好像是文档什么的part2

**1. Introduction**

虽然代码和文档都写的很乱，但还是描述一下。主要的部分有，一是model.py中的数据库的内容，即关于用户以及信件的数据存储部分，二是main/view\*.py中的route部分，即向用户展示的页面以及interaction等等。

这一部分主要描述的是实现的细节，不过是在测试前写的，之后会有一些简单的测试，既然写的这么乱，目测各种bug。

**2. Model**

2.0 logs

描述：记录系统数据（每一项为magic匹配记录）

字段：略（详见models.py以及magic-step4）

创建：/magic-step4

查询：/logs

修改：无

Todo：增加系统管理相关

2.1 users

描述：用户表

字段：【1】（ID段）id，username，passwd\_hash；【2】（info段）penname，gender，age，self\_intro；【3】（preference段）who\_can\_see（age/intro），matching\_gender

创建：/register（with RegisterForm）

查询：<ALL>；另外who\_can\_see用于/penpals/<uid>，matching\_gender可以用于匹配（目前未实现）

修改：/settings（with InfoForm）

条件-内部：id以及username都是唯一的

Todo：增加更多的用户识别，信息以及设定

2.2 letters\*

描述：信件相关，分为三种：普通信件，temp信件，system信件。这三种类似继承关系，即所有的system信件都是temp信件并在letter-temp表中有，所有的temp信件同样有普通信件的性质并且在letter表中有。

2.2.1 letters

描述：信件表（普通信件字段）

字段：【1】（ID段）id，send\_id，recv\_id（对于system信件一开始此段可以为None）；【2】（meta-info段）type（三种信件中哪一种），status（未派送，已派送，已打开），write\_time，sent\_time；【3】（content段）title，text

创建：/writings（with LetterForm）

查询：/sendings，/receivings，/sessions/<sid>，/letters/<lid>，/writings，/magic

修改：/letters/<lid>（状态变为已打开），/magic（发送以及匹配）

条件-内部：只有system信件未匹配前recv\_id可以为None，否则ID段必须非空

条件-外部：相当于temp与system信件的base-class

Todo：。。。

2.2.2 letters\_temp

描述：temp信件表

字段：【1】（ID段）id（同时作为外键指向letter表），sid（相关联的session）；【2】（info段）replied（是否被回复过），checked（发信人是否在该信件附带好友请求）

创建：/writings?lid=\*（回复system信或temp信）

查询：/letters/<lid>，/session/<sid>，/magic

修改：/writings（可修改replied）

条件-外部：id必须在letters表中有对应项

Todo：可回复期限，可能比较好的方式是回复session/user（在session中记录可回复数）

2.2.3 letters\_system

描述：system信件表

字段：【1】（ID段）id（同时作为外键指向letter\_temp表）；【2】（info段）matched（是否被匹配，实际上与Letter表中一些信息由冗余，但为了方便）

创建：/writings?system=1

查询：/letters/<lid>，/session/<sid>，/magic

修改：/magic

条件-外部：id必须在letters\_temp表中有对应项

2.3 sessions

描述：这可能是最奇怪的一个表了，本意是储存一次匹配的信件以及后续与次匹配相关的temp信件以及二者是否加好友的一些信息，不过似乎又可以将这些存储在friends表中。目前的实现中，即使是temp好友在friends表中也有一项，只不过status标记为temp（正式好友标记为ok）。虽然目前来看session有一些冗余，但如果用户之间可以有多个session（多次匹配）的话还是有作用的。

字段：【1】（ID段）id，uid1，uid2；【2】（info段）num（信件数量），u\*\_checked（二者是否提出请求），status（有效，无效，成功），last\_time（最近送信时间）

创建：/magic（匹配时）

查询：/sessions，/writings，/magic

修改：/magic

条件-内部：uid1<uid2（似乎并没用到）

条件-外部：为了方便，所有的session对应friends（friends表中状态可以是temp或是ok）

Todo：取消申请

2.4 friends

描述：好友表

字段：【1】（ID段）id，uid1，uid2，sid（最开始的session的sid，目前实现中是唯一的）；【2】（info段）status（temp临时，ok正式），num（信件数量）

