



網站的Session功能

1

學習目標

- Session簡介
 - 複製Django網站
 - Cookies簡介
 - 使用Cookie建立網站登入功能
 - 開始使用Session
- 活用Session
 - 建立使用者資料表
 - 使用Session建立網站登入功能
 - 整合Django的訊息顯示框架Messages Framework
- Django auth使用者驗證
 - 使用Django的使用者驗證系統
 - 顯示新增的User欄位
 - 應用auth使用者驗證存取資料庫

Session簡介

- ▶ Session的最重要目的就是讓網站記得使用者，也就是瀏覽這個網站的人
- ▶ 因為網際網路HTTP的特性，理論上每一次來自於瀏覽器的請求(request)都是獨立和前後次的請求沒有關係的
- ▶ 如果沒有特別的機制，網頁伺服器是沒有辦法辨識出前後次的瀏覽行為是不是來自於同一個人
- ▶ Session機制的目的就是為了解決這個問題

複製Django網站

- 使用上一堂課的HTML_Form網站，複製整個網站之後再做修改
- 把整個資料夾複製之後，再針對一些檔案的內容進行修改
- 有哪些檔案的哪些內容需要修改呢？
- 只有manage.py以及HTML_Form這個資料夾的少部份檔案
- 在Linux以及MacOS作業系統之下可以使用grep指令找出和專案有關的字串
- 在Windows作業系統之下也有類似的工具，那就是findstr，用法如下：

複製Django網站

- (VENV) I:\myDjango\ HTML_Form>findstr/s "HTML_Form" *.py
- manage.py:
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "HTML_Form.settings")
- HTML_Form\settings.py:Django settings for **HTML_Form** project.
- HTML_Form\settings.py:ROOT_URLCONF = '**HTML_Form**.urls'
- HTML_Form\settings.py:WSGI_APPLICATION = 'ch08www.wsgi.application'
- HTML_Form\urls.py:'''**HTML_Form** URL Configuration
- HTML_Form\wsgi.py:WSGI config for **HTML_Form** project.
- HTML_Form\wsgi.py:os.environ.setdefault("DJANGO_SETTINGS_MODULE", "**HTML_Form**.settings")

複製Django網站

- 全部要改的地方就這幾個檔案，在本例就是把所有的 `HTML_Form` 改為 `Session_cookies`，就算是完成了網

```
C:\Users\shiuny\Desktop\Session_cookies>findstr/s "HTML_Form" *.py
HTML_Form\settings.py:Django settings for HTML_Form project.
HTML_Form\settings.py:ROOT_URLCONF = 'HTML_Form.urls'
HTML_Form\settings.py:WSGI_APPLICATION = 'HTML_Form.wsgi.application'
HTML_Form\urls.py:'''HTML_Form URL Configuration
HTML_Form\wsgi.py:WSGI config for HTML Form project.
HTML_Form\wsg:Performing system checks...
manage.py:
System check identified no issues (0 silenced).
C:\Users\shiuu June 28, 2019 - 23:44:47
Django version 2.2.1, using settings 'Session_cookies.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

開始使用Session

- Session把所有的資料者放在**伺服器端**，客戶端只會記錄一個**識別的資訊**而已。
- Django在實作Session支援時有許多種方式，包括透過**Cookie**的方式或是把**識別字串**放在URL中編碼，而**識別資料**主要是放在settings.py中的**SECRET_KEY**這個常數中
- 網站實際上線時，這個**常數**就必須要另外放在**安全的地方**，以**開啟檔案**的方式讀取，或是以**環境變數**的方式操作，才不會讓惡意人士有**偽造Session連結**的可能。

開始使用Session

- 預設的Django的Session後端會使用到資料庫
- 而主要的操作也可以選擇使用cookie-based以及file-based的方式，使用預設的方式就可以了

開始使用Session

- ▶ 如何取得Session中的內容以及如何設定變數資料到Session中
 - ▶ 使用字典的方式操作如設定使用者的名稱 (`username`) ，只要使用以下的方式即可：
 - ▶ `request.session['username']='使用者名稱'`
- ▶ 而要取出也是：
 - ▶ `username = request.session['username']`

建立使用者資料表

- 在這一節中我們將以個人化網站為例子
 - 結合資料庫的功能
 - 透過Session變數的設定與提取，提供讓使用者可以登入的功能
 - 在登入之後可以依照自己權限，取得專屬的網頁的資料
- 建立使用者資料表，在models.py中建立一個User類別，如下：
 - `class User(models.Model):`
 - `name = models.CharField(max_length=20, null=False)`
 - `email = models.EmailField()`
 - `password = models.CharField(max_length=20, null=False)`
 - `enabled = models.BooleanField(default=False)`

 - `def __str__(self):`
 - `return self.name`

建立使用者資料表

- ▶ 在 `admin.py` 中要加入以下這一行：
 - ▶ `admin.site.register(models.User)`
- ▶ 把選單組織如下所示（選單的內容是放在 `header.html` 檔案中）：
 - ▶ `<!-- header.html (Session_cookies) -->`
 - ▶ `<nav class='navbar navbar-default'>`
 - ▶ `<div class='container-fluid'>`
 - ▶ `<div class='navbar-header'>`
 - ▶ `<div class='navbar-brand' align=center>`
 - ▶ `分享日記`
 - ▶ `</div>`
 - ▶ `</div>`

建立使用者資料表

```
➤ <ul class='nav navbar-nav'>
➤     <li class='active'><a href='/'>Home</a></li>
➤     {% if username %}
➤     <li><a href='/userinfo'>個人資料</a></li>
➤     <li><a href='/post'>寫日記</a></li>
➤     <li><a href='/contact'>連絡管理員</a></li>
➤     <li><a href='/logout'>登出</a></li>
➤     {% else %}
➤     <li><a href='/login'>登入</a></li>
➤     {% endif %}
➤     <li><a href='/admin'>後台管理</a></li>
➤ </ul>
➤ </div>
➤ </nav>
```

建立使用者資料表

- ▶ 讓網頁使用者登入需要有一個表單，因此在forms.py中需加入此一類別：
 - ▶ `class LoginForm(forms.Form):`
 - ▶ `username = forms.CharField(label='姓名', max_length=10)`
 - ▶ `password = forms.CharField(label='密碼', widget=forms.PasswordInput())`

建立使用者資料表

- ▶ 在login.html中要配合如下：
 - ▶ <!-- login.html (Session_cookies) -->
 - ▶ {% extends "base.html" %}
 - ▶ {% block title %}登入分享日記{% endblock %}
 - ▶ {% block content %}
 - ▶ <div class='container'>
 - ▶ {% if message %}
 - ▶ <div class='alert alert-warning'>{{ message }}</div>
 - ▶ {% endif %}
 - ▶ <div class='row'>
 - ▶ <div class='col-md-12'>
 - ▶ <div class='panel panel-default'>
 - ▶ <div class='panel-heading' align=center>
 - ▶ <h3>登入我的私人日記</h3>
 - ▶ </div>
 - ▶ </div>

建立使用者資料表

- `<form action='.' method='POST'>`
- `{% csrf_token %}`
- `<table>`
- `{{ login_form.as_table }}`
- `</table>`
- `<input type='submit' value='登入'>
`
- `</form>`
- `</div>`
- `{% endblock %}`

建立使用者資料表

➤ 在 `views.login` 中就可以透過以下所示的程式碼運作：

➤ `from django.shortcuts import redirect`

➤ `def login(request):`

➤ `if request.method == 'POST':`

➤ `login_form = forms.LoginForm(request.POST)`

➤ `if login_form.is_valid():`

➤ `login_name = request.POST['username'].strip()`

➤ `login_password = request.POST['password']`

➤ `try:`

➤ `user = models.User.objects.get(name = login_name)`

建立使用者資料表

