|  |  |
| --- | --- |
| 文档名称 | web\_demo设计文档 |
| 文档状态 | [初稿] |
| 版 本 | V1.0 |
| 审 核 |  |
| 密 级 | 内部有限公开 |

web\_demo设计文档



编写日期：2016年4月11日

v1.0

文档修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| **当前版本** | **修订时间** | **修订人** | **备注** |
| V1.0 | 2016年4月11日 | 王兴宇 | 建稿 |
| V1.0 | 2016年4月15日 | 王兴宇 | 修改类图等排版以及部分内容 |
|  |  |  |  |

编写说明：

编写本文档的主要目的在于提供web\_demo的设计说明；

本文档作为web\_demo 系统的附属文档；

本文档作为产品维护依据之一。

目录

[一、 web\_demo 系统介绍 4](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403260)

[1. 简介 4](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403261)

[2. 基本功能 4](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403262)

[3. 信息安全 5](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403263)

[二、 系统概述 5](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403264)

[1. 系统说明 5](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403265)

[2. 系统任务 5](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403266)

[3. 系统目标 6](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403267)

[4. 运行环境 6](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403268)

[三、 详细设计 6](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403269)

[1. 功能结构设计 6](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403270)

[2. 系统基本流程设计 7](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403271)

[3. 通信流程设计与webSocket原理分析 11](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403272)

[4. 数据结构设计 12](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403273)

[5. 接口设计 14](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403274)

[6. 系统出错处理 16](file:///C:\Users\ADMINI~1\AppData\Local\Temp\Rar$DIa0.786\web_demo设计文档.docx#_Toc448403275)

## web\_demo 系统介绍

### 简介

web\_demo 是一基于web端实现的聊天通信会话系统。

在该系统中，用户由管理员进行账号分配，然后用户使用该账号进行登录。在系统主页面上，普通用户可以进行个人信息的修改，包括基本信息、密码、头像删除更新；同时用户可以点击聊天室菜单进入聊天室，可以与当前在线用户进行群聊，也可以点击在线列表中的私聊按钮选择用户聊天；会话内容包括文字，图片及文件的交流。

管理员除包含普通用户的功能外，还拥有管理普通用户的菜单接口，包括用户管理、分组管理。用户管理包括创建用户，对所有用户的编辑，查询及删除操作；分组管理包括创建分组，对用户进行分组。

### 基本功能

普通用户的基本功能包含如下：

1. 登录系统；
2. 查看个人基本信息；
3. 编辑个人基本信息；
4. 上传头像、更新头像；
5. 进入聊天室；
6. 查看在线用户列表；
7. 与所有在线用户群聊；
8. 与单个在线用户私聊；

管理员的功能包含如下：

1. 包含普通用户基本功能下的所有功能；
2. 用户管理中创建新用户；
3. 用户管理中查看用户列表，修改用户信息，删除用户操作；
4. 分组管理中创建分组；
5. 分组管理中查看分组详情及组内用户列表；
6. 将用户加入组，删除分组操作；

有关具体的功能结构设计将在后文详细设计中提到。

### 信息安全

信息安全是现今系统不可不提的一个重要方面。在本系统设计中，也需要考虑到系统的信息安全。因此在设计本系统中，主要考虑到信息安全的地方如下：

1. 在数据库用户登录密码的加密；
2. 用户登录时防止SQL注入；
3. 防止机器程序恶意攻击，加入验证码；
4. 过滤器设置；
5. 请求方式的限制。

## 系统概述

### 系统说明

web\_demo 是一款基于web形式的即时聊天系统，在该系统中，用户不仅可以对自己的基本信息进行管理，还可以进行一对一聊天以及群聊。群聊即是说用户在会话中发送信息，所有在线成员均可以看到该信息，同时可以参与到该会话当中。

系统针对管理员附属了一些对系统的管理功能，比如说：管理员对组的管理，用户所属组，用户角色的管理，用户管理等常规操作。同样，该管理员具备有普通用户相同的参与聊天的功能。

