Lab05:
Django Tutorial III

ESM2014-41

객체지향프로그래밍 및 실습

SKKU 시스템경영공학과

조영일

Form

- Form : HTML 상에서 사용자가 서버로 특정 정보를 전송하기 위한 수단
- Form 에서 사용자가 내용을 선택하거나 작성하고 Form을 Submit하면 서버로 해당 내용을 포함한 GET, POST HTTP Request가 전송됨

- “polls/templates/polls/detail.html” 수정 : 설문조사 문항(Question)과 그에 속한 선택지(Choice)들을 보여주고, 사용자가 선택한 결과를 서버로 전송 할 수 있는 Form을 구현

Form

```
<!-- Question의 본문을 표시 -->
<h1>{{ question.question_text }}</h1>

<!-- context 에 error message가 있을 경우 -->
{% if error_message %}<p><strong>{{ error_message }}</strong></p>{% endif %}

<!--
form action = form을 submit할 대상 URL
form method = form의 내용을 전송할 방식 get/post 중 하나
-->
<form action="{% url 'polls:vote' question.id %}" method="post">
    <!-- CSRF 보안 문제를 방지하기 위해 넣어야하는 Template Tag -->
    {% csrf_token %}
    <!-- Question에 딸린 Choice들을 가져와서, for 문을 통해 radio button으로 표시 -->
    {% 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 %}
    <!-- 전송(Submit) 버튼 -->
    <input type="submit" value="Vote">
</form>
```

vote() view 구현

- Form 에서 전송 받은 데이터를 실제로 Model을 통해 Database에 반영하기 위한 View 구현
- “polls/views.py” vote method 수정

vote() view 구현

```python
from django.http import HttpResponse, HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse

from polls.models import Choice, Question
# ...
def vote(request, question_id):
    # question_id 에 해당하는 Question을 가져옴
    question = get_object_or_404(Question, pk=question_id)
    try:
        # request.POST는 dictionary이며, POST Method 를 통해 전달 받은 값들이 이름(Key)-값(Value) 형태로 저장되어 있음
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # POST로 전달 받은 Choice가 존재 하지 않을 경우, error_message와 함께 다시 원래 페이지를 표시하도록 함 .
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        # try 문에서 오류 없었을 경우, 선택지의 votes 값에 1을 더하고 저장
        selected_choice.votes += 1
        selected_choice.save()
        # 결과 화면으로 사용자를 이동 시킴
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

vote() view 구현

```python
from django.http import HttpResponse, HttpResponseRedirect
from django.shortcuts import get_object_or_404, render
from django.urls import reverse

from polls.models import Choice, Question
# ...
def vote(request, question_id):
    # question_id 에 해당하는 Question을 가져옴
    question = get_object_or_404(Question, pk=question_id)
    try:
        # request.POST는 dictionary이며, POST Method 를 통해 전달 받은 값들이 이름(Key)-값(Value) 형태로 저장되어 있음
        selected_choice = question.choice_set.get(pk=request.POST['choice'])
    except (KeyError, Choice.DoesNotExist):
        # POST로 전달 받은 Choice가 존재 하지 않을 경우, error_message와 함께 다시 원래 페이지를 표시하도록 함 .
        return render(request, 'polls/detail.html', {
            'question': question,
            'error_message': "You didn't select a choice.",
        })
    else:
        # try 문에서 오류 없었을 경우, 선택지의 votes 값에 1을 더하고 저장
        selected_choice.votes += 1
        selected_choice.save()
        # 결과 화면으로 사용자를 이동 시킴
        return HttpResponseRedirect(reverse('polls:results', args=(question.id,)))
```

results() view 구현

- 설문조사 결과를 보여주기 위한 view로 수정
- “polls/views.py” results method 수정

```
from django.shortcuts import get_object_or_404, render

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

results() view 구현

- “polls/templates/results.html” 템플릿 작성

```
<h1>{{ question.question_text }}</h1>

<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }} -- {{ choice.votes }} vote{{
choice.votes|pluralize }}</li>
{% endfor %}
</ul>

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

Static Files

- Static Files : Django에 의해 동적으로 생성하는 웹 페이지(Dynamic Resource)와 달리 이미 존재하는 파일을 그대로 클라이언트에게 전달

- Static File의 종류
  - Image : *.jpg, *.gif, *png 등 HTML 본문에 삽입되는 이미지 요소들
  - CSS(Cascading Style Sheet) : HTML 로 작성된 문서의 표현 방식을 결정하는 언어. HTML은 디자인적인 요소를 제외하고 웹 페이지의 구조화된 요소들을 작성하고, CSS를 통해 그 요소들의 표현 방식을 결정
  - JavaScript : 페이지 로드 이후 사용자와의 동적인 상호작용을 담당

Static Files

- Static Files : Django에 의해 동적으로 생성하는 웹 페이지(Dynamic Resource)와 달리 이미 존재하는 파일을 그대로 클라이언트에게 전달

- Static File의 종류
  - Image : *.jpg, *.gif, *png 등 HTML 본문에 삽입되는 이미지 요소들
  - CSS(Cascading Style Sheet) : HTML 로 작성된 문서의 표현 방식을 결정하는 언어. HTML은 디자인적인 요소를 제외하고 웹 페이지의 구조화된 요소들을 작성하고, CSS를 통해 그 요소들의 표현 방식을 결정
  - JavaScript : 페이지 로드 이후 사용자와의 동적인 상호작용을 담당

Static Files

- “settings.py”에 다음 내용 추가 : Project Root의 “static” 디렉토리에서 Django가 static file들을 찾도록 하기 위한 설정

```
STATIC_URL = '/static/'

STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static/'),
)
```

Static Files

- “static/polls/style.css” 파일 생성 후 아래와 같이 작성 : li > a 태그의 글자 색을 green으로 변경

polls/static/polls/style.css

```
li a {
    color: green;
}
```

- “polls/templates/polls/index.html”

polls/templates/polls/index.html

```
{% load static %}

<link rel="stylesheet" type="text/css" href="{% static 'polls/style.css' %}">
```

Static Files

- http://localhost:8000/polls/

Study Topic

- Django Tutorial Topic 4(https://docs.djangoproject.com/ko/2.2/intro/tutorial04/)
- Django Static File(https://docs.djangoproject.com/ko/2.2/howto/static-files/)
- Django Official Documentation(https://docs.djangoproject.com/en/2.2/)
- CSS Basics(https://developer.mozilla.org/ko/docs/Learn/Getting_started_with_the_web/CSS_%EA%B8%B0%EB%B3%B8)
- Javascript Basics(https://developer.mozilla.org/ko/docs/Learn/Getting_started_with_the_web/JavaScript_basics)