大厂面试课之Go语言

第20课：首页

同学们好！千里之行，始于足下。在这节课里，我们的任务是在浏览器里显示《我家租房网》的首页。

显示默认首页

获取Session

首先，我们将在前端服务器中增加显示默认首页的路由处理，其次借助浏览器的调试功能，修复程序运行过程中出现的错误，逐步完善其它功能。

显示默认首页

常识告诉我们，打开浏览器，在地址栏中输入某个应用的URL，浏览器窗口即会显示出该应用的默认首页。该页面是应用一切功能的入口，也是用户访问应用的起点。

知识讲解

将“/home”路径路由到“./view”目录

- router.Static("/home", "./view")

上一节课，我们在front-end工程目录的view子目录下，放置了一系列静态页面及其所依赖的JavaScript代码、层叠式样式单、图标图片等各种资源。其中就包括一个名为index.html的静态页面，这就是《我家租房网》应用的默认首页。我们所要达到的效果是，只要用户在浏览器的地址栏中输入localhost:8080/home，就能够显示该页面。为此，我们可以通过路由对象的Static方法，创建一条静态路由。该方法需要两个参数，第一个参数为路由路径，比如“/home”，第二个参数为路由目标，比如“./view/index.html”。关于路由目标，这里使用了相对路径，因为前端服务器的可执行程序就位于front-end工程目录下，与view子目录同级。我们甚至可以使用更简单的形式，比如“./view”。在不显式给出文件名的情况下，Gin框架会缺省选择index.html文件。

GOPATH/src/iHome/front-end/main.go

项目实战

```
package main

import "github.com/gin-gonic/gin"

func main() {
    // 初始化路由
    router := gin.Default()

    // 路由匹配
    router.Static("/home", "./view")

    // 运行
    router.Run(":8080")
}
```

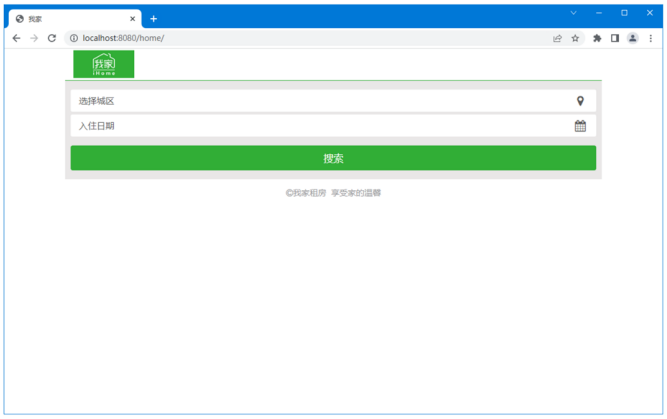
修改front-end工程目录下的main.go文件：

```
1  package main
2
3  import "github.com/gin-gonic/gin"
4
5  func main() {
6      // 初始化路由
7      router := gin.Default()
8
9      // 路由匹配
10     router.Static("/home", "./view")
11
12     // 运行
13     router.Run(":8080")
14 }
```

这里我们调用了路由对象的Static方法，创建了一条静态路由，将用户输入的资源路径“/home”，路由到指定的页面文件，当前目录下view子目录中的index.html。