### 系统任务

在该系统中，普通用户主要实现用户基本信息管理，进行一对一聊天通信功能；管理员主要实现基本通信功能之外的用户管理，组管理，用户组/角色管理等基本需求。

技术上要求采用SpringMVC进行处理。

### 系统目标

完成任务中要求实现的功能，同时考虑系统的可用性，设计出合理的系统架构，对系统的安全性做出一定的规划处理。

### 运行环境

JDK1.7；Tomcat 7; MySQL5.6.5

## 详细设计

### 功能结构设计

系统功能设计的区分主要是根据系统的两个重要角色进行的。分别为普通用户和管理员用户，因此在进行系统功能设计时，主要的区分点就在这里。

首先保证管理员在拥有普通用户角色的基础上还拥有其独有的管理功能，在本系统的功能设计中，根据需求分析出管理员的独有功能包含：用户管理，组管理等核心功能。而对于普通用户而言，需要有基本信息管理，通信等两个大方面的设计。

在大方面下，在进行细分，然后完成系统的整体要求。详细的功能设计结构图参见图2.1.1所示。在该图中，可以看到管理员对于普通用户多了对系统的用户管理，组管理以及用户组/角色管理等。

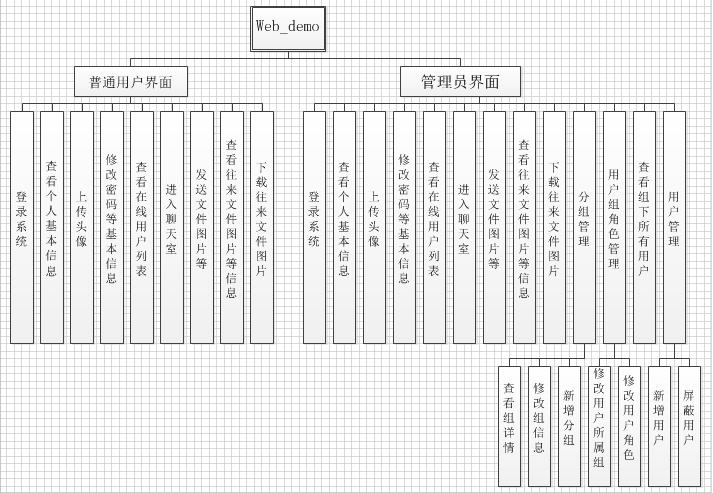


图2.1.1

### 系统基本流程设计

系统登录采用管理员和普通用户相同的页面，但是在后端查询到该用户为有效用户，且为管理员角色时，加载另外的功能。即时说：普通用户和管理员登录页面相同，并且登录成功后的主页面也是相同的，但是在主页面上显示的功能点是有区别的。

在主页面上，普通用户和管理员均可以通过显示在页面上的不同的功能模块进行操作，用户可以点击主页面左侧的“在线用户列表”查询具体用户列表信息，然后选择用户进行会话，完成会话后，只需要点击“返回主菜单”即可。当用户没有操作之后，可以点击主页面左侧的“退出系统”完成当前流程操作。

系统基本流程机构如图3.2.1所示。在本图中由于篇幅原因，未将功能子模块流程列入其中，但是可以从功能模块中看到关于子模块的相关信息。

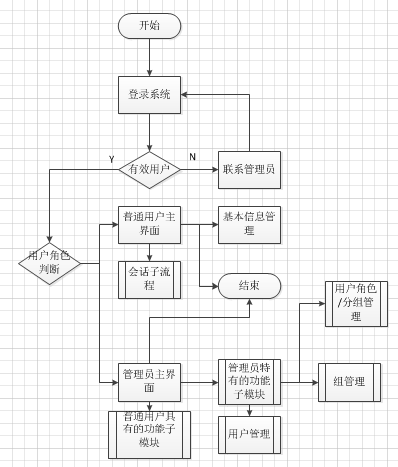


图3.2.1

在用户操作过程中，其状态也会产生相应的变化，如图3.2.2所示，在该系统中，用户的基本状态通过登录系统变为已登录状态，在已登录状态下，用户可以与在线用户进行聊天，其状态转变为相应的会话状态中，当完成会话状态，并且已经完成系统的基本操作，准备退出系统，此时系统进入退出处理，用户即将完成退出系统的操作中，之后，用户的基本状态转变为已退出状态。此时，用户如果还想继续进入会话，则必须进行重新登录方可以继续使用。

