django配置用户登陆界面+mysql使用

ubuntu下面安装mysql

`sudo apt install mysql-server`
报错
`django.db.utils.InternalError: (1698, u"Access denied for user 'root'@'localhost'")`

解决方法

怀疑是mysql的密码的问题

参考文章
使用如下方法更改
1. 停止mysql
`sudo service mysql stop` 或者 `sudo /etc/init.d/mysql stop`
`sudo /usr/bin/mysqld_safe --skip-grant-tables --skip-networking &`
这个地方会报错:
`mysql_safe Directory `/var/run/mysqld` for UNIX socket file don't exits`
解决方法如下:
`sudo mkdir -p /var/run/mysqld`
`sudo chown mysql:mysql /var/run/mysqld`
2. 然后再输入
`sudo /usr/bin/mysqld_safe --skip-grant-tables --skip-networking &`
这个时候可以打开另外一个terminal，无密码登陆MYSQL
`mysql -u root`
3. 在数据库里面改密码
`use mysql;`
`update user set authentication_string=PASSWORD("要改的密码")where User='root';`
`update user set plugin="mysql_native_password";`
`flush privileges;`
`quit;`
4. 重新启动mysql
`sudo /etc/init.d/mysql stop`
`sudo /etc/init.d/mysql start`

创建项目和基础设置

主要内容都是参考的第一篇博客，代码99.9%一摸一样。这一篇markdown主要是学习的随手记录。 参考内容

1 参考内容2

创建项目和app

`django-admin startproject mysite_login`
`python manage.py startapp login`

设置配置文件

文件目录

`mysite_login/mysite_login/settings.py`
在配置文件设置以下内容

- 设置时区和语言 默认使用的是美国时间和英语，在这个文件里面修改以下内容。

```
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
```

修改成亚洲/上海时间和中文

```
LANGUAGE_CODE = 'zh-hans'
TIME_ZONE = 'Asia/Shanghai'
USE_I18N = True
USE_L10N = True
USE_TZ = False
```

- 修改数据库，使用的是mysql数据库，用户和密码要对应，host写127.0.0.1本地的IP地址，端口为3306。

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'django',         #数据库名字
        'USER': 'root',           #账号
        'PASSWORD': '123456',       #密码
        'HOST': '127.0.0.1',    #IP
        'PORT': '3306',                    #端口
    }
}
```

- 注册APP，添加了之前创建的app。

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'login',
]
```

数据库模型设计

/login/models.py 添加如下文件

```
# login/models.py
from django.db import models
class User(models.Model):
    '''用户表'''

    gender = (
        ('male','男'),
        ('female','女'),
    )

    name = models.CharField(max_length=128,unique=True)
    password = models.CharField(max_length=256)
    email = models.EmailField(unique=True)
    sex = models.CharField(max_length=32,choices=gender,default='男')
    c_time = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.name

    class Meta:
        ordering = ['c_time']
        verbose_name = '用户'
        verbose_name_plural = '用户'
```

如果显示编码错误，在前面加上一行 `# -*- coding: utf-8 -*-` 其中各字段限制了字长长度，有的内容是需要输入的，有的内容是需要选择的，email使用了Django内置的邮箱类型。使用 `__str__` 帮助人性化显示对象信息。元数据里面定义用户按创建时间的反序排列，也就是最近的最先显示。

导入pymysql模块

文件目录

`/mysite_login/login/__init__.py`

添加内容

```
import pymysql
pymysql.install_as_MySQLdb()
```

Mysql创建数据库

报错：`django.db.utils.InternalError: (1049, u"Unknown database 'django'")`
`create database django;`

字符编码报错

```
django.db.utils.InternalError: (1366, u"Incorrect string value: '\\xE7\\x94\\xA8\\x
E6\\x88\\xB7' for column 'name' at row 1")
```

参考文献

- 首先修改django中的数据库配置信息，修改settings配置信息，加入

```
'TEST': {
    'CHARSET' : 'utf8',
    'COLLATION' : 'utf8_general_ci'
}
```

- 然后删除数据库，重新创建一个新的数据库，这次要指定编码格式 `create database django default character set utf8 collate utf8_general_ci;`