启动前端，用浏览器访问：http://localhost:8080/home

项目实战

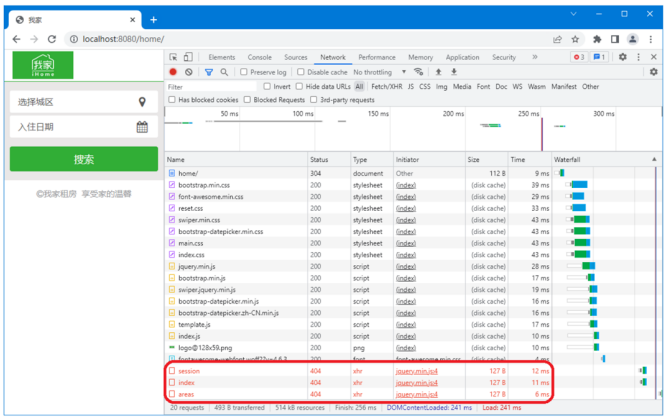


- 显示默认首页

启动前端，用浏览器访问localhost:8080/home，可以看到《我家租房网》的默认首页，已赫然显现在浏览器窗口中。

按F12键调试，点选Network标签

项目实战



- 三个错误：session、index、areas

在Chrome浏览器中按下F12功能键，会打开调试窗格，点选Network标签，会看到三个以红色显示的错误信息：session、index和areas。点击其中的“session”，选择Headers标签，会看到发生错误的请求URL为localhost:8080/api/v1.0/session，状态码为404。发生这样的错误并不奇怪，我们的前端服务器确实没有为路径“/api/v1.0/session”设置路由。

获取Session

GET方法结合/api/v1.0/session路径表示获取会话。具体如何获取会话，我们会在后续课程中为大家详细介绍。这节课，我们只是添加相应的路由处理，消灭错误。

添加路由

服务	请求	URL	函数
获取Session	GET	/api/v1.0/session	GetSession

这里我们要为获取会话添加一条路由，GET方法结合/api/v1.0/session路径，处理函数名为GetSession。

GOPATH/src/iHome/front-end/main.go

```
package main

import (
    "github.com/gin-gonic/gin"
    "iHome/front-end/controller"
)

func main() {
    // 初始化路由
    router := gin.Default()

    // 路由匹配
    router.Static("/home", "./view")
    router.GET("/api/v1.0/session", controller.GetSession)

    // 运行
    router.Run(":8080")
}
```


再次修改front-end工程目录下的main.go文件：

```
1  package main
2
3  import (
4      "github.com/gin-gonic/gin"
5      "iHome/front-end/controller"
6  )
7
8  func main() {
9      // 初始化路由
10     router := gin.Default()
11
12     // 路由匹配
13     router.Static("/home", "./view")
14     router.GET("/api/v1.0/session", controller.GetSession)
15
16     // 运行
17     router.Run(":8080")
18 }
```

这里我们调用了路由对象的GET方法，为获取会话添加了一条路由，将GET方法结合/api/v1.0/session路径，路由到controller包的GetSession函数。将GetSession函数定义在controller包里是因为该函数的主要任务是执行业务逻辑，属于MVC中的C，即控制器层的部分。

添加函数

当然，在controller包里真的得有GetSession函数。

GOPATH/src/iHome/front-end/controller/user.go

项目实战

```
package controller

import (
    "github.com/gin-gonic/gin"
    "iHome/front-end/utils"
    "net/http"
)

// 获取Session
func GetSession(ctx *gin.Context) {
    rsp := make(map[string]string)
    rsp["errno"] = utils.ERROR_USER_NOT_LOGIN
    rsp["errmsg"] = utils.StrError(rsp["errno"])

    ctx.JSON(http.StatusOK, rsp)
}
```


