Atitit session机制的实现

目录

[1. Web状态管理 Cookie 和 Session。 token 1](#_Toc16980)

[2. session 管理设计 1](#_Toc8398)

[2.1. session 的存储（可以存储到内存、文件、数据库等） 2](#_Toc4813)

[2.2. session 创建过程 2](#_Toc2170)

[3. Go 实现 session 管理 4](#_Toc10402)

[4. Php的实现 文件模式 4](#_Toc22545)

[5. Java net 的实现 4](#_Toc8131)

# Web状态管理 Cookie 和 Session。 token

Web 开发中一个很重要的议题就是如何做好用户整个浏览过程的控制，因为 HTTP 协议是无状态的，所以用户的每一次请求都是无状态的，不知道在整个 Web 操作过程中哪些连接与该用户有关。应该如何来解决这个问题呢？Web 里面经典的解决方案是 Cookie 和 Session。

# session 管理设计

我们知道 session 管理涉及到如下几个因素

* 全局 session 管理器
* 保证 sessionid 的全局唯一性
* 为每个客户关联一个 session

## session 的存储（可以存储到内存、文件、数据库等）

* session 过期处理

Cookie 机制是一种客户端机制，把用户数据保存在客户端，而 Session 机制是一种服务器端的机制，服务器使用一种类似于散列表的结构来保存信息，每一个网站访客都会被分配给一个唯一的标识符，即 sessionID。  
  
sessionID 的存放形式无非两种：要么经过 URL 传递，要么保存在客户端的 Cookie 里。当然，也可以将 Session 保存到数据库里，这样会更安全，但效率方面会有所下降。本节主要介绍 [Go语言](http://c.biancheng.net/golang/" \t "http://c.biancheng.net/view/_blank)使用 Cookie 的方法。

设置 Cookie

面《[Cookie设置与读取](http://c.biancheng.net/view/5560.html" \t "http://c.biancheng.net/view/_blank)》一节我们介绍了 Cookie 的应用，本节我们将讲解 session 的应用，我们知道 session 是在服务器端实现的一种用户和服务器之间认证的解决方案，目前 [Go语言](http://c.biancheng.net/golang/" \t "http://c.biancheng.net/view/_blank)标准包没有为 session 提供任何支持，接下来我们将会自己动手来实现 go 版本的 session 管理和创建。

## session 创建过程

session 的基本原理是由服务器为每个会话维护一份信息数据，客户端和服务端依靠一个全局唯一的标识来访问这份数据，以达到交互的目的。当用户访问 Web 应用时，服务端程序会随需要创建 session，这个过程可以概括为三个步骤：

* 生成全局唯一标识符（sessionid）；
* 开辟数据存储空间。一般会在内存中创建相应的[数据结构](http://c.biancheng.net/data_structure/" \t "http://c.biancheng.net/view/_blank)，但这种情况下，系统一旦掉电，所有的会话数据就会丢失，如果是电子商务类网站，这将造成严重的后果。所以为了解决这类问题，可以将会话数据写到文件里或存储在数据库中，当然这样会增加 I/O 开销，但是它可以实现某种程度的 session 持久化，也更有利于 session 的共享；
* 将 session 的全局唯一标示符发送给客户端。

以上三个步骤中，最关键的是如何发送这个 session 的唯一标识这一步上。考虑到 HTTP 协议的定义，数据无非可以放到请求行、头域或 Body 里，所以一般来说会有两种常用的方式：cookie 和 URL 重写。  
  
Cookie 服务端通过设置 Set-cookie 头就可以将 session 的标识符传送到客户端，而客户端此后的每一次请求都会带上这个标识符，另外一般包含 session 信息的 cookie 会将失效时间设置为 0（会话 cookie），即浏览器进程有效时间。至于浏览器怎么处理这个 0，每个浏览器都有自己的方案，但差别都不会太大（一般体现在新建浏览器窗口的时候）。  
  
URL 重写，所谓 URL 重写，就是在返回给用户的页面里的所有的 URL 后面追加 session 标识符，这样用户在收到响应之后，无论点击响应页面里的哪个链接或提交表单，都会自动带上 session 标识符，从而就实现了会话的保持。虽然这种做法比较麻烦，但是，如果客户端禁用了 cookie 的话，此种方案将会是首选。

# Go 实现 session 管理

通过上面 session 创建过程的讲解，大家应该对 session 有了一个大体的认识，但是具体到动态页面技术里面，又是怎么实现 session 的呢？下面我们将结合 session 的生命周期（lifecycle），来实现 Go语言版本的 session 管理。

session 管理设计

我们知道 session 管理涉及到如下几个因素

* 全局 session 管理器
* 保证 sessionid 的全局唯一性
* 为每个客户关联一个 session
* session 的存储（可以存储到内存、文件、数据库等）
* session 过期处理

接下来将讲解一下关于 session 管理的整个设计思路以及相应的 go 代码示例：

# Php的实现 文件模式

# Java net 的实现