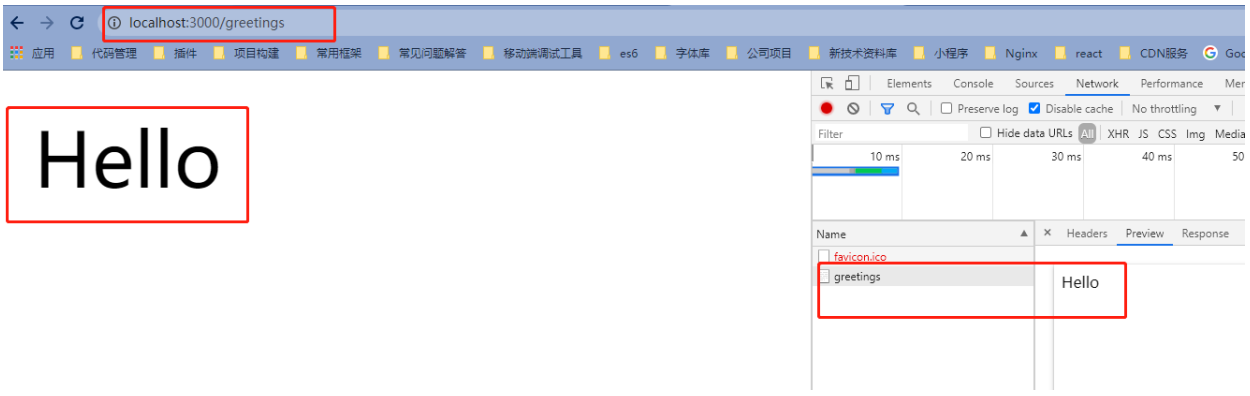
课程目标

Method和解析Body

查询GET /product

```
//使用node GET请求
const exprss = require('express')
const app = exprss()
app.get('/greetings',(req,res)=>{
 res.send('Hello')
})

app.listen(3000,(req,res)=>{
 console.log(23232)
 })
 //使用node 开启某个服务可以下载 nodemon
 //nodemen进行热加载(
 //不必每次修改都要去重新启动一次)
```

结果：



新增POST /product

```
//使用node GET请求 Post请求
const exprss = require('express')
const app = exprss()
```

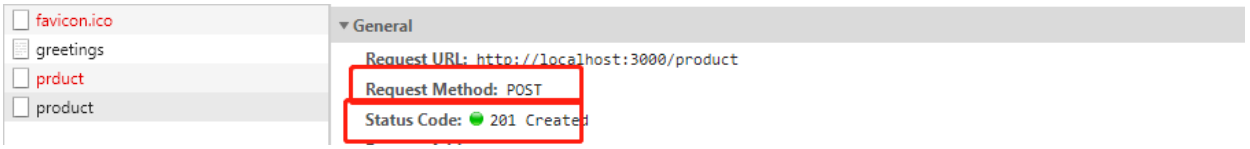
```
//get
app.get('/greetings',(req,res)=>{
 res.send('Hello')
})

//post
app.post('/product',(req,res)=>{
 //post在服务器创建一个资源
 res.set('Content-Type','application/json') //指定返回的资源格式为JSON
 res.status(201).send(JSON.stringify({data:'资源已经创建,但是未实现'}))
 //比较规范的做法，返回201 表示资源已经创建
})

app.listen(3000,(req,res)=>{
 console.log(23232)
})
```

POST product 返回201

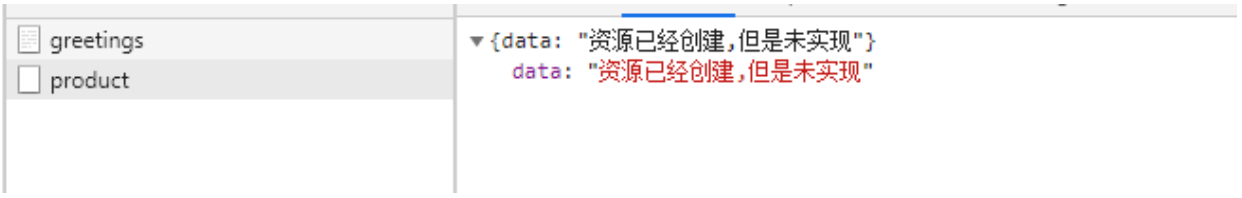
使用fetch直接调用(图一) (查看相关结果，图二，图三)