为此，我们在front-end工程目录下的controller子目录中创建一个名为user.go的文件：

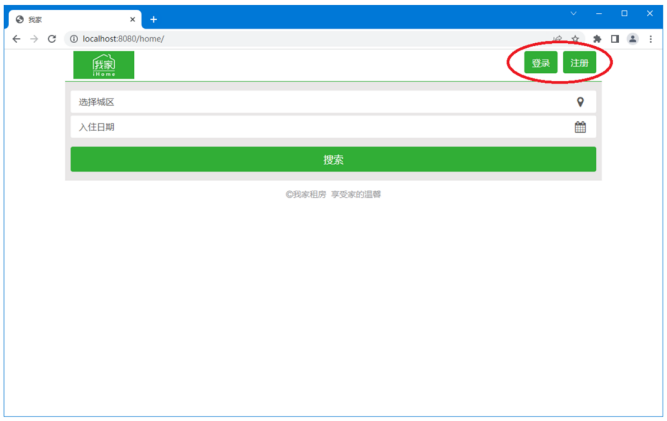
```
1  package controller
2
3  import (
4      "github.com/gin-gonic/gin"
5      "iHome/front-end/utils"
6      "net/http"
7  )
8
9  // 获取Session
10 func GetSession(ctx *gin.Context) {
11     rsp := make(map[string]string)
12     rsp["errno"] = utils.ERROR_USER_NOT_LOGIN
13     rsp["errmsg"] = utils.StrError(rsp["errno"])
```

```
14
15      ctx.JSON(http.StatusOK, rsp)
16  }
```

在这里我们创建了一个名为controller的包，并在包中定义了一个名为GetSession的函数。在GetSession函数里，我们创建了一个字符串到字符串的映射，并为其添加了两个键值对，一个表示错误代码，另一个表示错误描述。最后通过Gin框架调用该函数时传入的上下文对象的JSON方法，构造响应报文，响应头中的状态码为200，响应体为一个JSON字符串，源于对映射的序列化。

启动前端，用浏览器访问：http://localhost:8080/home

项目实战

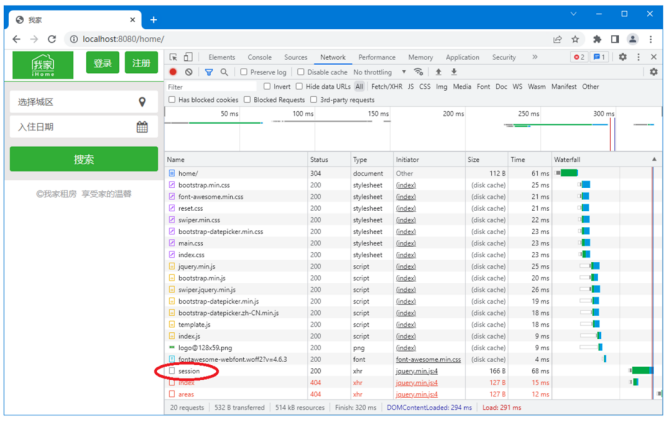


- 右上角出现“登录”和“注册”按钮

重启前端，用浏览器访问localhost:8080/home，右上角出现“登录”和“注册”按钮。

按F12键调试，点选Network标签

项目实战

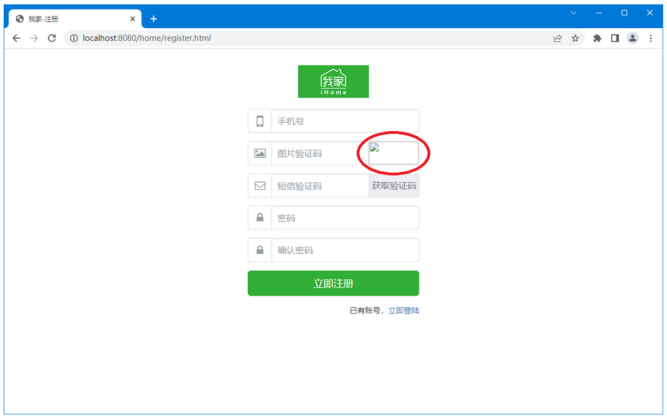


- sesion错误消失

在调试窗格中点选Network标签，会看到session错误消失了。点击“session”，选择Response标签，会看到JSON形式的响应体字符串，其中的错误代码为“4100”，错误描述为“用户未登录”，这与我们在GetSession函数中编写的代码是吻合的。

点击“注册”按钮，进入注册页面

项目实战



- 获取图片验证码失败

点击“注册”按钮，进入注册页面。这时我们会发现获取图片验证码失败，这正是我们在下节课需要解决的问题。

更多精彩，敬请期待

谢谢大家，我们下节课再见！