迁移到数据库

`python manage.py makemigrations` `python manage.py migrate`

admin后台

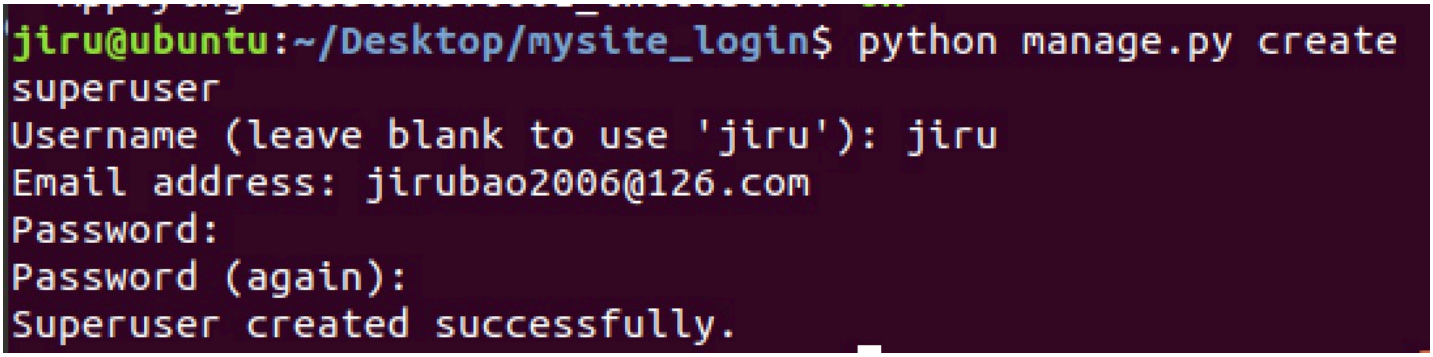
文件目录

`mysite_login/login/admin.py`

在admin中注册模型

```
from django.contrib import admin
from . import models
admin.site.register(models.User)
```

创建超级管理员



这个位置的密码要输入一个数字和字母结合的，用户是jrr，密码是jrr19991026，记在这里不知道后面用得上用不上。

url路由和视图

在根路由下面直接编写路由条目。

文件目录

`mysite_login/urls.py`

添加路由

```
# mysite_login/urls.py

from django.conf.urls import url
from django.contrib import admin
from login import views

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^index/', views.index),
    url(r'^login/', views.login),
    url(r'^register/', views.register),
    url(r'^logout/', views.logout),
]
```

初步简易视图

在 `login/views.py` 文件中写入如下内容

```
# login/views.py

from django.shortcuts import render,redirect

def index(request):
    pass
    return render(request,'login/index.html')

def login(request):
    pass
    return render(request,'login/login.html')

def register(request):
    pass
    return render(request,'login/register.html')

def logout(request):
    pass
    return redirect('/index/')
```

设计登陆页面

首先在 `login/templates/login` 目录中创建文件login.html然后写入如下内容

```
{#login/templates/login/login.html#}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>登录</title>
</head>
<body>
     <div style="margin: 15% 40%;">
        <h1>欢迎登录! </h1>
       <form action="/login/" method="post">
            <p>
                <label for="id_username">用户名: </label>
                <input type="text" id="id_username" name="username" placeholder="用户名
            </p>

            <p>
                <label for="id_password">密码: </label>
                <input type="password" id="id_password" placeholder="密码" name="passw
            </p>
            <input type="submit" value="确定">
        </form>
    </div>



</body>
</html>
```

引入Bootstrap

背景与下载

Bootstrap是基于HTML、CSS、JAVASCRIPT的一个前端框架。 下载地址，下载的是3.3.7的版本。

放到文件夹的位置

在根目录下面新建一个static目录，将解压后的目录整体拷贝到static下。 `/mysite_login/static`

引入Jquery文件

引入的文件目录

`/mysite_login/static/js`

背景与下载

JQuery是一个简洁的JavaScript框架。倡导写更少的代码，做更多的事情，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互。
由于Bootstrap依赖JQuery，下载并且引入JQuery，在static目录下，新建一个css和js目录，作为以后的样式文件和js文件的存放地，将jquery文件拷贝到static/js。

编辑settings文件

`/mysite_login/mysite_login/settings.py`
打开项目的settings文件，在最下面加入如下内容，用于指定静态文件的搜索目录。