图一

图二



图三



指定返回资源的格式res.set('Content-Type','application/json')，就可以查看response的Headers

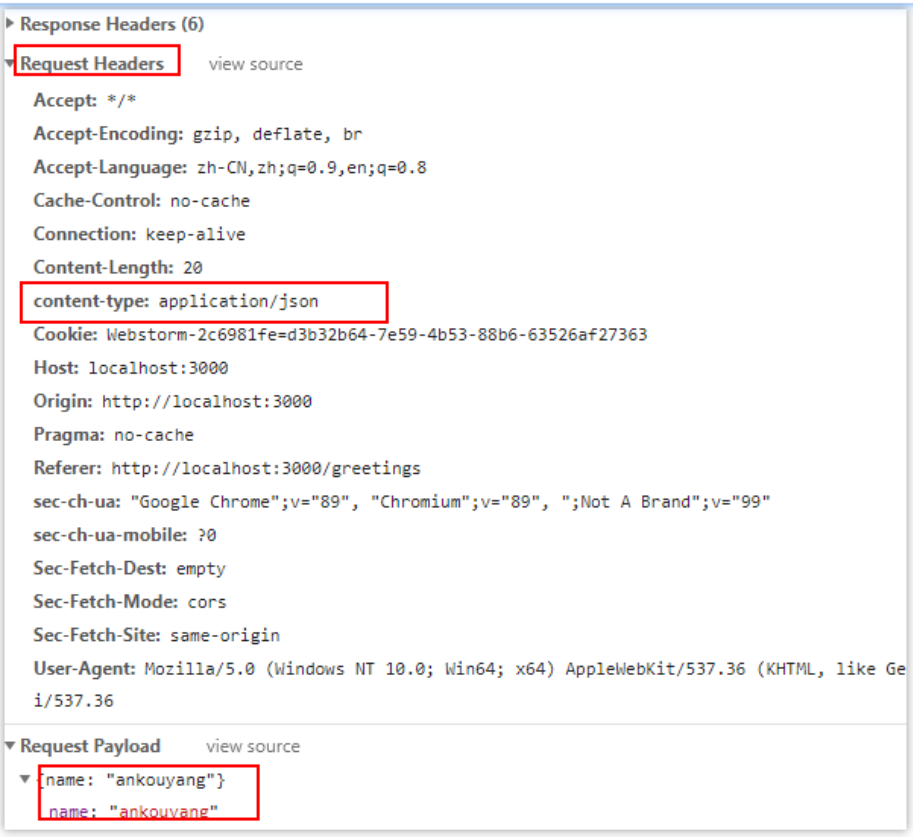
在客户端，可以获取到res的headers中Content-Type

```
> fetch('/product',{method:'POST'}).then(res=>console.log(res.headers.get('Content-type')))
< ▶ Promise {<pending>}
application/json; charset=utf-8
```

POST请求body传参，服务端如何解析body

客户端body传参

```
fetch('/product',{method:'POST',headers:{'content-type':'application/json'},body:JSON.stringify({name:'ankouyang'})})
▶ Promise {<pending>}
```



服务端解析Body, 这里用原生的写一下，其实是可以用到express的中间件去实现的

```
1 const exprss = require('express')
2 const app = exprss()
3 //get
4 app.get('/greetings',(req,res)=>{
```

```
 res.send('Hello')
})

//post
app.post('/product',(req,res)=>{
 //使用application/json
 const contentType = req.headers['content-type']
 let requestText = ""
 let body =null
 req.on('data',(buffer)=>{
 //8bit -byte 把流转化为utf-8数据格式
 requestText +=buffer.toString('utf-8')
 })
 //数据传输完毕后
 req.on('end',()=>{
 switch (contentType) {
 //通过客户端的请求头content-type
 case 'application/json':
 body = JSON.parse(requestText) //解析body
 res.set('Content-Type','application/json')
 res.status(201).send(JSON.stringify({data:body}))
 break
 }
 })

})

app.listen(3000,(req,res)=>{
 console.log(23232)
})
```

