# 钩子函数就是在做某个功能之前或者之后执行的指定函数，如在全局注册路由组件之前，可以传递一个函数，让app对象先执行它在注册路由

|  |
| --- |
|  |
| 在项目入口函数app.js中： |

# 示例，

## 1.把上面express\_hello\_art-template-router2项目复制一份，改名express\_hello\_art-template-router2-hook

|  |
| --- |
|  |

## 2.在项目根目录下面新建一个hooks文件夹，在里面写一个index.js文件，内容如下

|  |
| --- |
|  |

## 3.然后在01-server-art.js中app全局注册passportRouter路由组件之前，调用这个checkLogin函数

|  |
| --- |
| //express art-template的使用  let path = require('path')  let express = require('express')  let bodyParser = require('body-parser')  let utils = require('./hooks/index')  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) //使用路由文件导出的注册路由  app.use(bodyParser.json())   // 解析json格式    //全局注册passportRouter,可以在注册之前调用一个钩子或者中间件函数  app.use(utils.checkLogin,passportRouter)  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

### 当然，还可以全局注册这个钩子函数，这样子，无论使用声明动态路由都会执行这个函数，我们修改一下01-server-art.js的代码

|  |
| --- |
| //express art-template的使用  let path = require('path')  let express = require('express')  let bodyParser = require('body-parser')  let utils = require('./hooks/index')  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'))  //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(bodyParser.json())   // 解析json格式    //全局注册passportRouter,可以在注册之前调用一个钩子或者中间件函数  app.use(utils.checkLogin)  app.use(passportRouter)  app.use(indexRouter) //全局注册路由文件  //配置404,一定要在最后面  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>')  })  app.listen(PORT,()=>{      console.log(`url: http://localhost:${PORT}/`);  }) |

### 这样子，即使你使用indexRouter也会调用这个函数

|  |
| --- |
|  |
|  |

## 这个函数其实很有用，可以例如，可以用在某些特定的场合，如一些操作需要登录了才能够做，此时，我们就可以用这个函数来检测用户是否登录，如果没有登录，可以阻止他做相应的操作或者重定向到登录页面等等

|  |
| --- |
|  |