# JWT

**JSON Web Token（JWT）：**是一个开放的行业标准（RFC 7519），它定义了一种简介的、自包含的协 议格式，用于在通信双方传递json对象，传递的信息经过数字签名可以被验证和信任。JWT可以使用 HMAC算法或使用RSA的公钥/私钥对来签名，防止被篡改。

**官网：** https://jwt.io/

**标准：** https://tools.ietf.org/html/rfc7519

**JWT令牌的优点：**

1. jwt基于json，非常方便解析。
2. 可以在令牌中自定义丰富的内容，易扩展。
3. 通过非对称加密算法及数字签名技术，JWT防止篡改，安全性高。
4. 资源服务使用JWT可不依赖认证服务即可完成授权。

**缺点：** 1. JWT令牌较长，占存储空间比较大。

**JWT组成**

一个JWT实际上就是一个字符串，它由三部分组成，头部、载荷与签名。

**头部(Header)**

头部用于描述关于该JWT的最基本的信息，例如其类型（即JWT）以及签名所用的算法（如HMAC SHA256或RSA）等。这也可以被表示成一个JSON对象。

{  "alg": "HS256",

  "typ": "JWT"

}

**typ ：**是类型。

**alg ：**签名的算法，这里使用的算法是HS256算法

我们对头部的json字符串进行BASE64编码（网上有很多在线编码的网站），编码后的字符串如下： Base64 是一种基于64个可打印字符来表示二进制数据的表示方法。由于2的6次方等于64，所以每6 个比特为一个单元，对应某个可打印字符。三个字节有24个比特，对应于4个Base64单元，即3个字节 需要用4个可打印字符来表示。JDK 中提供了非常方便的 BASE64Encoder 和 BASE64Decoder ，用它们 可以非常方便的完成基于 BASE64 的编码和解码。

**负载(Payload)**

第二部分是负载，就是存放有效信息的地方。这个名字像是特指飞机上承载的货品，这些有效信息包含 三个部分：

**标准中注册的声明（建议但不强制使用）**

**iss:** jwt签发者

**sub:** jwt所面向的用户

**aud:** 接收jwt的一方

**exp:** jwt的过期时间，这个过期时间必须要大于签发时间

**nbf:** 定义在什么时间之前，该jwt都是不可用的.

**iat:** jwt的签发时间

**jti:** jwt的唯一身份标识，主要用来作为一次性token,从而回避重放攻击。

**签证、签名（signature）**

jwt的第三部分是一个签证信息，这个签证信息由三部分组成：

1. **header (base64后的)**
2. **payload (base64后的)**
3. **secret（盐，一定要保密）**

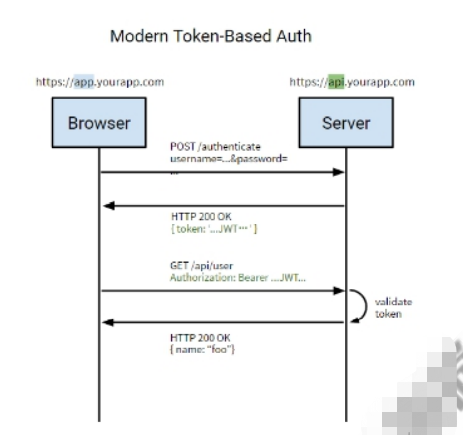
***这个部分需要base64加密后的header和base64加密后的payload使用.连接组成的字符串，然后通过 header中声明的加密方式进行加盐secret组合加密，然后就构成了jwt的第三部分.***

**JJWT简介**

JJWT是一个提供端到端的JWT创建和验证的Java库。永远免费和开源(Apache License，版本2.0)，JJW 很容易使用和理解。它被设计成一个以建筑为中心的流畅界面，隐藏了它的大部分复杂性。