```
➤ if user.password == login_password:  
➤     request.session['username'] = user.name  
➤     request.session['useremail'] = user.email  
➤     return redirect('/')  
➤ else:  
➤     message = "密碼錯誤，請再檢查一次"  
➤ except:  
➤     message = "找不到使用者"  
➤ else:  
➤     message = "請檢查輸入的欄位內容"  
➤ else:  
➤     login_form = forms.LoginForm()  
➤ return render(request, 'login.html', locals())
```

建立使用者資料表

- 非常重要的步驟如下，檢查密碼正確後
- 在`request.session`中設定`username`和`useremail`這兩個Session變數，這兩個變數就會在設定的Session存續時間內（預設是瀏覽器關閉之前）都可以被取得：
 - `if user.password == login_password:`
 - `request.session['username'] = user.name`
 - `request.session['useremail'] = user.email`
 - `return redirect('/')`

建立使用者資料表

- 在views.index函數設定程式如下
 - def `index`(request, pid=None, del_pass=None):
 - if 'username' in request.session:
 - username = request.session['username']
 - useremail = request.session['useremail']
 - return render(request, 'index.html', locals())
- 檢查' username'有沒有存在於Session中
- 如果有就把username以及useremail都取出來，再送去index.html中渲染網頁：

建立使用者資料表

- ▶ `index.html`如下所示：
 - ▶ `<!-- index.html (Session_cookies) -->`
 - ▶ `{% extends "base.html" %}`
 - ▶ `{% block title %}分享日記{% endblock %}`
 - ▶ `{% block content %}`
 - ▶ `<div class='container'>`
 - ▶ `{% if message %}`
 - ▶ `<div class='alert alert-warning'>{{ message }}</div>`
 - ▶ `{% endif %}`
 - ▶ `<div class='row'>`
 - ▶ `<div class='col-md-12'>`
 - ▶ `<div class='panel panel-default'>`
 - ▶ `<div class='panel-heading' align=center>`
 - ▶ `<h3>我的私人日記</h3>`
 - ▶ `</div>`
 - ▶ `</div>`
 - ▶ `{% if username %}`
 - ▶ `歡迎 : {{username}}`
 - ▶ `{% endif %}`
 - ▶ `</div>`
 - ▶ `{% endblock %}`

設定路徑

- ▶ 不要忘記編輯 `urls.py`
- ▶ `path('login/', views.login),`

建立使用者資料表



The screenshot shows a web browser window with the following elements:

- Browser Tab:** 登入分享日記
- Address Bar:** localhost:8000/login/
- Navigation:** Back, Forward, Refresh, Home icons.
- Page Navigation:** 分享日記, Home, 登入, 後台管理.
- Main Content:** 登入我的私人日記
- Form:** 姓名: , 密碼: , 登入 button.
- Footer:**  Copyright 2020 jfanc. All rights reserved. 阿傑股份有限公司 每天都不離上班

建立使用者資料表

The screenshot shows a web browser window with the following elements:

- Browser Tab:** 登入分享日記
- Address Bar:** localhost:8000/login/
- Navigation:** 分享日記, Home, 登入, 後台管理
- Message:** 找不到使用者 (User not found)
- Form:** 登入我的私人日記 (Login to my private diary). Fields include 姓名: 王大明 (Name: Wang Daming) and 密碼: (Password). A tooltip over the password field says "請填寫此欄位" (Please fill in this field).
- Footer:** Copyright 2020 jfanc. All rights reserved. 陶傑股份有限公司 (Tao Jie Co., Ltd.)

登入分享日記

Add user | Django site admin

localhost:8000/admin/mysite/user/

Django administration

WELCOME, ADMIN. VIEW SITE

Home > Mysite > Users > Add user

Add user

Name:

Email:

Password:

Enabled

登入分享日記

Add user | Django site admin

localhost:8000/admin/mysite/user/ 110%

Django administration

WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home > Mysite > Users > Add user

Add user

Please correct the error below.

Name:

Email: This field is required.

Password:

Enabled

Save and add another Save and continue editing SAVE

登入分享日記 x Select user to change | Django site x +

localhost:8000/admin/mysite/user/ 110% 搜尋

Django administration

WELCOME, **ADMIN**. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Home > Mysite > Users

✔ The user "Apple" was added successfully.

Select user to change

ADD USER +

Action: Go 0 of 2 selected

<input type="checkbox"/>	USER
<input type="checkbox"/>	Apple
<input type="checkbox"/>	王大明

2 users

建立使用者資料表

密碼錯誤時顯示的訊息



建立使用者資料表

順利登入網站時的首頁畫面



建立使用者資料表

- 如果是Session的話如何處理登出呢？只需將Session清除並且導回至login.html頁面即可，`views.logout`處理函數如下所示：
 - `from django.contrib.sessions.models import Session`
 - `def logout(request):`
 - `if 'username' in request.session:`
 - `Session.objects.all().delete()`
 - `return redirect('/login/')`
 - `return redirect('/')`


判斷如果為登入狀態則將**Session**中的所有資料清除且回到登入頁面，否則則導回首頁。

建立使用者資料表

➤ 顯示個人資料的網頁在這個範例中簡單設計如下：

```
➤ def userinfo(request):
➤     if 'username' in request.session:
➤         username = request.session['username']
➤     else:
➤         return redirect('/login/')
➤     try:
➤         userinfo = models.User.objects.get(name=username)
➤     except:
➤         pass
➤     return render(request, 'userinfo.html', locals())
```

資料庫中取得所有使用者的資訊

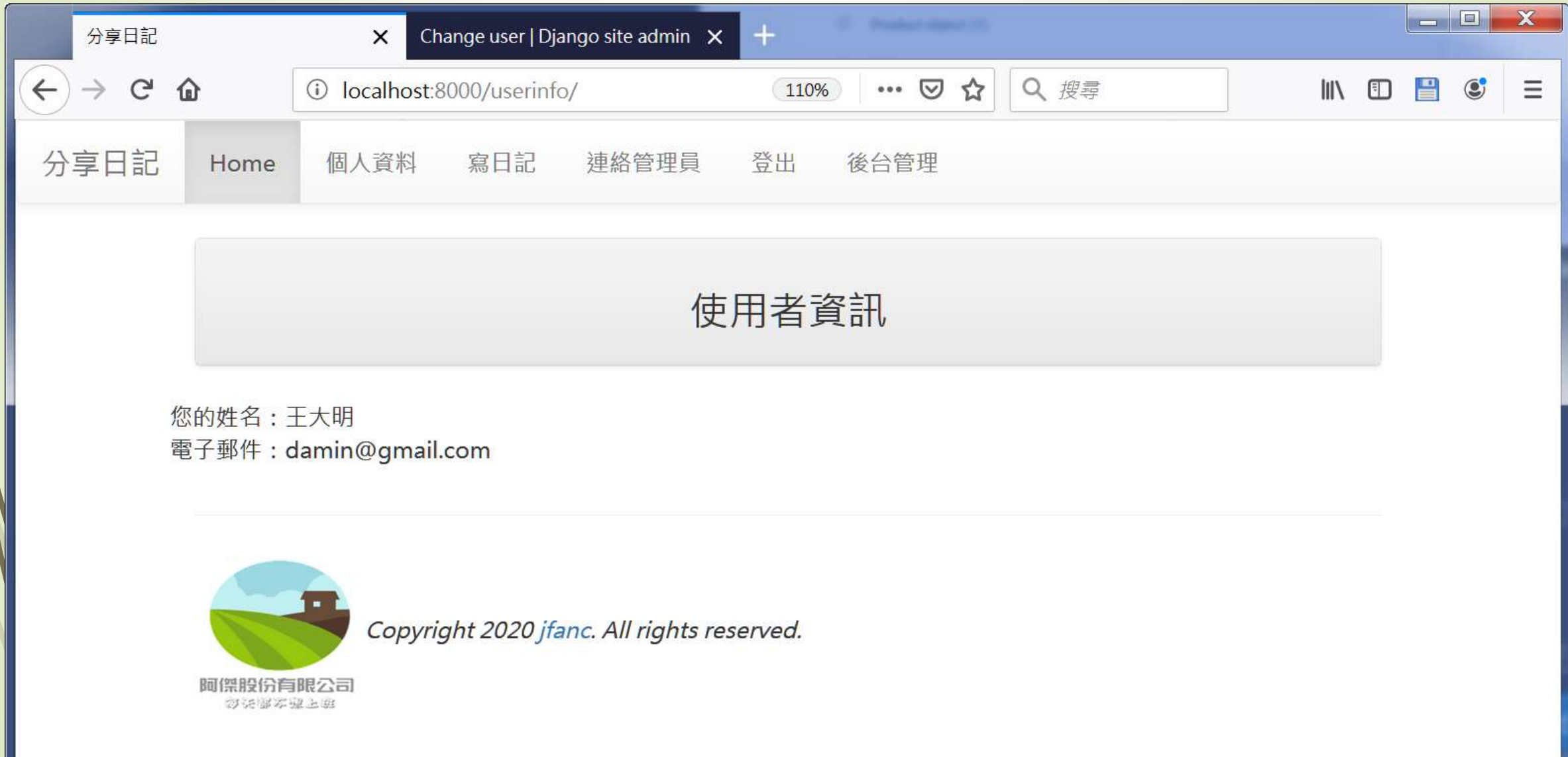


建立使用者資料表

- ▶ 至於 `userinfo.html` 的內容如下：
 - ▶ `{% extends "base.html" %}`
 - ▶ `{% block title %}分享日記{% endblock %}`
 - ▶ `{% block content %}`
 - ▶ `<div class='container'>`
 - ▶ `<div class='row'>`
 - ▶ `<div class='col-md-12'>`
 - ▶ `<div class='panel panel-default'>`
 - ▶ `<div class='panel-heading' align=center>`
 - ▶ `<h3>使用者資訊</h3>`
 - ▶ `</div>`
 - ▶ `</div>`
 - ▶ `</div>`
 - ▶ `<p>`
 - ▶ 您的姓名：`{{ userinfo.name }}``
`
 - ▶ 電子郵件：`{{ userinfo.email }}`
 - ▶ `</p>`
 - ▶ `</div>`
 - ▶ `{% endblock %}`

調整一下 `urls.py`

- `path('userinfo/', views.userinfo),`
- `path('logout/', views.logout),`



分享日記

Change user | Django site admin

localhost:8000/userinfo/


110%

搜尋

分享日記 Home 個人資料 寫日記 連絡管理員 登出 後台管理

使用者資訊

您的姓名：王大明
電子郵件：damin@gmail.com

 Copyright 2020 jfanc. All rights reserved.
阿傑股份有限公司
砂朥越不墜上座

整合 Django 的訊息顯示框架 Messages Framework

- Django 提供了一個 messages framework
- 只要引入 `from django.contrib import messages` 之後，就可以透過它提供的函數及框架，自動做到橫跨各網頁間的訊息顯示了。
- 它主要提供 2 個函數，分別是（請留意 message 後面有沒有加上 s）：
 - `from django.contrib import messages`
 - `messages.add_message(request, messages.INFO, '要顯示的字串')`
 - `messages.get_messages(request)`

整合Django的訊息顯示框架Messages Framework

- 其中add_message用來加上一段訊息，而訊息的內容型態預設還分成以下幾個等級：
 - DEBUG
 - INFO
 - SUCCESS
 - WARNING
 - ERROR
- 對應到以上的不同訊息等級也可以分別使用以下的函數來簡化：
 - messages.debug(request, '除錯訊息字串')
 - messages.info(request, '資訊訊息字串')
 - messages.success(request, '成功訊息字串')
 - messages.warning(request, '警告訊息字串')
 - messages.error(request, '錯誤訊息字串')

整合Django的訊息顯示框架Messages Framework

- 使用這個機制，我們在[views.login](#)中就可以修改如下：
 - `from django.contrib import messages`
 - `def login(request):`
 - `if request.method == 'POST':`
 - `login_form = forms.LoginForm(request.POST)`
 - `if login_form.is_valid():`
 - `login_name=request.POST['username'].strip()`
 - `login_password=request.POST['password']`
 - `try:`
 - `user = models.User.objects.get(name=login_name)`

整合 Django 的訊息顯示框架 Messages Framework

38