创建：/magic（二者都check）

查询：/writings，/penpals/<uid>，/magic

修改：/magic

条件-内部：uid1<uid2（似乎并没用到）

Todo：删除好友与找回机制

**3. Views**

3.0. /newworld

描述：创建数据库

Todo：可能需要增加更多的管理方法以及日志，比如设置管理员账户什么的

3.1. /login

描述：用户登录

方法：GET；POST

数据库-读：User（通过username寻找，并核对密码）

数据库-写：无

表单：LoginForm

页面：login.html

链接-来：主页/为此页alias

链接-去：/welcome（登录成功则前往主页）

错误：登录失败（返回此页）

Todo：其他登录方式（邮件，手机）；忘记密码；验证码；记录登录信息

3.2. /logout

描述：用户退出

方法：GET

链接-来：<ALL>（几乎所有）

链接-去：/login

3.3. /register

描述：用户注册

方法：GET；POST

数据库-读：User（通过username寻找是否重名）

数据库-写：User.add\_one（添加用户）

表单：RegisterForm

页面：register.html

链接-来：未登录页面

链接-去：/login（注册成功前往登录页面）

错误：注册失败（返回此页）

Todo：其他注册方式（邮件，手机）

/\* 以下页面都需要用户登录（读User表获取当前用户），以及功能栏都有相关链接（详见layout.html） \*/

3.4. /welcome

描述：用户主页

方法：GET

数据库-读：User（获取当前用户）

数据库-写：无

页面：welcome.html

链接-来：<layout-pages>（功能栏，详见layout.html）；/login

链接-去：<layout-pages>

错误：未登录（401）；id错误（500）

Todo：添加更多状态显示，比如通知

3.5. /settings

描述：用户信息设置

方法：GET；POST

数据库-读：User（获取当前用户及其信息）

数据库-写：User（修改用户信息）

表单：InfoForm

页面：settings.html

链接-来：<layout-pages>；/penpals/<self.id>（查看用户为自己）

链接-去：<layout-pages>

错误：未登录（401）；id错误（500）；表单invalid（返回此页）

Todo：添加更多个人信息以及偏好设置

3.6. /receivings?filter= ["all", "unread", "read", "frompenpals", "fromtemp"]

描述：收信箱（所有收信人为自己且已发送的信件）

参数：详见Arg\_receivings\_\* （main/helpers.py）

方法：GET

数据库-读：User（获取当前用户及其信息）；Letter（用户收到的信件）

数据库-写：无

页面：receivings.html

链接-来：<layout-pages>

链接-去：<layout-pages>，/letters/<lid>，/penpals/<uid>

错误：未登录（401）；id错误（500）

Todo：查询，收藏，标记，删除

3.7. /sendings?filter= ["all", "unsent", "sent", "topenpals", "totemp"]

