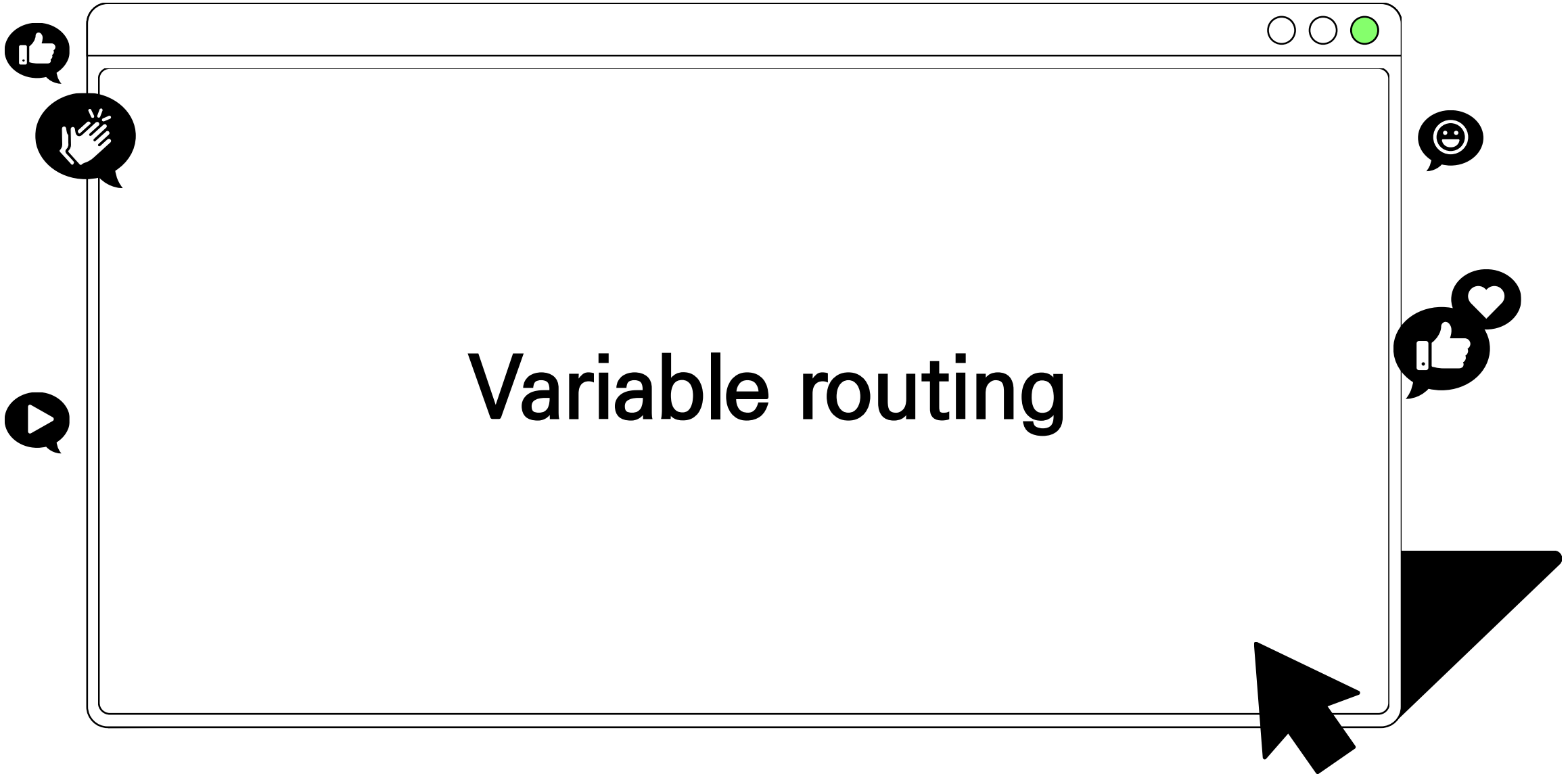
HAPPY HACKING



Variable routing

HAPPY HACKING

Variable routing의 필요성

- 템플릿의 많은 부분이 중복되고,

  일부분만 변경되는 상황에서 비슷한 URL과 템플릿을 계속해서 만들어야 할까?

