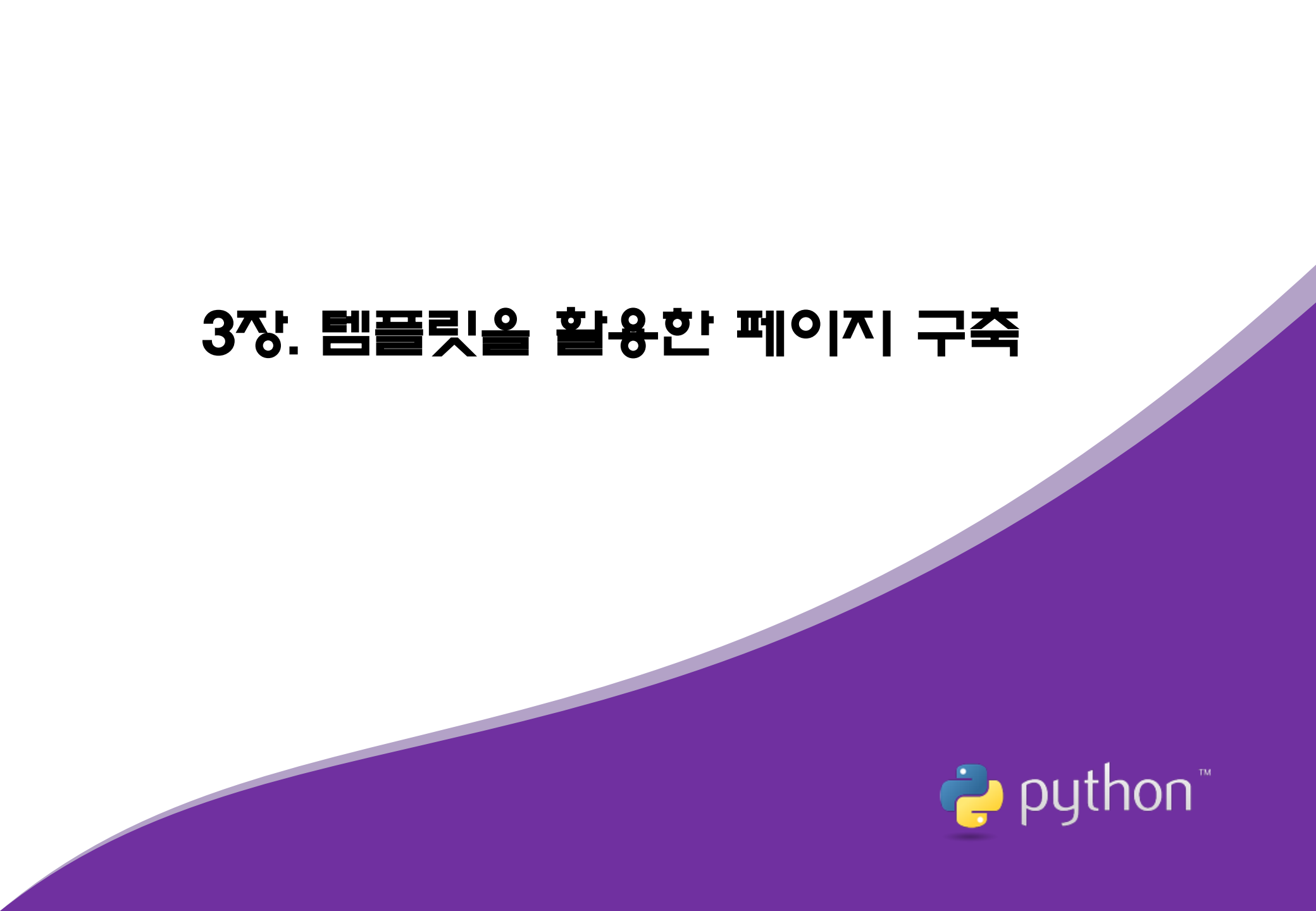
3장. 템플릿을 활용한 페이지 구축



목 차

템플릿으로 질문 목록 페이지 만들기

◆ 템플릿(template)