```
➤ if user.password == login_password:
➤     request.session['username'] = user.name
➤     request.session['useremail'] = user.email
➤     messages.add_message(request, messages.SUCCESS, '成功登入了')
➤     return redirect('/')
➤ else:
➤     messages.add_message(request, messages.WARNING, '密碼錯誤，請再檢查一次')
➤ except:
➤     messages.add_message(request, messages.WARNING, '找不到使用者')
➤ else:
➤     messages.add_message(request, messages.INFO, '請檢查輸入的欄位內容')
➤ else:
➤     login_form = forms.LoginForm()

➤ return render(request, 'login.html', locals())
```

整合Django的訊息顯示框架Messages Framework

- 為了方便和Bootstrap framework中使用的Alert訊息系統相同
- 我們只使用了SUCCESS、WARNING以及INFO
- 因此，在login.html中就可以修改為以下所示的樣子：
 - <!-- login.html (Session_Cookies) -->
 - {% extends "base.html" %}
 - {% block title %}登入分享日記{% endblock %}
 - {% block content %}
 - <div class='container'>
 - {% for message in messages %}
 - <div class='alert alert-{{message.tags}}'>{{ message }}</div>
 - {% endfor %}

整合 Django 的訊息顯示框架 Messages Framework

```
➤ <div class='row'>
➤   <div class='col-md-12'>
➤     <div class='panel panel-default'>
➤       <div class='panel-heading' align=center>
➤         <h3>登入我的私人日記</h3>
➤       </div>
➤     </div>
➤   </div>
➤ <form action='.' method='POST'>
➤   {% csrf_token %}
➤   <table>
➤     {{ login_form.as_table }}
➤   </table>
➤   <input type='submit' value='登入'><br/>
➤ </form>
➤ </div>
➤ {% endblock %}
```

整合Django的訊息顯示框架Messages Framework

➤ `index.html`前面加入messages，如下所示：

➤ `<!-- index.html (Sessions_Cookies) -->`

➤ `{% extends "base.html" %}`

➤ `{% block title %}分享日記{% endblock %}`

➤ `{% block content %}`

➤ `<div class='container'>`

➤ `{% for message in messages %}`

➤ `<div class='alert alert-{{message.tags}}'> {{ message }}</div>`

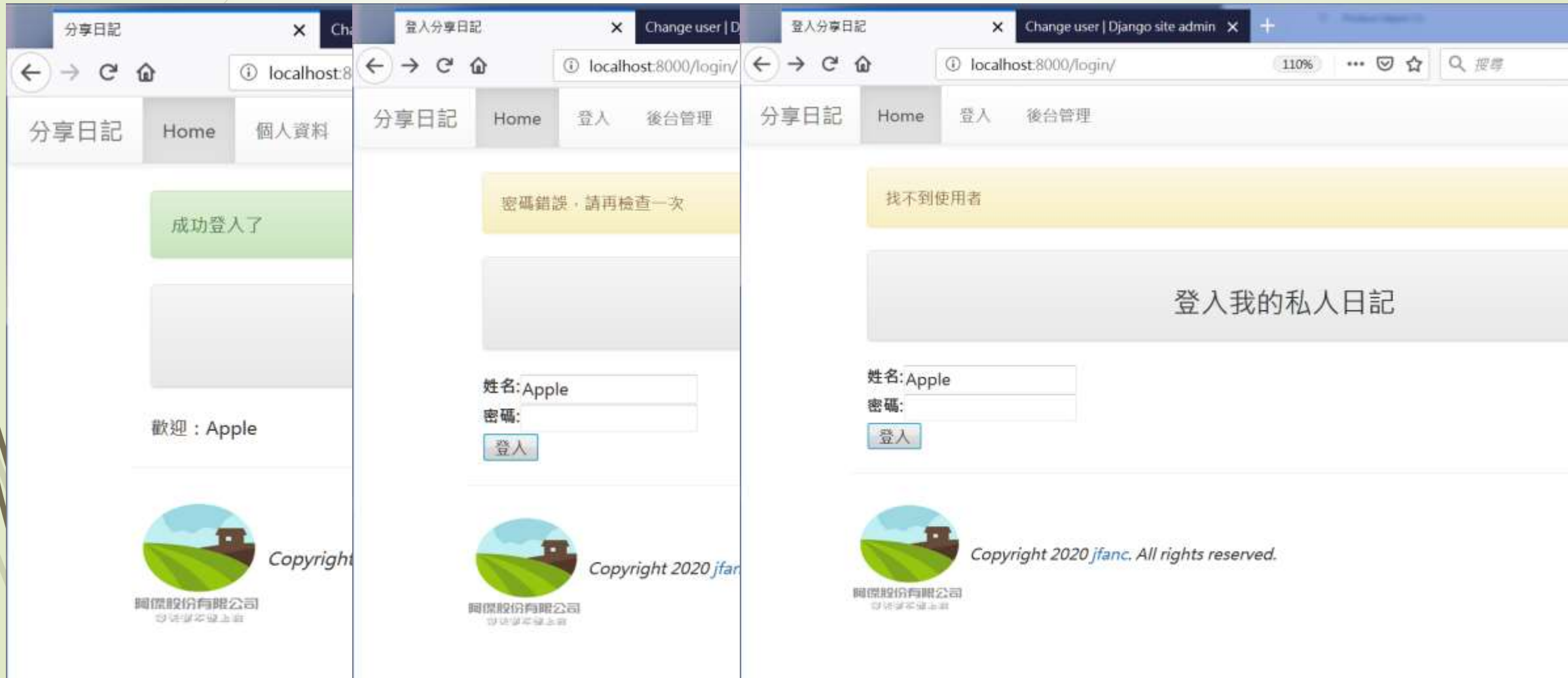
➤ `{% endfor %}`

整合Django的訊息顯示框架Messages Framework

- `<div class='row'>`
- `<div class='col-md-12'>`
- `<div class='panel panel-default'>`
- `<div class='panel-heading' align=center>`
- `<h3>我的私人日記</h3>`
- `</div>`
- `</div>`
- `{% if username %}`
- `歡迎 : {{username}}`
- `{% endif %}`
- `</div>`
- `{% endblock %}`

整合Django的訊息顯示框架Messages Framework

- 使用messages framework顯示登入成功的訊息

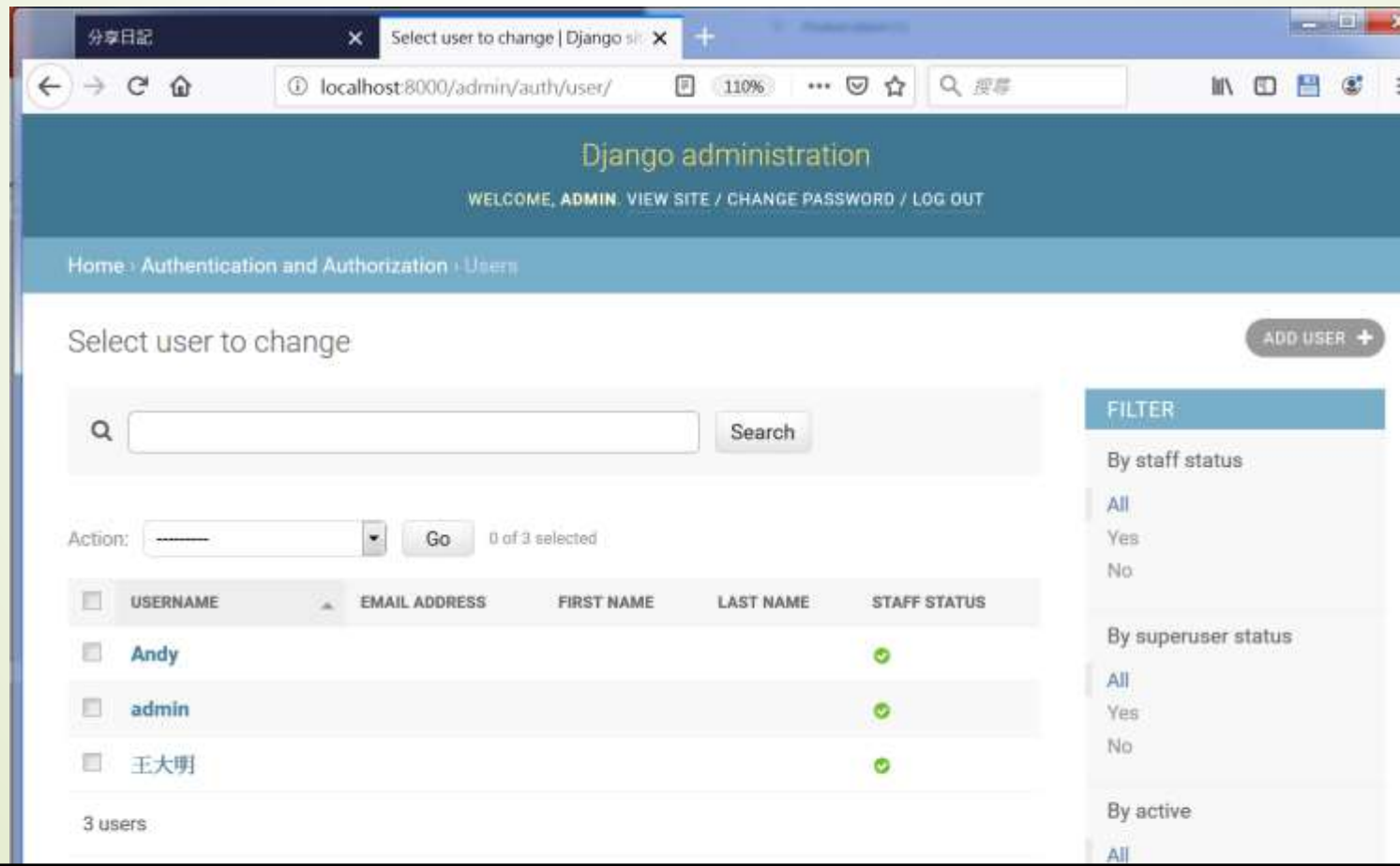


Django auth使用者驗證

- 在上一節中我們透過Session變數的操作設計出支援使用者登入以及登出，並可以依照登入與否的狀態來顯示出相對應的內容，而且可以防止未登入的訪客直接以網址的方式前往未經授權的介面。
- 但其實這些操作，如果使用Django使用在admin管理網頁的這套系統，則運作起來會更順暢，而且程式的設計也可以再進一步簡化，不需要再自行設定Session變數了。

使用Django的使用者驗證系統

- 在Django本身內建了使用者的登入/登出的功能
- admin管理網頁的Users介面，使用Django的使用者驗證系統



使用Django的使用者驗證系統

- Django物件是在auth.models中，所以要使用之前需引入如下：
 - `from django.contrib.auth.models import User`
- 在Django預設的User物件中欄位有：
 - username
 - password
 - email
 - first_name
 - last_name

使用Django的使用者驗證系統

- ▶ 使用以下的程式碼來[建立一個新的使用者](#)：
 - ▶ `from django.contrib.auth.models import User`
 - ▶ `user = User.objects.create_user('pikachu', 'pikachu@ntu.edu.tw', 'mypassword')`
- ▶ 需要修改其中的任一欄位資料的話，和之前操作Model實例變數的方法是一樣的，如下所示：
 - ▶ `user.last_name = 'Pokemon'`
 - ▶ `user.save()`

使用Django的使用者驗證系統

- 先使用admin管理網頁來修改以及新增使用者資料，接下來我們要做的是利用在User中的資料來做使用者登入以及登出的實作
- 請留意本小節的操作對象是User類別
- 是在django.contrib.auth.models中的類別
- 和在前一節中在models.py中定義的User是不同的。
- 在models.py中自訂的User類別在views.py中是以models.User來操作
- 而在auth.models中的則是直接用User拿來使用

使用Django的使用者驗證系統

- django.contrib.auth中提供了3個主要的函數
 - Authenticate
 - login
 - logout
- 利用Django的auth機制，views.login的內容修改為如下：
 - `from django.contrib.auth import authenticate`
 - `from django.contrib import auth`
 - `from django.contrib.auth.decorators import login_required`
 - `def login(request):`
 - `if request.method == 'POST':`
 - `login_form = forms.LoginForm(request.POST)`

使用Django的使用者驗證系統

