-jwt</artifactId>
    <version>3.8.1</version>
</dependency>
```

2、编写工具类

``

```
/**
 * @Author feri
 * @Date Created in 2019/7/11 15:04
 * 实现对JWT的封装处理
 */
public class Jwt_Util {
    //生成
    public static String createJWT(String id,int minutes,String content){
        //1、创建加密的技术-sha-256
        SignatureAlgorithm signatureAlgorithm=SignatureAlgorithm.HS256;
        //2、创建JWT建造者对象
        JwtBuilder jwtBuilder=Jwts.builder();
        //3、设置Jwt相关信息
        jwtBuilder.setId(id);//唯一
        jwtBuilder.setIssuedAt(new Date());//开始时间
        jwtBuilder.setExpiration(TimeUtil.getMinutes(minutes));//失效时间
    }
}
```

```

        jwtBuilder.setSubject(content);//设置JWT中的内容
        jwtBuilder.signWith(signatureAlgorithm,createKey());
        //4、生成JWT
        return jwtBuilder.compact();
    }

    //解析
    public static String parseJWT(String token){
        SecretKey key=createKey();
        Claims claims=Jwts.parser().setSigningKey(key).parseClaimsJws(token).getBody();
        return claims.getSubject();
    }

    //生成密钥
    private static SecretKey createKey(){
        byte[] keys=Jwt_Config.JWTKEY.getBytes();
        SecretKey key=new SecretKeySpec(keys,0,keys.length,"AES");
        return key;
    }
}

```

``

```

public class Jwt_Config {
    //JWT生成的KEY的初始字符串
    public static final String JWTKEY="pocketstate_1902";
}

```

3、测试

``

```

@Test
public void t1(){
    String m="18515990152";
    IdGenerator idGenerator=new IdGenerator();
    String pass=Jwt_Util.createJWT(idGenerator.nextId()+"",30,m);
    System.out.println("JWT: "+pass);
    System.out.println("JWT解析: "+Jwt_Util.parseJWT(pass));
}

```

官网校验JWT是否正确



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJxMjEyMTEiLCJpYXQiOiE1NjI4Mjk3NzAsImV4cCI6MTU2MjgzMTU3MCwic3ViIjoiaMTg1MTU5OTAxNTIifQ.CXQE  
KcjO6GJ_5RaJfP3akt7M0ZKNJY6dM4CKeXn-TtI
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
"alg": "HS256"
```

PAYLOAD: DATA

```
{  
  "jti": "121211",  
  "iat": 1562829770,  
  "exp": 1562831570,  
  "sub": "18515990152"  
}
```

VERIFY SIGNATURE

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  ...  
)
```

二、基于JWT实现微服务架构统一鉴权中心

2.1 多设备登录

登录接口需要传入：设备（手机(Android、IOS)、平板、PC、智能设备)

技术栈：

- 1、JWT 生成Token
- 2、Redis+Jedis 实现数据的共享

实现微服务架构统一鉴权中心

登录接口 多设备 一个设备只能登录一次

- 1、手机号和设备 令牌 String 验证当前是否可以登录 ZSet(分数： 值 令牌) Hash()

格式： jwt: phone:设备id 值： 令牌

- 2、令牌 用户信息 通过令牌获取用户信息 有效期 30分钟 String

格式： jwt:token 值 用户信息

注销接口 退出

找回密码 重新设置密码

类型名称	特点	使用场景
String	单条数据	优先考虑
List	保证添加顺序，元素可重复	榜单
Set	无法保证添加顺序，元素不可重复	考试题、
zSet	想对比Set,多个Score(分数，Double 可以重复)	榜单、成绩信息
Hash	键值对存储，键唯一 值可以重复	榜单，点赞

点赞功能的实现：hash结合Zset

Hash：存储 key=说说id 值=点赞数量

ZSet: 存储 分数=说说id 值=说说id:用户id

共享数据：

- 1、单个值 就考虑String
- 2、多个值 每个值都有有效期 String key必须有一定的规律
- 3、多个值，没有有效期，要么有效期一致

实现分布式Session共享的方式？

- 1、基于Redis 实现Session共享