修改PUT /product/:id (restful风格)

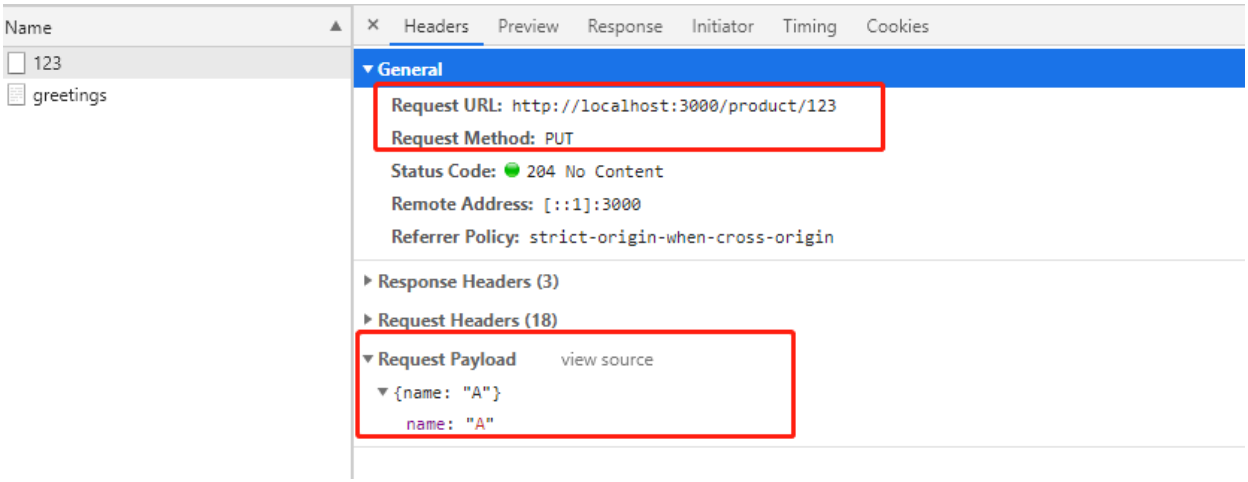
服务端：

```
//put
app.put('/product/:id',(req,res)=>{
    res.status(204).send() //204成功了 但是无内容返回给你
})
```

客户端请求：

```
> fetch('/product/123',{method:'PUT',headers:{'content-type':'application/json'},body:JSON.stringify({name:'A'})})
< ▶ Promise {<pending>}
>
```

服务端返回：



删除DELETE /product/:id (resful风格)

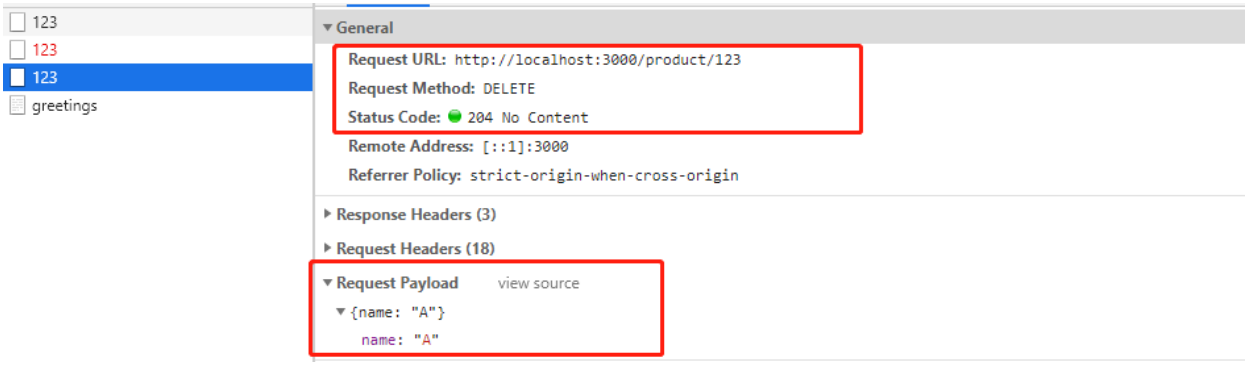
服务端：

```
//delete
app.put('/product/:id',(req,res)=>{
    res.status(204).send() //204成功了 但是无内容返回给你
})
```

