Django Form과 View

한국기술교육대학교
온라인평생교육원

학습내용

- View 작성
- Form 작성
- 클래스 기반 View

학습목표

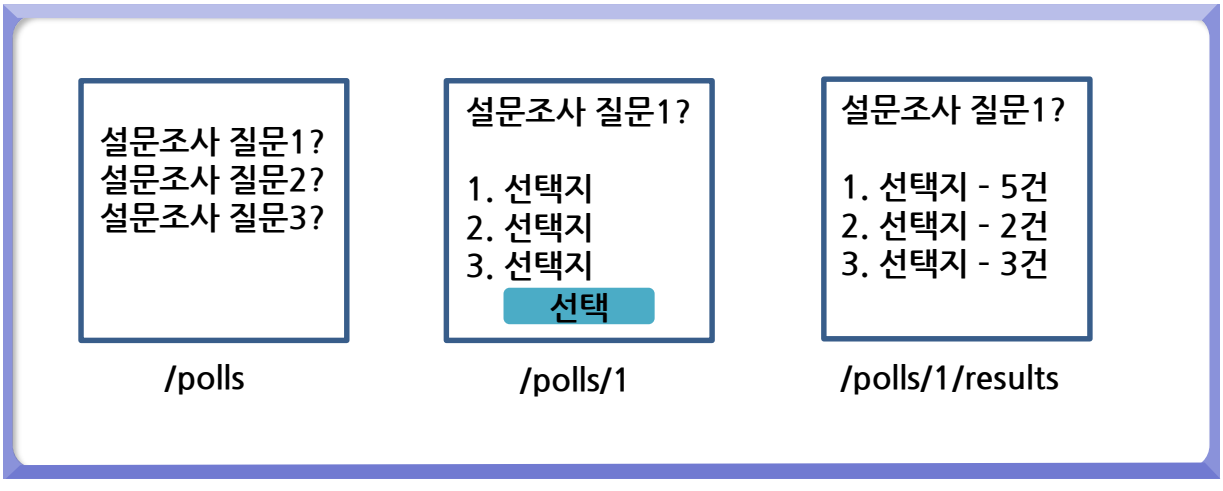
- Django에서 Get과 Post 요청을 처리하는 방법에 대해 설명할 수 있다.
- Django에서 Form을 작성하는 방법을 설명할 수 있다.
- Django에서 함수형 View와 클래스 기반 View의 차이를 설명할 수 있다.

View 작성하기

1 파라미터를 처리하는 View 구성

《1》 URLConf를 통해 URL 맵핑 구성

1. /polls 주소 : Index, 설문 리스트를 보여줌
2. /polls/1 주소 : Detail, 설문번호 1에 대한 질문과 선택지를 보여줌
3. /polls/1/vote 주소 : 설문번호 1에 대한 설문 결과를 서버에 전송
4. /polls/1/results 주소 : 설문번호 1에 대한 설문 결과를 보여줌



View 작성하기

1 파라미터를 처리하는 View 구성

2 mysite/polls/urls.py 구현

```
from django.conf.urls import include, url
from django.contrib import admin
from polls import views

urlpatterns = [
                url(r'^$', views.index, name='index'),
                url(r'^(?P<question_id>₩d+)/$', views.detail, name='detail'),
                url(r'^(?P<question_id>₩d+)/vote/$', views.vote,
name='vote'),
                url(r'^(?P<question_id>₩d+)/results/$', views.results,
name='results'),
]
```

3 View의 index() 구성

1 설문 목록을 데이터베이스로부터 가져옴

- 설문 목록 입력 날짜를 역순으로 정렬하여 최대 5개를 가져옴

2 html에 모델 데이터를 렌더링

- context라는 변수를 python 사전 타입으로 만들어서 render함수의 파라미터로 전달

3 Html 파일을 리턴하여 설문 목록을 보여줌

- mysite/polls/templates/polls/index.html파일을 리턴

View 작성하기

1 파라미터를 처리하는 View 구성

3 View의 index() 구성

```
from django.shortcuts import render
from .models import Question

def index(request):
    latest_question_list = Question.objects.order_by('-pub_date')[:5]
    context = {
        'latest_question_list': latest_question_list,
    }
    return render(request, 'polls/index.html', context)
```

