# 1.状态保持简介

|  |
| --- |
|  |

# 2.cookie

## 1）.cookie使用原理

|  |
| --- |
|  |

## 2）cookie的使用

|  |
| --- |
|  |
|  |

# 示例

## 1.把express\_pathinfo\_inherit项目复制一份，改名express\_pathinfo\_inherit-cookie，然后端口终端定位到这个路径下面

|  |
| --- |
|  |

## 2.安装cookie-parser：yarn add cookie-parser，安装完成后如图

|  |
| --- |
|  |

## 3.把03-art-template-filter-inherit.js复制一份改名：04-art-template-filter-inherit-cookies.js，我们这个项目就是使用这个文件

|  |
| --- |
|  |

## 4.打开04js文件，引入cookie-parser,然后需要全局注册这个中间件

|  |
| --- |
|  |

## 5.在04-art-template-filter-inherit-cookies.js中添加一个/set\_cookie路由用来学习cookie处理，我们在这个路由处理函数里面设置一个cookie

|  |
| --- |
|  |

## 6.然后我们可以添加一个路由来获取cookie，路由是/get\_cookie

|  |
| --- |
|  |

### 效果：

|  |
| --- |
|  |

### 注意：需要先范围/set\_cookie,然后再访问/get\_cookie,设置的时候需要一个一个啦设置，获取的时候可以一次获取所有cookies，设置cookie使用response对象的cookie方法，获取cookie使用request对象（全局注册了cookieParser中间件后）的cookies属性来获取。cookie是保存在浏览器中的。浏览器保存了cookie后每一次访问该服务器都会带着这个cookie去访问。

# 04-art-template-filter-inherit-cookies.js完整代码

|  |
| --- |
| let express = require('express')  const { template } = require('express-art-template')  let cookieParser = require('cookie-parser')  let path = require('path')  let port = 3500  let app = express()  //全局注册cookieParser中间件,注意：有的中间件需要小括号，有的中间件不需要  app.use(cookieParser())  app.use(express.static(\_\_dirname+'/public'))//静态服务器，根据需要配置  //定义art-template过滤器  //1.首字母大写过滤器  template.defaults.imports.Cap = (data)=>{      if(data){          return data.substring(0,1).toUpperCase()+data.substring(1)      }  }  //时间戳变为"yyyy-MM-dd hh:mm:ss" 格式的时间字符串过滤器  template.defaults.imports.dateFormat = (timestamp)=>{      let date = new Date(timestamp)      console.log(date);      let year = date.getFullYear()      let month = (date.getMonth()+1).toString().padStart(2,'0')      let day = date.getDay().toString().padStart(2,'0')      let h = date.getHours().toString().padStart(2,'0')      let m = date.getMinutes().toString().padStart(2,'0')      let s = date.getSeconds().toString().padStart(2,'0')      return `${year}-${month}-${day} ${h}:${m}:${s}`  }  //处理中文乱码  app.use(function (req, res, next) {      res.setHeader('Content-Type', 'text/html;charset=utf-8');      next();  });  //配置模板引擎,4步，一个engine和3个set  app.engine('html',require('express-art-template'))  app.set('view options',{      debug:process.env.NODE\_ENV!=='production'  })  app.set('views',path.join(\_\_dirname,'views'))  app.set('view engine','html')  app.get('/',(req,res)=>{       res.render('index',{title:"模板继承例子",link:'/list'})  })  app.get('/list',(req,res)=>{       res.render('list') //渲染列表页面  })  //测试过滤器的路由  app.get('/testfilter',(req,res)=>{      res.render('test',{msg: 'hello',tstamp:1699978584726})  })  //响应动态路由  app.get('/detail/:id/:type',(req,res)=>{      let {id,type} = req.params      res.end(`<h1>第${id}条新闻标题</h1> <hr> <h3>主题是：${type}</h3>`)  })  //处理设置cookie的路由  app.get('/set\_cookie',(req,res)=>{      res.cookie('name','node',{maxAge:1000\*60\*60\*2})      res.cookie('age',11)      res.send("设置了cookie")  })  app.get('/get\_cookie',(req,res)=>{      // console.log(req.cookies); //注意：这里是cookies      // res.send(`获取到cookie数据${JSON.stringify(req.cookies)}`)      res.send(`获取到cookie数据,name:${req.cookies['name']},age:${req.cookies['age']}`)  })  app.listen(port,()=>{       console.log('server is ready :http://localhost:'+port+'/');  }) |

