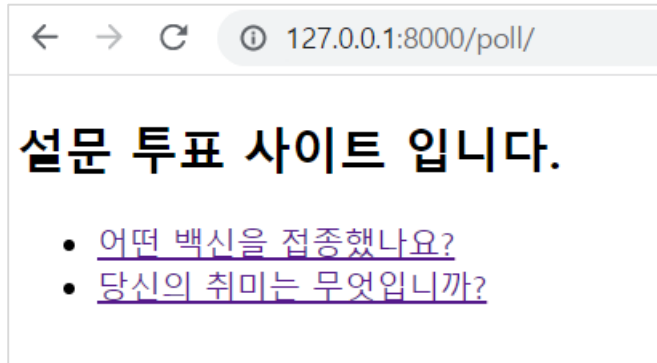


3장. 템플릿 - HTML



템플릿으로 질문 목록 페이지 만들기

index.html 만들기- templates/poll/index.html



```
{% if question_list %}
    <ul>
        {% for question in question_list %}
            <li>
                <a href="/poll/{{question.id}}">
                    {{ question.question_text }}
                </a>
            </li>
        {% endfor %}
    </ul>
{% else %}
    <p>설문이 없습니다.</p>
{% endif %}
```

템플릿으로 질문 목록 페이지 만들기

◆ 템플릿(template) 태그

파이썬을 웹에 적용한 언어로 **{% %}** 블록을 사용한다.

템플릿 태그	설 명
<code>{% if item_list %}</code> .. 내용 .. <code>{% endif %}</code>	item_list가 있다면(조건문)
<code>{% for item in item_list %}</code> .. 내용 .. <code>{% endfor %}</code>	item_list를 반복하며 순차적으로 item에 대입(반복문)
<code>{{ id }}</code>	id 출력(출력문)
<code>{{ forloop.counter0 }}</code> <code>{{ forloop.counter }}</code>	for 반복문의 인덱스로 0부터 시작 0, 1 , 2, 3 ...

질문 상세 페이지 만들기

◆ 설문 상세 페이지

1. detail 페이지 URL 매핑하기 – poll/urls.py

```
urlpatterns = [  
    path('', views.index),  
    path('<int:pk>/', views.detail), # 127.0.0.1:8000/poll/1  
]
```

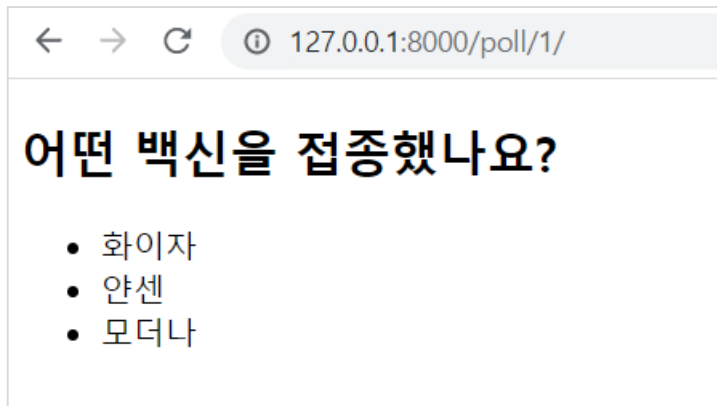
2. detail() 구현 – poll/views.py

```
def detail(request, pk): # pk는 question의 id  
    # 설문 상세 보기  
    question = Question.objects.get(id=pk)  
    return render(request, 'poll/detail.html', {'question':question})
```

설문 상세 페이지 만들기

◆ 설문 상세 페이지

3. poll/detail.html 작성



← → ↻ ⓘ 127.0.0.1:8000/poll/1/

어떤 백신을 접종했나요?

