大厂面试课之Go语言

第27课：用户退出

同学们好！成功登录《我家租房网》系统的用户，会在搜索页面的右上角看到自己的用户名，初始用户名为手机号。这时如果点击该用户名，则会进入用户页面。点击用户页面中的“退出”，将退出系统，回到未登录状态。退出操作的本质就是从Session中将当前用户的用户名删除。只要Session中没有用户名，即表示用户处于未登录状态。

从前面有关用户退出的描述中，可以看出，用户退出并不涉及到后端微服务，只需在前端服务器中处理好Session中的数据即可。我们将先在前端服务器中添加路由处理函数，完成从Session中删除用户名的操作，然后再就这项功能做一下测试。

前端

打开front-end前端工程。

删除Session

在模型层添加删除Session的函数。

GOPATH/src/iHome/front-end/model/session.go

项目实战

```
...
// 从Session中删除用户名
func DeleteUsername(ctx *gin.Context) error {
    // 获取Session
    session := sessions.Default(ctx)
    // 删除Session中指定的键值对
    session.Delete("username")
    // 保存Session
    return session.Save()
}
...
```

打开front-end工程目录下，model子目录中的session.go文件，添加有关从Session中删除用户名的代码：

```
...
// 从Session中删除用户名
func DeleteUsername(ctx *gin.Context) error {
    // 获取Session
    session := sessions.Default(ctx)
    // 删除Session中指定的键值对
    session.Delete("username")
    // 保存Session
    return session.Save()
}
...
```

这里定义了一个名为DeleteUsername的函数，用于从Session中删除用户名。该函数首先通过sessions包的Default方法，从上下文中获取Session。然后在Session中根据用户名的键，删除对应的键值对。最后通过Session的Save方法，保存所做的修改。

路由——控制器

服务	请求	URL	函数
用户退出	DELETE	/api/v1.0/session	UserLogout

这里我们要为用户退出添加一条路由，DELETE方法结合/api/v1.0/session路径，处理函数名为UserLogout。

项目实战

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

```
...
func main() {
    ...
    // 路由匹配
    ...
    apiv10.DELETE("/session", controller.UserLogout)
    ...
}
...
```

在front-end工程目录下的main.go文件中添加一条路由：

```
...
func main() {
    ...
    // 路由匹配
    ...
    apiv10.DELETE("/session", controller.UserLogout)
    ...
}
...
```

这里我们调用了路由对象的DELETE方法，为用户退出添加了一条路由，将DELETE方法结合/session路径，路由到controller包的UserLogout函数。将UserLogout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。

项目实战

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

```
...
// 用户退出
func UserLogout(ctx *gin.Context) {
    rsp := make(map[string]interface{})
    rsp["errno"] = utils.ERROR_OK

    // 从Session中读取用户名
    username := model.ReadUsername(ctx)
    if username != "" {
        // 从Session中删除用户名
        if err := model.DeleteUsername(ctx); err != nil {
            fmt.Println(err)
            rsp["errno"] = utils.ERROR_USER_LOGOUT
        }
    } else {
        rsp["errno"] = utils.ERROR_USER_NOT_LOGIN
    }

    rsp["errmsg"] = utils.StrError(rsp["errno"].(string))
    ctx.JSON(http.StatusOK, rsp)
}
...
```


当然，在controller包里真的得有UserLogout函数。为此，我们打开front-end工程目录下controller子目录中的user.go文件，定义UserLogout函数：

```
...
// 用户退出
func UserLogout(ctx *gin.Context) {
    rsp := make(map[string]interface{})
    rsp["errno"] = utils.ERROR_OK

    // 从Session中读取用户名
    username := model.ReadUsername(ctx)
    if username != "" {
        // 从Session中删除用户名
        if err := model.DeleteUsername(ctx); err != nil {
            fmt.Println(err)
            rsp["errno"] = utils.ERROR_USER_LOGOUT
        }
    } else {
        rsp["errno"] = utils.ERROR_USER_NOT_LOGIN
    }

    rsp["errmsg"] = utils.StrError(rsp["errno"].(string))
    ctx.JSON(http.StatusOK, rsp)
}
...
```

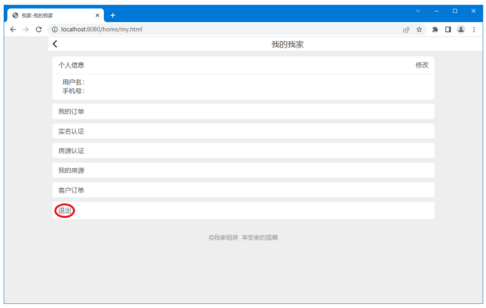
首先调用模型层的ReadUsername函数，从Session中读取用户名。如果读到了，说明该用户已登录，调用模型层的DeleteUsername函数，从Session中删除用户名键值对，没读到，则响应“用户未登录”错误。最后，将该响应对象序列化为一个JSON字符串，编码到HTTP响应中，回传给浏览器。

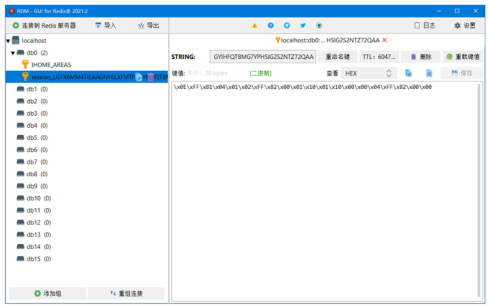
测试

至此，我们已经完成用户退出的全部开发工作。下面我们将对这部分功能进行测试。

执行如下操作，检查Redis数据库

项目实战





1. 启动Consul、后端（UserLogin）和前端
2. 通过登录页面登录系统
3. 点击右上角手机号，进入用户页面
4. 点击“退出”，退出系统

启动Consul服务器、UserLogin后端微服务和前端服务器。通过登录页面登录系统，点击位于搜索页面右上角的用户名，初始用户名即手机号，进入用户页面，点击“退出”，退出系统。检查Redis数据库中的Session，可以看到当前登录用户的用户名键值对已被删除。

更多精彩，敬请期待

谢谢大家，我们下节课再见！