```
➤ if login_form.is_valid():
➤     login_name=request.POST['username'].strip()
➤     login_password=request.POST['password']
➤     user = authenticate(username=login_name, password=login_password)
➤     if user is not None:
➤         if user.is_active:
➤             auth.login(request, user)
➤             print("success")
➤             messages.add_message(request, messages.SUCCESS, '成功登入了')
➤             return redirect('/')
➤         else:
➤             messages.add_message(request, messages.WARNING, '帳號尚未啟用')
➤     else:
➤         messages.add_message(request, messages.WARNING, '登入失敗')
➤     else:
➤         messages.add_message(request, messages.INFO, '請檢查輸入的欄位內容')
➤ else:
➤     login_form = forms.LoginForm()
➤ return render(request, 'login.html', locals())
```

使用Django的使用者驗證系統

- 上述程式重點處理步驟：
 - 把從表單中取得的login_name和login_password
 - 透過authenticate進行驗證的工作
 - 驗證成功此函數會傳回該使用者的資料放在user變
 - 驗證失敗則回傳None
 - 成功登入之後可以再使用user.is_active來檢查此帳號是否有效
 - (註：is_active 在Django 1.10之後authenticate(...)會自動檢查is_active是否是False，如果user不存在或is_active = False則回傳None。
<https://stackoverflow.com/questions/43184151/django-user-is-active>)
 - 如果一切都通過，使用auth.login(request, user)把此使用者的資料存入Session中，供接下來其它網頁中使用
 - (註：在views.py中使用了login以及logout這兩個自訂函數名稱，為了避免和auth中的兩個同名函數衝突，在這裡使用auth.login以及auth.logout)

使用Django的使用者驗證系統

- ▶ 在 `views.index` 函數中使用 `is_authenticated` 來檢查使用者是否有登入，如下所示：
 - ▶ `def index(request, pid=None, del_pass=None):`
 - ▶ `if request.user.is_authenticated:`
 - ▶ `username = request.user.username`
 - ▶ `messages.get_messages(request)`
 - ▶ `return render(request, 'index.html', locals())`

使用Django的使用者驗證系統

➤ 顯示個人資料的部份，`views.userinfo`的內容如下：

