# 为了更好的学习express路由，我们把express\_hello\_art-template项目复制一份改名：express\_hello\_art-template-router

|  |
| --- |
|  |

# 抽取路由其实比较简单，如我们把首页的路由抽取出来，步骤如下

## 1.新建一个routes文件夹，在里面新建一个index.js

|  |
| --- |
|  |

## 2.index.js的内容如下

|  |
| --- |
| let express = require('express')  let router = express.Router()  router.get('/',(req,res)=>{      let data=[          {name:'登录',href:"/login.html"},          {name:'注册',href:"/register.html"},      ]      res.render('index',{data}) //渲染首页  })  module.exports = router |

## 3.在01-server-art.js中导入这个router，并且把原来处理’/’路由的代码注释掉，然后使用indexRouter

|  |
| --- |
| //express art-template的使用  let path = require('path')  let express = require('express')  let bodyParser = require('body-parser')  let app = express()  let PORT = 3003  //导入首页路由文件  let indexRouter = require('./routes/index')  //模拟登录用数据  let USER = 'admin'  let PASS = '12345'  //设置art-template引擎,4个步骤  //1.引入art-template,使用对应的引擎  app.engine('html',require('express-art-template'))  //2.设置项目环境  app.set('view options',{      debug:process.env.NODE\_ENV!== "production"  })  //3.设置视图路径，也就是到哪里找模板文件  app.set('views',path.join(\_\_dirname,'views'))  //4.设置模板文件的后缀名  app.set('view engine','html')  //配置静态资源服务器  app.use(express.static('public'))  app.use('/',indexRouter) //使用导入的路由文件  // app.get('/',(req,res)=>{  //     let data=[  //         {name:'登录',href:"/login.html"},  //         {name:'注册',href:"/register.html"},  //     ]  //     res.render('index',{data}) //渲染首页  // })  //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('/register',(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')//路由重定向  })  app.post('/login',(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}/`);  }) |

### 其实这个路由文件有点像中间件，他的使用方法和中间件类似，都是用app.use来加载

## 4.测试一下，一切正常

|  |
| --- |
|  |
|  |

## 5.然后我们把login和注册也抽取路由

### ./routes/login.js

|  |
| --- |
| let express = require('express')  let router = express.Router()   router.post('/',(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>`)  })  module.exports = router |

### ./routes/register.js

|  |
| --- |
| let express = require('express')  let router = express.Router()   router.post('/',(req,res)=>{ //这里的路由和app里面的是不一样的，app里面是/register      // 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')//路由重定向  })  module.exports = router |

### 01-server-art.js

|  |
| --- |
| //express art-template的使用  let path = require('path')  let express = require('express')  let bodyParser = require('body-parser')  let app = express()  let PORT = 3003  //导入首页路由文件  let indexRouter = require('./routes/index')  let loginRouter = require('./routes/login')  let registerRouter = require('./routes/register')  //模拟登录用数据  let USER = 'admin'  let PASS = '12345'  //设置art-template引擎,4个步骤  //1.引入art-template,使用对应的引擎  app.engine('html',require('express-art-template'))  //2.设置项目环境  app.set('view options',{      debug:process.env.NODE\_ENV!== "production"  })  //3.设置视图路径，也就是到哪里找模板文件  app.set('views',path.join(\_\_dirname,'views'))  //4.设置模板文件的后缀名  app.set('view engine','html')  //配置静态资源服务器  app.use(express.static('public'))  app.use('/',indexRouter) //使用导入的路由文件  //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) //使用路由文件导出的注册路由  app.use("/register",registerRouter)//使用路由文件导出的登录路由  app.use('/login',loginRouter)  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

### 使用路由文件可以适当项目的可维护性更好，需要注意的是路径问题，在app文件中是/register,在注册的路由文件中应该是/,因为app文件中是/register是匹配这个路由文件的条件，然后注册路由文件的路由应该是根路由。

# 在实际开发中。我们把验证效果的路由放到一个叫做passport.js的文件中来管理，他里面就会有注册和登录等路由

# 如果一个路由文件里面有多条路由规则，可以在app中把它全局注册

|  |
| --- |
|  |

# 我们来改造一下项目

## 1我们把上面的express\_hello\_art-template-router复制一份，改名express\_hello\_art-template-router2

|  |
| --- |
|  |

## 2.在routes文件夹里面新建一个passport.js文件，把登录和注册的路由处理都写在里面，然后打login.js和register.js文件删除，passport.js的内容如下

|  |
| --- |
| let express = require('express')  let router = express.Router()  //模拟登录用数据  let USER = 'admin'  let PASS = '12345'  router.post('/register',(req,res)=>{//这里必须使用/register因为此时在app里面没有做路由匹配      // 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')//路由重定向  })  router.post('/login',(req,res)=>{//这里必须使用/login因为此时在app里面没有做路由匹配      // 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>`)  })  module.exports = router |

## 然后把01-server-art.js修改如下

|  |
| --- |
| //express art-template的使用  let path = require('path')  let express = require('express')  let bodyParser = require('body-parser')  let app = express()  let PORT = 3004  //导入首页路由文件  let indexRouter = require('./routes/index')  let passportRouter = require('./routes/passport')  //设置art-template引擎,4个步骤  //1.引入art-template,使用对应的引擎  app.engine('html',require('express-art-template'))  //2.设置项目环境  app.set('view options',{      debug:process.env.NODE\_ENV!== "production"  })  //3.设置视图路径，也就是到哪里找模板文件  app.set('views',path.join(\_\_dirname,'views'))  //4.设置模板文件的后缀名  app.set('view engine','html')  //配置静态资源服务器  app.use(express.static('public'))  app.use(indexRouter) //全局注册路由文件  //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) //全局使用中间件    //全局注册passportRouter  app.use(passportRouter)  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

### 效果是一样的

## 注意：全局注册路由组件和局部使用路由组件的路径匹配方式是不太一样的。全局注册的路由组件由于中app中没有指定路由规则，使用路由组件必须匹配完整路径，而局部使用的路由组件由于app里面已经做了匹配，路由需要减少一层，express\_hello\_art-template-router项目使用的就是局部路由的方式，而express\_hello\_art-template-router2则有使用全局注册路由的方式