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
]
```

创建base.html模版

文件目录

`/mysite_login/mysite_login/templates/base.html`

代码内容

这个代码是在Bootstrap的文档中提供的。

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap 101 Template</title>

    <!-- Bootstrap -->
    <link href="css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media querie
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://cdn.bootcss.com/html5shiv/3.7.3/html5shiv.min.js"></script
      <script src="https://cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>你好，世界！</h1>

    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="https://cdn.bootcss.com/jquery/1.12.4/jquery.min.js"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

创建页面导航条

Bootstrap提供了导航条组件如下

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <!-- Brand and toggle get grouped for better mobile display -->
    <div class="navbar-header">
      <button type="button" class="navbar-toggle collapsed" data-toggle="collapse" da
        <span class="sr-only">Toggle navigation</span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
        <span class="icon-bar"></span>
      </button>
      <a class="navbar-brand" href="#">Brand</a>
    </div>
```

```
    <!-- Collect the nav links, forms, and other content for toggling -->
    <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
      <ul class="nav navbar-nav">
        <li class="active"><a href="#">Link <span class="sr-only">(current)</span></a
        <li><a href="#">Link</a></li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" ar
          <ul class="dropdown-menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">Separated link</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">One more separated link</a></li>
          </ul>
        </li>
      </ul>
      <form class="navbar-form navbar-left">
        <div class="form-group">
          <input type="text" class="form-control" placeholder="Search">
        </div>
        <button type="submit" class="btn btn-default">Submit</button>
      </form>
      <ul class="nav navbar-nav navbar-right">
        <li><a href="#">Link</a></li>
        <li class="dropdown">
          <a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" ar
          <ul class="dropdown-menu">
            <li><a href="#">Action</a></li>
            <li><a href="#">Another action</a></li>
            <li><a href="#">Something else here</a></li>
            <li role="separator" class="divider"></li>
            <li><a href="#">Separated link</a></li>
          </ul>
        </li>
      </ul>
    </div><!-- /.navbar-collapse -->
  </div><!-- /.container-fluid -->
</nav>
```

进行修改获得如下的内容

```
<nav class="navbar navbar-default">
      <div class="container-fluid">
        <!-- Brand and toggle get grouped for better mobile display -->
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse
            <span class="sr-only">切换导航条</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Mysite</a>
        </div>

        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="my-nav">
          <ul class="nav navbar-nav">
            <li class="active"><a href="/index/">主页</a></li>
          </ul>
          <ul class="nav navbar-nav navbar-right">
            <li><a href="/login/">登录</a></li>
            <li><a href="/register/">注册</a></li>
          </ul>
        </div><!-- /.navbar-collapse -->
      </div><!-- /.container-fluid -->
    </nav>
```

将这部分内容加入到base.html中，最后得到的base.html的代码如下

```
{% load staticfiles %}

<!DOCTYPE html>
<html lang="zh-CN">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>{% block title %}base{% endblock %}</title>

    <!-- Bootstrap -->
    <link href="{% static 'bootstrap-3.3.7-dist/css/bootstrap.min.css' %}" rel="style

    <!-- HTML5 shim and Respond.js for IE8 support of HTML5 elements and media querie
    <!-- WARNING: Respond.js doesn't work if you view the page via file:// -->
    <!--[if lt IE 9]>
      <script src="https://cdn.bootcss.com/html5shiv/3.7.3/html5shiv.min.js"></script
      <script src="https://cdn.bootcss.com/respond.js/1.4.2/respond.min.js"></script>
    <![endif]-->
```

```
    {% block css %}{% endblock %}
  </head>
  <body>
    <nav class="navbar navbar-default">
      <div class="container-fluid">
        <!-- Brand and toggle get grouped for better mobile display -->
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse
            <span class="sr-only">切换导航条</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" href="#">Mysite</a>
        </div>

        <!-- Collect the nav links, forms, and other content for toggling -->
        <div class="collapse navbar-collapse" id="my-nav">
          <ul class="nav navbar-nav">
            <li class="active"><a href="/index/">主页</a></li>
          </ul>
          <ul class="nav navbar-nav navbar-right">
            <li><a href="/login/">登录</a></li>
            <li><a href="/register/">注册</a></li>
          </ul>
        </div><!-- /.navbar-collapse -->
      </div><!-- /.container-fluid -->
    </nav>

    {% block content %}{% endblock %}


    <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
    <script src="{% static 'js/jquery-3.2.1.js' %}"></script>
    <!-- Include all compiled plugins (below), or include individual files as needed
    <script src="{% static 'bootstrap-3.3.7-dist/js/bootstrap.min.js' %}"></script>
  </body>
</html>
```