```
➤ from django.contrib.auth.models import User
➤ @login_required(login_url='/login/')
➤ def userinfo(request):
➤     if request.user.is_authenticated:
➤         username = request.user.username
➤         try:
➤             userinfo = User.objects.get(username=username)
➤         except:
➤             pass
➤     return render(request, 'userinfo.html', locals())
```

使用Django的使用者驗證系統

- 上述程式碼第一行的decorator `@login_required`
- 是auth驗證機制提供的一個非常方便的使用法
- 用來告訴Django接下來的這個處理函數的內容是**需要登入過才能夠瀏覽的**
- 如果還沒有瀏覽就想要執行這一頁，請先到括弧中指定的 `login_url` 登入網址先做過登入再說

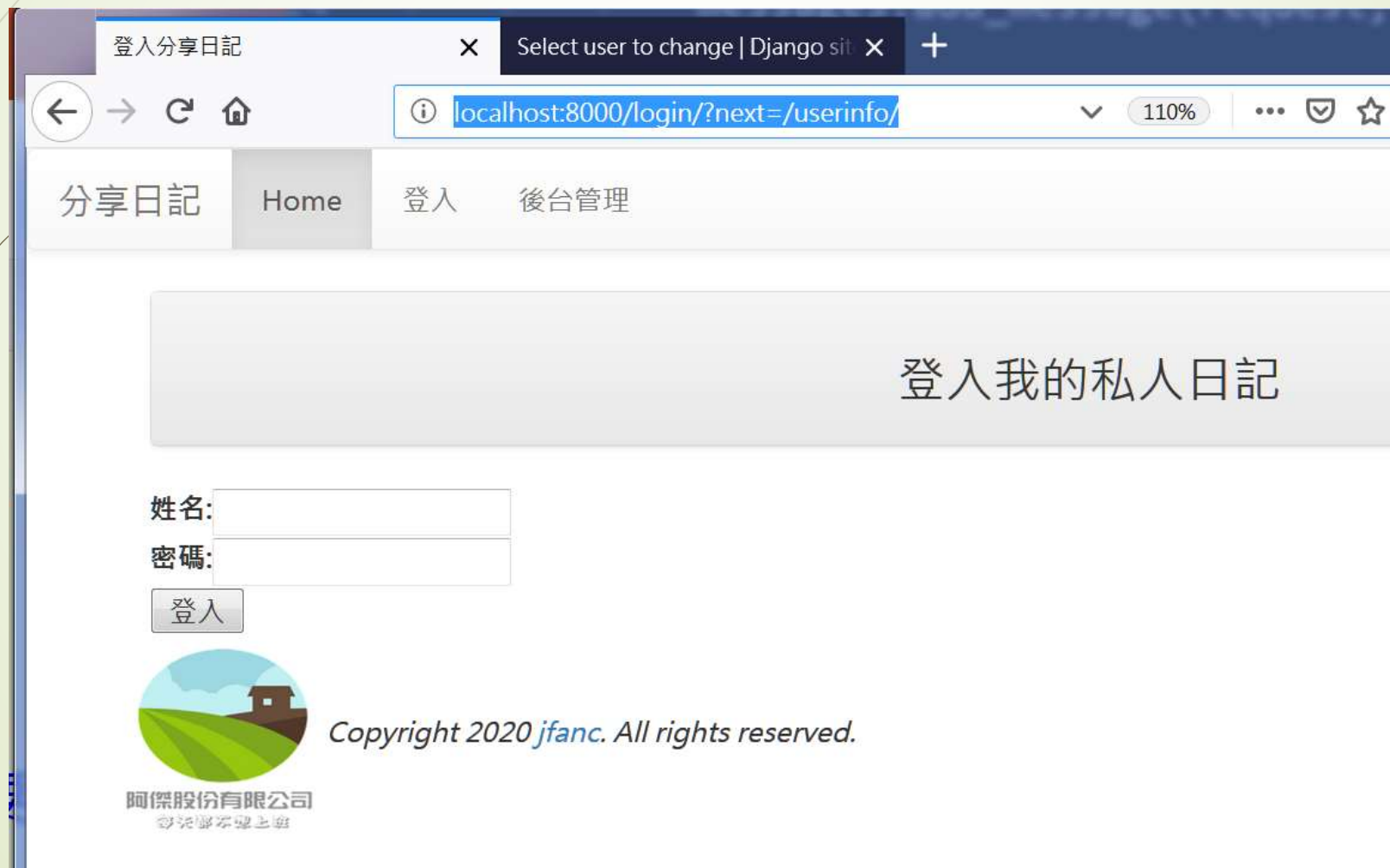
使用Django的使用者驗證系統

- 使用這個使用者驗證機制，要登出就非常簡單了，如下所示：
 - `def logout(request):`
 - `auth.logout(request)`
 - `messages.add_message(request, messages.INFO, "成功登出了")`
 - `return redirect('/')`



使用Django的使用者驗證系統

- 在未登入的情況下使用網址直接瀏覽userinfo的情形



增加User的欄位

- 前一節介紹的，在auth.User中預設的欄位只有
- username, password, email, first_name, 以及last_name等5個
- 就一般網站的應用上是不足的
- 解決方法：
 - 在models中建立一個新的Model
 - 然後把User使用一對一的對應連結在一起
 - 記得先把原本的 `class User` 刪除或註解掉

增加User的欄位

- ▶ 假設我們要創建一個使用者類別叫做Profile
- ▶ 並要增加身高、性別以及網站等3個欄位
- ▶ 那麼在models.py中可以這樣定義：
 - ▶ # `_*_ encoding: utf-8 *_*`
 - ▶ `from django.db import models`
 - ▶ `from django.contrib.auth.models import User`

 - ▶ `class Profile(models.Model):`
 - ▶ `user = models.OneToOneField(User, on_delete=models.CASCADE)`
 - ▶ `height = models.PositiveIntegerField(default=160)`
 - ▶ `male = models.BooleanField(default=False)`
 - ▶ `website = models.URLField(null=True)`
 - ▶ `def __str__(self):`
 - ▶ `return self.user.username`

增加User的欄位

- ▶ 在admin.py中加入管理網頁的登記作業，如下所示：
 - ▶ `from django.contrib import admin`
 - ▶ `from mysite import models`
 - ▶ `admin.site.register(models.Profile)`
 - ▶ `#admin.site.register(models.User) #註解或刪除`

The screenshot shows a web browser window with the Django administration interface. The browser's address bar shows the URL `localhost:8000/admin/`. The page title is "Django administration" and the user is logged in as "ADMIN". The breadcrumb trail is "Home > Mysite > Profiles > Add profile".

The "Add profile" form contains the following fields:

- User:** A dropdown menu with a pencil icon and a plus sign to the right.
- Height:** A text input field containing the value "160" and a small icon to the right.
- Male:** A checkbox that is currently unchecked.
- Website:** An empty text input field.

At the bottom of the form, there are three buttons: "Save and add another", "Save and continue editing", and "SAVE".

The screenshot shows a web browser window with two tabs. The active tab is titled "Select profile to change | Django". The address bar shows "localhost:8000/admin/mys" with a zoom level of 110%. The page content includes a header for "Django administration" with navigation links for "WELCOME, ADMIN.", "VIEW SITE", "CHANGE PASSWORD", and "LOG OUT". A breadcrumb trail reads "Home > Mysite > Profiles". A green success message states: "The profile 'Andy' was added successfully." Below this is a section titled "Select profile to change" with an "ADD PROFILE +" button. An "Action:" dropdown menu is set to "-----" with a "Go" button and "0 of 1 selected" text. A table lists profiles with checkboxes:

<input type="checkbox"/>	PROFILE
<input type="checkbox"/>	Andy

At the bottom, it indicates "1 profile".

顯示新增加的User欄位

- ▶ 利用Profile在User資料表增加了新的欄位，當使用者順利登入網站之後，在localhost:8000/userinfo中應該要顯示更多的資料
- ▶ 找到user，然後再以此找出Profile，views.userinfo的內容如下：
 - ▶ `from django.contrib.auth.models import User`
 - ▶ `@login_required(login_url='/login/')`
 - ▶ `def userinfo(request):`
 - ▶ `if request.user.is_authenticated:`
 - ▶ `username = request.user.username`
 - ▶ `try:`
 - ▶ `user = User.objects.get(username=username)`
 - ▶ `userinfo = models.Profile.objects.get(user=user)`
 - ▶ `except:`
 - ▶ `pass`
 - ▶ `template = get_template('userinfo.html')`
 - ▶ `html = template.render(locals())`
 - ▶ `return HttpResponse(html)`

顯示新增加的User欄位

➤ `userinfo.html`則如下所示：

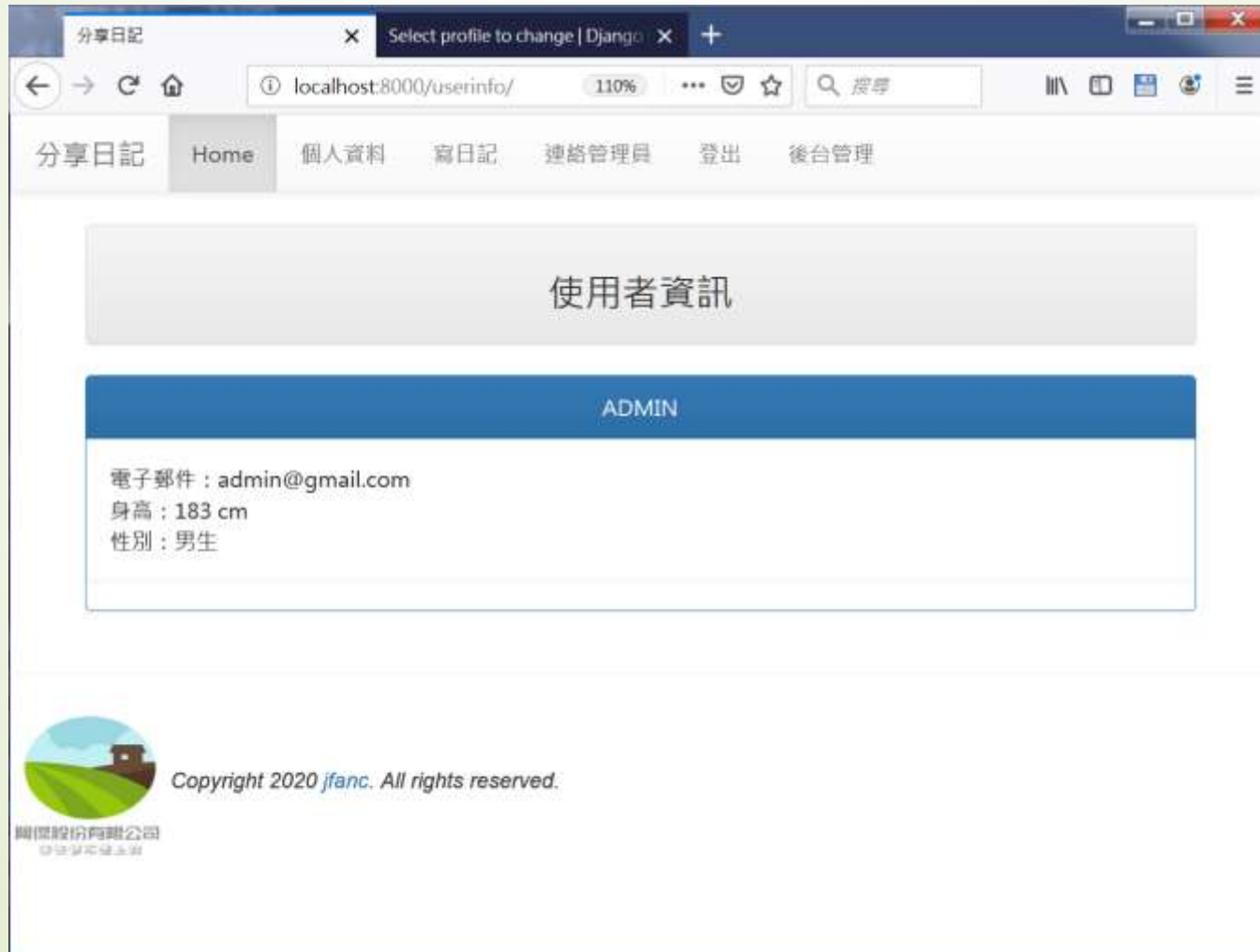
- `{% extends "base.html" %}`
- `{% block title %}分享日記{% endblock %}`
- `{% block content %}`
- `<div class='container'>`
- `<div class='row'>`
- `<div class='col-md-12'>`
- `<div class='panel panel-default'>`
- `<div class='panel-heading' align=center>`
- `<h3>使用者資訊</h3>`
- `</div>`
- `</div>`
- `</div>`
- `</div>`

增加顯示新增加的User欄位User的欄位

