用户身份认证

1.单一服务器模式

一般过程如下：

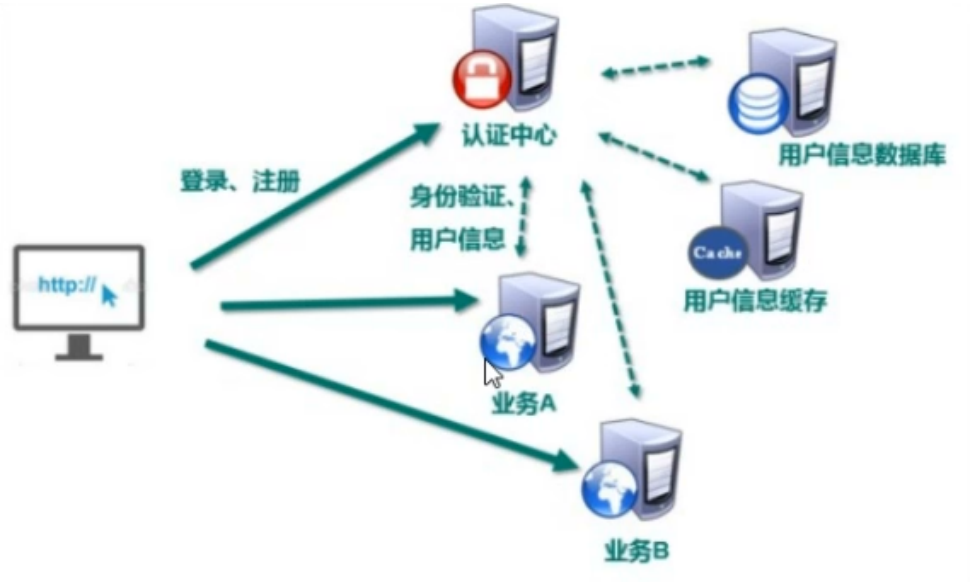
1. 用户向服务器发送用户名和密码。
2. 验证服务器后，相关数据（如用户名，用户角色等）将保存在当前会话（session）中。
3. 服务器向用户返回session_id，session信息都会写入到用户的Cookie。
4. 用户的每个后续请求都将通过在Cookie中取出session_id传给服务器。
5. 服务器收到session_id并对比之前保存的数据，确认用户的身份。

缺点：

- 单点性能压力，无法扩展。
- 分布式架构中，需要session共享方案，session共享方案存在性能瓶颈。

2. SSO（Single Sign On）单点登录模式

CAS单点登录、OAuth2



分布式，SSO(single sign on)模式：单点登录英文全称Single Sign On，简称就是SSO。它的解释是：在多个应用系统中，只需要登录一次，就可以访问其他相互信任的应用系统。

- 如图所示，图中有3个系统，分别是业务A、业务B、和SSO。
- 业务A、业务B没有登录模块。
- 而SSO只有登录模块，没有其他的业务模块。

一般过程如下：

1. 当业务A、业务B需要登录时，将跳到SSO系统。
2. SSO从用户信息数据库中获取用户信息并校验用户信息，SSO系统完成登录。
3. 然后将用户信息存入缓存（例如redis）。
4. 当用户访问业务A或业务B，需要判断用户是否登录时，将跳转到SSO系统中进行用户身份验证，SSO判断缓存中是否存在用户身份信息。
5. 这样，只要其中一个系统完成登录，其他的应用系统也就随之登录了。这就是单点登录（SSO）的定义。

