# 城见详细设计

[城见详细设计](#_Toc16038)

[1. 概述](#_Toc28490)

[1.1. 系统简述](#_Toc5445)

[1.2. 软件设计目标](#_Toc3725)

[1.3. 参考资料](#_Toc26683)

[1.4. 修订版本记录](#_Toc13578)

[2. 术语表](#_Toc453)

[3.设计概述](#_Toc17)

[3.1 复用计划](#_Toc7656)

[3.2. 系统接口设计](#_Toc26525)

[3.3 对象模型设计](#_Toc6273)

[3.4 动态](#_Toc13502)

[3.5 系统非功能性设计](#_Toc2164)

## 概述

### 1.1. 系统简述

　　我们的系统名为：城见，是一个匿名网页聊天工具。使用我们的软件，用户可以选择单独匹配陌生人进行匿名聊天，也可以选择对某一近期发生的大学城热点进行群聊。

### 1.2. 软件设计目标

　 软件设计目标能够为大学城学生提供一个匿名聊天环境，主要功能包括以下方面：

①. 匹配任意陌生人聊天

②. 向聊天对象推送笑话、图片

③. 浏览大学城近期发生的热点

④. 在某一热点进行群聊

⑤. 提供单聊消息下载功能

对于非功能性需求，我们有以下考虑：

　　①. 鲁棒性 - 系统应该尽可能的提供可靠服务，控制服务器宕机次数

②. 易用性 - 系统应该尽可能简单易用。主要要求包括下面：

　　　　　1. 首页提供单人匹配功能，减少用户点击次数

2. 热点展示应该控制展现数量，减少信息拥塞

3. 单聊或者群聊要保证消息准确无误的传送，对于无法正确传送的消息，应该可以友好的提示消息。

　　③. 安全性 - 系统应该保证用户消息不泄露，如果用户选择不下载消息记录，那么应该从系统中及时删除

　　④. 可扩展性 - 系统应该合理设计接口，使得能很好的修改或者添加功能

### 1.3. 参考资料

需求上，我们参考了国外著名的匿名聊天网站 Omegle 设计，在此基础上提出了我们自己的创新。技术上，我们主要参考 python 的 tornado 框架文档以及 前端Bootstrap 框架文档

### 1.4. 修订版本记录

我们计划在软件1.0 版本提供单聊功能和新闻热点功能，在软件 2.0 版本提供热点群聊和笑话推送功能

## 术语表

|  |  |
| --- | --- |
| 术语 | 解释 |
| 笑话推送 | 系统提供笑话推送功能。单人聊天时，聊天一方可以向另一方推送合适的笑话，增加聊天趣味性。 |

其他术语请参考需求文档。

## 3.设计概述

### 3.1 复用计划

后台框架以 tornado 为主，复用 tornado 的 RequestHandler、websocketHandler、Application等类。爬虫复用自定义基类 baseSpider、Table以及 selenium 库。前端复用 bootstrap 框架。

### 3.2. 系统接口设计

#### 3.2.2.内部接口：

**1. 聊天部分**

**1.1 返回用户ID**

url

119.29.161.184:8000?rand=xxx&sex=xxx&nameIndex=xxx

sex : 0为male 1为female

response

{

"status":1,

"id":"xsijfa-r21ja2j-asoj2aa-adh2saj"

}

**1.2 聊天**

url

119.29.161.184:8000/chat

params

id : string(用接口1.1返回的ID)

request

{

"type":"xxxx",

"text":"xxxxxxx",

"id":"xxxxxx-xxxx-xxxxx"

}

response

{

"res":{

"sender":"xxx",

"type":"xxx",

"content":"",

"time":"2016-10-16 13:45:20"

},

"status":1

}

该接口使用websocket协议

**1.3 返回昵称**

url

119.29.161.184:8000/name?sex=xxx

response

{

res: {

7: "郭芙",

11: "李莫愁",

16: "李沅芷",

24: "方怡",

28: "何红药",

31: "安小惠",

37: "杨不悔",

41: "袁紫衣"

},

status: 1

}

sex : 0为male 1为female

**2. 热点新闻部分**

**2.1 返回热点新闻**

url

119.29.161.184:8000?page=xxx

(page表示第几页，从 1 开始计数，每页返回十条热点新闻,按照 pv 进行降序排列)

response

success:

{

"status": 1,

"data": [{"id":xxx,"url":xxx,"title":xxx,"visit\_cnt":xxx},...]

fail:

{"status": 0, "data": "page exceed limits"}

**2.2 热点新闻访问量更新**

**url**

119.29.161.184:8000/news (post)

数据格式：

{

"id":XXX

}

**response** success:

{

"status":1

}

fail:

{

"status":0,

"data":"XXX" (给出失败原因)

}

**2.3 返回笑话**

**url**

119.29.161.184:8000/joke?num=XXX

(返回 num 个笑话)

**response** success:

{

"status":1,

"data":[

{"id":"XXX",

"content":"XXX"

},

{

...