4 View의 detail() 구성

1. URLConf로부터 question_id를 파라미터로 전달받음
2. 전달받은 question_id를 model의 Question 객체로 PK(Primary Key)로 넘겨준 후 객체로 가져옴
   - 만약 없다면 404 에러 발생
   - 만약 있다면 polls/detail.html에 객체를 전달하여 render함수 호출

View 작성하기

1 파라미터를 처리하는 View 구성

4 View의 detail() 구성

```
from django.http import Http404
from django.shortcuts import render
from .models import Question

def index(request):
...생략...

def detail(request, question_id):
    try:
        question = Question.objects.get(pk=question_id)
    except Question.DoesNotExist:
        raise Http404("Question does not exist")
    return render(request, 'polls/detail.html', {'question': question})
```

5 View의 results() 구성

1. get_object_or_404 함수를 이용하면 훨씬 간편화되어 객체를 가져올 수 있음

2. 이 함수는 해당 object가 없을 시 자동으로 404 에러를 반환

View 작성하기

1 파라미터를 처리하는 View 구성

4 View의 results() 구성

```
from django.shortcuts import render
from django.shortcuts import get_object_or_404
from .models import Question

def index(request):
...생략...

def detail(request, question_id):
 ...생략...

def results(request, question_id):
    question = get_object_or_404(Question, pk=question_id)
    return render(request, 'polls/results.html', {'question': question})
```

View 작성하기

2 Template 구성

⟨1⟩ index.html 구성

앞서 View에서 전달된 'latest_question_list'라는 이름의 Question 객체 목록을 다음과 같이 화면에 나타냄

```
{% if latest_question_list %}
    <ul>
    {% for question in latest_question_list %}
        <li><a
href="/polls/{{ question.id }}/">{{ question.question_text }}</a></li>
    {% endfor %}
    </ul>
{% else %}
    <p>No polls are available.</p>
{% endif %}
```

⟨2⟩ results.html 구성

앞서 View에서 전달된 'question'이라는 이름의 Question 객체와 객체에 대한 Choice 선택 목록을 다음과 같이 화면에 나타냄

```
<h1>{{ question.question_text }}</h1>
<ul>
{% for choice in question.choice_set.all %}
  <li>{{ choice.choice_text }} : {{ choice.votes }}</li>
{% endfor %}
</ul>

<a href="{% url 'polls:detail' question.id %}">Vote again?</a>
```

Form 작성하기

1 Django에서의 Form 작성

⟨1⟩ index.html 구성

View에서 전달하는
Question 객체와
Question 객체와 연관된 Choice 객체
목록을 보여줌

사용자가 Choice 객체 목록 중 하나를
라디오버튼으로
선택한 후 버튼을 누르면
POST 메소드로 결과 전달

```
<h1>{{ question.question_text }}</h1>
<form action="{% url 'polls:vote' question.id %}" method="post">
{% csrf_token %}
{% for choice in question.choice_set.all %}
  <input type="radio" name="choice" id="choice{{ forloop.counter }}"
value="{{ choice.id }}" />
  <label
for="choice{{ forloop.counter }}">{{ choice.choice_text }}</label><br/>
{% endfor%}
<input type="submit" value="Vote"/>
</form>
```

⟨2⟩ detail.html 파악하기

1 get_object_or_404 함수를 이용하면 훨씬 간편화되어 객체를 가져올 수 있음

- template 문법인 {% url %} 태그에 의해 데이터를 전달받을 URL을 'polls:vote'로 지정
- 이 때, 'polls:vote'는 polls 폴더 밑의 urls.py에 있는 'vote'라는 이름의 URL을 뜻함

