简易微信 server 开发文档

Liyh

第一部分 实现原理

1.1 概述

本次的简易微信 server 端采用 Tcp 作为唯一的运输层协议，HTTP 为唯一的应用层协议，通过与客户端交换 json 格式数据文档的方式来进行数据交换。底层使用 sqlite 数据库储存用户信息。

服务器搭建在本地，监听 2333 端口。在本地使用服务器需要连接本地 ip，即 127.0.0.1。

服务器程序会一直监听发起请求的客户端，并与之保持连接状态，直到客户端主动断开连接或者客户端请求离线接口。在此期间，服务器将持续接受客户端请求并返回响应信息。

1.2 数据库格式

底层数据库共储存两张表，一张储存用户信息，另一张储存用户好友关系。

```
CREATE TABLE IF NOT EXISTS UserData(
        userName VARCHAR(20) PRIMARY KEY,
        ID INT,
        passWord VARCHAR(20),
        isOnline INT,
        token INT
)
```

显而易见。UserData 表储存 userName(用户名)，ID(用户 ID)，passWord（密码）以及 isOnline(是否在线)和 token 信息。其中 userName 为主键，这意味着用户信息是不可重复的。但实际上 ID 也具有这样的主键性质，它的引入目的是用于快速索引 socket。

好友关系表：

```
CREATE TABLE IF NOT EXISTS UserFriends(
        userName VARCHAR(20),
        friendName VARCHAR(20),
```

```
        ID INT PRIMARY KEY"
)
```

好友关系表中储存用户名和它的一个好友的用户名。显然一对好友会对应两个元组。这里的主键 ID 没有实际意义。

服务器的数据库文件实体为同一目录下的 userData.db。当需要完全重置服务器时可以直接删除这个文件。

任何时候服务器重启都会使所有用户强制登出。

1.3 网络请求

网络方面，服务器只接受 POST 请求，对于其余的请求不接受处理。服务器总是会返回 200 OK 的状态码，而错误信息会包含在附带在报文中的 json 里。

第二部分 接口规范

2.0 报头

服务器只接受 POST，可以使用下面格式的 HTTP 报头：

```
POST /%1 HTTP/1.1\r\n
Content-Type: application/json;charset=utf-8\r\n
Connection: keep-alive\r\n
Content-Length: %2\r\n\r\n
```

这里的%1 和%2 按照请求的数据而定。

2.1 注册

注册的请求接口为：IP/register

请求 json 格式：

```
{
    “username”:”name”,
    “pwd”:”password”
}
```

这里的 userName 为需要注册的用户名。服务器会按照下面的格式返回数据。

```
{
    “code”:0,
    “msg”:”msg”,
    “type”:”register”
}
```

服务器所有的返回数据都会至少包含这三个参数。其中 code 表示基本的返回码，msg 为返回信息，type 指定这是属于哪一种请求的返回类型。在 register 请求中，type 总是为 register。

返回的信息如下表所示，我们用粗体字表示成功的请求：

code	msg
0	**注册成功**
-1	重复的用户名

2.2 登录

注册的请求接口为：IP/login。

请求 json 格式：

```
{
    “username”:”name”,
    “pwd”:”password”
}
```

返回数据格式：

```
{
    “code”:0,
    “msg”:”msg”,
    “type”:”login”,
    “token”:12345
}
```

这里的 token 出于网络安全方面的考量。服务器除了注册之外的所有接口都必须正确的 token，请求登录接口是获取 token 的唯一方式。这里的 token 在 windows 平台上是不超过 65535 的 int 值。

返回信息表：

code	msg
0	**OK**
2	**已经登录了**
-1	无此用户
1	密码错误

code 为 0 或者 2 时返回的 token 有效，其余为随机数，没有实际意义。

2.3 获取好友列表

请求接口：IP/friends

请求 json 格式：

```
{
    “username”:”name”,
    “token”:12345
}
```

返回数据格式：

```
{
    “code”:0,
    “msg”:”msg”,
    “type”:” friends”,
    “friends”:[“friend1”,”friend2”]
}
```

friends 是一个储存字符串类型的 json 数组。

返回信息表:

code	msg
0	**OK**
-1	用户不存在
1	token 无效

2.4 添加好友

请求接口：IP/ makefriend

请求 json 格式：

```
{
    “username”:”name”,
    “token”:12345,
    “who”:”who”
}
```

who 为好友的用户名。

返回数据格式：

```
{
    “code”:0,
    “msg”:”msg”,
    “type”:” makefriend”,
}
```

返回信息表：

code	msg
0	**请求发送成功**
-1	用户不存在
1	token 无效
2	已经是好友了
3	目标用户不存在
-2	目标用户不在线

服务器不支持向离线用户发送请求。

2.5 好友请求回复

请求接口：IP/ result

请求 json 格式：

```
{
    “from”:”A”,
    “to”:”B”,
    “token”:12345,
    “isOk”:1
}
```

from 通常就是当前的用户名，to 是回复对象。isOk 是一个整型，0 表示拒绝，1 表示同意。

返回数据格式：

```
{
```

“code”:0,
“msg”:”msg”,
“type”:” result”,
}

返回信息表：

code	msg
0	**发送结果成功**
-1	用户不存在
1	token 无效
2	已经是好友了
3	目标用户不存在
-2	目标用户不在线

服务器不支持对离线用户发送请求。

2.6 删除好友

请求接口：IP/delete

请求 json 格式：

{
“username”:”name”,
“token”:12345,
“who”:”who”
}

返回数据格式：

{
“code”:0,
“msg”:”msg”,
“type”:” delete”,
}

返回信息表：

code	msg
0	**删除好友成功**
1	token 无效
2	没有这个好友
-1	没有这个用户

2.7 [服务器]好友请求

{
“code”:0,
“msg”:”msg”,
“type”:” friendrequest”,
“from”:”from”
}

当有用户发送好友请求时，服务器主动发出该 json 数据，其中 from 标识了好友请求的来源用户名。

2.8 [服务器]好友请求结果

{
“code”:0,
“msg”:”msg”,
“type”:” makefriendres”
}

当好友请求对象发送 result 请求时，服务器会同时向发送方发送结果。该 json 结果由服务器主动发送。其中 code 表示对方是否同意（返回 0/1），msg 标识对方的用户名。

2.9 发送消息

请求接口：IP/ sendmsg
发送数据格式：

{
“userName”:”name”,
“token”:12345,
“to”:”to”,
“msg”:”data”
}

其中 to 表示发送对象用户名，msg 为消息本体。
返回数据格式：

{
“code”:0,
“msg”:”msg”,
“type”:” sendmsgres”
}

返回信息表：

code	msg
0	**发送成功**
-1	没有这个用户
1	token 无效
2	没有这个好友
-2	当前用户不在线

服务器不支持向离线用户发送消息。

2.10 [服务器]接受消息

{
“code”:0,
“msg”:”msg”,
“type”:” msg”,
“from”:”from”
}

当接受到消息时，该 json 由服务器主动发出。其中 code 总是 0，from 表示

发送方用户名，msg 为消息本体。

2.11 离线

请求接口：IP/offline

发送数据格式：

```
{
    “userName”:”name”,
    “token”:12345
}
```

返回数据格式：

```
{
    “code”:0,
    “msg”:”msg”,
    “type”:” offline”
}
```

返回信息表：

code	msg
0	**OK**
1	已经离线了
2	token 无效
-1	用户不存在

请求完成后，无论返回什么结果，连接都会断开。