

# Go培训第16天

tony

## Outline

1. 聊天室模块划分
2. 登陆&注册功能实现
3. 房间服务实现
4. 群聊功能实现

## 聊天室模块划分

### 1. 聊天室主要功能

- a. 用户注册和登陆
- b. 聊天房间模块
- c. 聊天房间进入&群聊

## 登陆注册功能实现

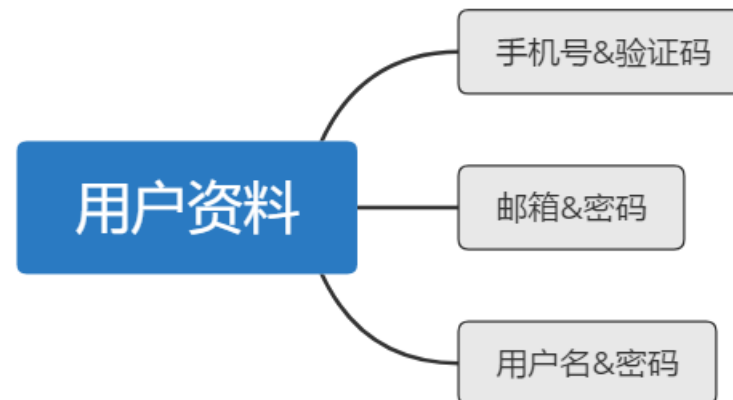
### 1. 为什么需要账号系统？

- a. http协议是无状态的，服务端需要确认每次访问者的身份。
- b. 有些功能必须确认身份之后，才能使用。比如转账、购物等

## 账号模块简介

2. 如何唯一标识一个用户的身份？

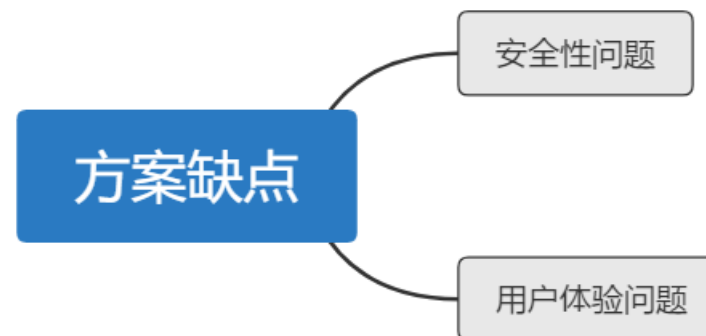
a. 通过用户提供一些资料，这些资料能够表明用户的身份。



## 账号模块简介

### 3. 怎么进行身份验证？

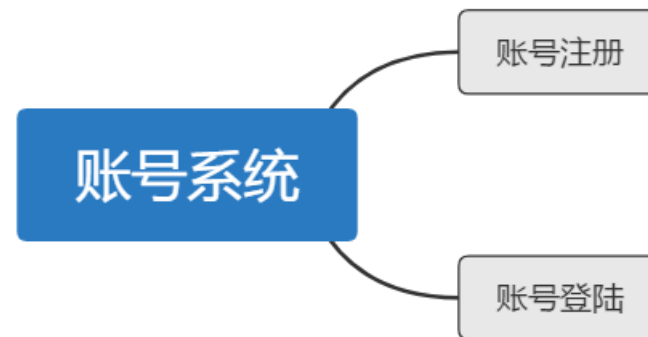
a. 每次请求的时候都把表明身份的《用户资料》带上。



## 账号模块简介

### 4. 如何解决这个问题？

- a. 使用账号系统进行同一管理和鉴权。
- b. 通过账号注册把用户的身份信息存储起来。
- c. 通过用户登陆进行身份鉴权，鉴权通过，则以后再也不需要登陆。



## Cookie&session机制

### 5. Cookie机制

- a. Cookie, 中文意思是小甜饼。
- b. 存储在用户本地（客户端）的一个数据文件。
- c. Cookie机制1：浏览器发送请求的时候，自动把cookie给带上。
- d. Cookie机制2：服务端可以设置cookie。
- e. Cookie机制3：cookie是针对单个域名的，不同域名之间的cookie是独立的。



## Cookie&session机制

### 6. Cookie与登陆鉴权

- a. 用户登陆校验账号、密码通过之后，设置一个cookie: username=shaolin
- b. 用户每次请求时，会自动把cookie: username=shaolin，发送到服务端
- c. 服务端收到这个请求之后，从cookie里面取出username，然后查询该用户是否已经登陆。
- d. 如果登陆的话，则鉴权通过。没有登陆则重定向到注册页面。

## Cookie&session机制

### 7. 方案缺陷

- a. Cookie容易被伪造，因为用户名都是有规律的，很容易被人猜到
- b. 猜到用户名之后，只需要把用户名带到请求上，就被攻破了。

## Cookie&session机制

### 8. 方案改进

- a. 既然生成的cookie直接存username容易被识破，因此需要生成一个随机id来代替
- b. 比如生成一个32位的uuid，比如：id=f7bc50d60641337e3b7061e23b264971
- c. 用户再次请求的时候，会自动把id= f7bc50d60641337e3b7061e23b264971，自动带到服务器。
- d. 服务端程序通过这个id，然后查询这个id对应的用户信息，就搞定了。