1. Templates 디렉터리를 루트 디렉터리 바로 밑에 만든다.

```
(mysite) C:\projects\mysite>mkdir  templates
```

2. 템플릿 디렉터리 위치를 config/settings.py에 등록하기

```
TEMPLATES = [
    ……
    'DIRS': [BASE_DIR / 'templates'],
    ……
]
```

템플릿으로 질문 목록 페이지 만들기

◆ 템플릿(template)

3. render로 화면 출력하기 – views.py

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

def index(request):
    # 질문 목록 조회
    question_list = Question.objects.order_by('create_date')
    context = {'question_list': question_list}
    return render(request, 'pybo/question_list.html', context)
```

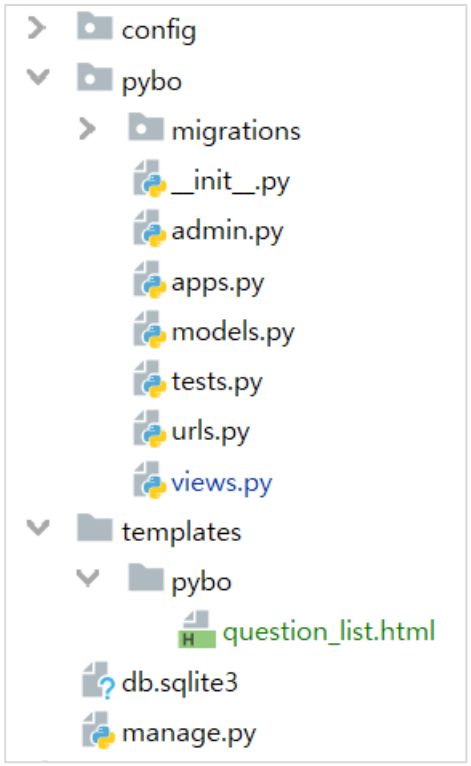
-create_date: 작성한 날짜의 역순으로 정렬

context : render 함수가 템플릿을 HTML로 변환하는 데이터(딕셔너리)

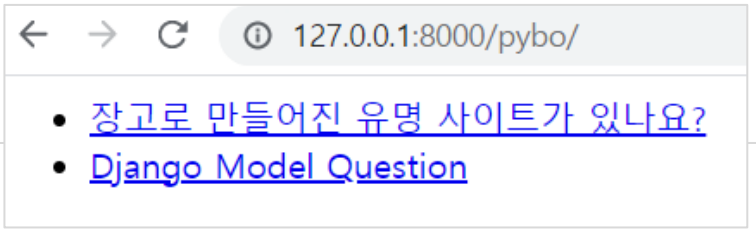
템플릿으로 질문 목록 페이지 만들기

4. 템플릿 파일 만들기

– 경로 : C:/projects/mysite/templates/pybo/question_list.html




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



템플릿으로 질문 목록 페이지 만들기

◆ 템플릿(template) 태그

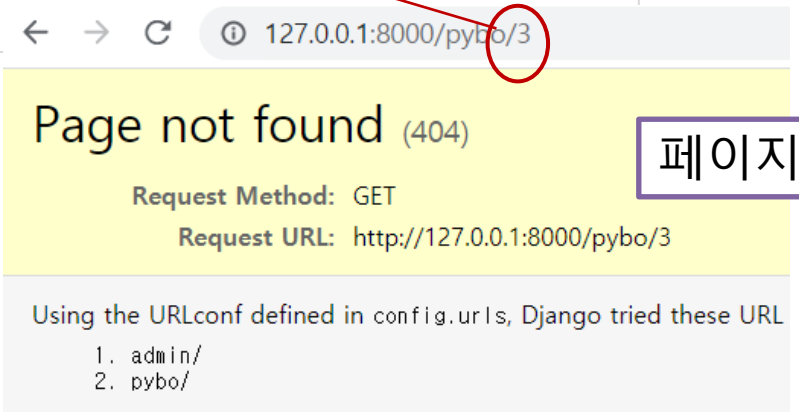
템플릿 태그	설 명
{% if question_list %}	question_list가 있다면(조건문)
{% for question in question_list %}	question_list를 반복하면 순차적으로 question에 대입(반복문)
{{ question.id }}	question 객체의 id 출력(출력문)
{{ question.subject }}	question 객체의 subject 출력

질문 상세 페이지 만들기

◆ 질문 상세 페이지

1. detail 페이지 URL 매핑하기 – pybo/urls.py