Variable routing

- URL 주소를 변수로 사용하는 것을 의미
- URL의 일부를 변수로 지정하여 view 함수의 인자로 넘길 수 있음
- 즉, 변수 값에 따라 하나의 path()에 여러 페이지를 연결 시킬 수 있음

HAPPY HACKING

Variable routing 작성

- 변수는 “〈〉”에 정의하며 view 함수의 인자로 할당됨
- 기본 타입은 string이며 5가지 타입으로 명시할 수 있음

1. str
   - `/` 를 제외하고 비어 있지 않은 모든 문자열
   - 작성하지 않을 경우 기본 값
2. int
   - 0 또는 양의 정수와 매치
3. slug
4. uuid
5. path

```
# urls.py

urlpatterns = [
    ...,
    # path('hello/<str:name>/', views.hello),
    path('hello/<name>/', views.hello),
]
```

HAPPY HACKING

View 함수 작성

- variable routing으로 할당된 변수를 인자로 받고 템플릿 변수로 사용할 수 있음

```python
# articles/views.py

def hello(request, name):
    context = {
        'name': name,
    }
    return render(request, 'hello.html', context)
```

```
<!-- articles/templates/hello.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>만나서 반가워요 {{ name }}님!</h1>
{% endblock %}
```

HAPPY HACKING

Django Template

HAPPY HACKING

Django Template

- “데이터 표현을 제어하는 도구이자 표현에 관련된 로직”
- Django Template을 이용한 HTML 정적 부분과 동적 컨텐츠 삽입
- Template System의 기본 목표를 숙지

- Django Template System
  - 데이터 표현을 제어하는 도구이자 표현에 관련된 로직을 담당

HAPPY HACKING

Django Template Language (DTL)

- Django template에서 사용하는 built-in template system
- 조건, 반복, 변수 치환, 필터 등의 기능을 제공
  - Python처럼 일부 프로그래밍 구조(if, for 등)를 사용할 수 있지만
    이것은 Python 코드로 실행되는 것이 아님
  - Django 템플릿 시스템은 단순히 Python이 HTML에 포함 된 것이 아니니 주의
- 프로그래밍적 로직이 아니라 프레젠테이션을 표현하기 위한 것임을 명심할 것

DTL Syntax

- Variable
- Filters
- Tags
- Comments

HAPPY HACKING

Variable

```
{{ variable }}
```

- 변수명은 영어, 숫자와 밑줄(_)의 조합으로 구성될 수 있으나 밑줄로는 시작 할 수 없음
  - 공백이나 구두점 문자 또한 사용할 수 없음
- dot(.)를 사용하여 변수 속성에 접근할 수 있음
- render()의 세번째 인자로 {'key': value} 와 같이 딕셔너리 형태로 넘겨주며,
  여기서 정의한 key에 해당하는 문자열이 template에서 사용 가능한 변수명이 됨

HAPPY HACKING

Filters

```
{{ variable|filter }}
```

- 표시할 변수를 수정할 때 사용
- 예시) `{{ name|lower }}`
  - name 변수를 모두 소문자로 출력
- 60개의 built-in template filters를 제공
- chained가 가능하며 일부 필터는 인자를 받기도 함

```
{{ name|truncatewords:30 }}
```

HAPPY HACKING

Tags

```
{% tag %}
```

- 출력 텍스트를 만들거나, 반복 또는 논리를 수행하여 제어 흐름을 만드는 등 변수보다 복잡한 일들을 수행
- 일부 태그는 시작과 종료 태그가 필요 `{% if  %}{% endif %}`
- 약 24개의 built-in template tags를 제공

Comments

```
{# #}
```

- Django template에서 라인의 주석을 표현하기 위해 사용
- 한 줄 주석에만 사용할 수 있음 (줄 바꿈이 허용되지 않음)
- 여러 줄 주석은 {% comment %}와 {% endcomment %} 사이에 입력

```
{% comment %}
  여러 줄
  주석
{% endcomment %}
```

Django Template

HAPPY HACKING

- [실습] DTL Syntax (1/5) – “Variable”

```python
# urls.py

urlpatterns = [
    path('admin/', admin.site.urls),
    path('index/', views.index),
    path('greeting/', views.greeting),
]
```

```python
# articles/views.py

def greeting(request):
    return render(request, 'greeting.html', {'name': 'Alice'})
```

```html
<!-- articles/templates/greeting.html -->

<!DOCTYPE html>
<html lang="en">
<head>
  …
</head>
<body>
  <p>안녕하세요 저는 {{ name }} 입니다.</p>
</body>
</html>
```

HAPPY HACKING

[실습] DTL Syntax (2/5) – “Variable”