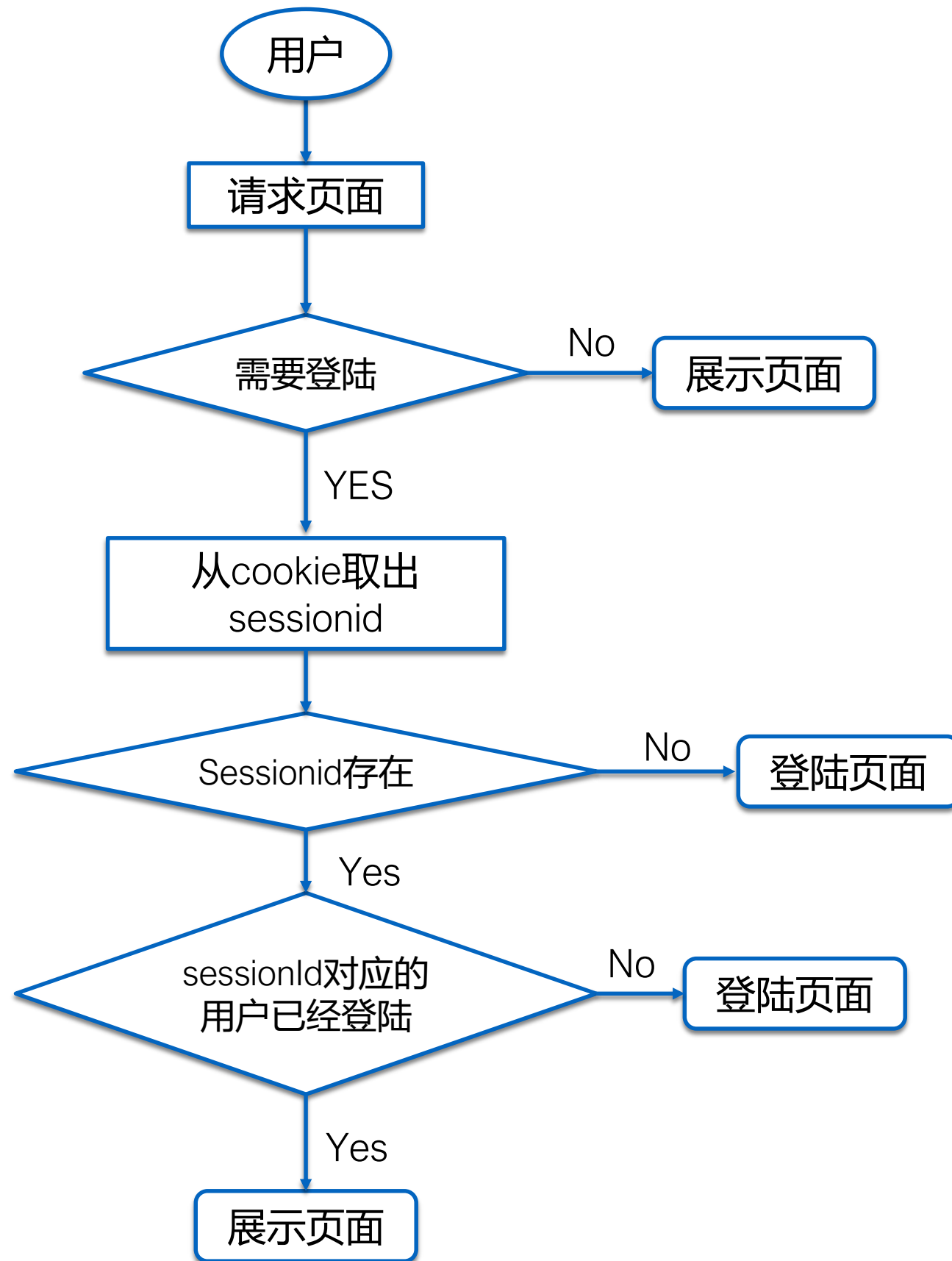
## Cookie&session机制

### 9. Session机制

- a. 我们把上面在服务端生成的id以及保存id对应用户信息的机制，叫做session机制
- b. Session和cookie共同构建了我们账号鉴权体系。
- c. Cookie是保存在客户端的， session是保存在服务端。
- d. 当服务端登陆校验成功之后，就分配一个无法伪造的id，存储在用户的机器上，以后每次请求的时候都带上这个id，就能够达到鉴权的目的了

## Cookie&session机制

### 10. 登陆流程



## 账号模块开发

### 24. 账号模块功能介绍

- a. 主要提供登陆和注册两个功能
- b. 登陆支持账号和密码登陆
- c. 注册只收集用户名、昵称、性别以及密码等信息
- d. user\_id通过全局唯一id生成器生成