- 화이자
- 얀센
- 모더나

```
<h2>{{ question.question_text }}</h2>
<ul>
{% for choice in question.choice_set.all %}
    <li>{{ choice.choice_text }}</li>
{% endfor %}
</ul>
```

URL 네임 스페이스

◆ URL 별칭 사용하기 > 네임 스페이스

```
app_name = 'poll'

urlpatterns = [
    path('', views.index, name='index'),          # 127.0.0.1:8000/poll
    path('<int:pk>/', views.detail, name='detail'), # 127.0.0.1:8000/poll/1
]
```

<h1>설문 조사</h1>

{% for question in question_list %}

{{ question.question_text }}-->

{{ question.question_text }}

{% endfor %}

poll/index.html

설문 투표하기

어떤 백신을 접종했나요?

☐ 화이자

☐ 얀센

☐ 모더나

투표

← → ↻ ⓘ 127.0.0.1:8000/poll/1/vote/

투표 결과

어떤 백신을 접종했나요?

- 화이자 -- 1votes
- 얀센 -- 1votes
- 모더나 -- 1votes

어떤 백신을 접종했나요?

선택을 확인하세요

☐ 화이자

☐ 얀센

☐ 모더나

투표

선택하지 않은 경우 에러처리

설문 투표하기

◆ 투표하기

1. vote URL 매핑 추가하기 – poll/urls.py

```
app_name = 'poll' #네임 스페이스(소속)

urlpatterns = [
    path('', views.index, name='index'),
    path('<int:pk>/', views.detail, name='detail'),
    path('<int:pk>/vote/', views.vote, name='vote'),
]
```


설문 투표하기

2. vote 함수 정의 – poll/views.py

```
def vote(request, pk):  
    # 투표하기  
    question = Question.objects.get(id=pk)    # 질문 가져옴  
  
    choice_id = request.POST['choice']        # 선택자료 전달 받음  
    sel_choice = question.choice_set.get(id=choice_id)    # 선택 id로 db에서 검색  
    sel_choice.votes = sel_choice.votes + 1    # 1 더하기  
    sel_choice.save()                        # 저장  
    return render(request, 'poll/result.html', {'question': question})
```

설문 투표하기

3. 투표 양식 만들기 – poll/detail.html

```
<form action="{% url 'poll:vote' question.id %}" method="post">
    {% csrf_token %}
    <fieldset>
        <legend>{{ question.question_text }}</legend>
        {% for choice in question.choice_set.all %}
        <p>
            <input type="radio" name="choice" value="{{ choice.id }}">
            <label>{{ choice.choice_text }}</label>
        </p>
        {% endfor %}
    </fieldset>
    <p><input type="submit" value="투표"></p>
</form>
```

설문 투표하기

4. 투표 결과 페이지 – poll/results.html

```
<h2>{{ question.question_text }}</h2>
<ul>
    {% for choice in question.choice_set.all %}
        <li>
            {{ choice.choice_text }} -- {{ choice.votes }}votes
        </li>
    {% endfor %}
</ul>
```

설문 투표하기

5. 예러 처리하기 – poll/detail.html

```
<form action="{% url 'poll:vote' question.id %}" method="post">
    {% csrf_token %}
    <fieldset>
        <legend>{{ question.question_text }}</legend>
        {% if error %}
        <p class="error">{{ error}}</p>
        {% endif %}
        {% for choice in question.choice_set.all %}
        <p>
            <input type="radio" name="choice" value="{{ choice.id }}">
            <label>{{ choice.choice_text }}</label>
        </p>
        {% endfor %}
    </fieldset>
    <p><input type="submit" value="투표"></p>
</form>
```

설문 투표하기

5. 에러 처리하기 – poll/views.py

```
def vote(request, pk):
    # 투표하기
    question = Question.objects.get(id=pk)    # 질문 가져옴
    try:
        choice_id = request.POST['choice']    # 선택자료 전달 받음
        sel_choice = question.choice_set.get(id=choice_id)    # 선택 id로 db에서
    except:
        error = "선택을 하지 않았습니다."
        return render(request, 'poll/detail.html',
                      { 'question':question, 'error':error})
    else:
        sel_choice.votes = sel_choice.votes + 1    # 1 더하기
        sel_choice.save()    # 저장
        return render(request, 'poll/result.html', {'question':question})
```

설문 투표하기

➤ csrf_token 이 없는 경우 에러 발생

Forbidden (403)

CSRF 검증에 실패했습니다. 요청을 중단하였습니다.