**规范官网：**https://jwt.io/

下面是基于jwt的token验证：

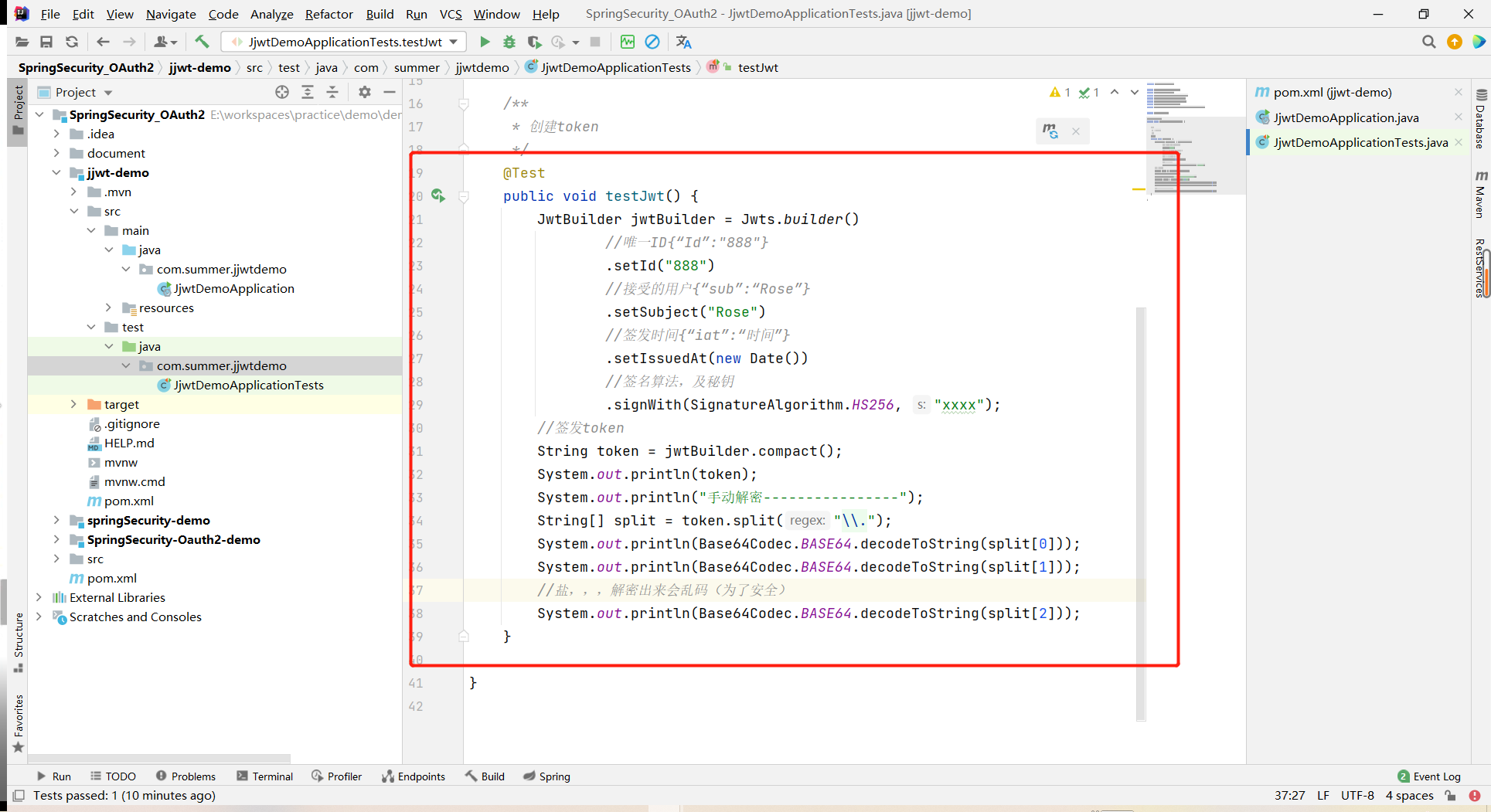


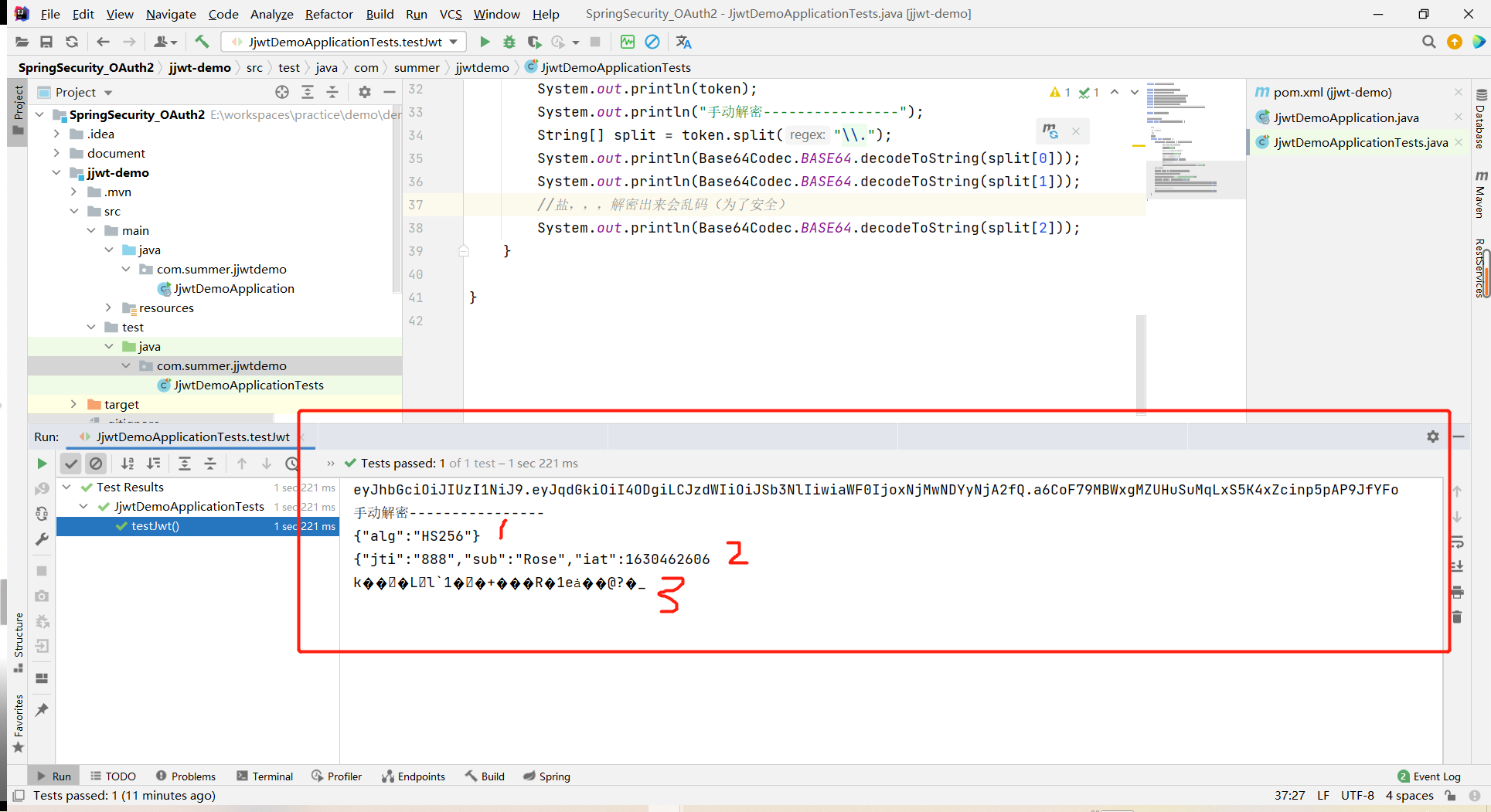
验证流程：

1. 浏览器输入用户名密码发送一个请求到服务器；
2. 服务器返回状态码200，表示请求成功，并携带一个基于jwt的token；
3. 浏览器在cookie中存储了整个token，接下来每次去访问服务器都会携带token；
4. 服务器会校验后续的token，如果和之前颁发的相同，则表示是同一个用户，就不需要再去登录。

