auth 앱 기본 설정

URL Pattern	View Name	Template File Name
/login/	LoginView (FormView)	login.html
/logout/	LogoutView (TemplateView)	logged_out.html
/password_change/	PasswordChangeView (FormView)	password_change_form.html
/password_change/done/	PasswordChangeDoneView (TemplateView)	password_change_done.html
/password_reset/	PasswordResetView (FormView)	password_reset_form.html password_reset_email.html password_reset_subject.txt
/password_reset/done/	PasswordResetDoneView (TemplateView)	password_Reset_done.html
/reset/<uidb64>/<token>/	PasswordResetConfirmView (FormView)	password_reset_confirm.html
/reset/done/	PasswordResetCompleteView (TemplateView)	password_reset_complete.html
/regiser/	UserCreateView (CreateView)	register.html
/register/done/	UserCreateDoneTV (TemplateView)	register_done.html

View Name	Form Name
LoginView (FormView)	class AuthenticationForm(forms.Form)
UserCreateView (CreateView)	class UserCreationForm(forms.ModelForm):

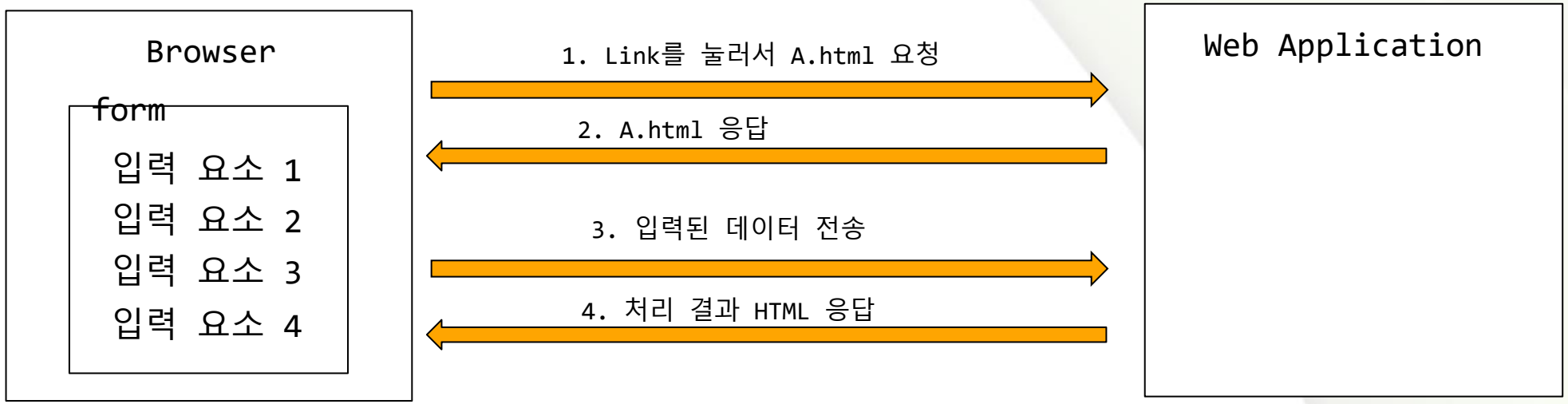
auth 앱 기본 설정

- URL Pattern은 사용자 정의 경로와 결합해서 사용
  - /accounts/login/ → accounts (사용자 정의 경로) + login(auth 앱 경로)

- Template File 경로는 공통적으로 registration 경로 하위에 구성
  - login.html → registration/login.html

HTML에서의 폼

- 우리는 웹 사이트를 개발할 때 사용자로부터 입력을 받기 위해서 폼을 사용.
- HTML로 표현 하면 폼은 <form>…</form> 사이에 있는 엘리먼트들의 집합.
- 웹 사이트 사용자는 폼을 통 해 텍스트를 입력할 수도 있고, 항목을 선택할 수도 있음
- form 태그에 action 속성으로 전송 대상을 지정
- form 태그에 method 속성으로 데이터 전송방법을 지정
  - post : 메시지 본문에 데이터를 기록하고 전송
  - get : 주소 뒤에 ?이름=값&이름=값2의 형식으로 데이터 전송