}

...

]

}

fail:

{

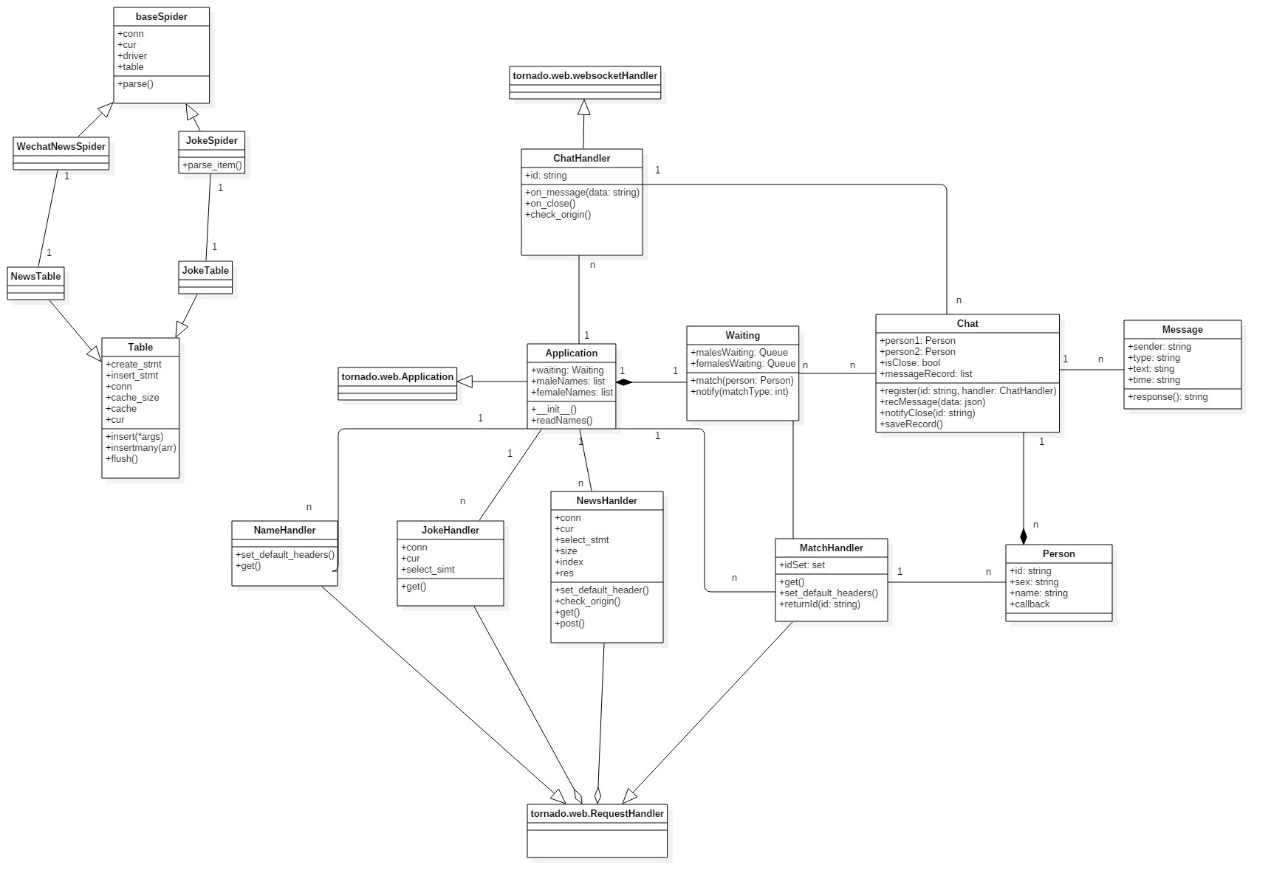
"status":0,

"data":"XXX"