- question.id는 Form으로 전달할 데이터이며, 이어서 쓸 수 있음

Form 작성하기

1 Django에서의 Form 작성

2 detail.html 파악하기

2 {% csrf_token %}

- CSRF(Cross Site Request Forgery) 공격을 방지하기 위한 기능으로 Django에서는 간단히 한 줄만 입력하면 됨

3 {% for choice in question.choice_set.all %}

View로부터 전달된
Question 객체와 이 객체를 참조하는
Choice 객체 목록을 가져오는 문법

Django에서는 1:N 관계에서 연결된
N개의 모든 항목들을 기본으로
question_set과 같이 _set 속성으로
가짐

Form 작성하기

2 View에서의 처리

1 View의 vote( ) 구성

POST로 전달되는 데이터 확인

해당 데이터가 정상적이면 choice에 대한 카운트 증가 후 results 페이지로 이동

해당 데이터가 정상적이지 않다면 에러 발생 후 detail 페이지로 이동

```
from django.http import HttpResponse
from django.http import HttpResponseRedirect
from django.core.urlresolvers import reverse
from django.shortcuts import render
from django.shortcuts import get_object_or_404

from .models import Question, Choice
def index(request):
...생략...
def detail(request, question_id):
 ...생략...
def results(request, question_id):
...생략...
def vote(request, question_id):
  question = get_object_or_404(Question, pk=question_id)
  try:
    selected_choice = question.choice_set.get(pk=request.POST['choice'])
  except (KeyError, Choice.DoesNotExist):
    return render(request, 'polls/detail.html', {'question': question})
  else:
    selected_choice.votes += 1
    selected_choice.save()
    return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

Form 작성하기

2 View에서의 처리

2 View의 vote( ) 살펴보기

1 selected_choice = question.choice_set.get(pk=request.POST['choice'])

POST 메소드로 전달된 'choice'라는 이름의 파라미터의 값을 PK로 하는 객체를 Question 객체의 choice_set으로부터 찾음

이 'choice' 파라미터는 detail.html에서 각각의 선택 문항의 아이디를 값으로 갖는 라디오버튼으로부터 전달됨

2 selected_choice.save()

- 변경사항을 Choice 테이블에 저장하는 문장으로 Django에서는 간편하게 데이터베이스에 반영함

3 return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))

우선 reverse 함수로부터 polls:results라는 URL이름을 바탕으로 URL을 생성해내며, 이 때 question.id를 인수로 전달하여 URL을 만들어냄

예 : question.id가 2일 경우 /polls/2/results라는 URL을 만들어냄

만들어진 URL을 HttpResponseRedirect 객체로 전달하며, 해당 주소로 이동함

클래스 기반 View

1 제너릭 View란?

Generic View

Django에서는 자주 사용되는 용도에 따라 기본적인 클래스를 View로 제공

이런 공통된 View를 Generic View라 하며, 함수가 아닌 클래스로 구현되어 있음

View	설명
View	가장 기본이 되는 View
TemplateView	템플릿을 렌더링해 줌
RedirectView	URL이 주어지면 redirect함
DetailView	객체에 대한 상세 정보를 보여줌
ListView	여러 개의 객체를 보여줌

자주 쓰이는 Generic View

클래스 기반 View

2 제너릭 View 적용하기

⟨1⟩ View의 index 구성

1. 기존의 index 함수를 class로 변환, generic.ListView 를 상속
2. list로 보여줄 데이터는 get_queryset 함수를 통해 지정
3. ListView에서는 <app name>/<model name>_list.html을 기본으로 사용하기 때문에 기존의 'polls/index.html'을 사용하기 위해서 별도 지정
4. 컨텍스트 변수도 'question_list'와 같이 자동 생성되기 때문에 기존의 'latest_question_list'라는 이름을 사용하기 위해서는 별도 지정