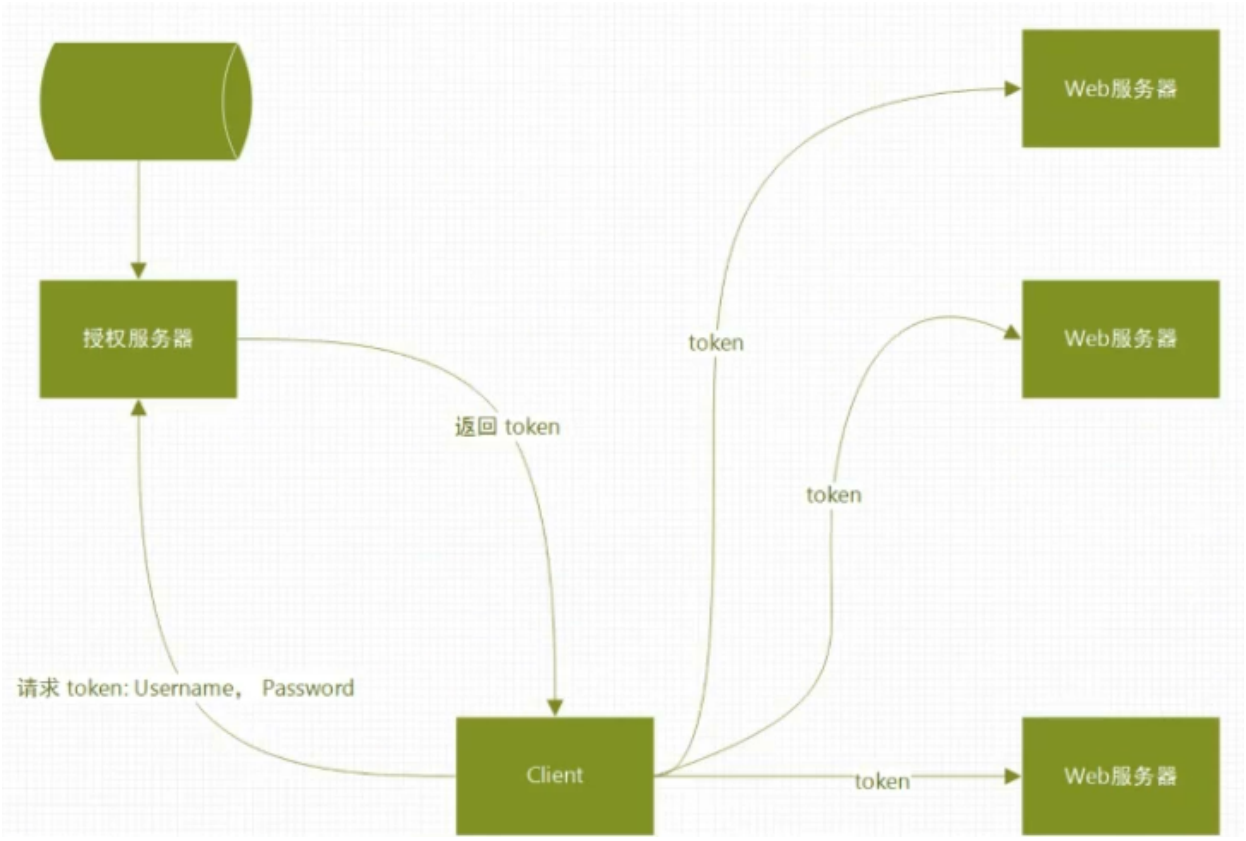
优点：

用户身份信息独立管理，更好的分布式管理。可以自己扩展安全策略

缺点：

认证服务器访问压力较大。

3. Token单点登录模式



token就是访问令牌

优点：

- 无状态： token是无状态，session是有状态的
- 基于标准化：你的API可以采用标准化的 JSON Web Token (JWT)