- context 데이터가 많아질 경우를 생각하면 다음과 같이 작성하는 것이 바람직
- context라는 이름은 다른 이름으로 사용 가능하지만 관행적으로 context를 사용

```python
# views.py

def greeting(request):
    foods = ['apple', 'banana', 'coconut',]
    info = {
        'name': 'Alice'
    }
    context = {
        'foods': foods,
        'info': info,
    }
    return render(request, 'greeting.html', context)
```

```html
<!-- articles/templates/greeting.html -->

<p>저는 {{ foods.0 }}을 가장 좋아합니다.</p>
<p>안녕하세요 저는 {{ info.name }} 입니다.</p>

<a href="/index/">뒤로</a>
```

❖ dot-lookup (dot-notaion)으로 배열의 인덱스 및 딕셔너리의 키 값에 접근 할 수 있음

HAPPY HACKING

[실습] DTL Syntax (3/5) – “Filters”

```python
# urls.py

urlpatterns = [
        ...,
        path('dinner/', views.dinner),
]
```

```python
# articles/views.py

import random
from django.shortcuts import render

...

def dinner(request):
    foods = ['족발', '햄버거', '치킨', '초밥',]
    pick = random.choice(foods)
    context = {
        'pick': pick,
        'foods': foods,
    }
    return render(request, 'dinner.html', context)
```

```html
<!-- articles/templates/dinner.html -->

<!DOCTYPE html>
<html lang="en">
<head>
...
</head>
<body>
  <p>{{ pick }}은 {{ pick|length }}글자</p>
  <p>{{ foods|join:", "}}</p>

  <a href="/index/">뒤로</a>
</body>
</html>
```

HAPPY HACKING

[실습] DTL Syntax (4/5) – “Filters”

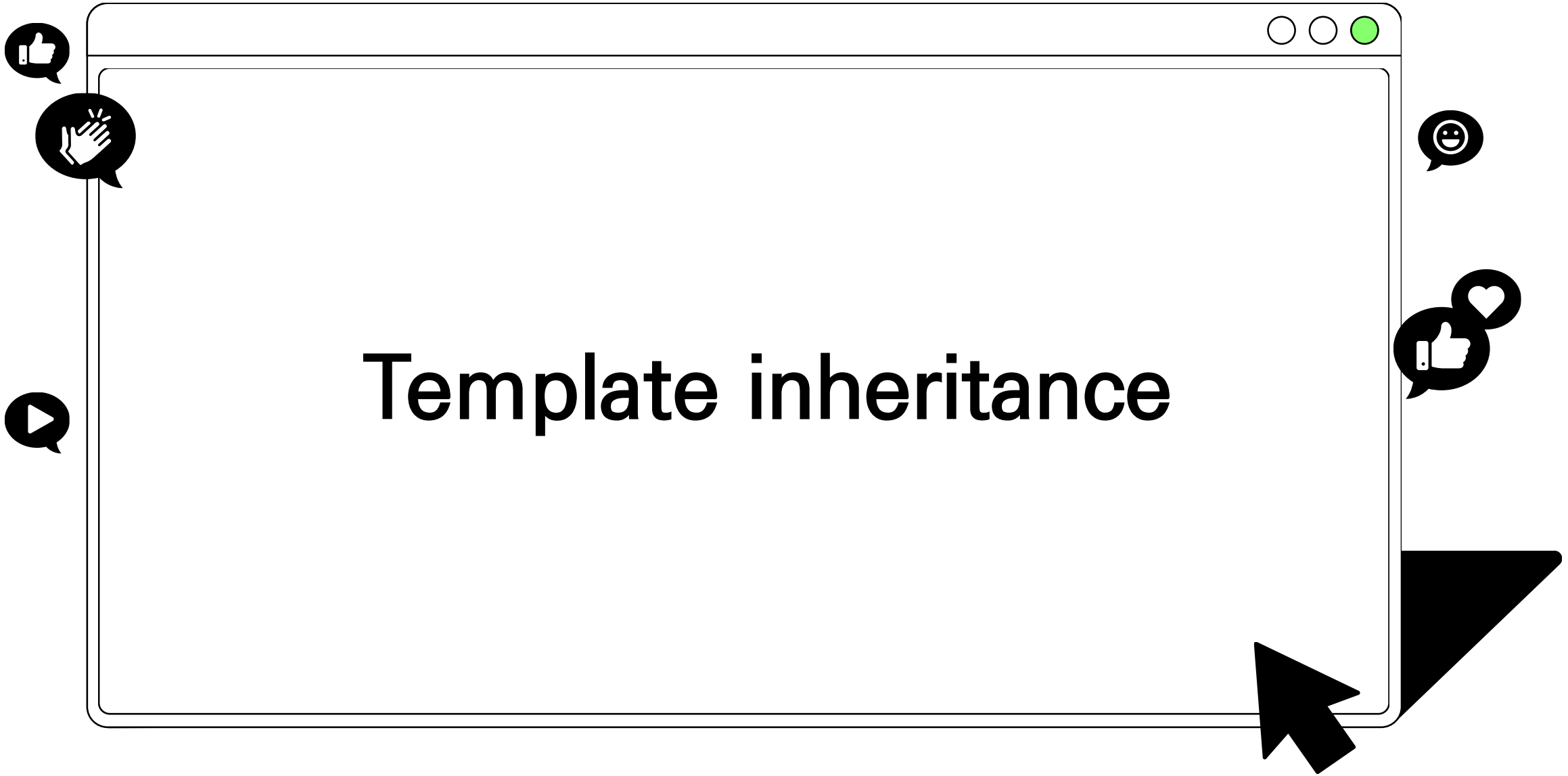
```
<!-- dinner.html -->

<!DOCTYPE html>
<html lang="en">
<head>
  ...
</head>
<body>
  <p>{{ pick }}은 {{ pick|length }}글자</p>
  <p>{{ foods|join:", "}}</p>

  <p>메뉴판</p>
  <ul>
    {% for food in foods %}
      <li>{{ food }}</li>
    {% endfor %}
  </ul>

  <a href="/index/">뒤로</a>
</body>
</html>
```