}

### 3.3 对象模型设计

对象模型：



对象描述：

1. Application

属性：

1. waiting : Waiting
2. maleNames：list

存储男的名字的一个list

1. femaleNames：list

存储女的名字的一个list

方法：

1. \_\_init\_\_()

重载构造函数

1. readNames()

读取名字存到maleNames和femaleNames

1. NameHandler

方法：

1. set\_default\_headers()

设置返回的header，解决跨域问题

1. get()

响应对其相应的url的get请求

1. MatchHandler

属性：

1. idSet : set

存储已分配出去的id，避免重复

方法：

1. set\_default\_headers()

设置返回的header，解决跨域问题

1. get()

响应对其相应的url的get请求

1. returnId(id:string)

返回ID给前端

1. ChatHandler

属性：

1. id : string

用户在MatchHandler的returnId中返回的id

方法：

1. on\_message(data: string)

实现websocket的on\_message方法，data为在连接过程中前端向服务器发送的消息

1. on\_close()

实现websocket的on\_close()方法，这是当连接断开时执行的函数。

1. check\_origin()

解决跨域问题

1. Waiting

属性:

1. malesWaiting: Queue

存储未匹配的女生的队列

1. femalesWaiting: Queue

存储未匹配的男生的队列

方法：

1. match(person:Person)

当一个新的用户进入时，若已有正在等待的用户，则进行匹配，若没有则将用户加入队列进行等待

1. notify(matchType: int)

用来通知被匹配的用户，matchType表示匹配的类型。

1. Chat

属性：

1. person1: Person

用户1

1. person2 : Person

用户2

1. isClose : bool

表示是否有人退出聊天

1. messageRecord : list

存储聊天记录的一个list

1. Person

属性：

1. Id : string

用户的id

1. Sex: string

用户的性别

1. Name : string

用户的名字

1. Callback

该用户建立连接对应的MatchHandler实例中的returnId这个回调函数

1. Message

属性：

1. Sender : string

消息发送者的名字

1. Type : string

发送的消息的类型

1. Text : string

发送的消息的内容

1. Time : string

发送消息的时间

方法：

1. response(): string

返回给另一个用户的消息

1. Table

属性：

1. conn: pymysql.cnnections.Connection

与数据库的连接

1. cur: pymysql.cursors.Cursor

游标

1. cache\_size: int

缓存大小

1. cache: list

缓存

1. spider\_name: string

爬虫名字

1. create\_stmt: string

创建数据库表语句

1. insert\_stmt: string

插入数据库表语句

方法：

(1). Insert(\*args): void

向缓存或者数据库插入记录

(2). Insertmany(arr): void

向数据库插入多条记录

(3). Flush(): void

将缓存写入数据库

1. baseSpider

属性：

1. conn: pymysql.cnnections.Connection

与数据库的连接

(2). cur: pymysql.cursors.Cursor

游标

(3). driver: selenium.webdriver

浏览器内核

(4). table: Table

table 实例

方法：

(1). Parse(OA\_name): void

给定公众号名字，爬取该公众号最近十条文章

1. NewsHandler

属性：

1. conn: pymysql.cnnections.Connection

与数据库的连接

1. cur: pymysql.cursors.Cursor

游标

1. select\_stmt: string

选择语句

1. size: int

数据库记录大小

1. index: tuple(int,int)

一个tuple,用来读取长度为10的文章记录

1. res: list[list[string]]

数据库文章

方法：

1. set\_default\_header()

设置允许跨域

1. check\_origin

检查请求来源

1. get()

响应客户端发起的 get 请求

1. post()