缺点：

- 占用带宽
- 无法在服务器端销毁

一、访问令牌的类型

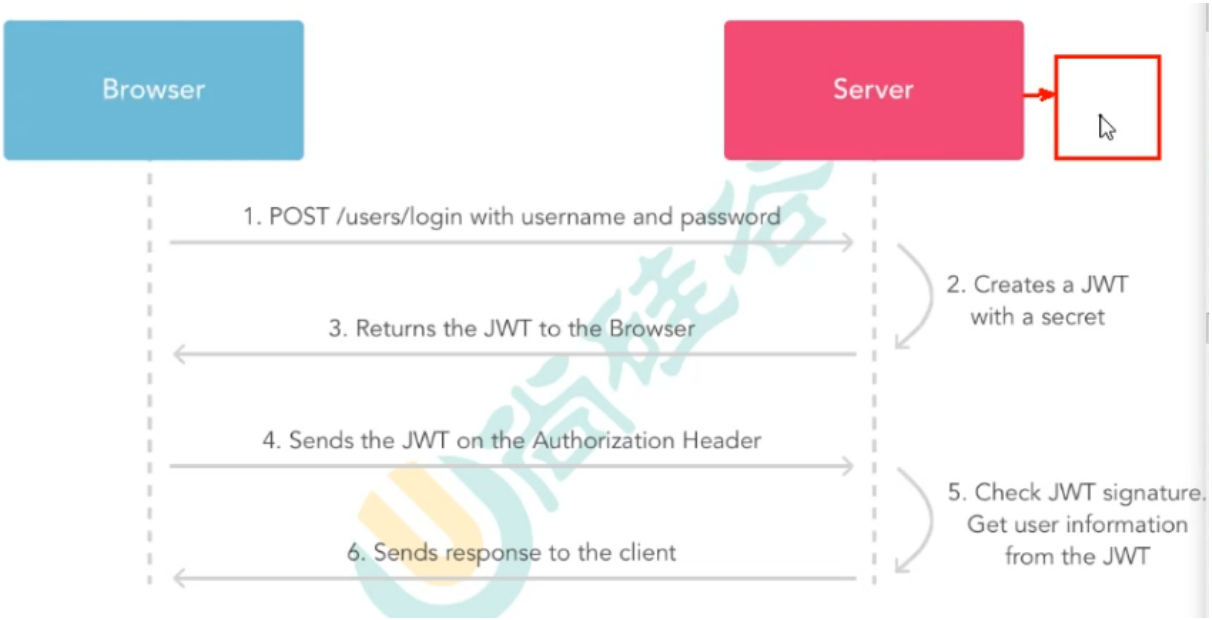
二、JWT令牌

1、什么是JWT令牌

JWT是JSON Web Token的缩写，即JSON Web令牌，是一种自包含令牌。

JWT的使用场景：

- 一种情况是webapi，类似之前的阿里云播放凭证的功能
- 另一种情况是多web服务器下实现无状态分布式身份验证
    - JWT官网有一张图描述了JWT的认证过程