其中

- `{% static '相对路径' %}` 为我们提供静态文件加载方法，可以将页面与静态文件链接起来。链接之后才可以使用static方法。
- `{% block title %}base{% endblock %}`，设置了一个动态的页面title块。
- `{% block css %}{% endblock %}`，设置了一个动态的css加载块。
- `{% block content %}{% endblock %}`，为具体页面的主体内容留下接口。
- `{% static 'bootstrap-3.3.7-dist/css/bootstrap.min.css' %}` 将样式文件指向了静态文件，下面的js脚本也是同样的道理。

结合Bootstrap修改登陆界面

文件目录

`login/templates/login/login.html`

```
{% extends 'login/base.html' %}
{% load staticfiles %}
{% block title %}登录{% endblock %}
{% block css %}
    <link rel="stylesheet" href="{% static 'css/login.css' %}">
{% endblock %}


{% block content %}
    <div class="container">
        <div class="col-md-4 col-md-offset-4">
          <form class='form-login' action="/login/" method="post">
              <h2 class="text-center">欢迎登录</h2>
              <div class="form-group">
                <label for="id_username">用户名：</label>
                <input type="text" name='username' class="form-control" id="id_userna
              </div>
              <div class="form-group">
                <label for="id_password">密码：</label>
                <input type="password" name='password' class="form-control" id="id_pa
              </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 %}
```

其中，

- 通过 `{% extends 'base.html' %}` 继承了 `base.html` 模版的内容。
- 通过 `{% block title %}` 登陆 `{% endblock %}` 设置了专门的title。
- 通过 `{% block css %}` 引入了针对性的 `login.css` 样式文件。
- 主题内容定义在 `block content` 内部。
- 添加了一个重置按钮。

添加登陆界面的css内容

在 `static/css` 中新建一个login.css文件。

```
body {
  background-color: #eee;
}
.form-login {
  max-width: 330px;
  padding: 15px;
  margin: 0 auto;
}
.form-login .form-control {
  position: relative;
  height: auto;
  -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
          box-sizing: border-box;
  padding: 10px;
  font-size: 16px;
}
.form-login .form-control:focus {
  z-index: 2;
}
.form-login input[type="text"] {
  margin-bottom: -1px;
  border-bottom-right-radius: 0;
  border-bottom-left-radius: 0;
}
.form-login input[type="password"] {
  margin-bottom: 10px;
  border-top-left-radius: 0;
  border-top-right-radius: 0;
}
```

登陆视图

用户通过login.html中的表单填写用户名和密码，并以POST的方式发送到服务器的 `/login/` 地址，服务器通过 `/login.views.py` 中的login()视图函数，接受并处理这一请求。 可以通过如下的方法接收和处理请求：

```
def login(request):
    if request.method == "POST":
        username = request.POST.get('username')
        password = request.POST.get('password')        return redirect('/index/')
    return render(request, 'login/login.html')
```

在前端页面的form表单里面添加一个 `{% csrf_token %}` 标签，是用来防止跨站请求伪造的。

数据验证

修改login视图

文件目录 `login/views.py`

- 通过用户名，使用Django的ORM去数据库中查询用户数据，如果有匹配项，进行密码对比，如果没有匹配项，说明用户名不存在，如果密码对比错误，说明密码不正确。
- 在用户登陆成功或失败的时候，给用户返回提示信息。修改login函数如下： #### 改变的位置 增加了message变量，用于保存提示信息，当有错误信息的时候，将错误信息打包成一个字典，然后作为第三个参数提供给render()方法。这个数据字典在渲染模版的时候会传递到模版里供调用。 ``` {% extends 'login/base.html' %} {% load staticfiles %} {% block title %}登录{% endblock %} {% block css %} {% endblock %}

{% block content %}

```
            {% if message %}
                <div class="alert alert-warning">{{ message }}</div>
            {% endif %}

            {% csrf_token %}
            <h2 class="text-center">欢迎登录</h2>
            <div class="form-group">
                <label for="id_username">用户名：</label>
                <input type="text" name='username' class="form-control" id="id_userna
                       autofocus required>
            </div>
            <div class="form-group">
                <label for="id_password">密码：</label>
                <input type="password" name='password' class="form-control" id="id_pa
                       required>
            </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 %} ```

为在前端正确显示，修改login.html

其中