客户端请求：

```
fetch('/product/123',{method:'DELETE',headers:{'content-type':'application/json'},body:JSON.stringify({name:'A'})})
▶ Promise {<pending>}
```

服务端返回：



跳转Header和3XX状态码

实战-重定向

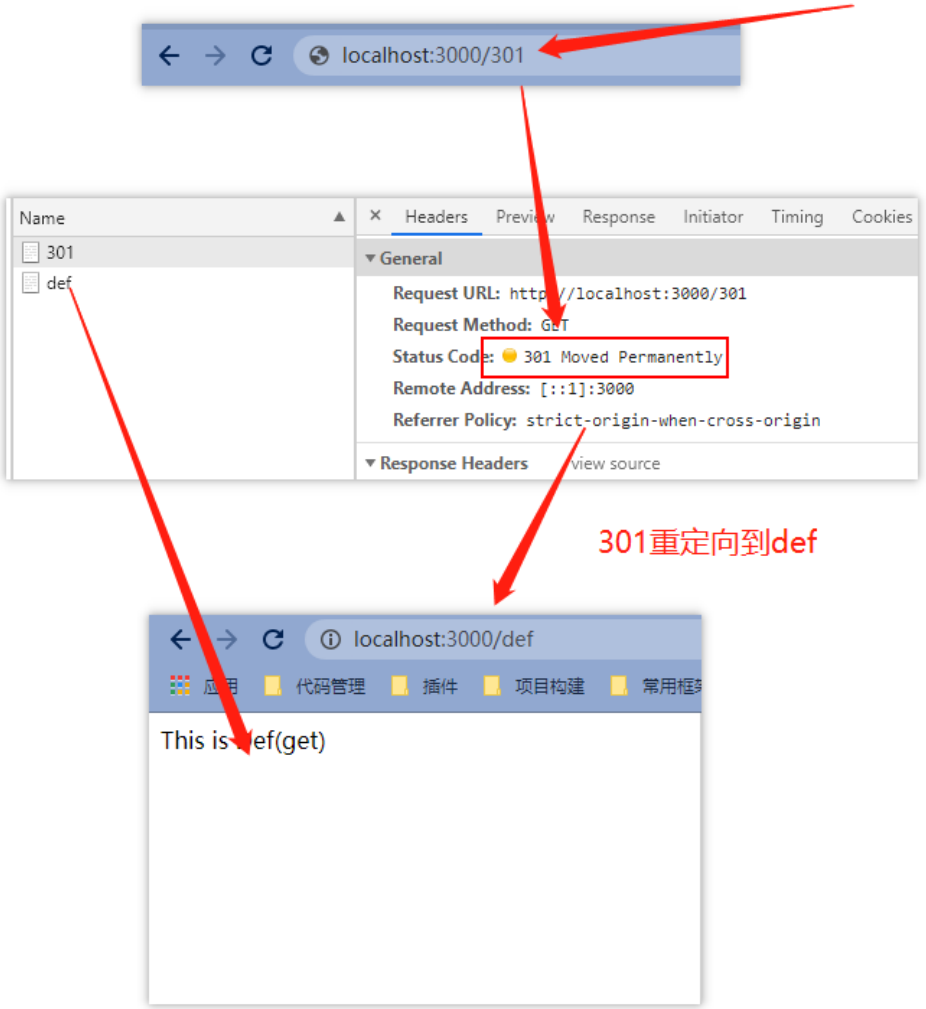
- 301
- 302
- 303
- 307
- 308

重定向301

服务端：

```
const exprss = require('express')
const app = exprss()
//get
app.get('/301',(req,res)=>{
 res.redirect(301,'/def')
})
//重定向接口
app.get('/def',(req,res)=>{
 res.send('This is Def(get)')
 })


 app.listen(3000,(req,res)=>{
  console.log(23232)
 })
```