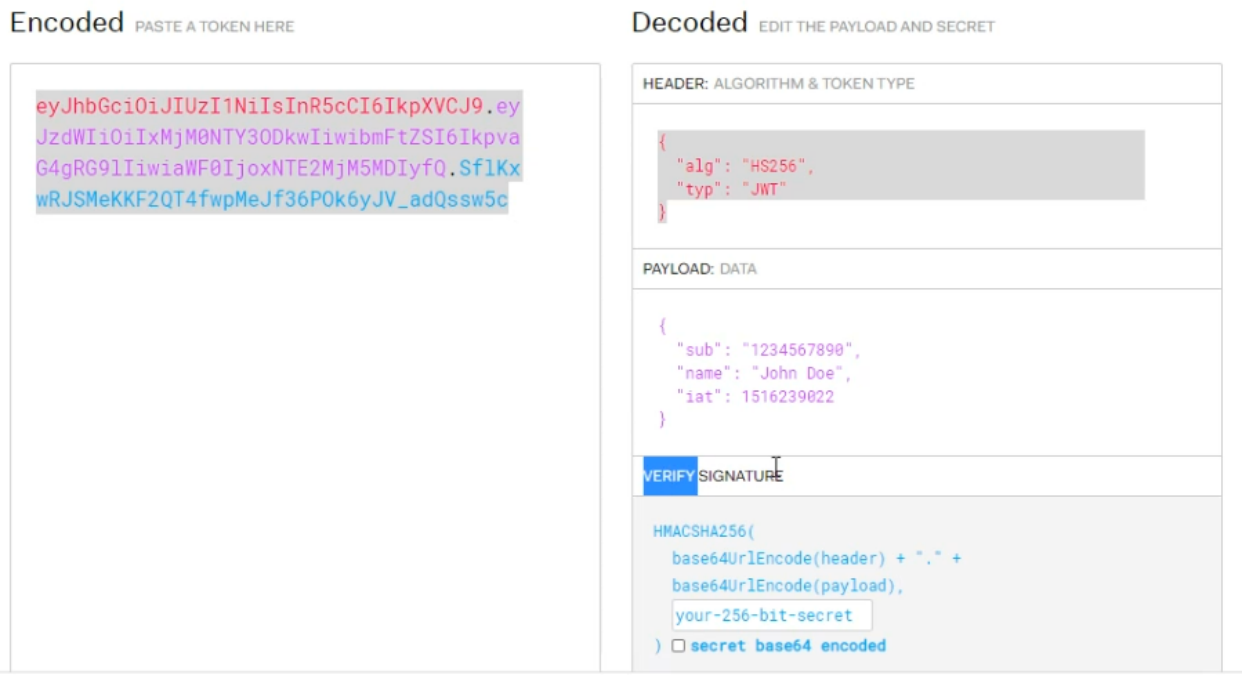
JWT的作用：

- JWT 最重要的作用就是对 token信息的防伪作用

JWT的原理：

- 一个JWT由三个部分组成：JWT头、有效载荷、签名哈希
- 最后由这三者组合进行base64编码得到JWT

JWT的组成



该对象为一个很长的字符串，字符之间通过"."分隔符分为三个子串。

每一个子串表示了一个功能块，总共有以下三个部分：JWT头、有效载荷和签名

JWT头

JWT头部分是一个描述JWT元数据的JSON对象，通常如下所示。

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

在上面的代码中，alg属性表示签名使用的算法，默认为HMAC SHA256（写为HS256）；typ属性表示令牌的类型，JWT令牌统一写为JWT。最后，使用Base64 URL算法将上述JSON对象转换为字符串保存。

只会对有效载荷进行编码，不会进行加密，所以不要存放敏感信息

有效载荷

有效载荷部分，是JWT的主体内容部分，也是一个JSON对象，包含需要传递的数据。JWT指定七个默认字段供选择。

```
sub: 主题
iss: jwt签发者
aud: 接收jwt的一方
iat: jwt的签发时间
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.
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。
```

除以上默认字段外，我们还可以自定义私有字段，如下例：

```
{
  "name": "Helen",
  "admin": true,
  "avatar": "helen.jpg"
}
```

请注意，默认情况下JWT是未加密的，任何人都可以解读其内容，因此不要构建隐私信息字段，存放保密信息，以防止信息泄露。

JSON对象也使用Base64 URL算法转换为字符串保存。

签名哈希

签名哈希部分是对上面两部分数据签名，通过指定的算法生成哈希，以确保数据不会被篡改。

首先，需要指定一个密码（secret）。该密码仅仅为保存在服务器中，并且不能向用户公开。然后，使用标头中指定的签名算法（默认情况下为HMAC SHA256）根据以下公式生成签名。

```
HMACSHA256(base64UrlEncode(header) + "." + base64UrlEncode(claims), secret)    ==>    签名hash
```

在计算出签名哈希后，JWT头，有效载荷和签名哈希的三个部分组合成一个字符串，每个部分用"."分隔，就构成整个JWT对象。

Base64URL算法

如前所述，JWT头和有效载荷序列化的算法都用到了Base64URL。该算法和常见Base64算法类似，稍有差别。

作为令牌的JWT可以放在URL中（例如api.example/?token=xxx）。 Base64中用的三个字符是"+"，"/"和"="，由于在URL中有特殊含义，因此Base64URL中对他们做了替换："="去掉，"+"用"-"替换，"/"用"_"替换，这就是Base64URL算法。

注意：base64编码，并不是加密，只是把明文信息变成了不可见的字符串。但是其实只要用一些工具就可以把base64编码解成明文，所以不要在JWT中放入涉及私密的信息。

3、JWT的用法

客户端接收服务器返回的JWT，将其存储在Cookie或localStorage中。

此后，客户端将在与服务器交互中都会带JWT。如果将它存储在Cookie中，就可以自动发送，但是不会跨域，因此一般是将它放入HTTP请求的Header Authorization字段中。

当跨域时，也可以将JWT放置于POST请求的数据主体中。

三、JWT问题和趋势

1、JWT默认不加密，但可以加密。生成原始令牌后，可以使用该令牌再次对其进行加密。

2、当JWT未加密时，一些私密数据无法通过JWT传输。

3、JWT不仅可用于认证，还可用于信息交换。善用JWT有助于减少服务器请求数据库的次数。

4、JWT的最大缺点是服务器不保存会话状态，所以在使用期间不可能取消令牌或更改令牌的权限。也就是说，一旦JWT签发，在有效期内将会一直有效。

5、JWT本身包含认证信息，因此一旦信息泄露，任何人都可以获得令牌的所有权限。为了减少盗用，JWT的有效期不宜设置太长。对于某些重要操作，用户在使用时应该每次都进行身份验证。

6、为了减少盗用和窃取，JWT不建议使用HTTP协议来传输代码，而是使用加密的HTTPS协议进行传输。

jwt测试：

引入maven依赖

```
<dependencies>
    <dependency>
        <groupId>io.jsonwebtoken</groupId>
        <artifactId>jjwt</artifactId>
        <version>0.7.0</version>
    </dependency>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```