link: null
title: 珠峰架构师成长计划
description: 在header中通常包含了两部分：token类型和采用的加密算法。
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=57 sentences=171, words=1070

1.JWT

- 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。
- 因为数字签名的存在，这些信息是可信的，JWT可以使用HMAC算法或者是RSA的公私秘钥对进行签名

2.主要应用场景

- 身份认证在这种场景下，一旦用户完成了登陆，在接下来的每个请求中包含JWT，可以用来验证用户身份以及对路由，服务和资源的访问权限进行验证。
- 信息交换在通信的双方之间使用JWT对数据进行编码是一种非常安全的方式，由于它的信息是经过签名的，可以确保发送者发送的信息是没有经过伪造的**3.JWT的结构#**JWT包含了使用 .分隔的三部分
- Header 头部
- Payload 负载
- Signature 签名

在header中通常包含了两部分：token类型和采用的加密算法。

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

接下来对这部分内容使用 Base64Url编码组成了 JWT结构的第一部分。

3.2 Payload

负载就是存放有效信息的地方。这个名字像是指货车上承载的货物，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

3.2.1 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,这是一个秒数
- 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

3.2.2 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

3.2.3 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

3.2.4 负载使用的例子

```
{ "sub": "1234567890", "name": "zfpx", "admin": true}
```

上述的负载需要经过 Base64Url编码后作为JWT结构的第二部分

3.3 Signature

- 创建签名需要使用编码后的header和payload以及一个秘钥
- 使用header中指定签名算法进行签名
- 例如如果希望使用HMAC SHA256算法，那么签名应该使用下列方式创建

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

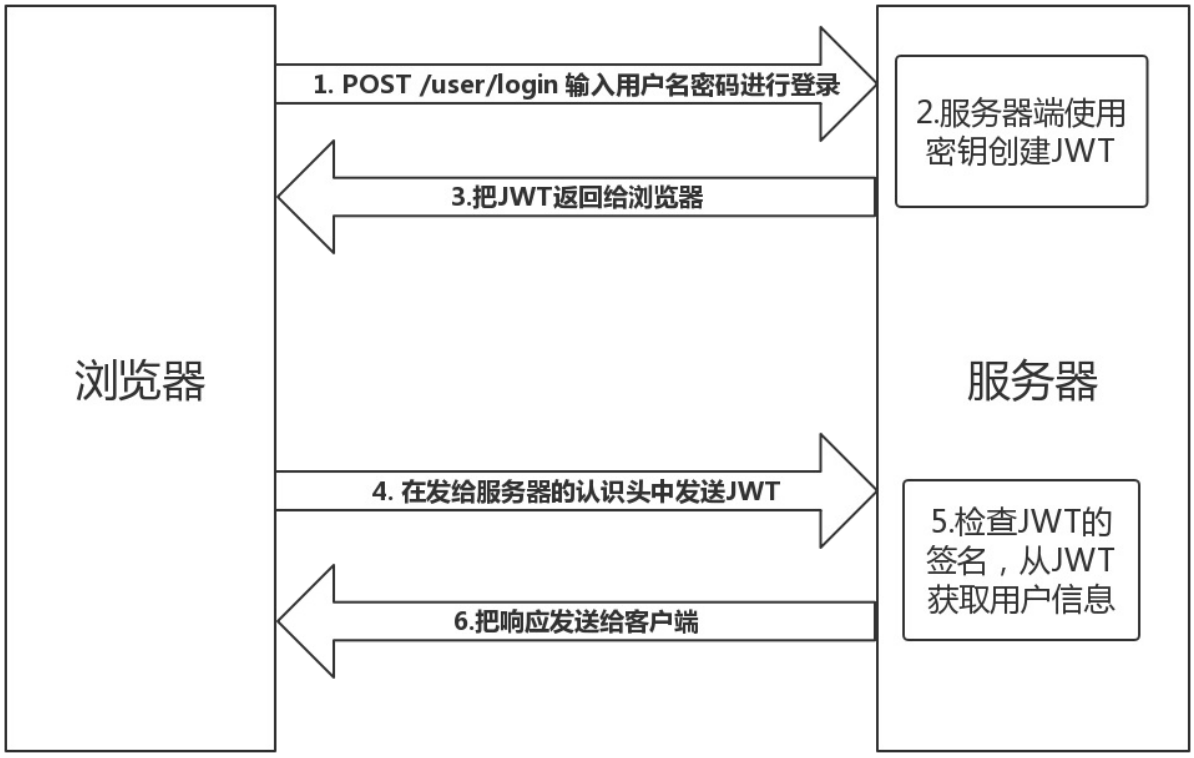
- 签名用于验证消息的发送者以及消息是没有经过篡改的
- 完整的JWT 完整的JWT格式的输出是以. 分隔的三段Base64编码
- 密钥secret是保存在服务端的，服务端会根据这个密钥进行生成token和验证，所以需要保护好。