```
from django.urls import path
from pybo import views
urlpatterns = [
    path('', views.index),
    path('<int:question_id>/', views.detail),
]
```



페이지 만들기 전 링크 오류

질문 상세 페이지 만들기

◆ 질문 상세 페이지

2. detail 함수 구현 – pybo/views.py

```
def detail(request, question_id):
    question = Question.objects.get(id=question_id)
    context = {'question': question}
    return render(request, 'pybo/question_detail.html', context)
```

3. pybo/question_detail.html 작성

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

<div>
  {{ question.content }}
</div>
```

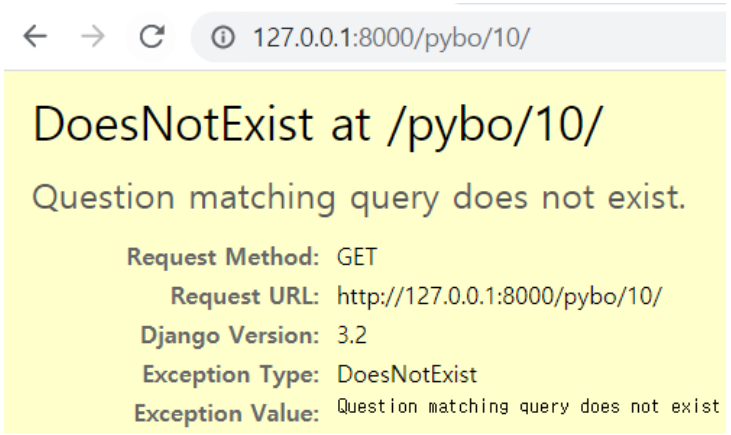
오류 화면 처리

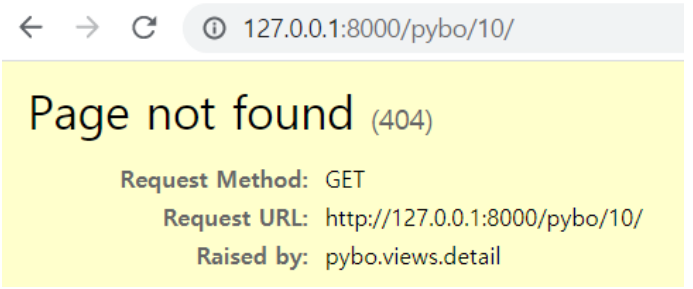
◆ 오류 화면 처리

- 존재하지 않는 페이지에 접속하면 오류 대신 404 페이지를 출력함

  get_object_or_404()를 사용

200은 성공
500은 코드 오류






```python
def detail(request, question_id):
    #question = Question.objects.get(id=question_id)
    question = get_object_or_404(Question, pk=question_id)
    context = {'question': question}
    return render(request, 'pybo/question_detail.html', context)
```

URL 네임 스페이스

◆ URL 별칭 사용하기 〉 네임 스페이스

1. URL에 실제 주소가 아닌 매핑된 주소를 사용 – pybo/urls.py

```
app_name = 'pybo'

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

2. 템플릿에서 URL 네임 스페이스 사용하기 – question_list.html