```
{% extends 'login/base.html' %}
{% load staticfiles %}
{% block title %}登录{% endblock %}
{% block css %}
    <link rel="stylesheet" href="{% static 'css/login.css' %}">
{% endblock %}


{% block content %}
    <div class="container">
        <div class="col-md-4 col-md-offset-4">
            <form class='form-login' action="/login/" method="post">

                {% if message %}
                    <div class="alert alert-warning">{{ message }}</div>
                {% endif %}

                {% csrf_token %}
                <h2 class="text-center">欢迎登录</h2>
                <div class="form-group">
                    <label for="id_username">用户名: </label>
                    <input type="text" name='username' class="form-control" id="id_us
                           autofocus required>
                </div>
                <div class="form-group">
                    <label for="id_password">密码: </label>
                    <input type="password" name='password' class="form-control" id="i
                           required>
                </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 %}
```

修改 `index.html` 主页模版

```
{#login/templates/login/index.html#}

{% extends 'login/base.html' %}
{% block title %}主页{% endblock %}
{% block content %}
    <h1>欢迎回来! </h1>
{% endblock %}
```

修改视图

- Django的表单提供了以下功能：准备和重构数据用于页面渲染，为数据创建HTML表单元素，接受和处理用户从表单发过来的数据。 ### 手动渲染表单 {{login_form}}的默认界面很丑，如果要使用CSS和JS，比如要引入Bootstrap框架，需要对表单内的元素进行额外控制 ### 图片验证码 `pip install django-simple-captcha` 在settings中，将'captcha'注册到app列表里。 captcha需要在数据库中建立自己的数据表，所以需要执行 `python manage.py migrate` 来生成数据表。 在 `/mysite_login/urls.py` 添加 `url(r'^captcha', include('captcha.urls'))`

主页修改

文件位置

`mysite_login/login/templates/login/index.html`

添加如下代码：

```
{#login/templates/login/index.html#}

{% extends 'login/base.html' %}
{% block title %}主页{% endblock %}
{% block content %}
    <h1>欢迎回来！</h1>
{% endblock %}
```

Django表单

Django表单的功能

- 准备和重构数据用于页面渲染
- 为数据创建HTML表单元素
- 接受和处理用户从表单发送过来的数据 #### 文件位置 `/mysite_login/login/forms.py`

```
from django import forms


class UserForm(forms.Form):
    username = forms.CharField(label="用户名", max_length=128)
    password = forms.CharField(label="密码", max_length=256, widget=forms.PasswordInpu
```

代码说明

- 首先要导入forms模块
- 所有表单类都要继承forms.form类

- 每一个表单字段都有自己的字段类型，比如CharField，分别对应一种HTML语言中内的一个input元素。
- label参数用于设置标签
- max_length限制字段输入的最大长度，同时起到两个作用，第一个是在浏览器页面限制用户输入不可超过字符数，二是在后端服务器验证用户输入的长度不可超过字符数。
- widget=forms.PasswordInput 用于指定该字段在form表单里面表现为 `<input type='password' />`，也就是一个密码输入框。

修改视图

使用了Django的表单，需要在视图中进行相应的修改。 文件位置 `login/views.py`

```
from django.shortcuts import render,redirect
from . import models
from .forms import UserForm

def index(request):
    pass
    return render(request,'login/index.html')

def login(request):
    if request.method == "POST":
        login_form = 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 == password:
                    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 = UserForm()
    return render(request, 'login/login.html', locals())
```

修改login界面

文件位置

`/mysite_login/login/templates/login/login.html`

```
{% extends 'base.html' %}
{% load staticfiles %}
{% block title %}登录{% endblock %}
{% block css %}<link href="{% static 'css/login.css' %}" rel="stylesheet"/>{% endbl
ock %}


{% block content %}
    <div class="container">
        <div class="col-md-4 col-md-offset-4">
          <form class='form-login' action="/login/" method="post">

              {% if message %}
                  <div class="alert alert-warning">{{ message }}</div>
              {% endif %}
              {% csrf_token %}
              <h2 class="text-center">欢迎登录</h2>