在用户刚被管理员创建，第一次使用该账号登录系统时，页面会自动跳转到提示用户修改初始密码的页面进行修改密码的操作。待操作完成后，页面会让用户使用新密码重新进行登录。然后进入系统主页。

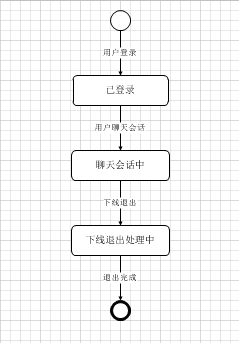


图3.2.2

为了更加清晰的阐述系统的整个流程设计，如图3.2.3所示为系统的时序图，在该图中，可以看到系统的整个序列结构。系统用户输入必备信息之后，进入主页面，然后根据用户选择进行不同的功能子模块，包括基本信息的管理，以及通信会话等操作。而管理员则是在此基础上又增加了一定的对系统的基本管理功能。而图3.2.3只是作为系统核心点：聊天通信的序列设计。

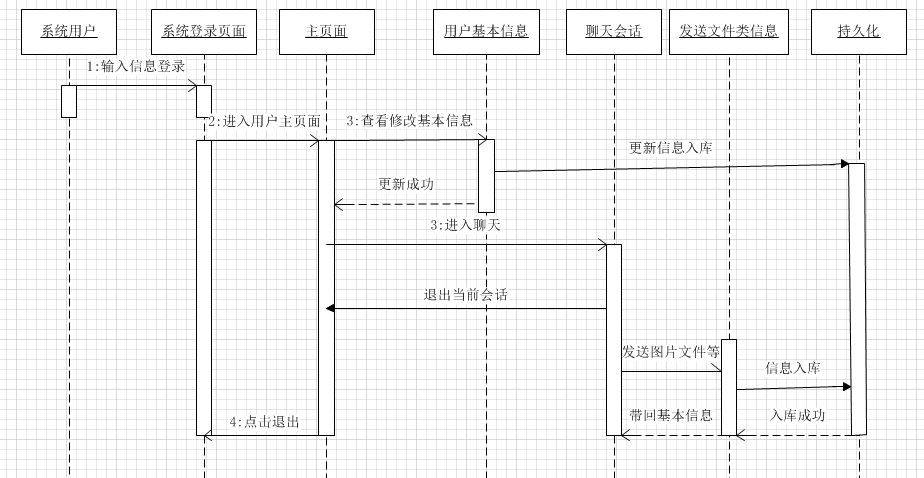


图3.2.3

系统类图设计如图3.2.4所示，BaseController作为控制层的基类，获取当前session以及当前会话的用户对象。InfoUserController、InfoGroupController、InfoMsgController继承自BaseController，这样，他们可以直接获取当前会话session和用户基本信息。BaseDAO作为数据访问的基类，内部包含sessionFactory和一些最基本的通用方法，实现IBaseDAo接口。对于其他接口，则是继承BaseDAO，实现自己定义的接口中的方法。

