cookie 和 session 的区别以及 session 的实现原理

作者：450370050
原文链接：https://ld246.com/article/1524558277212
来源网站：链滴

记得以前面试会有被问到session和cookie的区别，今天咱们梳理下cookie和session到底有什么区别。

常见解释：

1. session 在服务器端，cookie 在客户端（浏览器）
2. session 默认被存在在服务器的一个文件里（不是内存）
3. session 的运行依赖 session id，而 session id 是存在 cookie 中的，也就是说，如果浏览器禁用了 cookie ，同时 session 也会失效（但是可以通过其它方式实现，比如在 url 中传递 session_id）
4. session 可以放在 文件、数据库、或内存中都可以。

我们通过分析beego的session实现来具体验证一下以上几点

测试验证：

beego session使用

beego 内置了 session 模块，目前 session 模块支持的后端引擎包括 memory、cookie、file、mysq、redis、couchbase、memcache、postgres，用户也可以根据相应的 interface 实现自己的引擎。

beego 中使用 session 相当方便，只要在 main 入口函数中设置如下：

```
beego.BConfig.WebConfig.Session.SessionOn = true
```

或者通过配置文件配置如下：

```
sessionon = true
```

通过这种方式就可以开启 session，如何使用 session，请看下面的例子：

```
func (this *MainController) Get() {
   v := this.GetSession("asta")
   if v == nil {
      this.SetSession("asta", int(1))
      this.Data["num"] = 0
  } else {
      this.SetSession("asta", v.(int)+1)
      this.Data["num"] = v.(int)
   }
   this.TplName = "index.tpl"
}
```

源码分析

controller实现的GetSession、SetSession方法可以读取、设置session信息，那么它做了什么呢。

```
// StartSession starts session and load old session data info this controller.
func (c *Controller) StartSession() session.Store {
   if c.CruSession == nil {
      c.CruSession = c.Ctx.Input.CruSession
   }
```

原文链接：cookie 和 session 的区别以及 session 的实现原理

```
    return c.CruSession
}

// SetSession puts value into session.
func (c *Controller) SetSession(name interface{}, value interface{}) {
    if c.CruSession == nil {
        c.StartSession()
    }
    c.CruSession.Set(name, value)
}

// GetSession gets value from session.
func (c *Controller) GetSession(name interface{}) interface{} {
    if c.CruSession == nil {
        c.StartSession()
    }
    return c.CruSession.Get(name)
}
```

我们可以看到 session的数据是通过CruSession(session仓库)进行存取

beego的router路由处理请求时SessionStart设置了CruSession

```
// session init
if BConfig.WebConfig.Session.SessionOn {
    var err error
   context.Input.CruSession, err = GlobalSessions.SessionStart(rw, r)
    if err != nil {
        logs.Error(err)
        exception("503", context)
        goto Admin
    }
    defer func() {
        if context.Input.CruSession != nil {
            context.Input.CruSession.SessionRelease(rw)
        }
    }()
}
```

SessionStart 类似于 php的**session_start**此方法是session依赖cookie实现的关键，大家看下代码我的注释，了解下代码运行逻辑就明白了。

```
func (manager *Manager) SessionStart(w http.ResponseWriter, r *http.Request) (session Store,
err error) {
    //获取当前用户的sid  sid会通过用户的cookie，请求参数 以及header 中读取
    sid, errs := manager.getSid(r)
    if errs != nil {
        return nil, errs
    }

    // sid不为空并且存在此sid对应的session，返回session存储数据结构
    if sid != "" && manager.provider.SessionExist(sid) {
        return manager.provider.SessionRead(sid)
    }
```

```
    // 因为sid为空生成一个 sid
    sid, errs = manager.sessionID()
    if errs != nil {
        return nil, errs
    }
    //根据sid生成一个session存储数据结构
    session, err = manager.provider.SessionRead(sid)
    if err != nil {
        return nil, err
    }
    //组装sid的cookie信息并在返回header中设置此cookie
    cookie := &http.Cookie{
        Name:     manager.config.CookieName,
        Value:    url.QueryEscape(sid),
        Path:     "/",
        HttpOnly: !manager.config.DisableHTTPOnly,
        Secure:   manager.isSecure(r),
        Domain:   manager.config.Domain,
    }
    if manager.config.CookieLifeTime > 0 {
        cookie.MaxAge = manager.config.CookieLifeTime
        cookie.Expires = time.Now().Add(time.Duration(manager.config.CookieLifeTime) * time.Se
ond)
    }
    if manager.config.EnableSetCookie {
        http.SetCookie(w, cookie)
    }
    r.AddCookie(cookie)

    if manager.config.EnableSidInHTTPHeader {
        r.Header.Set(manager.config.SessionNameInHTTPHeader, sid)
        w.Header().Set(manager.config.SessionNameInHTTPHeader, sid)
    }

    return
}

func (manager *Manager) getSid(r *http.Request) (string, error) {
    //cookie中读取sid
    cookie, errs := r.Cookie(manager.config.CookieName)
    if errs != nil || cookie.Value == "" {
        var sid string
    //请求参数中读取cookie
    if manager.config.EnableSidInURLQuery {
            errs := r.ParseForm()
            if errs != nil {
                return "", errs
            }

            sid = r.FormValue(manager.config.CookieName)
    }

    // 如果cookie 请求参数中都没有sid 那就从请求的header中读取
    if manager.config.EnableSidInHTTPHeader && sid == "" {
```

```
        sids, isFound := r.Header[manager.config.SessionNameInHTTPHeader]
        if isFound && len(sids) != 0 {
            return sids[0], nil
        }
    }

    return sid, nil
}

// HTTP Request contains cookie for sessionid info.
return url.QueryUnescape(cookie.Value)
}
```

我们了解以上代码的话，可以总结流程为：

1. 读取用户的session_id

session_id 可以通过 cookie/请求参数/header参数 读取

2. 通过session_id获取用户的session存储

```
type SessionStore struct {
    sid          string
    lock         sync.RWMutex
    values       map[interface{}]interface{}
    maxlifetime int64
}
```

sid(session_id)、lock(读写锁)、values(session存储的数据)、maxlifetime(session的有效期) sessio
的存储由以上几个基本元素构成，我们也可以实现一个自己的存储。