** 4.如何使用JWT #**

1. 当用户使用它的认证信息登陆系统之后，会返回给用户一个JWT
2. 用户只需要本地保存该token（通常使用local storage，也可以使用cookie）即可
3. 当用户希望访问一个受保护的路由或者资源的时候，通常应该在Authorization头部使用Bearer模式添加JWT，其内容看起来是下面这样

```
Authorization: Bearer
```

4. 因为用户的状态在服务端的内存中是不存储的，所以这是一种无状态的认证机制
5. 服务端的保护路由将会检查请求头Authorization中的JWT信息，如果合法，则允许用户的行为。
6. 由于JWT是自包含的，因此减少了需要查询数据库的需要
7. JWT的这些特性使得我们可以完全依赖其无状态的特性提供数据API服务，甚至是创建一个下载流服务。
8. 因为JWT并不使用Cookie的，所以你可以使用任何域名提供你的API服务而不需要担心跨域资源共享问题（CORS）



** 5. JWT实战 #**

5.1 config.js

```
module.exports = {
    dbUrl: 'mongodb://127.0.0.1/jwt',
    secret: 'zfpx'
}
```

5.2 app.js

momentjs (http://momentjs.cn/)

```
const express = require('express');
const jwt = require('jwt-simple');
const bodyParser = require('body-parser');
const moment = require('moment');
const User = require('./model/user');
const jwtWare = require('./jwt');
const { secret } = require('./config');
const app = express();
app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());
app.post('/signup', async function (req, res) {
    let user = req.body;
    user = await User.create(user);
    if (user) {
        res.json({
            code: 0,
            data: {
                user
            }
        });
    } else {
        res.json({
            code: 1,
            data: '用户注册失败'
        });
    }
});
app.post('/login', async function (req, res) {
    let user = req.body;
    user = await User.findOne(user);
    if (user) {
        let expires = moment().add(7, 'days').valueOf();
        let userInfo = {
            id: user._id,
            username: user.username
        };
        let token = jwt.encode({
            user: userInfo,
            exp: expires
        }, secret);
        res.json({
            code: 0,
            data: {
                token,
                expires,
                user: userInfo
            }
        });
    } else {
        res.json({
            code: 1,
            data: '用户名或密码错误'
        });
    }
});
app.get('/user', jwtWare, function (req, res) {
    res.json({
        code: 0,
        data: {
            user: req.user
        }
    });
});
app.listen(8080);
```

5.3 jwt.js

```
const { secret } = require('./config');
const jwt = require('jwt-simple');
const User = require('./model/user');
module.exports = async function (req, res, next) {
    let authorization = req.headers['authorization'];
    if (authorization) {
        try {
            let decoded = jwt.decode(authorization.split(' ')[1], secret);
            req.user = decoded.user;
            next();
        } catch (err) {
            console.log(err);
            res.status(401).send('Not Allowed');
        }
    } else {
        res.status(401).send('Not Allowed');
    }
}
```

