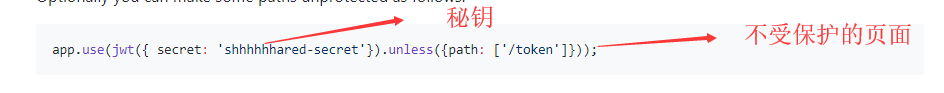
详见 github <https://github.com/auth0/express-jwt>

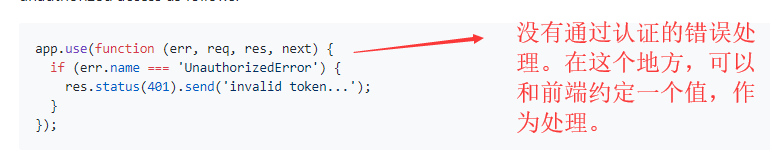
secret 秘钥



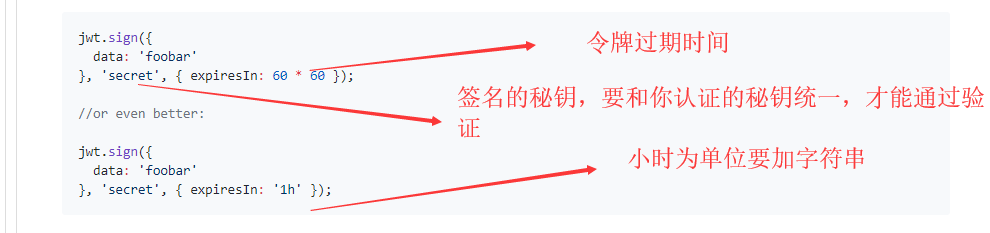
下面的是个自定义写法



错误处理



data中应该写的是 userID 以下是两种写法，区别在于过期时间



**实际应用：**

**routes / index.js中 写入**

**router.get("/getToken",(req, res) => {**

**var token = jwt.sign({**

**data: 'userId123456789'**

**}, 'zhang', { expiresIn: 60\*60 });**

**res.json({**

**data: {**

**token: token** //从一个中东的路由发送给前端token令牌，在这哥地方做密码验证

**}**

**})**

**})**

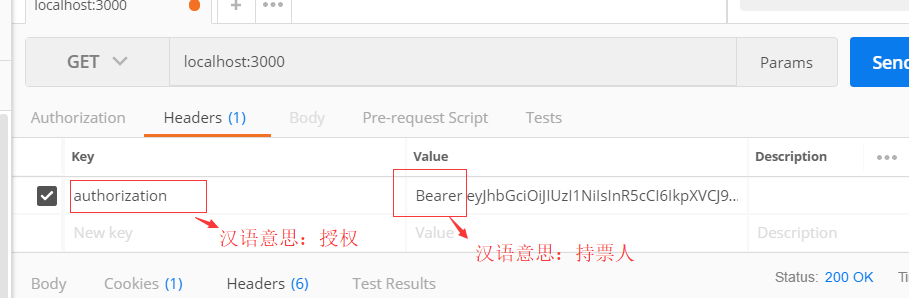
**app.js中 写入**

**var jwt = require("express-jwt");**

下面的代码放到需要登录判断的代码上，因为下面代码执行完成后会跳到下面紧挨着的的路由页面

app.use(jwt({ secret: 'yao'}).unless({path: ['/getToken']}));

Postman 中



**理论知识：**

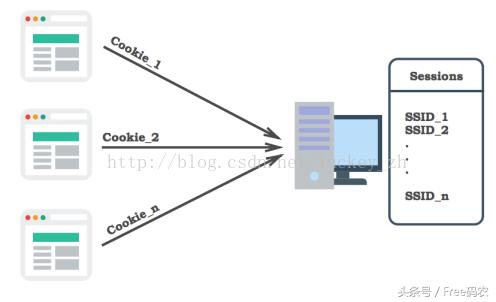
**JWT简介**

JWT(json web token)是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准。

JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源。比如用在用户登录上。

**基于session的登录认证**

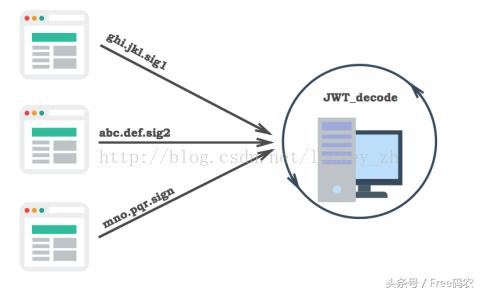
在传统的用户登录认证中，因为http是无状态的，所以都是采用session方式。用户登录成功，服务端会保证一个session，当然会给客户端一个sessionId，客户端会把sessionId保存在cookie中，每次请求都会携带这个sessionId。



国服最强JWT生成Token做登录校验讲解，看完保证你学会！

图片来源于网络博客

cookie+session这种模式通常是保存在内存中，而且服务从单服务到多服务会面临的session共享问题，随着用户量的增多，开销就会越大。而JWT不是这样的，只需要服务端生成token，客户端保存这个token，每次请求携带这个token，服务端认证解析就可。



国服最强JWT生成Token做登录校验讲解，看完保证你学会！

图片来源于网络博客

**JWT生成Token后的样子**

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJvcmciOiLku4rml6XlpLTmnaEiLCJuYW1lIjoiRnJlZeeggeWGnCIsImV4cCI6MTUxNDM1NjEwMywiaWF0IjoxNTE0MzU2MDQzLCJhZ2UiOiIyOCJ9.49UF72vSkj-sA4aHHiYN5eoZ9Nb4w5Vb45PsLF7x\_NY

**JWT的构成**

**secret： ‘yao’**

**userId ： 123456789**

**加密：按照一定编码格式去加密。**

**加盐加密：**

第一部分我们称它为头部（header),第二部分我们称其为载荷（payload)，第三部分是签证（signature)。

header

jwt的头部承载两部分信息：

·        声明类型，这里是jwt

·        声明加密的算法 通常直接使用 HMAC SHA256

完整的头部就像下面这样的JSON：

{

"typ": "JWT",

"alg": "HS256"

}

然后将头部进行base64加密（该加密是可以对称解密的),构成了第一部分：

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

**playload**

载荷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

·        公共的声明

·        私有的声明

**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。

**公共的声明 ：**

公共的声明可以添加任何的信息，一般添加用户的相关信息或其他业务需要的必要信息.但不建议添加敏感信息，因为该部分在客户端可解密.

**私有的声明 ：**

私有声明是提供者和消费者所共同定义的声明，一般不建议存放敏感信息，因为base64是对称解密的，意味着该部分信息可以归类为明文信息。

定义一个payload：

{

"name":"Free码农",

"age":"28",

"org":"今日头条"

}

然后将其进行base64加密，得到Jwt的第二部分：

eyJvcmciOiLku4rml6XlpLTmnaEiLCJuYW1lIjoiRnJlZeeggeWGnCIsImV4cCI6MTUxNDM1NjEwMywiaWF0IjoxNTE0MzU2MDQzLCJhZ2UiOiIyOCJ9

**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：

·        header (base64后的)

·        payload (base64后的)

·        secret

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：

49UF72vSkj-sA4aHHiYN5eoZ9Nb4w5Vb45PsLF7x\_NY

密钥secret是保存在服务端的，服务端会根据这个密钥进行生成token和验证，所以需要保护好。