JWT实战

一、Token验证

最近了解下基于 Token 的身份验证，很多大型网站也都在用，比如 Facebook，Twitter，Google+，Github 等等，比起传统的身份验证方法，Token 扩展性更强，也更安全点，非常适合用在 Web 应用或者移动应用上。Token 的中文有人翻译成 “令牌” ，意思就是，你拿着这个令牌，才能过一些关卡。

二、传统的Token验证

HTTP 是一种没有状态的协议，也就是它并不知道是谁是访问应用。这里我们把用户看成是客户端，客户端使用用户名还有密码通过了身份验证，不过下回这个客户端再发送请求时候，还得再验证一下。

解决的方法就是，当用户请求登录的时候，如果没有问题，我们在服务端生成一条记录，这个记录里可以说明一下登录的用户是谁，然后把这条记录的 ID 号发送给客户端，客户端收到以后把这个 ID 号存储在 Cookie 里，下次这个用户再向服务端发送请求的时候，可以带着这个 Cookie ，这样服务端会验证一个这个 Cookie 里的信息，看看能不能在服务端这里找到对应的记录，如果可以，说明用户已经通过了身份验证，就把用户请求的数据返回给客户端。

上面说的就是 Session，我们需要在服务端存储为登录的用户生成的 Session ，这些 Session 可能会存储在内存，磁盘，或者数据库里。我们可能需要在服务端定期的去清理过期的 Session 。

基于 Token 的身份验证方法

使用基于 Token 的身份验证方法，在服务端不需要存储用户的登录记录。大概的流程是这样的：

客户端使用用户名跟密码请求登录
服务端收到请求，去验证用户名与密码
验证成功后，服务端会签发一个 Token，再把这个 Token 发送给客户端
客户端收到 Token 以后可以把它存储起来，比如放在 Cookie 里或者 Local Storage 里
客户端每次向服务端请求资源的时候需要带着服务端签发的 Token
服务端收到请求，然后去验证客户端请求里面带着的 Token，如果验证成功，就向客户端返回请求的数据。

三、JWT+HA256验证

实施 Token 验证的方法挺多的，还有一些标准方法，比如 JWT(JSON Web Tokens) 。JWT 标准的 Token 有三个部分：

header
payload
signature

中间用点分隔开，并且都会使用 Base64 编码，所以真正的 Token 看起来像这样：

eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI4MDQ5NzAyNCIsImlhdCI6MTUxNzU1Mzg4Niwic3ViIjoiMTIzIiwiaXNzIjoiTVlKIiwiZXhwIjoxNTE3NjQwMjg2LCJyZXNwRGF0YSI6bnVsbCwicmVzcERlcyI6IuaIkOWKnyIsInJlc3BDb2RlIjoiMTAwIn0.kNgPmlCtgVcPZbJSMM2J30raRTmL557cZ-gFIcabhns

四、Java代码

需要使用JWT，则需要引用maven依赖包：

<dependency>

```
 <groupId>io.jsonwebtoken</groupId>
 <artifactId>jjwt</artifactId>
 <version>0.9.0</version>
</dependency>
```

加密密码类：

```
public class ApiKey {
    /**
     * 加密密码
     * @return
     */
    public static String getSecret() {
        return "secret";
    }
}
```

创建JWT和解析JWT的类：

```
import io.jsonwebtoken.*;

import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;
import java.security.Key;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;

/**
 * Created by MYJ on 2018/2/1.
 */
public class JWTUtil {

    /**
     * @param id
     * @param issuer    Issuer，发行者
     * @param subject   Subject，主题
     * @param ttlMillis Expiration time，过期时间
     * @return
     */
    public String createJWT(String id, String issuer, String subject, long ttlMillis) {

        //使用JWT签名算法来签署令牌
        SignatureAlgorithm signatureAlgorithm = SignatureAlgorithm.HS256;

        long nowMillis = System.currentTimeMillis();
        Date now = new Date(nowMillis);

        //我们将用ApiKey secret来签名JWT
        byte[] apiKeySecretBytes = DatatypeConverter.parseBase64Binary(ApiKey.getSecret());
        Key signingKey = new SecretKeySpec(apiKeySecretBytes, signatureAlgorithm.getJcaName());

        //设置JWT Claims
        JwtBuilder builder = Jwts.builder().setId(id)
                .setIssuedAt(now) //发行时间
                .setSubject(subject)
```

```
            .setIssuer(issuer)
            .signWith(signatureAlgorithm, signingKey);

        //设置过期时间
        if (ttlMillis >= 0) {
            long expMillis = nowMillis + ttlMillis;
            Date exp = new Date(expMillis);
            builder.setExpiration(exp);
        }

        Map<String, Object> map = new HashMap<>();
        map.put("respCode", "100");
        map.put("respDes", "成功");
        map.put("respData", null);
        builder.addClaims(map);

        //Builds the JWT and serializes it to a compact, URL-safe string
        return builder.compact();
    }

    /**
     * 验证JWT的方法
     * @param jwt
     */
    public void parseJWT(String jwt) {
        Jws<Claims> claimsJws = Jwts.parser()
                .setSigningKey(DatatypeConverter.parseBase64Binary(ApiKey.getSecret()))
                .parseClaimsJws(jwt);
        Claims claims = claimsJws.getBody();

        /*System.out.println("ID: " + claims.getId());
        System.out.println("Subject: " + claims.getSubject());
        System.out.println("Issuer: " + claims.getIssuer());
        System.out.println("Expiration: " + claims.getExpiration());
        System.out.println("respCode:" + claims.get("respCode"));*/
        System.out.println("header:" + claimsJws.getHeader());
        System.out.println("payload:" + claims);
        System.out.println("signature:" + claimsJws.getSignature());
    }

}
```