5.4 user.js

```
let mongoose = require('mongoose');
let Schema = mongoose.Schema;
let ObjectId = Schema.Types.ObjectId;
let { dbUrl } = require('../config');
let conn = mongoose.createConnection(dbUrl);
let UserSchema = new Schema({
    username: String,
    password: String
});
module.exports = conn.model("User", UserSchema);
```

** 6. 前端 #**

```
create-react-app front
cd front
cnpm i react react-dom react-router-dom axios -S
```

6.1 index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import { Router, Route, Switch, Redirect } from 'react-router-dom';
import createHashHistory from 'history/createHashHistory';
import Login from './components/Login';
import User from './components/User';
const history = createHashHistory();
ReactDOM.render(
    <Router history={history}>
        <Switch>
            <Route exact path="/" component={Login} />
            <Route path="/user" component={User} />
            <Redirect to="/" />
        Switch>
    Router>, document.querySelector('#root')
);
```

6.2 api.js

```
import axios from 'axios';
import createHashHistory from 'history/createHashHistory';
const history = createHashHistory();
axios.interceptors.request.use(config => {
    if (localStorage.token) {
        config.headers.Authorization = 'Bearer ' + localStorage.token
    }
    return config
}, error => {
    return Promise.reject(error)
})

axios.interceptors.response.use(res => {
    if (res.data.code != 0) {
        return Promise.reject(res);
    }
    return res;
}, error => {
    if (error.response.status == 401) {
        history.push('/');
    }
    return Promise.reject(error.response.data);
});

export function login(data) {
    return axios({
        url: 'http://localhost:8080/login',
        method: 'post',
        data
    }).then(response => {
        let data = response.data;
        localStorage.setItem('token', data.data.token);
        return data;
    })
}
export function getUser(data) {
    return axios({
        url: 'http://localhost:8080/user',
        method: 'get'
    }).then(response => {
        return response.data;
    })
}
```

6.3 Login.js

```
import React, { Component } from 'react';
import { login } from '../api';
export default class Login extends Component {
    handleSubmit = (event) => {
        event.preventDefault();
        let username = this.username.value;
        let password = this.password.value;
        login({ username, password }).then(data => {
            if (data.code == 0) {
                this.props.history.push('/user');
            }
        });
    }
    render() {
        return (
            <form onSubmit={this.handleSubmit}>
                用户名<input required ref={ref => this.username = ref} />
                密码<input required ref={ref => this.password = ref} />
                <input type="submit" />
            form>
        )
    }
}
```

6.4 User.js

```
import React, { Component } from 'react';
import { getUser } from '../api';
export default class User extends Component {
    state = {
        user: {}
    }
    componentDidMount() {
        getUser().then(res => {
            if (res && res.code == 0) {
                this.setState({ user: res.data.user });
            }
        });
    }
    render() {
        return (
            <div>
                欢迎 {this.state.user.username}
            div>
        )
    }
}
```

** 7.原理实现 #**

jwt.js

```
const crypto = require('crypto');
function encode(payload, key) {
    let header = { type: 'JWT', alg: 'sha256' };
    var segments = [];
    segments.push(base64urlEncode(JSON.stringify(header)));
    segments.push(base64urlEncode(JSON.stringify(payload)));
    segments.push(sign(segments.join('.'), key));
    return segments.join('.');
}
function sign(input, key) {
    return crypto.createHmac('sha256', key).update(input).digest('base64');
}

function decode(token, key) {
    var segments = token.split('.');
    var headerSeg = segments[0];
    var payloadSeg = segments[1];
    var signatureSeg = segments[2];

    var header = JSON.parse(base64urlDecode(headerSeg));
    var payload = JSON.parse(base64urlDecode(payloadSeg));

    if (signatureSeg != sign([headerSeg, payloadSeg].join('.'), key)) {
        throw new Error('verify failed');
    }

    if (payload.exp && Date.now() > payload.exp * 1000) {
        throw new Error('Token expired');
    }
    return payload;
}

function base64urlEncode(str) {
    return new Buffer(str).toString('base64');
}

function base64urlDecode(str) {
    return new Buffer(str, 'base64').toString();
}

module.exports = {
    encode,
    decode
}
```