# 还是express\_hello项目，我们来学习路由重定向

|  |
| --- |
|  |

# 上一节我们成功用express接收到了用户数据，然后其实我们需要校验这个表单数据，如果校验成功，需要重定向到一个路径。为了方便学习我们把03文件复制一份，起名：04-server-redirect.js，然后我们启动这个文件作为服务器，在这里学习重定向

|  |
| --- |
|  |

## 注意，04文件的端口改为3003

## 1.在public文件夹下面新建一个login.html文件

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>User Login</title>  </head>  <body>      <h3>用户登录</h3>      <form action="/dologin" method="post">          <p>用户姓名：<input type="text" name="username" placeholder="输入用户名"></p>          <p>用户密码：<input type="password" name="password"></p>          <p><input type="submit" value="注册"></p>      </form>  </body>  </html> |

## 2. 为了方便，我们改为全局注册析表单数据的中间件

|  |
| --- |
| //express处理get请求  let express = require('express')  let bodyParser = require('body-parser')  let app = express()  let PORT = 3003  app.get('/',(req,res)=>{      res.send('<h1>Welcome to index</h1>')  })  app.get('/get\_reg',(req,res)=>{      console.log(req.query);      res.send('<h1>login</h1>')  })  //配置静态服务器  app.use(express.static('public'))  //配置404  app.get('\*',(req,res)=>{ //如果我们没有为两个路径配置路由，就会被他匹配到      res.send('<h1>Not Found</h1>')  })  //post方法  //1.需要利用bodyParser来生成一个能够解析表单数据的中间件  let urlParser = bodyParser.urlencoded({extended:false}) //false接收字符串或者数组，true接收任意类型  // //2.全局注册解析表单数据的中间件  app.use(urlParser)  app.post('/post\_reg',(req,res)=>{      // console.log(req.body);      let {username,email,password} = req.body      console.log(username,email,password);        //业务逻辑：      //校验数据，是否为空，是否符合规定      //查询数据库看看用户名是否已经被注册了      //...      //假设数据已经通过上面的校验了，用户成功注册，现在需要跳转到      res.send(`<h4>welcome ${username}</h4>`)  })  app.post('/post\_login',(req,res)=>{      // console.log(req.body);      let {username,email,password} = req.body      console.log(username,email,password);        //业务逻辑：      //校验数据，是否为空，是否符合规定      //查询数据库看看用户名是否已经被注册了      //...      //假设数据已经通过上面的校验了，用户成功注册，现在需要跳转到      res.send(`<h4>welcome ${username}</h4>`)  })  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

## 3.定义两个常量作为模拟登录的数据相当于查询数据库操作

|  |
| --- |
|  |

## 4.在04-server-redirect.js中添加对/dologin路由的处理函数，然后在注册函数里面添加跳转处理

|  |
| --- |
| //express处理get请求  let express = require('express')  let bodyParser = require('body-parser')  let app = express()  let PORT = 3003  //模拟登录用数据  let USER = 'admin'  let PASS = '12345'  app.get('/',(req,res)=>{      res.send('<h1>Welcome to index</h1>')  })  app.get('/get\_reg',(req,res)=>{      console.log(req.query);      res.send('<h1>login</h1>')  })  //配置静态服务器  app.use(express.static('public'))  //配置404  app.get('\*',(req,res)=>{ //如果我们没有为两个路径配置路由，就会被他匹配到      res.send('<h1>Not Found</h1>')  })  //post方法  //1.需要利用bodyParser来生成一个能够解析表单数据的中间件  let urlParser = bodyParser.urlencoded({extended:false}) //false接收字符串或者数组，true接收任意类型  // //2.全局注册解析表单数据的中间件  app.use(urlParser)  app.post('/post\_reg',(req,res)=>{      // console.log(req.body);      let {username,email,password} = req.body      console.log(username,email,password);        //业务逻辑：      //校验数据，是否为空，是否符合规定      //查询数据库看看用户名是否已经被注册了      //...      //假设数据已经通过上面的校验了，用户成功注册，现在需要跳转到登录页面      res.redirect('/login.html')  })  app.post('/dologin',(req,res)=>{//模拟登录处理      // console.log(req.body);      let {username,password} = req.body      console.log(username,password);      if(username !== USER || password !== PASS){          res.send(`登录失败，用户名或者密码错误！！！`)  return      }      res.send(`<h4>登录成功，欢迎 ${username}</h4>`)  })  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

### 效果：

|  |
| --- |
|  |
|  |
|  |

### 当然这个服务器现在还是非常简陋的，我们慢慢改进