响应客户端发起的 post 请求

1. JokeHandler

属性：

1. conn: pymysql.cnnections.Connection

与数据库的连接

1. cur: pymysql.cursors.Cursor

游标

1. select\_stmt: string

选择语句

方法：

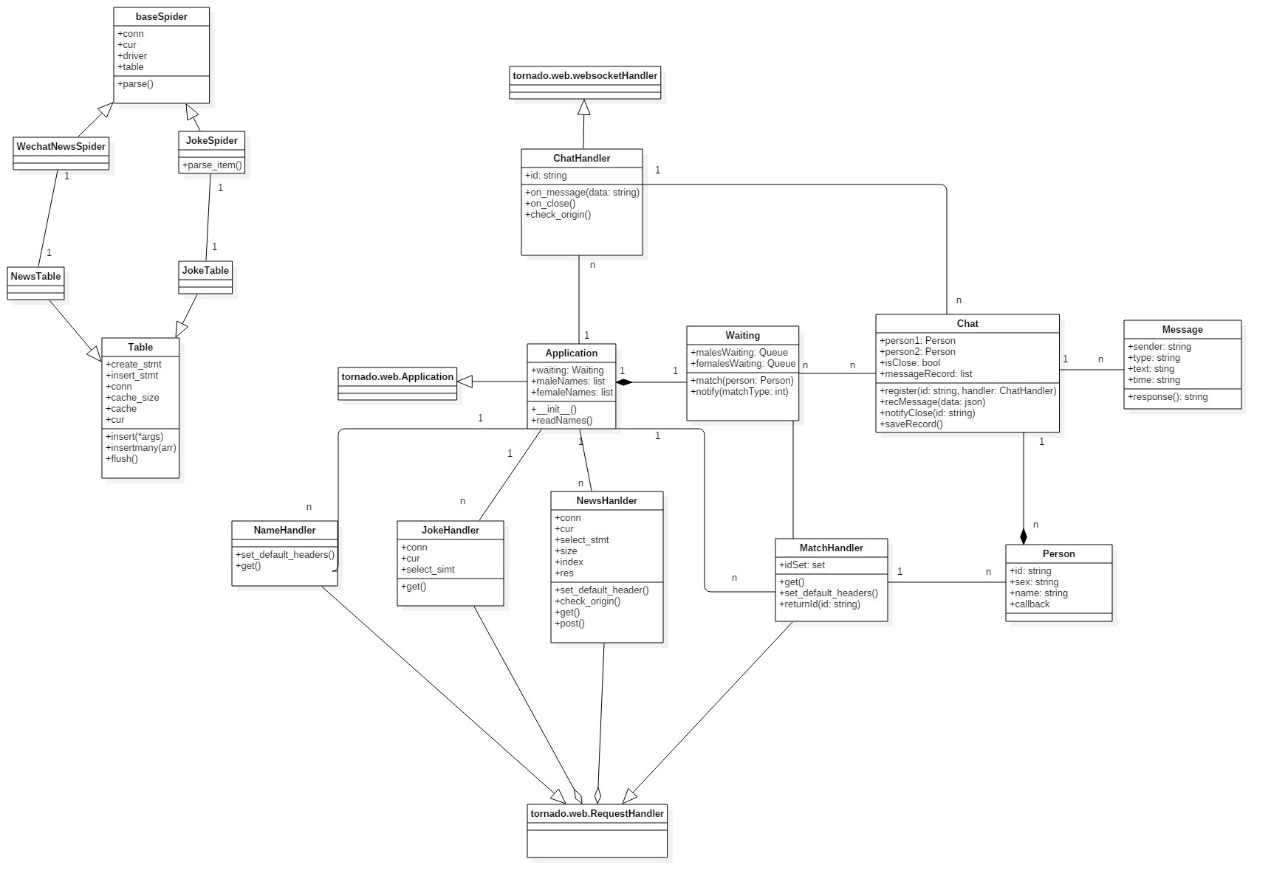
1. get()