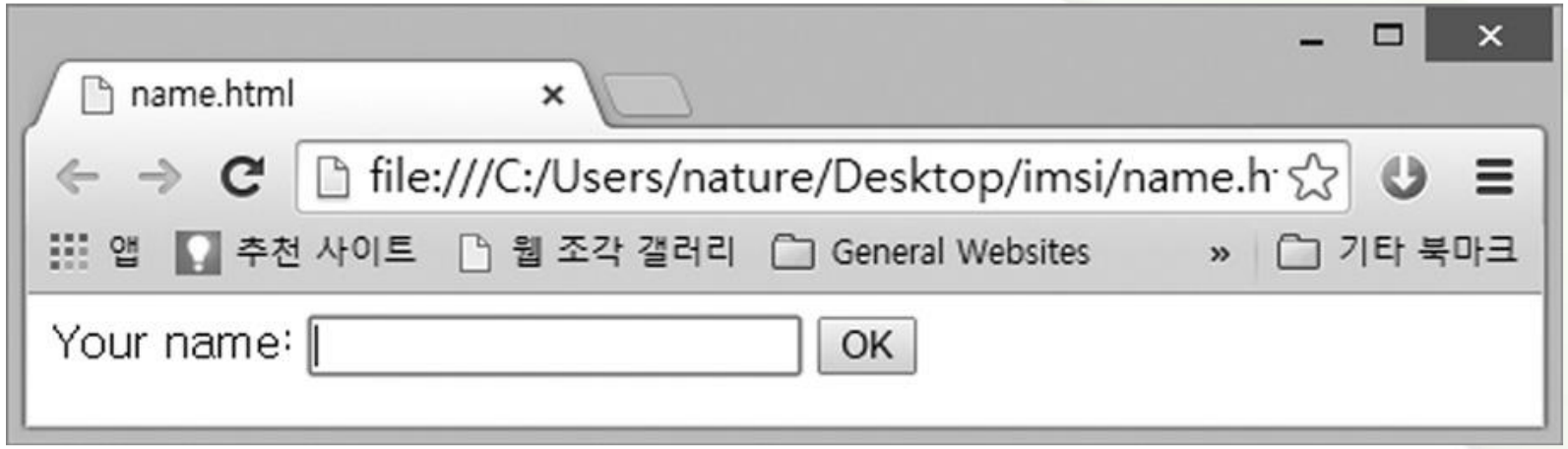


장고의 폼 기능

- 여러 가지 타입의 많은 위젯이 폼 화면 출력용으로 준비되어야 하고, HTML로 렌더링되며, 적절한 인터페이스를 사용하여 입력 및 수정되고, 서버로 보내져서 데이터가 유효한지 검증을 거친 후에 적절한 처리를 위해 저장되거나 전달.
- 장고는 이러한 폼 기능들을 단순화하고 자동화해서 개발자가 직접 코딩하는 것보다 훨씬 안전하게 처리.
- 장고는 폼 처리를 위하여 다음의 3가지 기능을 제공
  - 폼 생성에 필요한 데이터를 폼 클래스로 구조화하기
  - 폼 클래스의 데이터를 렌더링하여 HTML 폼 만들기
  - 사용자로부터 제출된 폼과 데이터를 수신하고 처리하기
- 폼도 결국은 템플릿의 일부이므로 템플릿 코드에 포함되어서 렌더링 절차에 결합
  - 렌더링할 객체를 뷰로 가져오기(예를 들어, 데이터베이스로부터 객체를 추출하기)
  - 그 객체를 템플릿 시스템으로 넘겨주기
  - 템플릿 문법을 처리해서 HTML 마크업 언어로 변환하기

폼 클래스로 폼 생성

```
<form action="/your-name/" method="post">
    <label for="your_name">Your name: </label>
    <input id="your_name" type="text" name="your_name" value="{{ current_name }}">
    <input type="submit" value="OK">
```



예제 4-18 폼 클래스 정의

```
from django import forms

class NameForm(forms.Form):
    your_name = forms.CharField(label='Your name', max_length=100)
```

뷰에서 폼 클래스 처리

예제 4-21 뷰에서 폼 클래스를 처리하는 방식

```
from django.shortcuts import render
from django.http import HttpResponseRedirect

def get_name(request):                                                    ❶
    # POST 방식이면, 데이터가 담긴 제출된 폼으로 간주합니다.
    if request.method == 'POST':                                          ❷, ❹
        # request에 담긴 데이터로, 클래스 폼을 생성합니다.
        form = NameForm(request.POST)                                     ❺
        # 폼에 담긴 데이터가 유효한지 체크합니다.
        if form.is_valid():                                               ❻
            # 폼 데이터가 유효하면, 데이터는 cleaned_data로 복사됩니다.
            new_name = form.cleaned_data['name']
            # 로직에 따라 추가적인 처리를 합니다.                          ❽

            # 새로운 URL로 리다이렉션시킵니다.
            return HttpResponseRedirect('/thanks/')

    # POST 방식이 아니면(GET 요청임),
    # 빈 폼을 사용자에게 보여줍니다.
    else:                                                                 ❷
        form = NameForm()                                                 ❸

    return render(request, 'name.html', {'form': form})                   ❼, ❽
```

폼 클래스를 템플릿으로 변환

- {{ form.as_table }} : <tr> 태그로 감싸서 테이블 셀로 렌더링됩니다. {{form}}과 동일함
- {{ form.as_p }} : <p> 태그로 감싸도록 렌더링됩니다.
- {{ form.as_ul }} : <li> 태그로 감싸도록 렌더링됩니다.

예제 4-22 ContactForm 폼 클래스 정의

```
from django import forms

class ContactForm(forms.Form):
    subject = forms.CharField(max_length=100)
    message = forms.CharField(widget=forms.Textarea)
    sender = forms.EmailField()
    cc_myself = forms.BooleanField(required=False)
```

예제 4-23 ContactForm 폼 클래스의 {{ form.as_p }} 렌더링 결과

```
<p><label for="id_subject">Subject:</label>
    <input id="id_subject" type="text" name="subject" maxlength="100" /></p>
<p><label for="id_message">Message:</label>
    <input type="text" name="message" id="id_message" /></p>
<p><label for="id_sender">Sender:</label>
    <input type="email" name="sender" id="id_sender" /></p>
<p><label for="id_cc_myself">Cc myself:</label>
    <input type="checkbox" name="cc_myself" id="id_cc_myself" /></p>
```