**一、生成token**

接下里，我们手写一个jwt的token：

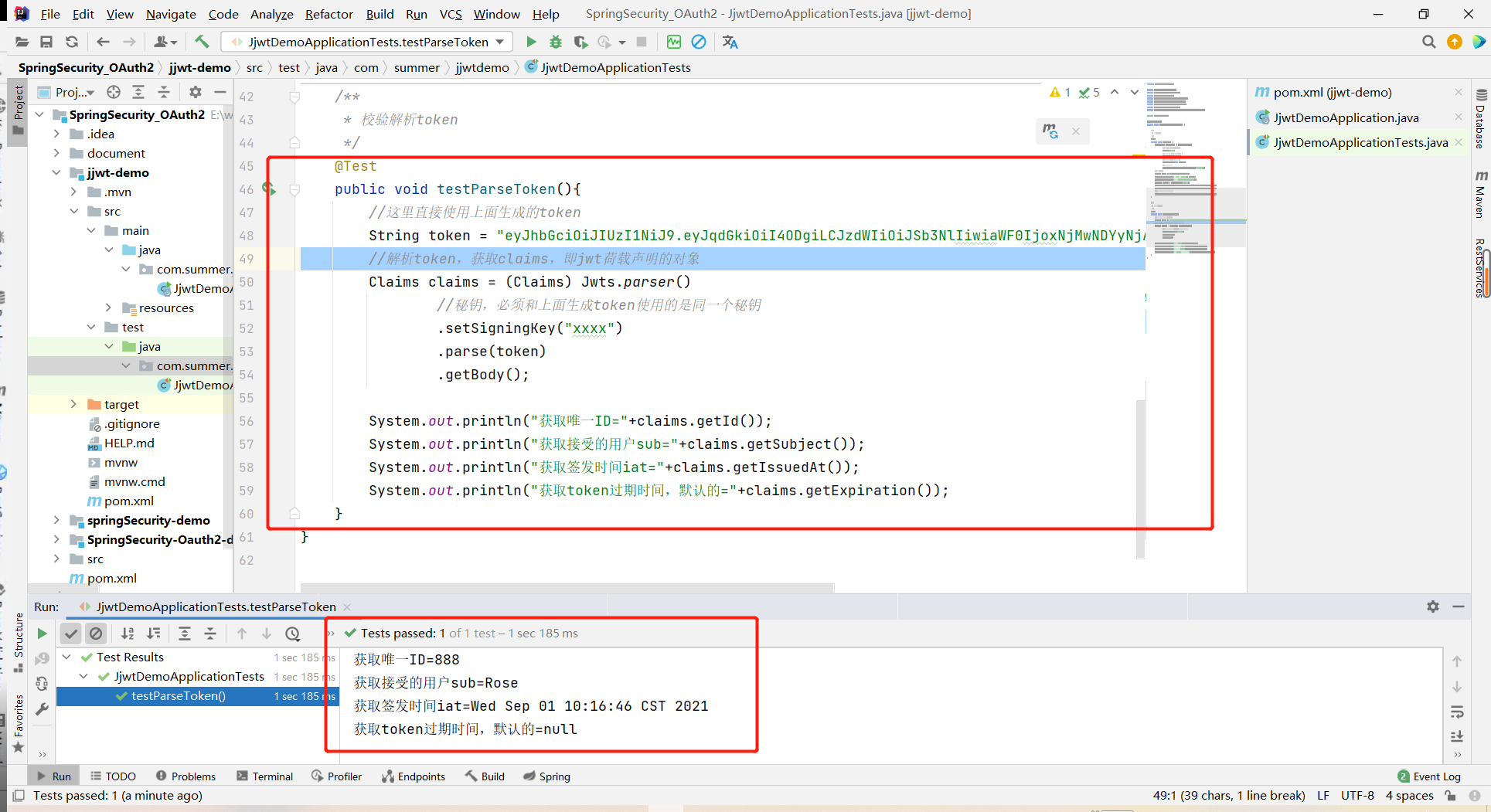




**可以看出，结果也是三部分组成，header、Payload、signature。