**一、创建forms**

显而易见，我们的注册页面也需要一个form表单。同样地，在/login/forms.py中添加一个新的表单类：

**class** **RegisterForm**(forms**.**Form):

gender **=** (

('male', "男"),

('female', "女"),

)

username **=** forms**.**CharField(label**=**"用户名", max\_length**=**128, widget**=**forms**.**TextInput(attrs**=**{'class': 'form-control'}))

password1 **=** forms**.**CharField(label**=**"密码", max\_length**=**256, widget**=**forms**.**PasswordInput(attrs**=**{'class': 'form-control'}))

password2 **=** forms**.**CharField(label**=**"确认密码", max\_length**=**256, widget**=**forms**.**PasswordInput(attrs**=**{'class': 'form-control'}))

email **=** forms**.**EmailField(label**=**"邮箱地址", widget**=**forms**.**EmailInput(attrs**=**{'class': 'form-control'}))

sex **=** forms**.**ChoiceField(label**=**'性别', choices**=**gender)

captcha **=** CaptchaField(label**=**'验证码')

说明：

* gender字典和User模型中的一样，其实可以拉出来作为常量共用，为了直观，特意重写一遍；
* password1和password2，用于输入两遍密码，并进行比较，防止误输密码；
* email是一个邮箱输入框；
* sex是一个select下拉框；

**二、完善register.html**

同样地，类似login.html文件，我们手工在register.html中编写forms相关条目：

***{%*** **extends** 'base.html' ***%}***

***{%*** **block** title ***%}***注册***{%*** **endblock** ***%}***

***{%*** **block** content ***%}***

<div class="container">

<div class="col-md-4 col-md-offset-4">

<form class='form-register' action="/register/" method="post">

***{%*** **if** message ***%}***

<div class="alert alert-warning">***{{*** message ***}}***</div>

***{%*** **endif** ***%}***

***{%*** **csrf\_token** ***%}***

<h2 class="text-center">欢迎注册</h2>

<div class="form-group">

***{{*** register\_form.username.label\_tag ***}}***

***{{*** register\_form.username***}}***

</div>

<div class="form-group">

***{{*** register\_form.password1.label\_tag ***}}***

***{{*** register\_form.password1 ***}}***

</div>

<div class="form-group">

***{{*** register\_form.password2.label\_tag ***}}***

***{{*** register\_form.password2 ***}}***

</div>

<div class="form-group">

***{{*** register\_form.email.label\_tag ***}}***

***{{*** register\_form.email ***}}***

</div>

<div class="form-group">

***{{*** register\_form.sex.label\_tag ***}}***

***{{*** register\_form.sex ***}}***

</div>

<div class="form-group">

***{{*** register\_form.captcha.errors ***}}***

***{{*** register\_form.captcha.label\_tag ***}}***

***{{*** register\_form.captcha ***}}***

</div>

<button type="reset" class="btn btn-default pull-left">重置</button>

<button type="submit" class="btn btn-primary pull-right">提交</button>

</form>

</div>

</div> *<!-- /container -->*

***{%*** **endblock** ***%}***

需要注意的是form标签的action地址为/register/。

**三、实现注册视图**

进入/login/views.py文件，现在来完善我们的register()视图：

**def** **register**(request):

**if** request**.**session**.**get('is\_login', None):

*# 登录状态不允许注册。你可以修改这条原则！*

**return** redirect("/index/")

**if** request**.**method **==** "POST":

register\_form **=** forms**.**RegisterForm(request**.**POST)

message **=** "请检查填写的内容！"

**if** register\_form**.**is\_valid(): *# 获取数据*

username **=** register\_form**.**cleaned\_data['username']

password1 **=** register\_form**.**cleaned\_data['password1']

password2 **=** register\_form**.**cleaned\_data['password2']

email **=** register\_form**.**cleaned\_data['email']

sex **=** register\_form**.**cleaned\_data['sex']

**if** password1 **!=** password2: *# 判断两次密码是否相同*

message **=** "两次输入的密码不同！"

**return** render(request, 'login/register.html', locals())

**else**:

same\_name\_user **=** models**.**User**.**objects**.**filter(name**=**username)

**if** same\_name\_user: *# 用户名唯一*

message **=** '用户已经存在，请重新选择用户名！'

**return** render(request, 'login/register.html', locals())

same\_email\_user **=** models**.**User**.**objects**.**filter(email**=**email)

**if** same\_email\_user: *# 邮箱地址唯一*

message **=** '该邮箱地址已被注册，请使用别的邮箱！'

**return** render(request, 'login/register.html', locals())

*# 当一切都OK的情况下，创建新用户*

new\_user **=** models**.**User()

new\_user**.**name **=** username

new\_user**.**password **=** password1

new\_user**.**email **=** email

new\_user**.**sex **=** sex

new\_user**.**save()

**return** redirect('/login/') *# 自动跳转到登录页面*

register\_form **=** forms**.**RegisterForm()