              {{ login_form }}

              <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 %}
```

手动渲染表单

要想对表单内的input元素进行额外控制，需要手动渲染字段。从 `login_form.name_of_field` 获取每一个字段，然后分别渲染。

文件位置

`/mysite_login/login/templates/login/login.html`
添加如下内容：

```
<div class="form-group">
  {{ login_form.username.label_tag }}
  {{ login_form.username}}
</div>
<div class="form-group">
  {{ login_form.password.label_tag }}
  {{ login_form.password }}
</div>
```

在form类里面添加attr属性

文件位置 `/mysite_login/login/forms.py`

```
from django import forms

class UserForm(forms.Form):
    username = forms.CharField(label="用户名", max_length=128, widget=forms.TextInput(
    password = forms.CharField(label="密码", max_length=256, widget=forms.PasswordInpu
```

图片验证码

安装captcha

`pip install django-simple-captcha`

注册captcha

文件位置 `mysite_login/mysite_login/settings.py`
在INSTALLED_APPS后面加上'captcha'

生成captcha数据表

因为captcha需要在数据库中建立自己的数据表，所以需要执行migrate `python manage.py migrate`

添加url路由

文件位置 `mysite_login/mysite_login/urls.py`
添加内容
`from django.conf.urls import include` `url(r'^captcha', include('captcha.urls'))`

修改forms.py

文件位置 `mysite_login/login/forms.py`

添加
`captcha=CaptchaField(label='验证码')` `from captcha.fields import CaptchaField`

在login.html中加入captcha相关的内容

```
{% extends 'login/base.html' %}
{% load staticfiles %}
{% block title %}登录{% endblock %}
{% block css %}
    <link rel="stylesheet" href="{% static 'css/login.css' %}">
{% endblock %}


{% block content %}
    <div class="container">
        <div class="col-md-4 col-md-offset-4">
          <form class='form-login' action="/login/" method="post">

              {% if message %}
                  <div class="alert alert-warning">{{ message }}</div>
              {% endif %}
              {% csrf_token %}
              <h2 class="text-center">欢迎登录</h2>
              <div class="form-group">
                  {{ login_form.username.label_tag }}
                  {{ login_form.username}}
              </div>
              <div class="form-group">
                  {{ login_form.password.label_tag }}
                  {{ login_form.password }}
              </div>

              <div class="form-group">
                  {{ login_form.captcha.errors }}
                  {{ login_form.captcha.label_tag }}
                  {{ login_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 %}
```

session会话

根据我的理解，这一部分的内容是用来保持用户的连接状态，因为HTTP的特性是每一次来自于用户浏览器的请求都是无状态、独立的，也就无法保存用户的状态，使用cookie可以保护用户状态。但是由于cookie不够安全（保存在用户的机器上），现代网站都用cookie来保存一些不重要的内容，实际的用户数据和状态以session会话的方式保存在服务器端。通常情况下，session保存在数据库内。

修改login/views中login()

加入

```
if request.session.get('is_login',None):
    return redirect("/index/")
```

不允许重复登录。

```
request.session['is_login'] = True
request.session['user_id'] = user.id
request.session['user_name'] = user.name
```

向session字典里面写入用户状态和数据。

修改logout()

```
def logout(request):
    if not request.session.get('is_login', None):
        # 如果本来就未登录，也就没有登出一说
        return redirect("/index/")
    request.session.flush()
    # 或者使用下面的方法
    # del request.session['is_login']
    # del request.session['user_id']
    # del request.session['user_name']
    return redirect("/index/")
```

完善页面

通过判断用户登陆与否，展示不同的页面。 修改base.html

```
<div class="collapse navbar-collapse" id="my-nav">
          <ul class="nav navbar-nav">
            <li class="active"><a href="/index/">主页</a></li>
          </ul>
          <ul class="nav navbar-nav navbar-right">
              {% if request.session.is_login %}
                  <li><a href="#">当前在线：{{ request.session.user_name }}</a></li>
                  <li><a href="/logout/">登出</a></li>
              {% else %}
                  <li><a href="/login/">登录</a></li>
                  <li><a href="/register/">注册</a></li>
              {% endif %}
          </ul>
        </div><!-- /.navbar-collapse -->
      </div><!-- /.container-fluid -->
```

Markdown正常显示html代码

知乎问题