link: null
title: 珠峰架构师成长计划
description: 客户端第一次访问服务器的时候服务器通过响应头向客户端发送Cookie,属性之间用分号空格分隔
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=46 sentences=103, words=588

1. cookie是什么

- HTTP1.0中协议是无状态的，但在WEB应用中，在多个请求之间共享会话是非常必要的，所以出现了Cookie
- cookie是为了辩别用户身份，进行会话跟踪而 *存储在客户端*上的数据

2. Cookie的处理流程

3. 使用步骤

客户端第一次访问服务器的时候服务器通过响应头向客户端发送Cookie,属性之间用分号空格分隔

```
Set-Cookie:name=zfpx; Path=/
```

客户端接收到Cookie之后保存在本地

以后客户端再请求服务器的时候会把此Cookie发送到服务器端

```
Cookie:name=zfpx
```

4. cookie重要属性

属性 说明 name=value 键值对，可以设置要保存的 Key/Value Domain 域名，默认是当前域名 maxAge 最大失效时间(毫秒),设置在多少后失效 secure 当 secure 值为 true 时，cookie 在 HTTP 中是无效，在 HTTPS 中才有效 Path 表示 cookie 影响到的路，如 path=/。如果路径不能匹配时，浏览器则不发送这个Cookie Expires 过期时间(秒)，在设置的某个时间点后该 Cookie 就会失效，如 expires=Money, 05-Dec-11 11:11:11 GMT httpOnly 如果在COOKIE中设置了

属性，则通过程序(JS脚本)将无法读取到COOKIE信息，防止XSS攻击产生

5. 在express中向客户端发送cookie

```
res.cookie(name,value,[,options]);
```

参数 chrome对应属性 类型 说明 示例 domain Domain String 域名，默认是当前域名 {domain:'a.zfpx.cn'} path Path String 路径，默认是/ {path:'/visit'} expires Expires Date 过期时间，如果没以有指定或为0表示当前会话有效 {expires:new Date(Date.now()+20*1000)} *maxAge Max-Age Number 有效时间(单位是毫秒) {maxAge:20*1000}* httpOnly HTTP Boolean 不能通过浏览器javascript访问 {httpOnly:true} secure Secure String 只通过https协议访问

使用cookie-parser中间件

```
$ npm install cookie-parser --save
```

```
app.use(require('cookie-parser')());
response.cookie(key,value)
request.cookies
response.clearCookie('username')
```

```
var express = require('express');
var cookieParser = require('cookie-parser');
var app = express();

app.use(cookieParser('zfpx'));
app.get('/write',function(req,res){

    res.cookie('age','123',{signed:true});
    res.end('ok');
});

app.get('/read',function(req,res){
    console.log(req.signedCookies);
    res.send(req.cookies);
});

app.get('/visit',function(req,res){
    res.cookie('count',isNaN(req.cookies.count)?0:parseInt(req.cookies.count)+1);
    res.send(req.cookies);
});

app.listen(9090);
```

```
function cookieParser(secret){
  return function cookieParser(req, res, next){
    req.secret = secret;
    if (!req.headers.cookie) {
        return next();
    }
    req.cookies =  require('querystring').parse(req.headers.cookie,'; ','=');
    if(req.secret){
        req.signedCookies = {};
        for(let attr in req.cookies){
                let val = req.cookies[attr];
                req.signedCookies[attr] = unsign(val, secret);
            }
        }
    }
    next();
  }
}

function cookie(name, val, options) {
    var opt = options || {};
    val = encodeURIComponent(val);
    if(opt.secret){
        var secret = this.req.secret;
        val = sign(val,this.secret);
    }

    var pairs = [name + '=' + value];

    if (null != opt.maxAge) {
        var maxAge = opt.maxAge - 0;
        if (isNaN(maxAge)) throw new Error('maxAge should be a Number');
        pairs.push('Max-Age=' + Math.floor(maxAge));
    }

    if (opt.domain) {
        pairs.push('Domain=' + opt.domain);
    }

    if (opt.path) {
        pairs.push('Path=' + opt.path);
    }

    if (opt.expires) pairs.push('Expires=' + opt.expires.toUTCString());
    if (opt.httpOnly) pairs.push('HttpOnly=true');
    if (opt.secure) pairs.push('Secure=true');

    return pairs.join('; ');
}

var crypto = require('crypto');
exports.sign = function(val, secret){
  return val + '.' + crypto
    .createHmac('sha256', secret)
    .update(val)
    .digest('base64')
    .replace(/\=+$/, '');
};

exports.unsign = function(val, secret){
  var str = val.slice(0, val.lastIndexOf('.'))
    , mac = exports.sign(str, secret);
  return mac == val ? str : false;
};
```

6. 权限控制

```
var express = require('express');
var cookieParser = require('cookie-parser');
var app = express();
app.set('view engine','html');
app.engine('html',require('ejs').__express);
app.set('views',__dirname);

app.use(cookieParser());

function checkUser(req,res,next){
    if(req.cookies && req.cookies.username)
      next();
    else
      res.redirect('/');
}

app.get('/',function(req,res){
    res.render('index');
});

app.get('/login',function(req,res){
    res.cookie('username',req.query.username,{httpOnly:true});
    res.redirect('/user');
});

app.get('/user',checkUser,function(req,res){
    res.render('user',{username:req.cookies.username});
});

app.get('/logout',function(req,res){
    res.clearCookie('username');
    res.redirect('/');
});

app.listen(8080);
```

7.cookie使用注意事项

- 可能被客户端篡改，使用前验证合法性
- 不要存储敏感数据，比如用户密码，账户余额
- 使用httpOnly保证安全
- 尽量减少cookie的体积
- 设置正确的domain和path，减少数据传输