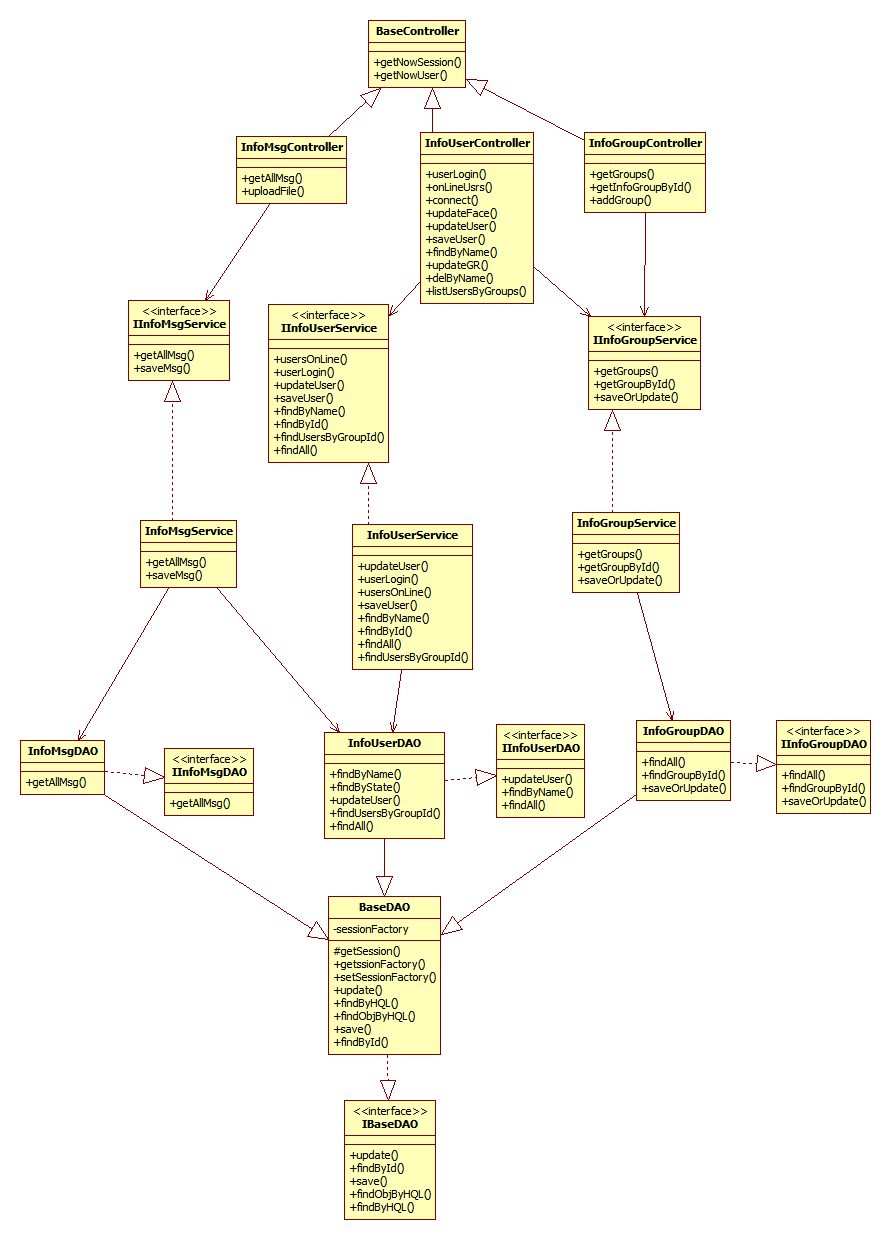


图3.2.4

### 通信流程设计与WebSocket原理分析

WebSocket约定了一个通信的规范，通过一个握手的机制，客户端和服务器之间能建立一个类似 TCP 的连接，从而方便它们之间的通信。实现了浏览器与服务器全双工通信，能更好的节省服务器资源和带宽并达到实时通讯，同 HTTP 一样通过 TCP 来传输数据，但是它和 HTTP 最大不同是：

1. WebSocket 是一种双向通信协议，在建立连接后，WebSocket 服务器和 Browser/Client Agent 都能主动的向对方发送或接收数据，就像 Socket 一样；
2. WebSocket 需要类似 TCP 的客户端和服务器端通过握手连接，连接成功后才能相互通信。