Help

Reason given for failure:
CSRF token missing or incorrect.

In general, this can occur when there is a genuine Cross Site Request Forgery, or when [Django's CSRF mechanism](#) is

➤ csrf_token 템플릿 태그 사용하기

csrf_token

This tag is used for CSRF protection, as described in the documentation for [Cross Site Request Forgeries](#).

```
<form method="post">{% csrf_token %}
```

CSRF(cross-site request forgery)

➤ csrf

사이트 간 요청 위조

한글 20개 언어 ▼

위키백과, 우리 모두의 백과사전.

사이트 간 요청 위조(또는 **크로스 사이트 요청 위조**, **영어**: Cross-site request forgery, **CSRF**, **XSRF**)는 **웹사이트 취약점 공격**의 하나로, 사용자가 자신의 의지와는 무관하게 공격자가 의도한 행위(수정, 삭제, 등록 등)를 특정 웹사이트에 요청하게 하는 공격을 말한다.

유명 경매 사이트인 옥션에서 발생한 개인정보 유출 사건에서 사용된 공격 방식 중 하나다.

사이트 간 스크립팅(XSS)을 이용한 공격이 사용자가 특정 웹사이트를 신용하는 점을 노린 것이라면, 사이트간 요청 위조는 특정 웹사이트가 사용자의 **웹 브라우저**를 신용하는 상태를 노린 것이다. 일단 사용자가 웹사이트에 **로그인**한 상태에서 사이트간 요청 위조 공격 코드가 삽입된 페이지를 열면, 공격 대상이 되는 웹사이트는 위조된 공격 명령이 믿을 수 있는 사용자로부터 발송된 것으로 판단하게 되어 공격에 노출된다.

공격 과정 [편집]

1. 이용자는 웹사이트에 로그인하여 정상적인 **쿠키**를 발급받는다
2. 공격자는 다음과 같은 **링크**를 이메일이나 게시판 등의 경로를 통해 이용자에게 전달한다.

`http://www.geocities.com/attacker`

3. 공격용 **HTML** 페이지는 다음과 같은 이미지태그를 가진다.

```
<img src= "https://travel.service.com/travel_update?.src=Korea&.dst=Hell">
```

해당 링크는 클릭시 정상적인 경우 출발지와 도착지를 등록하기위한 링크이다. 위의 경우 도착지를 변조하였다.

4. 이용자가 공격용 페이지를 열면, 브라우저는 이미지 파일을 받아오기 위해 공격용 URL을 연다.
5. 이용자의 승인이나 인지 없이 출발지와 도착지가 등록됨으로써 공격이 완료된다. 해당 서비스 페이지는 등록 과정에 대해 단순히 쿠키를 통한 본인확인 밖에 하지 않으므로 공격자가 정상적인 이용자의 수정이 가능하게 된다.

csrf_token

➤ csrf_token

django는 csrf 공격에 대한 방어로 csrf_token을 발급 체크한다.

동작 과정

1. 사용자가 해당 페이지에 접속하면 Django에서 자동으로 csrf_token을 클라이언트로 보내어 cookie에 저장
2. 사용자가 form을 모두 입력한 후 제출버튼을 클릭한다.
3. form과 cookie의 csrf_token을 함께 POST로 전송한다.
4. 전송된 token의 유효성을 검증
5. 유효한 요청이면 요청을 처리
 1. token이 유효하지 않거나(없거나 값이 잘못된 경우) 검증 오류 시에는 403 Forbidden Response 반환