```
{% for question in question_list %}
  <li>
      <a href="{% url 'pybo:detail' question.id %}">{{question.subject}}</a>
  </li>
{% endfor %}
```

답변 등록 폼 만들기

◆ 답변 저장하고 표시하기

1. 질문 상세 페이지에 답변 등록 버튼 만들기 – question_detail.html

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

<div>
  {{ question.content }}
</div>

<form action="{% url 'pybo:answer_create' question.id %}" method="post">
  {% csrf_token %}
  <textarea name="content" id="content" rows="15"></textarea>
  <input type="submit" value="답변 등록">
</form>
```

<form action="이동할 페이지" method="post or get"></form>

csrf(cross-site request forgery: 사이트간 요청 위조>

답변 등록 폼 만들기

2. 답변 등록을 위한 URL 매핑 등록하기 – pybo/urls.py

```
urlpatterns = [
    path('', views.index, name='index'),
    path('<int:question_id>/', views.detail, name='detail'),
    path('answer/create/<int:question_id>', views.answer_create, name='answer_create')
]
```

3. answer_create 함수 정의하기 – pybo/views.py

```
def answer_create(request, question_id):
    # 답변 등록
    question = get_object_or_404(Question, pk=question_id)
    question.answer_set.create(content=request.POST.get('content'),
                               create_date=timezone.now())
    return redirect('pybo:detail', question_id=question.id)
```

답변 등록 폼 만들기

4. 등록된 답변 표시하기

```
<h2>{{ question.subject }}</h2>
<div>
    <p>{{ question.content }}</p>
</div>

<h5>{{ question.answer_set.count }}개의 답변이 있습니다.</h5>
<div>
    <ul>
    {% for answer in question.answer_set.all %}
        <li>{{ answer.content }}</li>
    {% endfor %}
    </ul>
</div>
```

답변 등록 폼 만들기


127.0.0.1:8000/pybo/1/
pybo란 무엇인가요
pybo가 무엇인지 설명해 주세요
2개의 답변이 있습니다.
pybo란 파이썬 장고 프레임워크로 만든 게시판입니다.
pybo란 pyboard의 약자로 파이썬 장고로 만든 게시판 사이트입니다.
답변 등록

CSRF(cross-site request forgery)

◆ csrf

사이트 간 요청 위조

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 공격의 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 브라우저를 신용하는 상태를 노린 것이다. 일단 사용자가 웹사이트에 로그인한 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 쿠키를 발급받는다
2. 공격자는 다음과 같은 링크를 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

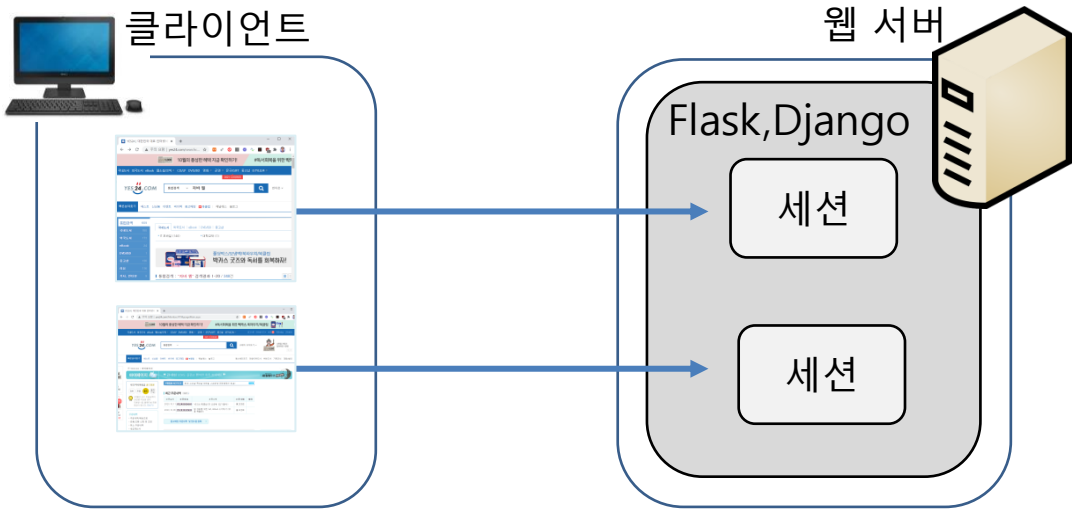
1. 사용자가 해당 페이지에 접속하면 Django에서 자동으로 csrf_token을 클라이언트로 보내어 cookie에 저장
2. 사용자가 form을 모두 입력한 후 제출버튼을 클릭한다.
3. form과 cookie의 csrf_token을 함께 POST로 전송한다.
4. 전송된 token의 유효성을 검증
5. 유효한 요청이면 요청을 처리
    1. token이 유효하지 않거나(없거나 값이 잘못된 경우) 검증 오류 시에는 403 Forbidden Response 반환