通信流程与原理分析如图3.3.1所示。为了建立一个 WebSocket 连接，客户端浏览器首先要向服务器发起一个 HTTP 请求，这个请求和通常的 HTTP 请求不同，包含了一些附加头信息，其中附加头信息“Upgrade: WebSocket”表明这是一个申请协议升级的 HTTP 请求，服务器端解析这些附加的头信息然后产生应答信息返回给客户端，客户端和服务器端的 WebSocket 连接就建立起来了，双方就可以通过这个连接通道自由的传递信息，并且这个连接会持续存在直到客户端或者服务器端的某一方主动的关闭连接。

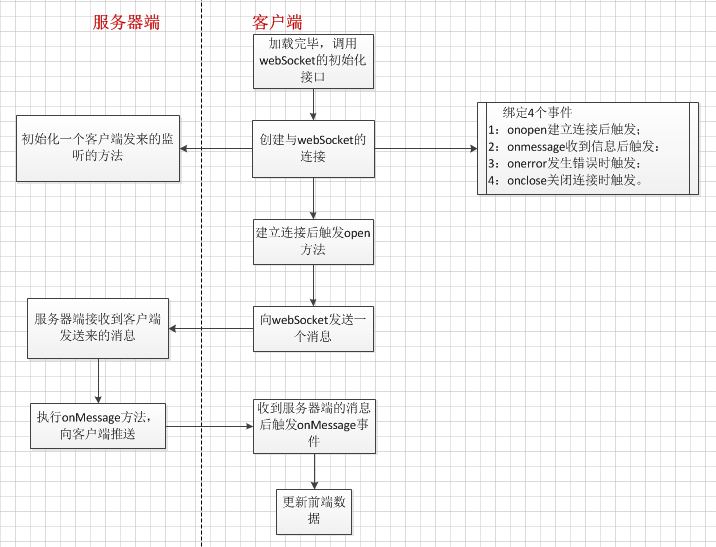


图3.3.1

### 数据结构设计

考虑到系统的核心在于通信聊天，数据库设计只是作为系统的一种辅助支持。可以根据系统的需求要求以及功能结构设计抽象中系统的实体类模型，如图3.3.1所示。

在设计上，将普通用户和管理员用户保存到同一张表中，只是根据其中的字段来判断其不同的角色等级，然后加载不同的功能模块。

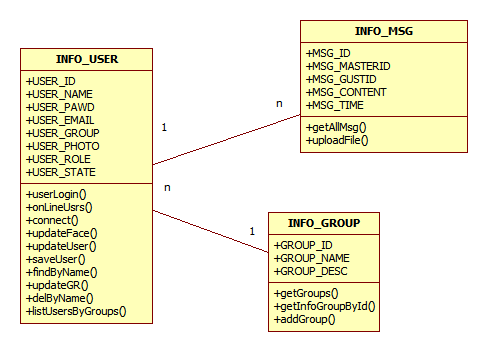


图3.3.1

表结构设计如下:

用户基本信息表，info\_user：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **主键** | **字段类型** | **长度** | **为空** | **注释** |
| USER\_ID | 是 | int | 11 | NOT NULL | 用户信息标识ID |
| USER\_NAME | 否 | varchar | 45 | NOT NULL | 用户登录名称 |
| USER\_PAWD | 否 | varchar | 45 | NOT NULL | 用户登录密码 |
| USER\_EMAIL | 否 | varchar | 45 | NOT NULL | 用户邮箱 |
| USER\_PHOTO | 否 | varchar | 45 | NOT NULL | 用户头像 |
| USER\_GROUP | 否 | int | 11 | NOT NULL | 用户所属分组 |
| USER\_ROLE | 否 | int | 11 | NOT NULL | 用户角色 |
| USER\_STATE | 否 | int | 11 | NOT NULL | 用户状态 |