描述：发信箱（所有发信人为自己的信件）

参数：详见Arg\_sendings\_\* （main/helpers.py）

方法：GET

数据库-读：User（获取当前用户及其信息）；Letter（用户发送的信件）

数据库-写：无

页面：sendings.html

链接-来：<layout-pages>

链接-去：<layout-pages>，/letters/<lid>，/penpals/<uid>

错误：未登录（401）；id错误（500）

Todo：查询，收藏，标记，删除

3.8 /sessions

描述：所有的匹配记录（system匹配以及后续temp信件都在一个session中）

方法：GET

数据库-读：User（获取当前用户及其信息）；Session（所有当前用户参与的session）

数据库-写：无

页面：sessions.html

链接-来：<layout-pages>

链接-去：<layout-pages>，/sessions/<sid>，/penpals/<uid>

错误：未登录（401）；id错误（500）

Todo：查询，收藏，标记，删除

3.9 /sessions/<sid>

描述：查看具体一个session的信件

方法：GET

数据库-读：User（获取当前用户及其信息）；Session（当前session）；Letter（其中的信件）

数据库-写：无

页面：one\_session.html

链接-来：/sessions

链接-去：<layout-pages>，/letters/<lid>，/penpals/<uid>

错误：未登录（401）；无权限（401）；id错误（500）

Todo：更多关于该匹配的描述，实际上这也相当于信件箱；直接回复session，直接在session中记录可以回复的次数

3.10 /penpals

描述：好友列表

方法：GET

数据库-读：User（获取当前用户及其信息）；Friend（获取所有ok好友）

数据库-写：无

页面：penpals.html

链接-来：<layout-pages>

链接-去：<layout-pages>，/penpals/<uid>，/writings?uid=u.id

错误：未登录（401）；id错误（500）

Todo：好友管理，。。。

3.11 /penpals/<uid>

描述：查看好友信息

方法：GET

数据库-读：User（获取当前用户及其信息）；Friend（获取所有ok好友）

数据库-写：无

页面：one\_penpal.html

链接-来：<almost-ALL>

链接-去：<layout-pages>，/writings?uid=u.id

错误：未登录（401）；无权限（401）；id错误（500）

Note：这里的if\_display和if\_write指的是是否可以查看好友信息以及是否可以写信（信息需要看对方设置，只有正式好友可以写信）

Todo：历史记录

/\* 以下route在views2文件中，大部分与信件相关，认为是比较重要且稍微复杂一些 \*/

3.12 /letters/<lid>

描述：查看具体信件

方法：GET

数据库-读：User（获取当前用户及其信息）；Letter（信件信息）；Friend（是否可回复）；Letter\_temp（如果是temp信件，是否已经回复）

数据库-写：Letter（信件状态变为opened）

页面：one\_letter.html

链接-来：/sendings，/receivings，/sessions

链接-去：<layout-pages>，/writings，/penpals/<uid>

错误：未登录（401）；无权限（401）；id错误（500）

Note：只有sender或receiver（并且已发送）才可以查看；get\_info\_reply查看是否可以回复

Todo：代码这里就开始有点乱了，调用的是get\_info\_reply

3.13 /writings?system=\*&uid=<uid>&lid=<lid>

描述：写信（三种写信模式）

方法：GET；POST

数据库-读：<ALL>（几乎所有）

数据库-写：Letter\*（创建信件）

页面：writings.html

链接-来：/penpals；/sessions；/letters/<lid>

链接-去：<layout-pages>

错误：未登录（401）；无权限（401）；id错误（500）

Note：三种写信方式，不过此处只是添加信件，大部分数据库写还是在magic送信的时候：【1】发送给system待匹配，【2】发送给好友提供uid，【3】回复临时信提供lid。其中第三种最为复杂，需要查看session中的check和reply。

Todo：对于每封信可以有匹配设置，延时发送设置，发送给自己

/\* 以下route并非用户可访问，应该添加管理员控制，其中magic为送信method最重要 \*/

3.14 /magic

描述：送信

方法：GET

数据库-读：<ALL>（几乎所有）

数据库-写：<ALL>（几乎所有）

Note：还是根据三种模式的信进行发送：普通信件直接发送并，temp信件发送并检查session以及可能的添加好友（实际上为了方便session开始以及添加Friend表选项，不过并不是ok状态），匹配system信件并添加session以及temp-friend。目前为随机匹配（match.\_\_init\_\_. NaiveMatcher）

Todo：匹配算法以及匹配学习；未匹配的system信件应该添加相关记录并告知用户当天没有相应匹配

3.15 /logs

描述：每次magic的一些记录以及当前系统状态

Todo：系统管理

**4. Rethink about the implementation**

4.1 maybe design problems

最大的问题可能是session与friend的问题，session表是否应该存在，目前看来可以由friend表代替；不过每次匹配维护一个专门的table也许应该还是必要的。

然后就是views2.py中的几个route实现的有点乱。。。

4.2 Todo List

之所以称目前这个为reference是因为todo-list有太多，这个基本是按照《flask-book》的part2实现的，很多功能没有实现，而且一些必备的扩展也没有，只能作为version-0.1了。

【必备扩展】

* 用户认证以及安全相关，例如使用Flask-Login扩展
* 系统管理，数据库修复，手工匹配信件等等，现在如果要做这些只能手动改数据库，很容易造成不一致
* 错误处理，目前错误主要有401权限错误和500内部错误，需要更完备的处理方式（对于用户更多错误通知，对于系统防止崩溃）
* Html同样可以写macro子函数，html完善界面

【未实现功能】

* 手机邮箱注册，登录验证码，找回密码
* 更多用户信息，头像爱好地区等等
* 虚拟地址以及相关
* 系统通知
* 信件限制：每天可发送信件数，信件字数，信件关键字检测，垃圾信件检测
* 匹配策略，soft&hard constraints
* 匹配满意度以及机器学习
* 系统信件多天未匹配可提高优先级
* 临时信件延时没有回复机会，临时信件列表限制
* 信件删除，好友删除，好友找回，信件管理
* 。。。

【当前实现】

* Views2中的route实现问题
* Test & Debug