세션(session)

세션(session)

- 클라이언트와 웹 서버 간의 상태를 지속적으로 유지하는 방법을 말한다.
- 사용자 인증을 통해 특정 페이지를 사용할 수 있도록 권한 상태 유지.

  예) 웹 쇼핑몰 – 장바구니나 주문 처리와 같은 회원 전용 페이지 로그인 후 다른 웹 페이지에 갔다가 돌아와도 로그인 상태 유지됨
- 세션은 오직 웹 서버에 존재하는 객체로 웹 브라우저마다 하나씩 존재하므로 브라우저를 닫기 전까지 웹 페이지를 이동하더라도 사용자 정보가 유지된다.

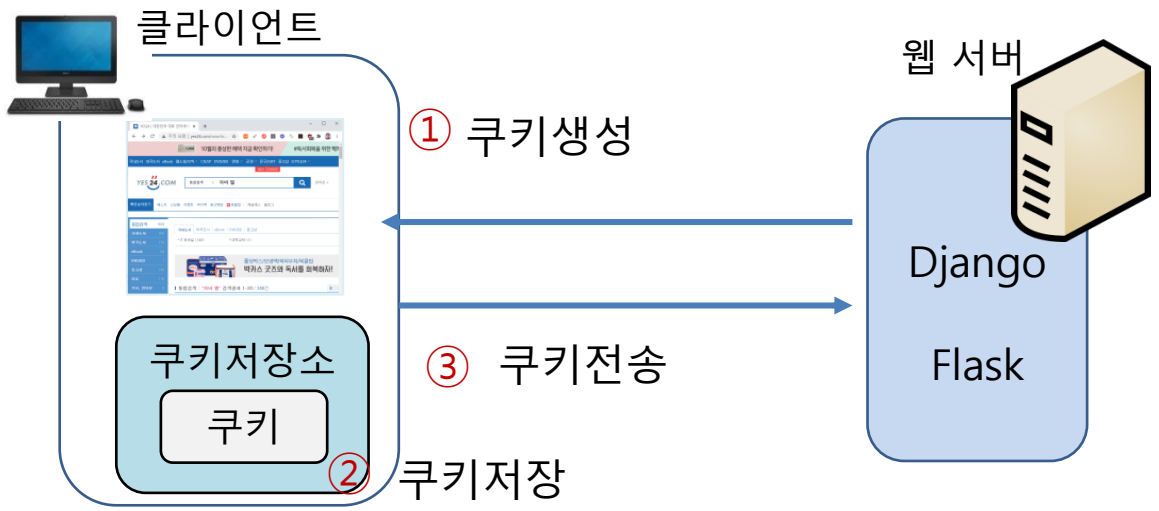


쿠키(cookie)

쿠키(cookie)

- 클라이언트와 웹 서버간의 상태를 지속적으로 유지하는 방법으로 쿠키와 세션이 있다.
- 쿠키는 세션과 달리 상태 정보를 웹 서버가 아닌 클라이언트에 저장한다.