```
➤ <div class='row'>
➤   <div class='col-md-12'>
➤     <div class='panel panel-primary'>
➤       <div class='panel-heading' align=center>
➤         {{ userinfo.user.username | upper }}
➤       </div>
➤       <div class='panel panel-body'>
➤         電子郵件 : {{ userinfo.user.email }}<br/>
➤         身高 : {{ userinfo.height }} cm<br/>
➤         性別 : {{ userinfo.male | yesno:"男生,女生"}}
➤       </div>
➤     </div>
➤   </div>
➤ </div>
➤ </div>
➤ {% endblock %}
```

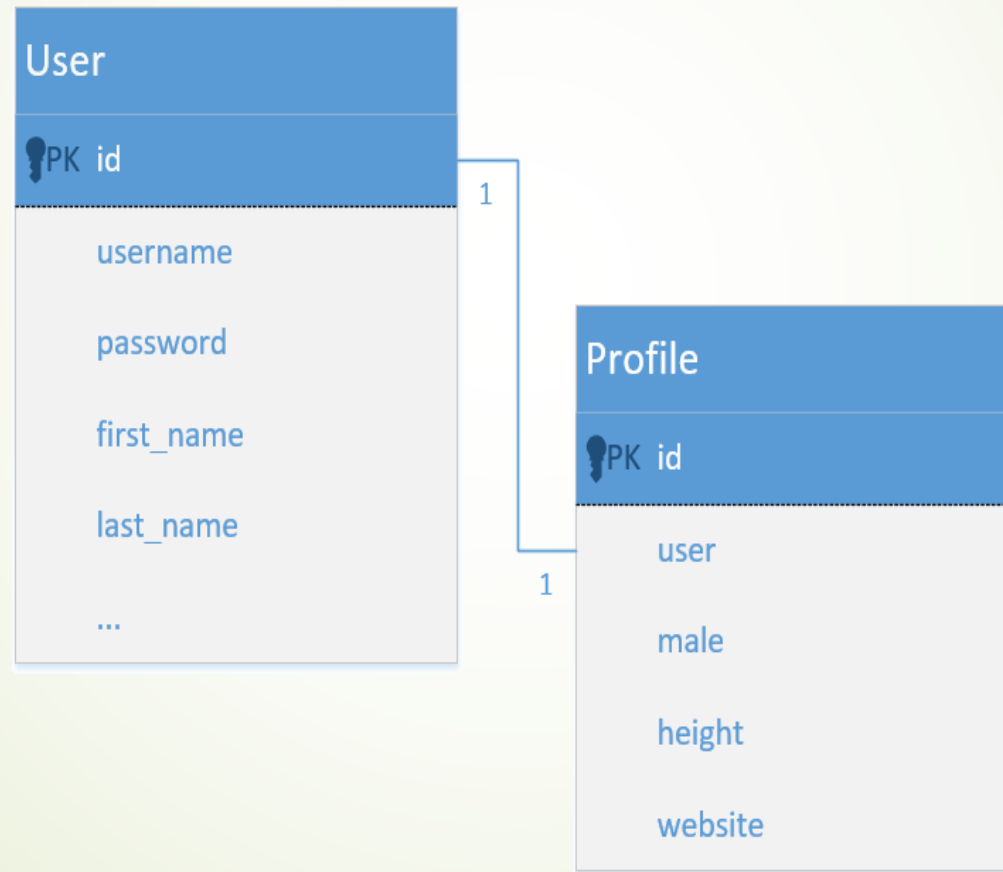
顯示新增加的User欄位

- ➡ 新版的使用者資訊網頁內容



顯示新增加的User欄位

➔ User和Profile資料表的關係

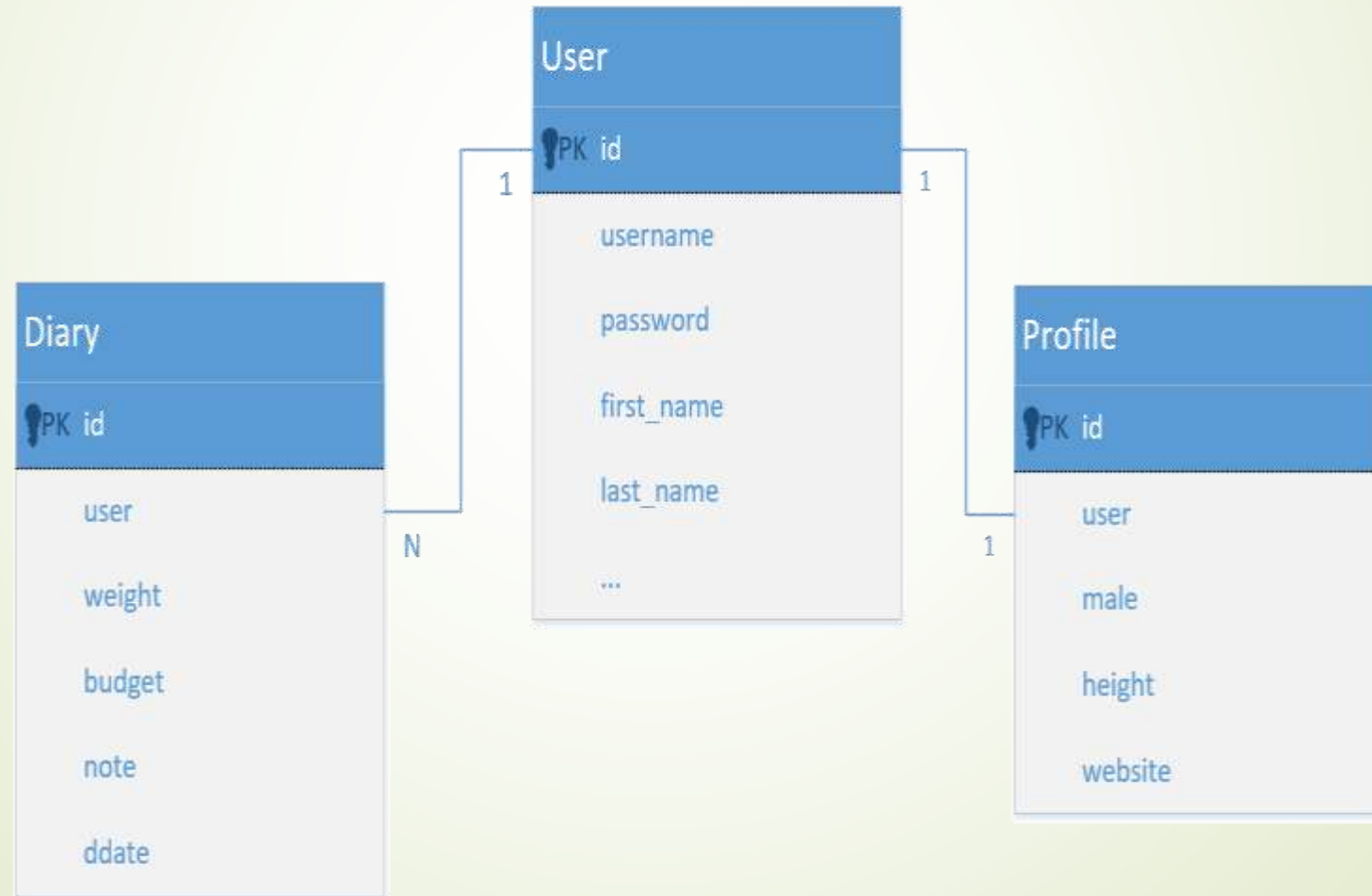


應用auth使用者驗證存取資料庫

- 更進一步地運用auth.User的驗證，新增另外一張資料庫用來儲存更多的內容，並完成本堂課的分享日記範例檔案的初步程式，基本功能如下：
 - 使用者登入
 - 登入之後，可以張貼自己的日記
- 因此需要有一張記錄每天日記的資料表Diary
- 它和User以及Profile的關係如下圖

應用auth使用者驗證存取資料庫

➔ Diary、User和Profile之間的關係



應用auth使用者驗證存取資料庫

- 在 `models.py` 中建立一個新的模型 `Diary`，如下所示：
 - `class Diary(models.Model):`
 - `user = models.ForeignKey(User, on_delete=models.CASCADE)`
 - `budget = models.FloatField(default=0)`
 - `weight = models.FloatField(default=0)`
 - `note = models.TextField()`
 - `ddate = models.DateField()`
 - `def __str__(self):`
 - `return "{}({})".format(self.ddate, self.user)`

應用auth使用者驗證存取資料庫

- 寫日記需要使用表單才能夠輸入到資料庫中，因此使用 `ModelForm`，在 `forms.py` 中編寫，其內容如下：
 - `class DateInput(forms.DateInput):`
 - `input_type = 'date'`

 - `class DiaryForm(forms.ModelForm):`
 - `class Meta:`
 - `model = models.Diary`
 - `fields = ['budget', 'weight', 'note', 'ddate']`
 - `widgets = {`
 - `'ddate': DateInput(),`
 - `}`

應用auth使用者驗證存取資料庫

- `def __init__(self, *args, **kwargs):`
- `super(DiaryForm, self).__init__(*args, **kwargs)`
- `self.fields['budget'].label = '今日花費(元)'`
- `self.fields['weight'].label = '今日體重(KG)'`
- `self.fields['note'].label = '心情留言'`
- `self.fields['ddate'].label = '日期'`

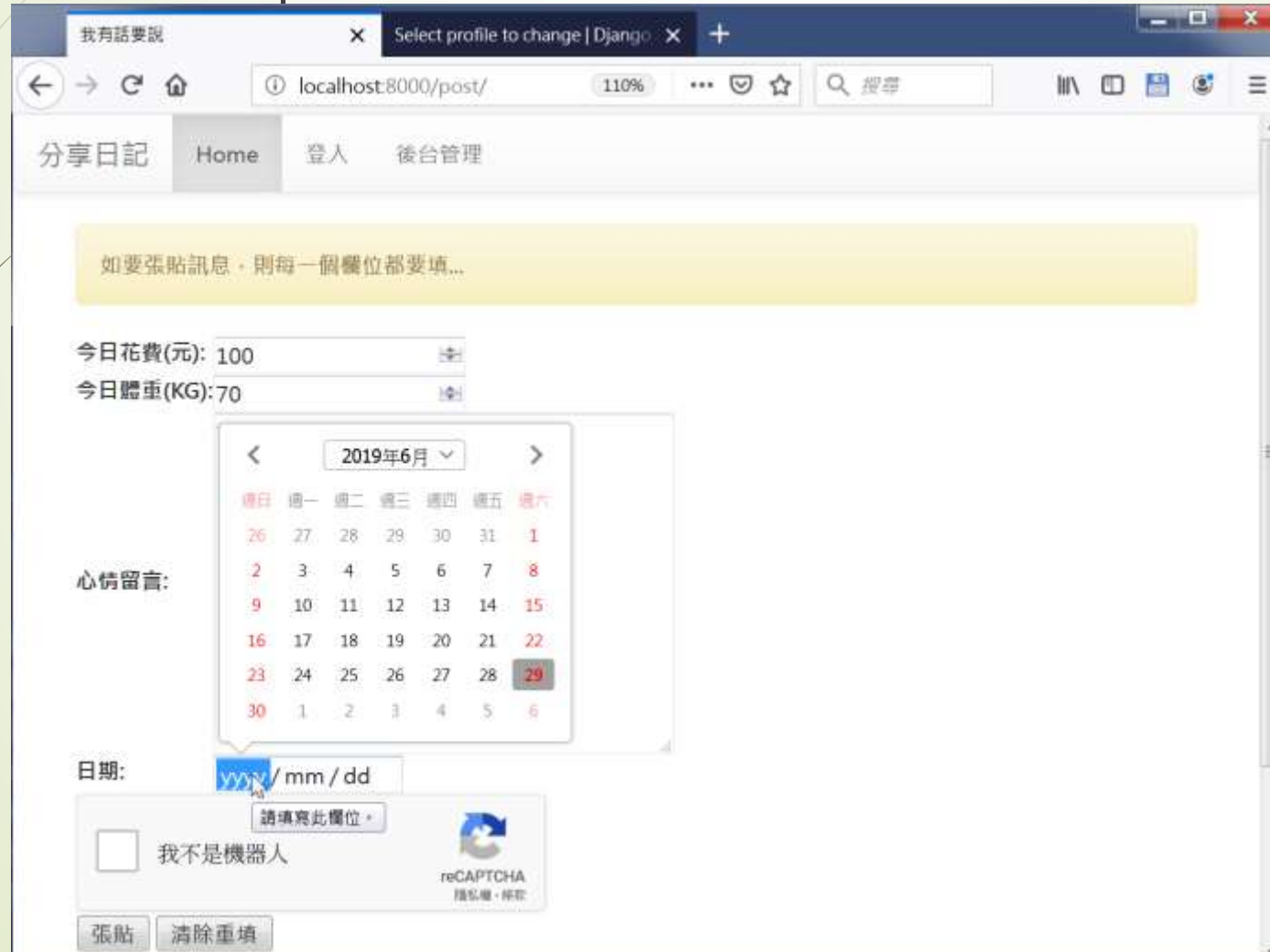
小練習

- 修改 `views.py` 的 `def posting(request)`:
- 仿造 `post2db` 修改如下：
- `diary_form = forms.DiaryForm(request.POST)`
- `diary_form.save()`
- `diary_form = forms.DiaryForm()`
- `return render(request, 'posting.html', locals())`

- 修改 `posting.html` 仿造 `post2db.html` 修改如下：
- `{{ diary_form.as_table }}`

應用auth使用者驗證存取資料庫

► DateInput的介面

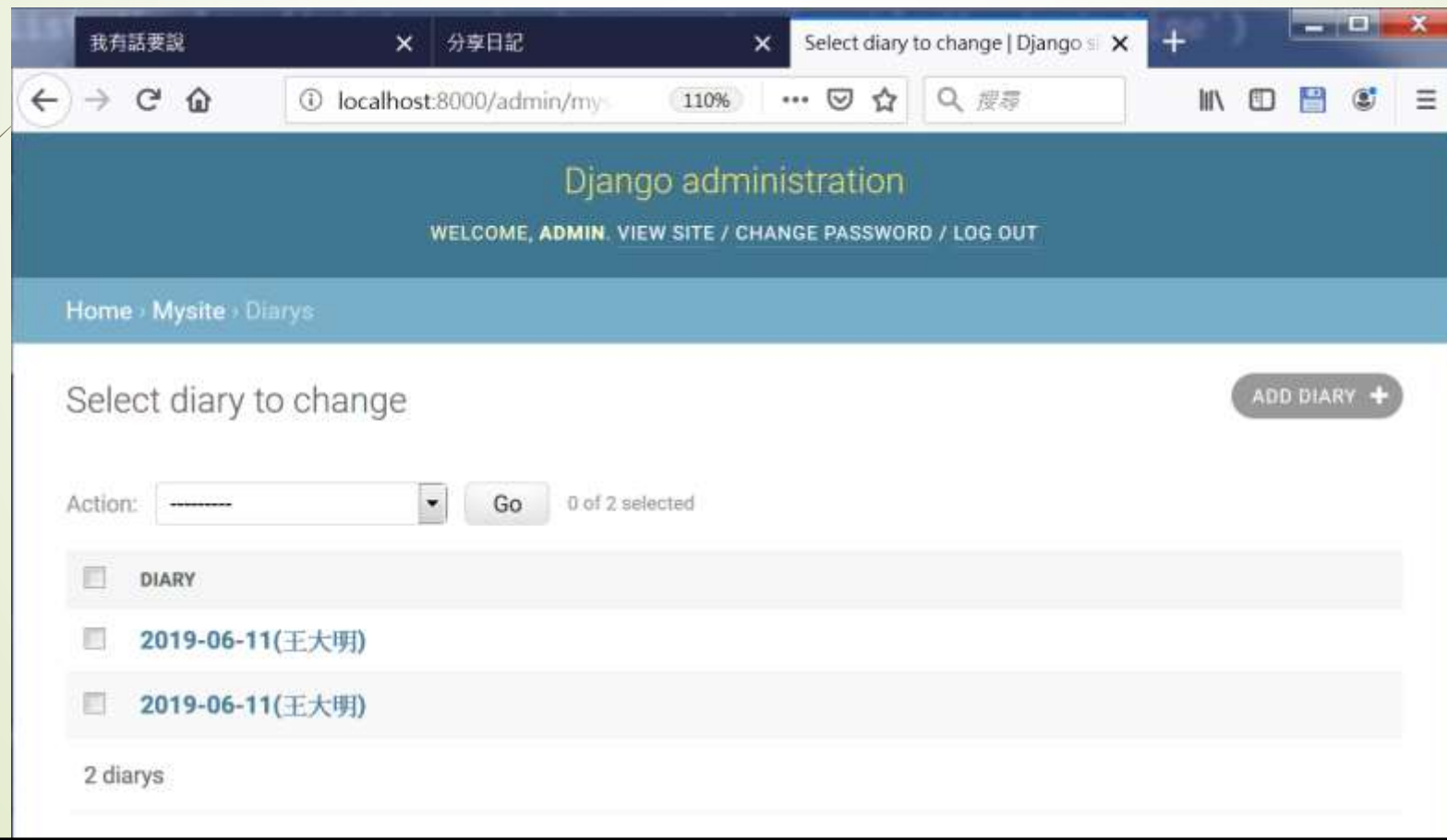


應用auth使用者驗證存取資料庫

- 在DiaryForm增加user欄位的結果
- `fields = ['user', 'budget', 'weight', 'note', 'ddate']`
- `self.fields['user'].label = 'User'`



- `admin.py`中新增
- `admin.site.register(models.Diary)`



應用auth使用者驗證存取資料庫

- ▶ 如箭頭所指的地方所示，要張貼自己的日記，卻還可以選擇張貼人，這不是一件很奇怪的事嗎？
- ▶ 因此在DiaryForm中不需要增加user這個欄位，這個欄位是要在views.posting函數中接到表單之後再加上去的（因為user在登入之後即為已知）