**

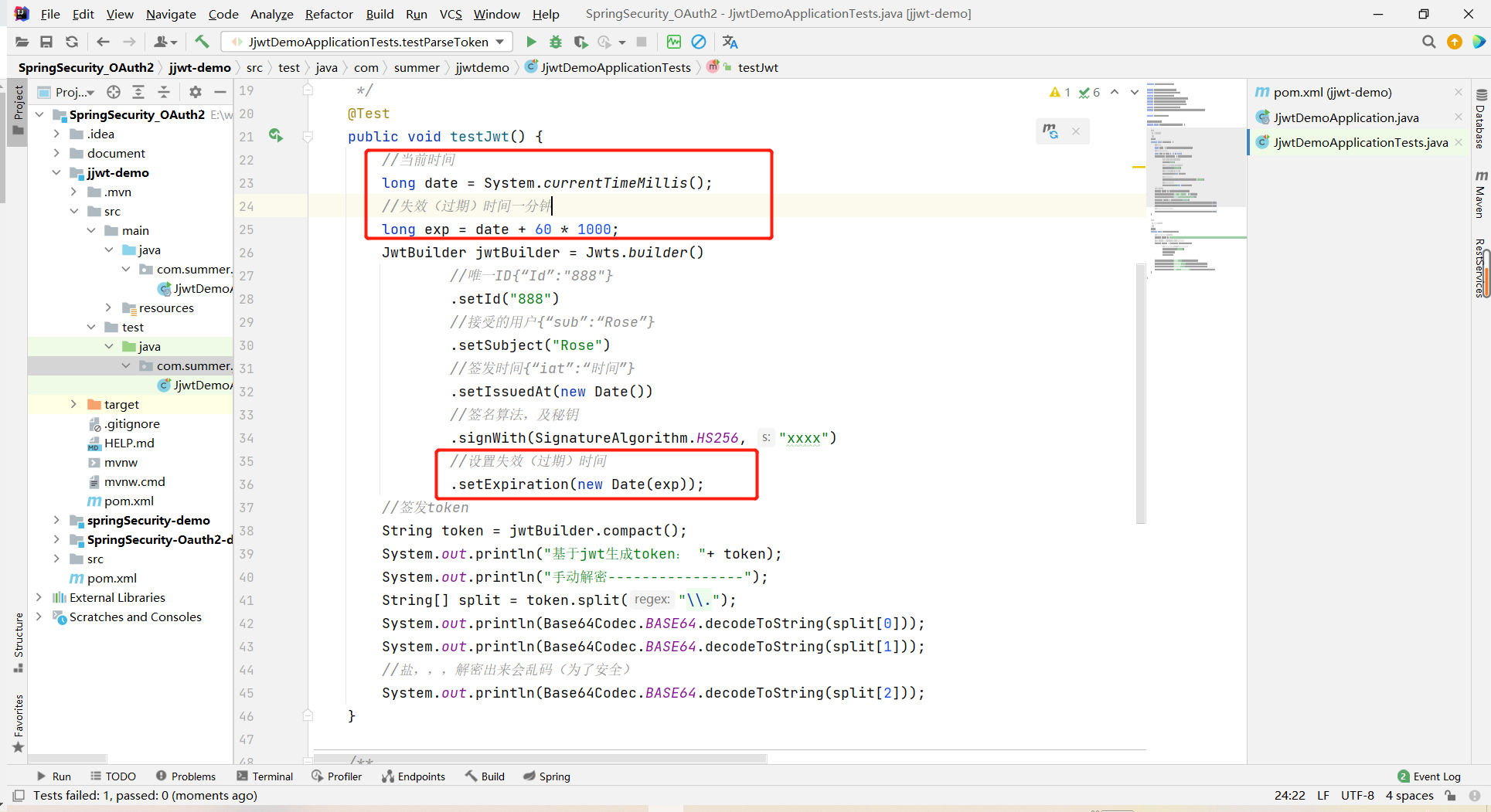
第三部分里边有“盐”，所以解密不了。提高了jwt的安全性。

**二、学习如何校验token？**