响应客户端的 get 请求

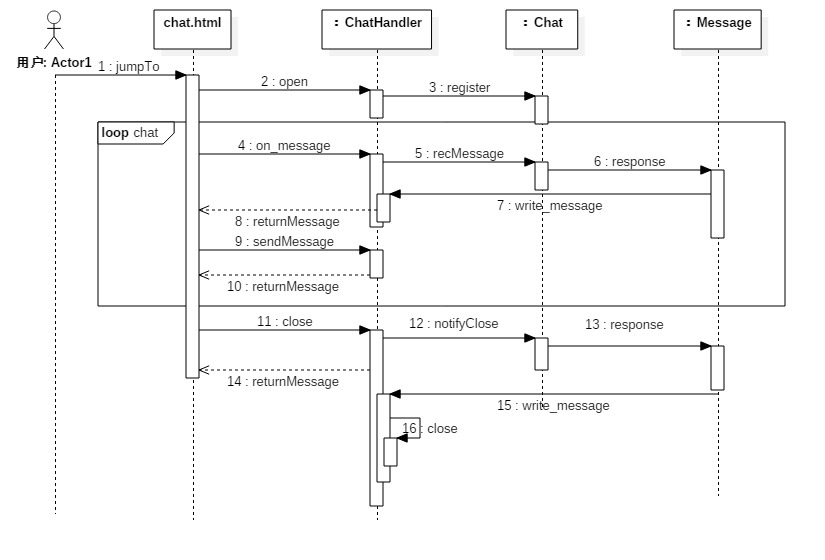
### 3.4 动态

系统顺序图：

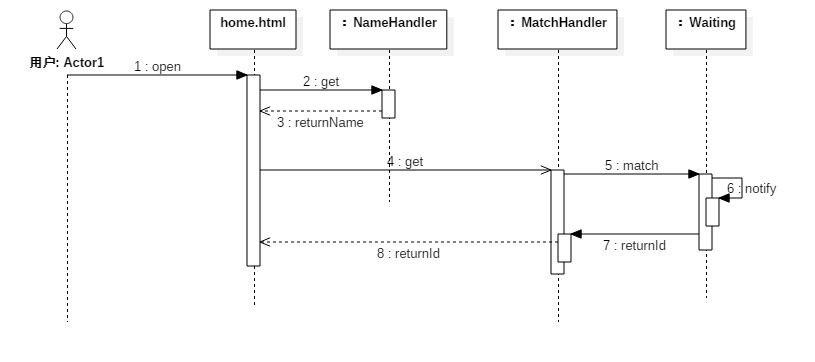
类图：



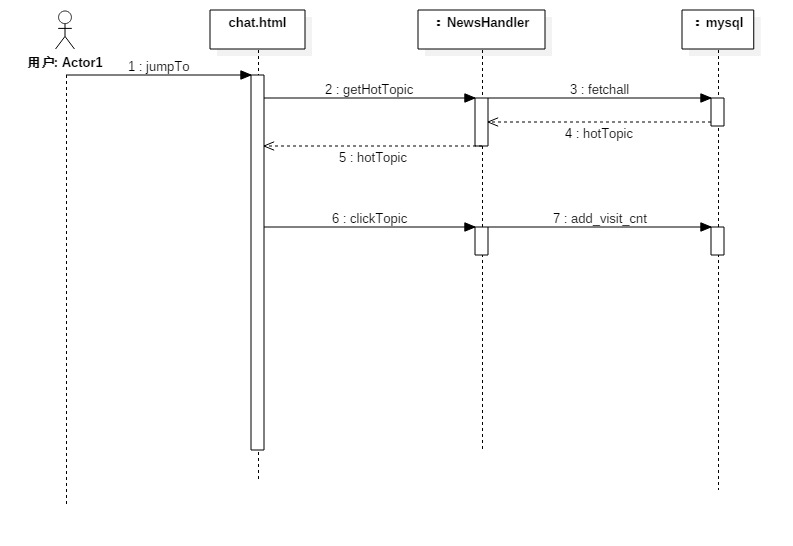
聊天时序图：



匹配时序图：



热点时序图：



### 3.5 系统非功能性设计

系统非功能设计：

（1）性能设计

用户在打开网页时响应时间不大于1秒

聊天的发送时间和接收时间间隔应不大于2秒

聊天记录存储：能够存储所有用户24小时内的聊天记录

用户临时ID存储：能存储所有用户在使用网页时分配的ID数据

（2）安全性需求

网页运行时应该有日志记录

用户管理以及权限管理应保证数据库和网页的安全性

充分利用主机、网络以及数据库管理系统提供的安全保障机制，确保系统数据的绝对安全

（3）易用性需求

未经训练的用户都可以轻易地掌握网页的操作

所有按钮都设置在显眼的位置

提供响应式网页设计，确保不同设备浏览器都可以有好的用户体验

前端界面风格统一、美观、人性化，确保网页用户友好

（4）可用性需求

连续7x24小时不间断工作

每周故障中断时间不应该超过3小时

（5）可靠性需求

网页必须能在主流浏览器和电脑、手机设备上稳定工作

确保数据的一致性

开发中充分利用数据库的安全机制和错误恢复机制，保证数据准确性和完整性

系统应能一天24小时连续不间断工作

出现故障时，能及时检修并恢复

（6）可拓展性需求

网页必须保证当聊天人数出现大幅增加时，能顺利扩容

当网页要添加新功能时，必须使代价尽可能小

（7）可伸缩性需求

部分功能的故障不会影响主要功能的进行

（8）运行环境约束

在个人电脑、安卓和IOS手机上各种网页浏览器中都能运行