- ▶ `posting.html`中如何顯示「寫日記」的網頁顯示：
 - ▶ `<!-- posting.html (Session_Cookies) -->`
 - ▶ `{% extends "base.html" %}`
 - ▶ `{% block title %}我有話要說{% endblock %}`
 - ▶ `{% block content %}`
 - ▶ `<div class='container'>`
 - ▶ `{% for message in messages %}`
 - ▶ `<div class='alert alert-{{message.tags}}'>{{ message }}</div>`
 - ▶ `{% endfor %}`

應用auth使用者驗證存取資料庫

- `<form name='my form' action='.' method='POST'>`
- `{% csrf_token %}`
- `<table>`
- `{{ post_form.as_table }}`
- `</table>`
- `<input type='submit' value='張貼'>`
- `<input type='reset' value='清除重填'>`
- `</form>`
- `</div>`
- `{% endblock %}`

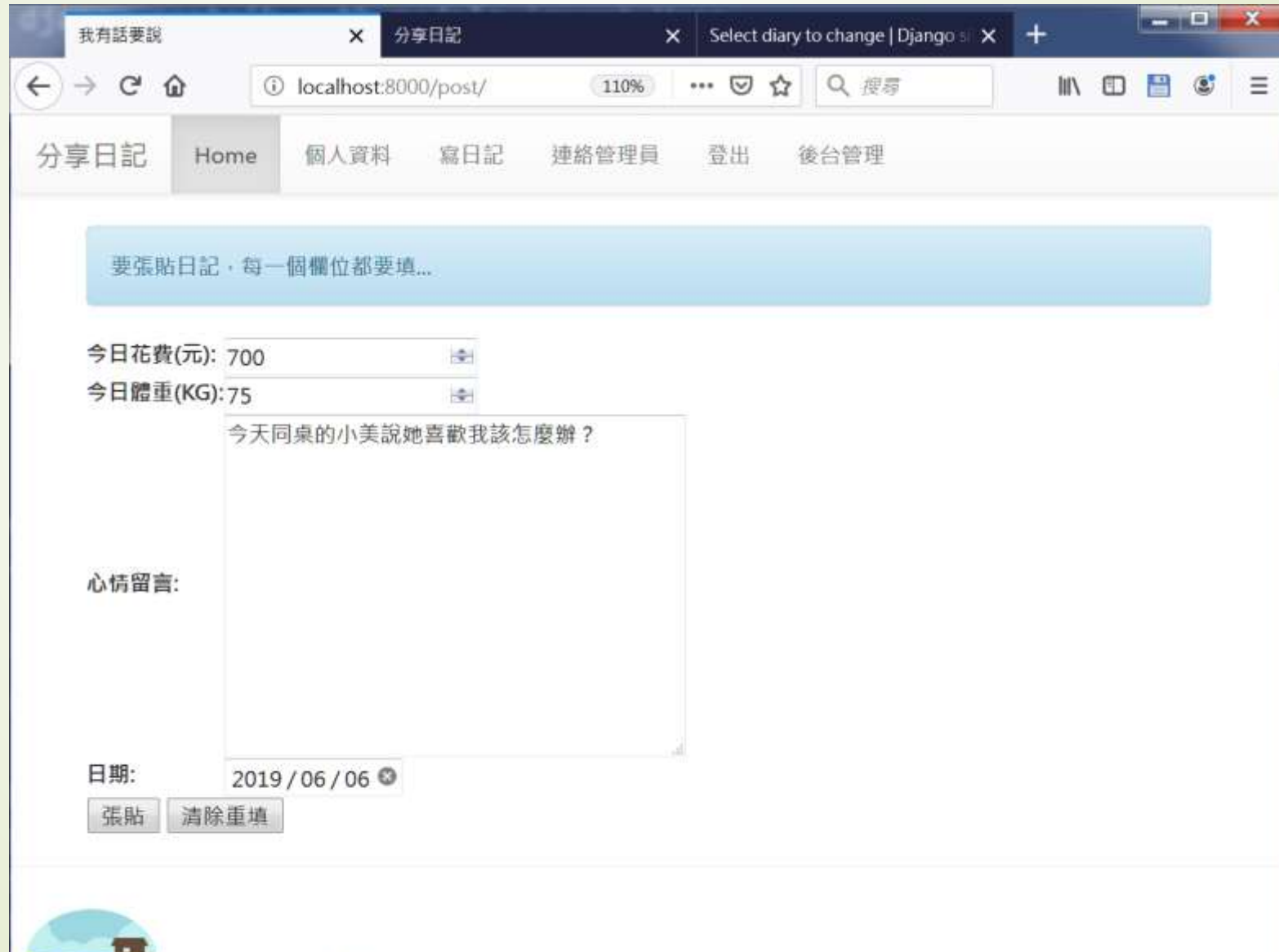
應用auth使用者驗證存取資料庫

- `views.posting`中，如下所示：
 - `@login_required(login_url='/login/')`
 - `def posting(request):`
 - `if request.user.is_authenticated:`
 - `username = request.user.username`
 - `useremail = request.user.email`
 - `messages.get_messages(request)`
- `if request.method == 'POST':`
 - `user = User.objects.get(username=username)`
 - `diary = models.Diary(user=user)`
 - `post_form = forms.DiaryForm(request.POST, instance=diary)`

應用auth使用者驗證存取資料庫

- `if post_form.is_valid():`
- `messages.add_message(request, messages.INFO, "日記已儲存")`
- `post_form.save()`
- `return HttpResponseRedirect('/')`
- `else:`
- `messages.add_message(request, messages.INFO, '要張貼日記，每一個欄位都要填...')`
- `else:`
- `post_form = forms.DiaryForm()`
- `messages.add_message(request, messages.INFO, '要張貼日記，每一個欄位都要填...')`
- `return render(request, 'posting.html', locals())`

這樣就可以po文了



The screenshot shows a web browser window with the URL `localhost:8000/post/`. The page has a navigation bar with links for "分享日記", "Home", "個人資料", "寫日記", "連絡管理員", "登出", and "後台管理". A blue banner at the top of the form area says "要張貼日記，每一個欄位都要填...". The form contains the following fields:

- "今日花費(元):" with a value of 700 and a spinner control.
- "今日體重(KG):" with a value of 75 and a spinner control.
- A text area containing the text "今天同桌的小美說她喜歡我該怎麼辦?".
- "心情留言:" with an empty text area below it.
- "日期:" with a date picker set to 2019/06/06.

At the bottom of the form, there are two buttons: "張貼" (Post) and "清除重填" (Clear and Re-enter).

應用auth使用者驗證存取資料庫

- ▶ 在index.html中就要能夠使用者登入之後把使用者自己的日記顯示在畫面上，因此index.html的內容就必須改寫如下：
 - ▶ <!-- index.html (Session_Cookies) -->
 - ▶ {% extends "base.html" %}
 - ▶ {% block title %}分享日記{% endblock %}
 - ▶ {% block content %}
 - ▶ <div class='container'>
 - ▶ {% for message in messages %}
 - ▶ <div class='alert alert-
{{message.tags}}'>{{ message }}</div>
 - ▶ {% endfor %}

應用auth使用者驗證存取資料庫