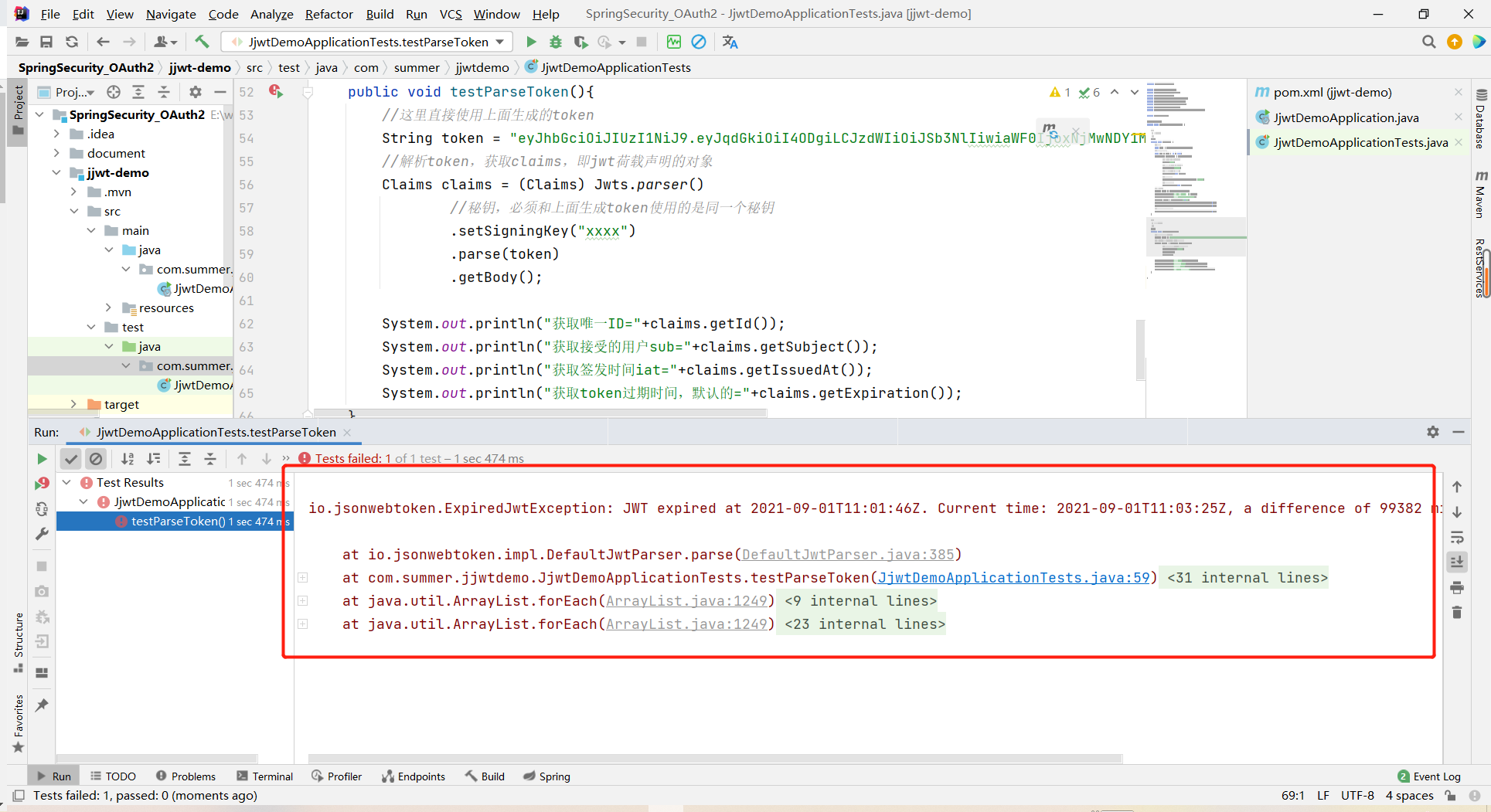


可以看出，解析完的token和生成的token内容是一样的；

1. **设置token过期时间**



1分钟后，再次解析，发现已经过期：



1. **自定义声明claims**