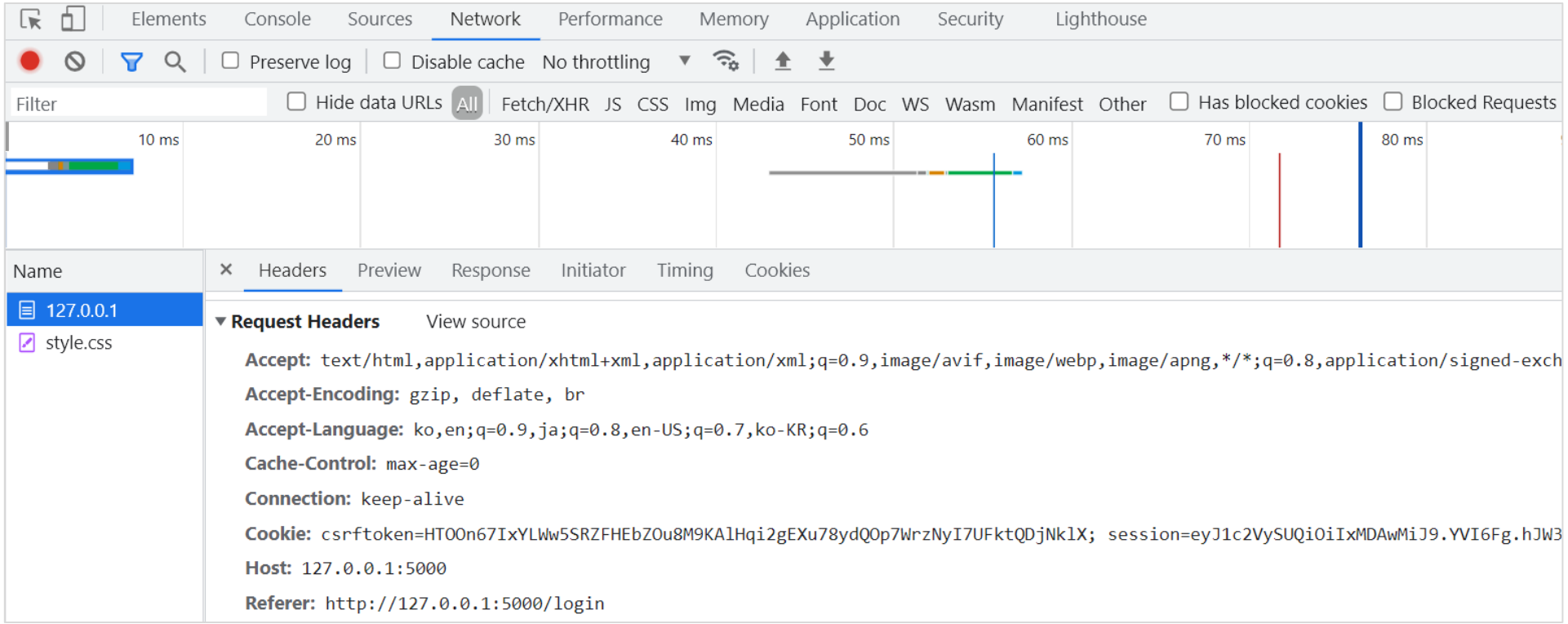
  예) 어떤 웹 사이트를 처음 방문한 사용자가 로그인 인증을 하고 나면 아이디와 비밀번호를 기록한 쿠키가 만들어지고, 그 다음부터 사용자가 그 사이트에 접속하면 별도의 절차를 거치지 않고 쉽게 접속할 수 있다.
- 쿠키는 클라이언트의 일정 폴더에 정보를 저장하므로 웹 서버의 부하를 줄일 수 있으나 개인 정보 기록이 남기 때문에 보안에 문제가 있다.



Session & Cookie

◆ 세션과 쿠키 확인

개발자도구 > Network



Session & Cookie

◆ 세션과 쿠키 확인

개발자도구 > Application