```
➤ <div class='row'>
➤   <div class='col-md-12'>
➤     <div class='panel panel-default'>
➤       <div class='panel-heading' align=center>
➤         <h3>{{ username | default:"我"}}的私人日記</h3>
➤       </div>
➤     </div>
➤   </div>
➤ </div>
➤ {% for diary in diaries %}
➤   {% cycle "<div class='row'>" "" "" %}
➤     <div class='col-md-4'>
➤       <div class='panel panel-primary'>
➤         <div class='panel-heading' align=center>
➤           {{ diary.ddate }}
➤         </div>
```

應用auth使用者驗證存取資料庫

- `<div class='panel-body'>`
- `{{ diary.note | linebreaks }}`
- `</div>`
- `<div class='panel-footer'>`
- `今日花費：{{ diary.budget }}元，體重：`
- `{{ diary.weight }}公斤`
- `</div>`
- `</div>`
- `</div>`
- `{% cycle "" "" "</div>" %}`
- `{% empty %}`
- `<h3>登入網站才能夠使用日記功能</h3>`
- `{% endfor %}`
- `</div>`
- `{% endblock %}`

應用auth使用者驗證存取資料庫

- ▶ 上面的程式碼主要處理diaries這個變數的輸出，提供這個變數內容的則是在[views.index](#)，如下所示：
 - ▶ `def index(request, pid=None, del_pass=None):`
 - ▶ `if request.user.is_authenticated:`
 - ▶ `username = request.user.username`
 - ▶ `useremail = request.user.email`
 - ▶ `try:`
 - ▶ `user = models.User.objects.get(username=username)`
 - ▶ `diaries = models.Diary.objects.filter(user=user).order_by('-ddate')`
 - ▶ `except:`
 - ▶ `pass`
 - ▶ `messages.get_messages(request)`
 - ▶ `return render(request, 'index.html', locals())`

應用auth使用者驗證存取資料庫

未登入帳號的首頁畫面



應用auth使用者驗證存取資料庫

- ➔ 已登入帳號之後的首頁畫面