**return** render(request, 'login/register.html', locals())

从大体逻辑上，也是先实例化一个RegisterForm的对象，然后使用is\_valide()验证数据，再从cleaned\_data中获取数据。

重点在于注册逻辑，首先两次输入的密码必须相同，其次不能存在相同用户名和邮箱，最后如果条件都满足，利用ORM的API，创建一个用户实例，然后保存到数据库内。

对于注册的逻辑，不同的生产环境有不同的要求，请跟进实际情况自行完善，这里只是一个基本的注册过程，不能生搬照抄。

让我们看一下注册的页面：



你可以尝试用不同的情况进行注册，然后观察错误信息的提示，最后进行一次成功地注册:



页面自动跳转到登录页面。我们进入admin后台，查看一下用户列表：





可以看到一切OK！大吉大利，今晚吃鸡！

**四、密码加密**

等等！我们好像忘了什么！我们到现在都还一直在用明文的密码！

对于如何加密密码，有很多不同的途径，其安全程度也高低不等。这里我们使用Python内置的hashlib库，使用哈希值的方式加密密码，可能安全等级不够高，但足够简单，方便使用，不是么？

首先在login/views.py中编写一个hash函数：

**import** hashlib

**def** **hash\_code**(s, salt**=**'mysite'):*# 加点盐*

h **=** hashlib**.**sha256()

s **+=** salt

h**.**update(s**.**encode()) *# update方法只接收bytes类型*

**return** h**.**hexdigest()

使用了sha256算法，加了点盐。具体的内容可以参考站点内的Python教程中hashlib库章节。

然后，我们还要对login()和register()视图进行一下修改：

**def** **login**(request):

**if** request**.**session**.**get('is\_login', None):

**return** redirect("/index/")

**if** request**.**method **==** "POST":

login\_form **=** forms**.**UserForm(request**.**POST)

message **=** "请检查填写的内容！"

**if** login\_form**.**is\_valid():

username **=** login\_form**.**cleaned\_data['username']

password **=** login\_form**.**cleaned\_data['password']

**try**:

user **=** models**.**User**.**objects**.**get(name**=**username)

**if** user**.**password **==** hash\_code(password): *# 哈希值和数据库内的值进行比对*

request**.**session['is\_login'] **=** True

request**.**session['user\_id'] **=** user**.**id

request**.**session['user\_name'] **=** user**.**name

**return** redirect('/index/')

**else**:

message **=** "密码不正确！"

**except**:

message **=** "用户不存在！"

**return** render(request, 'login/login.html', locals())

login\_form **=** forms**.**UserForm()

**return** render(request, 'login/login.html', locals())

**def** **register**(request):

**if** request**.**session**.**get('is\_login', None):

*# 登录状态不允许注册。你可以修改这条原则！*

**return** redirect("/index/")

**if** request**.**method **==** "POST":

register\_form **=** forms**.**RegisterForm(request**.**POST)

message **=** "请检查填写的内容！"

**if** register\_form**.**is\_valid(): *# 获取数据*

username **=** register\_form**.**cleaned\_data['username']

password1 **=** register\_form**.**cleaned\_data['password1']

password2 **=** register\_form**.**cleaned\_data['password2']

email **=** register\_form**.**cleaned\_data['email']

sex **=** register\_form**.**cleaned\_data['sex']

**if** password1 **!=** password2: *# 判断两次密码是否相同*

message **=** "两次输入的密码不同！"

**return** render(request, 'login/register.html', locals())

**else**:

same\_name\_user **=** models**.**User**.**objects**.**filter(name**=**username)

**if** same\_name\_user: *# 用户名唯一*

message **=** '用户已经存在，请重新选择用户名！'

**return** render(request, 'login/register.html', locals())

same\_email\_user **=** models**.**User**.**objects**.**filter(email**=**email)

**if** same\_email\_user: *# 邮箱地址唯一*

message **=** '该邮箱地址已被注册，请使用别的邮箱！'

**return** render(request, 'login/register.html', locals())

*# 当一切都OK的情况下，创建新用户*

new\_user **=** models**.**User()

new\_user**.**name **=** username

new\_user**.**password **=** hash\_code(password1) *# 使用加密密码*

new\_user**.**email **=** email

new\_user**.**sex **=** sex

new\_user**.**save()

**return** redirect('/login/') *# 自动跳转到登录页面*

register\_form **=** forms**.**RegisterForm()

**return** render(request, 'login/register.html', locals())

注意其中关于密码处理的部分！

好了，我们可以来验证一下了!但是，**请先在admin后台里，把我们前面创建的测试用户全部删除！**因为它们的密码没有使用哈希算法加密，已经无效了。

重启服务器，进入注册页面，新建一个用户，然后进入admin后台，查看用户的密码情况：



再使用该用户登录一下，大功告成！

可以看到密码长度根据你哈希算法的不同，已经变得很长了，所以前面model中设置password字段时，不要想当然的将max\_length设置为16这么小的数字。