# 3.session

## 1.session的运行原理

|  |
| --- |
|  |

## 可见：session是依赖于cookie的， 不同浏览器的session互不干扰。

## 2.session的特点

|  |
| --- |
|  |

## 3.session的使用，使用session之前需要安装cookie-session插件

|  |
| --- |
|  |
|  |

# 示例，设置和获取session信息，先安装cookie-session，使用命令：yarn add cookie-session，安装完后如图，可以在package.json里面找得到

|  |
| --- |
|  |

# 1.把express\_pathinfo\_inherit-cookie项目复制一份，改名：express\_pathinfo\_inherit-session，然后04-art-template-filter-inherit-cookies.js复制一份改名：05-art-template-filter-inherit-session.js

|  |
| --- |
|  |

# 2.引入cookie-session，并且全局注册中间件

|  |
| --- |
|  |

# 3.添加设置session的测试路由，这里是/set\_session

|  |
| --- |
|  |

# 4.添加获取session数据的测试路由，这里是/get\_session

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |
|  |

## 用调试工具查看，在cookie里面有一个my\_session,它就是session在cookie中保持的标识

|  |
| --- |
|  |

# 05-art-template-filter-inherit-session.js的完整代码：

|  |
| --- |
| let express = require('express')  const { template } = require('express-art-template')  let cookieParser = require('cookie-parser')  let cookieSession = require('cookie-session')  let path = require('path')  let port = 3500  let app = express()  //全局注册cookieParser中间件,注意：有的中间件需要小括号，有的中间件不需要  app.use(cookieParser())  //全局注册cookieSession中间件  app.use(cookieSession({      name:'my\_session',      keys:['adefg%%78$$','o9l1##dd09=='] ,//这个其实就是加密用的盐      maxAge:2\*24\*1000\*60\*60 //过期时间为2天  }))  app.use(express.static(\_\_dirname+'/public'))//静态服务器，根据需要配置  //定义art-template过滤器  //1.首字母大写过滤器  template.defaults.imports.Cap = (data)=>{      if(data){          return data.substring(0,1).toUpperCase()+data.substring(1)      }  }  //时间戳变为"yyyy-MM-dd hh:mm:ss" 格式的时间字符串过滤器  template.defaults.imports.dateFormat = (timestamp)=>{      let date = new Date(timestamp)      console.log(date);      let year = date.getFullYear()      let month = (date.getMonth()+1).toString().padStart(2,'0')      let day = date.getDay().toString().padStart(2,'0')      let h = date.getHours().toString().padStart(2,'0')      let m = date.getMinutes().toString().padStart(2,'0')      let s = date.getSeconds().toString().padStart(2,'0')      return `${year}-${month}-${day} ${h}:${m}:${s}`  }  //处理中文乱码  app.use(function (req, res, next) {      res.setHeader('Content-Type', 'text/html;charset=utf-8');      next();  });  //配置模板引擎,4步，一个engine和3个set  app.engine('html',require('express-art-template'))  app.set('view options',{      debug:process.env.NODE\_ENV!=='production'  })  app.set('views',path.join(\_\_dirname,'views'))  app.set('view engine','html')  app.get('/',(req,res)=>{       res.render('index',{title:"模板继承例子",link:'/list'})  })  app.get('/list',(req,res)=>{       res.render('list') //渲染列表页面  })  //测试过滤器的路由  app.get('/testfilter',(req,res)=>{      res.render('test',{msg: 'hello',tstamp:1699978584726})  })  //响应动态路由  app.get('/detail/:id/:type',(req,res)=>{      let {id,type} = req.params      res.end(`<h1>第${id}条新闻标题</h1> <hr> <h3>主题是：${type}</h3>`)  })  //处理设置cookie的路由  app.get('/set\_cookie',(req,res)=>{      res.cookie('name','node',{maxAge:1000\*60\*60\*2})      res.cookie('age',11)      res.send("设置了cookie")  })  //获取cookie的路由  app.get('/get\_cookie',(req,res)=>{      // console.log(req.cookies); //注意：这里是cookies      // res.send(`获取到cookie数据${JSON.stringify(req.cookies)}`)      res.send(`获取到cookie数据,name:${req.cookies['name']},age:${req.cookies['age']}`)  })  //设置session的路由  app.get('/set\_session',(req,res)=>{      req.session['name'] = 'session\_node'      req.session['age'] = 20       res.send('成功设置了session数据')  })  //获取session的路由  app.get('/get\_session',(req,res)=>{      let name = req.session['name']      let  age = req.session['age']       res.send(`获取session数据，name:${name},age:${age}`)  })  app.listen(port,()=>{       console.log('server is ready :http://localhost:'+port+'/');  }) |