客户端：



注意：这个重定向是浏览器干的，我们可以通过curl http://localhot:3000/301了可以看出，只是提示Redirecting to /def

```
C:\Users\yangke>curl http://localhost:3000/301
Moved Permanently. Redirecting to /def
C:\Users\yangke>
```

重定向302/303/307

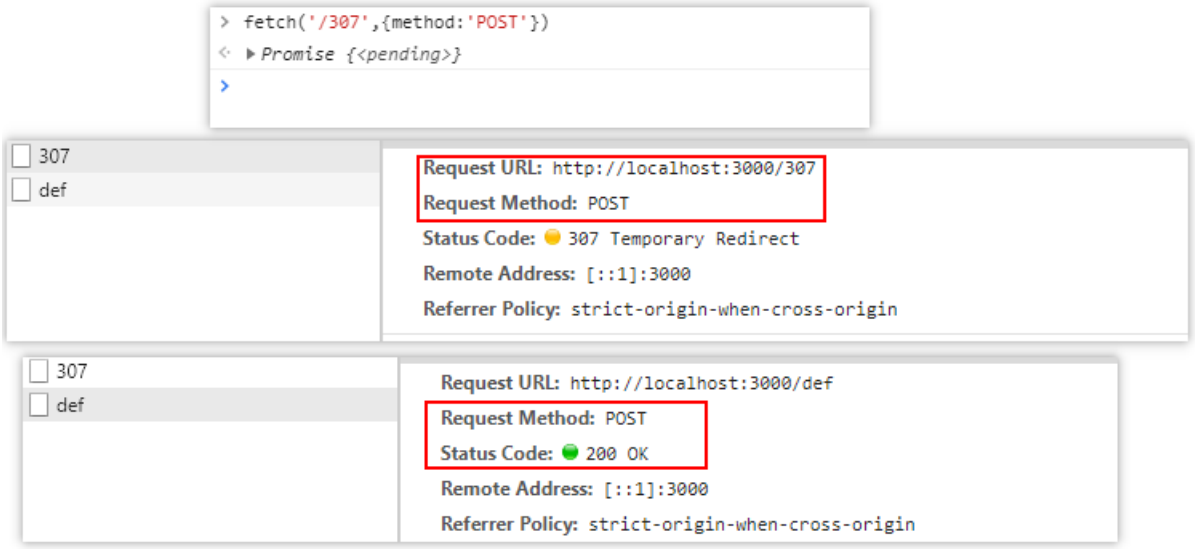
服务端:

```
1 const exprss = require('express')
2 const app = exprss()
3 //get post
4 //302/303/307
5 app.get('/301',(req,res)=>{
```

```
 res.redirect(301,'/def')
})
app.post('/301',(req,res)=>{
 res.redirect(301,'/def')
 })
 //302
 app.get('/302',(req,res)=>{
  res.redirect(302,'/def')
 })
 app.post('/302',(req,res)=>{
  res.redirect(302,'/def')
 })
 //303
 app.get('/303',(req,res)=>{
  res.redirect(303,'/def')
 })

 app.post('/303',(req,res)=>{
  res.redirect(303,'/def')
 })

 //get 直接报错 307
 app.get('/307',(req,res)=>{
  res.redirect(307,'/def')
 })
 //post 307
 app.post('/307',(req,res)=>{
  res.redirect(307,'/def')
 })

 //重定向接口
 app.get('/def',(req,res)=>{
  res.send('This is Def(get)')
 })

 app.post('/def',(req,res)=>{
  res.send('This is Def(post)')
 })

 app.listen(3000,(req,res)=>{
```

```
46   console.log(23232)
47 })
```