信息发送表，info\_msg：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **主键** | **字段类型** | **长度** | **为空** | **注释** |
| MSG\_ID | 是 | int | 11 | NOT NULL | 发送信息标识ID |
| MSG\_MASTERID | 否 | int | 11 | NOT NULL | 信息发送者ID |
| MSG\_GUSTID | 否 | int | 11 | NOT NULL | 信息接收者ID |
| MSG\_CONTENT | 否 | varchar | 45 | NOT NULL | 信息内容 |
| MSG\_TIME | 否 | int | 10 | NOT NULL | 信息发送时间戳 |

用户组信息表，info\_group：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **字段名称** | **主键** | **字段类型** | **长度** | **为空** | **注释** |
| GROUP\_ID | 是 | int | 11 | NOT NULL | 用户分组ID |
| GROUP\_NAME | 否 | varchar | 45 | NOT NULL | 用户分组名称 |
| GROUP\_DESC | 否 | varchar | 45 | NULL | 用户分组描述 |

### 接口设计

在本系统中，主要涉及到三个接口，分别是用户信息接口，发送信息接口，组信息接口。下面将逐个进行阐述。

1. 用户信息接口：
2. 方法名称：查询所有在线用户

返回类型：集合List<InfoUser>

**public** List<InfoUser> usersOnLine();

1. 方法名称：用户登录

参数名称：用户名，用户密码

返回类型：对象InfoUser

**public** InfoUser userLogin(String userName, String userPwd);

1. 方法名称：更新用户基本信息

参数名称：InfoUser对象

返回类型：InfoUser对象

**public** InfoUser updateUser(InfoUser infoUser);

1. 方法名称：添加保存用户

参数名称：InfoUser对象

返回类型：布尔值

**public** **boolean** saveUser(InfoUser infoUser);

1. 方法名称：根据用户名称查询用户

参数名称：用户名称

返回类型：InfoUser对象

**public** InfoUser findByName(String name);

1. 方法名称：根据用户ID查询用户

参数名称：用户ID

返回类型：InfoUser对象

**public** InfoUser findById(**int** id);

1. 方法名称：查询组下所有用户

参数名称：组ID

返回类型：集合List<GrpUsersVO>

**public** List<GrpUsersVO> findUsersByGroupId(**int** groupId);

1. 方法名称：查询所有用户

返回类型：集合List<InfoUser>

**public** List<InfoUser> findAll();

1. 信息接口如下：
2. 方法名称：查询往来所有信息记录

返回类型：集合List<MsgVO>

**public** List<MsgVO> getAllMsg();

1. 方法名称：添加保存信息记录

参数名称：InfoMsg对象

返回类型：布尔值

**public** **boolean** saveMsg(InfoMsg infoMsg);

1. 组信息接口如下：
2. 方法名称：查询所有组信息

返回类型：集合List<InfoGroup>

**public** List<InfoGroup> getGroups();

1. 方法名称：根据组ID查询组信息

参数名称：组ID

返回类型：集合List< InfoGroup>

**public** InfoGroup getGroupById(**int** groupId);

1. 方法名称：保存或更新组信息

参数名称：InfoGroup对象

返回类型：布尔值

**public** **boolean** saveOrUpdate(InfoGroup infoGroup);

### 系统出错处理

系统采用Log4j记录并保存日志信息，包含INFO和ERROR信息。在用户操作系统过程中，会将一些信息保存到指定的日志文件中以便于维护系统，同时，在系统设计过程中也加入了一定的过滤设置，只有通过系统过滤器的用户才能进入系统。系统中出现的异常也会被纳入到日志记录文件中。

在该系统中也加入了对404、405页面的处理，当访问到系统中不存在的页面或是不支持该请求方式时，系统进行自动处理并且提示用户访问的页面不存在或不支持用户的当前请求方式。

断链处理采取的解决方案是客户端浏览器不断的post其用户ID和当前时间戳到后台服务器，然后服务器更新其time值，由于后台有一个定时任务在不断的进行探测，当客户端传过来的time值超过预设的默认值时，则判断失效，更新该客户ID的状态。