## 账号模块开发

### 25. 账号模块数据库设计

```
CREATE TABLE `user` (  
  `id` bigint(20) NOT NULL,  
  `user_id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `username` varchar(64) COLLATE utf8mb4_general_ci NOT NULL,  
  `nickname` varchar(64) COLLATE utf8mb4_general_ci NOT NULL,  
  `password` varchar(64) COLLATE utf8mb4_general_ci NOT NULL,  
  `sex` tinyint(4) NOT NULL DEFAULT '0',  
  `create_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP,  
  `update_time` timestamp NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,  
  PRIMARY KEY (`user_id`),  
  UNIQUE KEY `idx_username` (`username`) USING BTREE  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_general_ci;
```

## 账号模块开发

26. 为什么要用全局唯一id生成器

- a. 单机系统生成唯一id非常简单，比如使用mysql的自增id
- b. 分布式系统多个分片插入数据时，如何保证插入的id唯一？



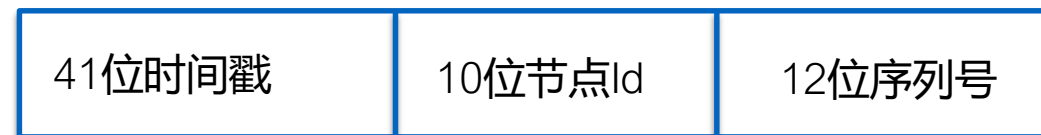
### 27. 全局唯一id生成器要求

- a. 生成的 ID 全局唯一
- b. 生成的 ID 最好不大于 64 bits
- c. 高性能要求
- d. 整个系统没有单点

## 账号模块开发

### 28. tweet Snowflake算法介绍

a. Github地址: <https://github.com/twitter/snowflake>



unique ID 生成过程:

- 10 bits 的机器号, 在 ID 分配 Worker 启动的时候, 从一个 Zookeeper 集群获取 (保证所有的 Worker 不会有重复的机器号)
- 41 bits 的 Timestamp: 每次要生成一个新 ID 的时候, 都会获取一下当前的 Timestamp, 然后分两种情况生成 sequence number:
- 如果当前的 Timestamp 和前一个已生成 ID 的 Timestamp 相同 (在同一毫秒中), 就用前一个 ID 的 sequence number + 1 作为新的 sequence number (12 bits); 如果本毫秒内的所有 ID 用完, 等到下一毫秒继续 (这个等待过程中, 不能分配出新的 ID)
- 如果当前的 Timestamp 比前一个 ID 的 Timestamp 大, 随机生成一个初始 sequence number (12 bits) 作为本毫秒内的第一个 sequence number

# 账号模块开发

## 29. tweet Snowflake算法异常情况分析

整个过程中, 只是在 Worker 启动的时候会对外部有依赖 (需要从 Zookeeper 获取 Worker 号), 之后就可以独立工作了, 做到了去中心化.

异常情况讨论:

- 在获取当前 Timestamp 时, 如果获取到的时间戳比前一个已生成 ID 的 Timestamp 还要小怎么办? Snowflake 的做法是继续获取当前机器的时间, 直到获取到更大的 Timestamp 才能继续工作 (在这个等待过程中, 不能分配出新的 ID)

从这个异常情况可以看出, 如果 Snowflake 所运行的那些机器时钟有大的偏差时, 整个 Snowflake 系统不能正常工作 (偏差得越多, 分配新 ID 时等待的时间越久)

从 Snowflake 的官方文档 (<https://github.com/twitter/snowflake/#system-clock-dependency>) 中也可以看到, 它明确要求 "You should use NTP to keep your system clock accurate". 而且最好把 NTP 配置成不会向后调整的模式. 也就是说, NTP 纠正时间时, 不会向后回拨机器时钟.

## 账号模块开发

### 30. 我们采用的id生成器

a. github: [github.com/sony/sonyflake](https://github.com/sony/sonyflake)

```
package main

import (
    "fmt"

    "github.com/sony/sonyflake"
)

func main() {
    //生产环境一定要设置machineID, 使用zk或者etcd
    st := sonyflake.Settings{}
    sk := sonyflake.NewSonyflake(st)
    fmt.Println(sk.NextID())
}
```

## 房间模块开发

### 24. 房间模块功能介绍

- a. 房间基本信息，比如房间名字、描述、人员上限
- b. 房间计数信息，比如当前在线人数

## 房间模块开发

### 24. 房间表结构

```
CREATE TABLE `room` (  
  `id` bigint(20) NOT NULL AUTO_INCREMENT,  
  `room_id` bigint(20) NOT NULL,  
  `room_name` varchar(64) NOT NULL,  
  `cap` int(10) unsigned NOT NULL DEFAULT '1000',  
  `create_time` timestamp NULL DEFAULT  
CURRENT_TIMESTAMP,  
  `update_time` timestamp NULL DEFAULT  
CURRENT_TIMESTAMP ON UPDATE  
CURRENT_TIMESTAMP,  
  PRIMARY KEY (`id`),  
  UNIQUE KEY `idx_room_id` (`room_id`)  
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4  
COLLATE=utf8mb4_0900_ai_ci;
```

# 课后作业

1. 把今天的聊天室代码，自己实现一遍