客户端：302 303方式,不管是GET还是POST请求方式，重定向都是GET返回。
307方式，只能支持post方式，重定向方式也是POST 如何所示:



307get方式 直接404

(遵守这个协议的话，可以有利于seo优化)

错误处理4XX和5XX

为下列场景返回不同的错误码

- 用户没有登录
- 服务器报错
- 内容没有找到
- 不支持POST方法

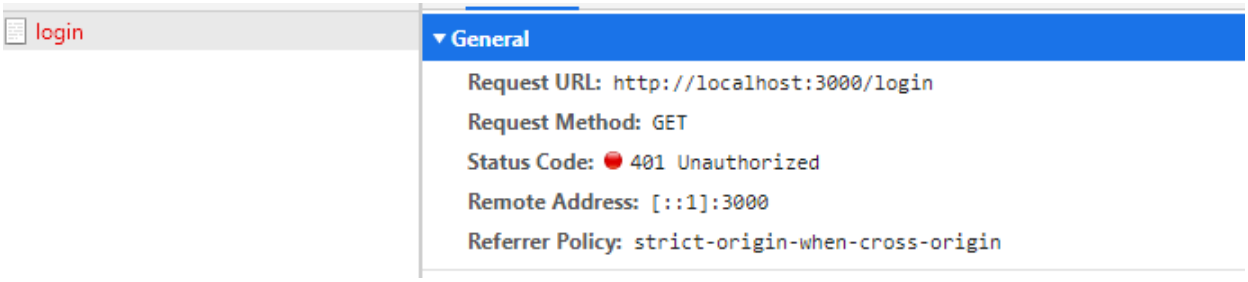
用户没有登录

服务端:

```
1 // 前面省略
```

```
app.get('/login',(req,res)=>{
 res.sendStatus(401)//未授权
 // res.sendStatus(403) 禁止访问
 // res.sendStatus(404) 未找到资源，内容未找到
 // res.sendStatus(405) 方法不被允许
})
```

客户端：

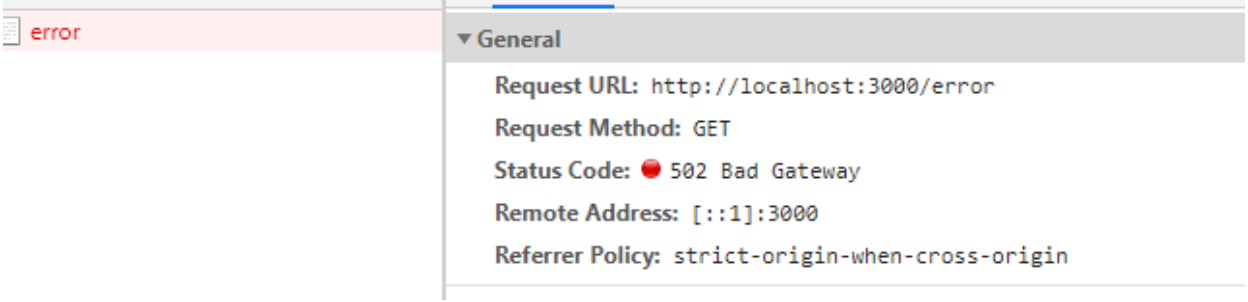


服务器报错

服务端：

```
app.get('/error',(req,res)=>{
 res.sendStatus(502) //网关错误
 // res.sendStatus(500) 服务错误
 // res.sendStatus(504) 网关超时
 // res.sendStatus(501) 没有实现，这个接口已经存在
 // res.sendStatus(503) 服务不可用
})
```

客户端：



。。。。等等

课程小结

Body解析过程中的协商技巧很有用

认真对待3XX状态码（谁开发,谁运维，前端工程师要配置反向代理，负载均衡）

错误处理遵循HTTP协议