```
from django.shortcuts import get_object_or_404, render
from django.http import HttpResponseRedirect
from django.urls import reverse
from django.views import generic

class IndexView(generic.ListView):
    template_name = 'polls/index.html'
    context_object_name = 'latest_question_list'

    def get_queryset(self):
        return Question.objects.order_by('-pub_date')[:5]
```

클래스 기반 View

2 제너릭 View 적용하기

2 View의 detail과 results 구성

1. 기존의 detail 함수와 results 함수를 class로 변환, generic.DetailView를 상속
2. DetailView에서는 URL로 전달되는 기본 키 값이 'pk'라고 인식되기 때문에 추후 urls.py에서 파라미터 이름을 'pk'로 변경해야 함
3. 객체는 model이라는 변수에 저장
4. DetailView에서는 <app name>/<model name>_detail.html을 기본으로 사용하기 때문에 다른 템플릿 이름을 쓰기 위해서는 별도 지정

```
class DetailView(generic.DetailView):
    model = Question
    template_name = 'polls/detail.html'


class ResultsView(generic.DetailView):
    model = Question
    template_name = 'polls/results.html'
```

클래스 기반 View

2 제너릭 View 적용하기

3 URLConf 재구성

클래스형 View를 지정하기 위해서는 as_view()라는 함수를 호출함

index, detail, results 주소의 경우 Generic View를 통해 클래스형 View로 전환하였기 때문에 as_view() 함수 호출을 통해 View를 지정

```
from django.conf.urls import url
from polls import views

urlpatterns = [
    url(r'^$', views.IndexView.as_view(), name='index'),
    url(r'^(?P<pk>[0-9]+)/$', views.DetailView.as_view(), name='detail'),
    url(r'^(?P<pk>[0-9]+)/results/$', views.ResultsView.as_view(),
name='results'),
    url(r'^(?P<question_id>[0-9]+)/vote/$', views.vote, name='vote'),
]
```

학습정리

1. View 작성하기

- 파라미터를 처리하는 View 구성

  - URLConf를 통해 URL 맵핑 구성

  ① /polls 주소 : Index, 설문 리스트를 보여줌

  ② /polls/1 주소 : Detail, 설문번호 1에 대한 질문과 선택지를 보여줌

  ③ /polls/1/vote 주소 : 설문번호 1에 대한 설문 결과를 서버에 전송

  ④ /polls/1/results 주소 : 설문번호 1에 대한 설문 결과를 보여줌

- Template 구성

  - index.html 구성 : View에서 전달된 'latest_question_list'라는 이름의 Question 객체 목록을 화면에 나타냄
  - results.html 구성 : View에서 전달된 'question'이라는 이름의 Question 객체와 객체에 대한 Choice 선택 목록을 화면에 나타냄

학습정리

2. Form 작성하기

- Django에서의 Form 작성

  - detail.html 구성
    - View에서 전달하는 Question 객체와, Question 객체와 연관된 Choice 객체 목록을 보여줌
    - 사용자가 Choice 객체 목록 중 하나를 라디오버튼으로 선택한 후 버튼을 누르면 POST 메소드로 결과를 전달함

  - View의 vote() 구성
    - post로 전달되는 데이터 확인
    - 해당 데이터가 정상적이면 choice에 대한 카운트 증가 후 results 페이지로 redirect
    - 해당 데이터가 정상적이지 않다면 에러 발생 후 detail 페이지로 이동

학습정리

3. 클래스 기반 View

- Generic View란?
  - Django에서는 자주 사용되는 용도에 따라 기본적인 클래스를 View로 제공
  - 공통된 View를 Generic View라 하며, 함수가 아닌 클래스로 구현

- 제너릭 View 적용하기
  - View의 index 구성
  - View의 detail과 results 구성
  - URLConf 재구성