HAPPY HACKING

[실습] DTL Syntax (5/5) – “Comments”

```
<!-- dinner.html -->

<!DOCTYPE html>
<html lang="en">
<head>
  ...
</head>
<body>
  ...

  {# 이것은 주석입니다. #}

  {% comment %}
    <p>여러 줄</p>
    <p>주석</p>
    <p>입니다.</p>
  {% endcomment %}

  <a href="/index/">뒤로</a>
</body>
</html>
```

HAPPY HACKING



Template inheritance

템플릿 상속

- 템플릿 상속은 기본적으로 코드의 재사용성에 초점을 맞춤
- 템플릿 상속을 사용하면 사이트의 모든 공통 요소를 포함하고,
  하위 템플릿이 재정의(override) 할 수 있는 블록을 정의하는
  기본 ‘skeleton’ 템플릿을 만들 수 있음

- 만약 모든 템플릿에 부트스트랩을 적용하려면 어떻게 해야 할까?
  - 모든 템플릿에 부트스트랩 CDN을 작성해야 할까?

HAPPY HACKING

템플릿 상속에 관련된 태그

```
{% extends '' %}
```

- 자식(하위)템플릿이 부모 템플릿을 확장한다는 것을 알림

❖ 반드시 템플릿 최상단에 작성 되어야 함 (즉, 2개 이상 사용할 수 없음)

```
{% block content %}{% endblock content %}
```

- 하위 템플릿에서 재지정(overridden)할 수 있는 블록을 정의
- 즉, 하위 템플릿이 채울 수 있는 공간
- 가독성을 높이기 위해 선택적으로 endblock 태그에 이름을 지정할 수 있음

HAPPY HACKING

템플릿 상속 예시 (1/2)

- base라는 이름의 skeleton 템플릿을 작성
- Bootstrap CDN 작성

```
<!-- articles/templates/base.html -->

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!-- bootstrap CDN 작성 -->
  <title>Document</title>
</head>
<body>
  {% block content %}
  {% endblock content %}
  <!-- bootstrap CDN 작성 -->
</body>
</html>
```

HAPPY HACKING

템플릿 상속 예시 (2/2)

- index 템플릿에서 base 템플릿을 상속받음
- Bootstrap이 적용되었는지 확인

```
<!-- index.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>만나서 반가워요!</h1>
  <a href="/greeting/">greeting</a>
  <a href="/dinner/">dinner</a>
{% endblock content %}
```

HAPPY HACKING

추가 템플릿 경로 추가하기

- base.html의 위치를 앱 안의 template 디렉토리가 아닌 프로젝트 최상단의 templates 디렉토리 안에 위치하고 싶다면 어떻게 해야 할까?
- 기본 template 경로가 아닌 다른 경로를 추가하기위해 다음과 같은 코드를 작성

```python
# settings.py

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [BASE_DIR / 'templates',],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
```

HAPPY HACKING

추가 템플릿 경로

- app_name/templates/ 디렉토리 경로 외 추가 경로를 설정한 것
- base.html의 위치를 다음과 같이 이동 후 상속에 문제가 없는지 확인

```
articles/
firstpjt/
templates/
    base.html
```

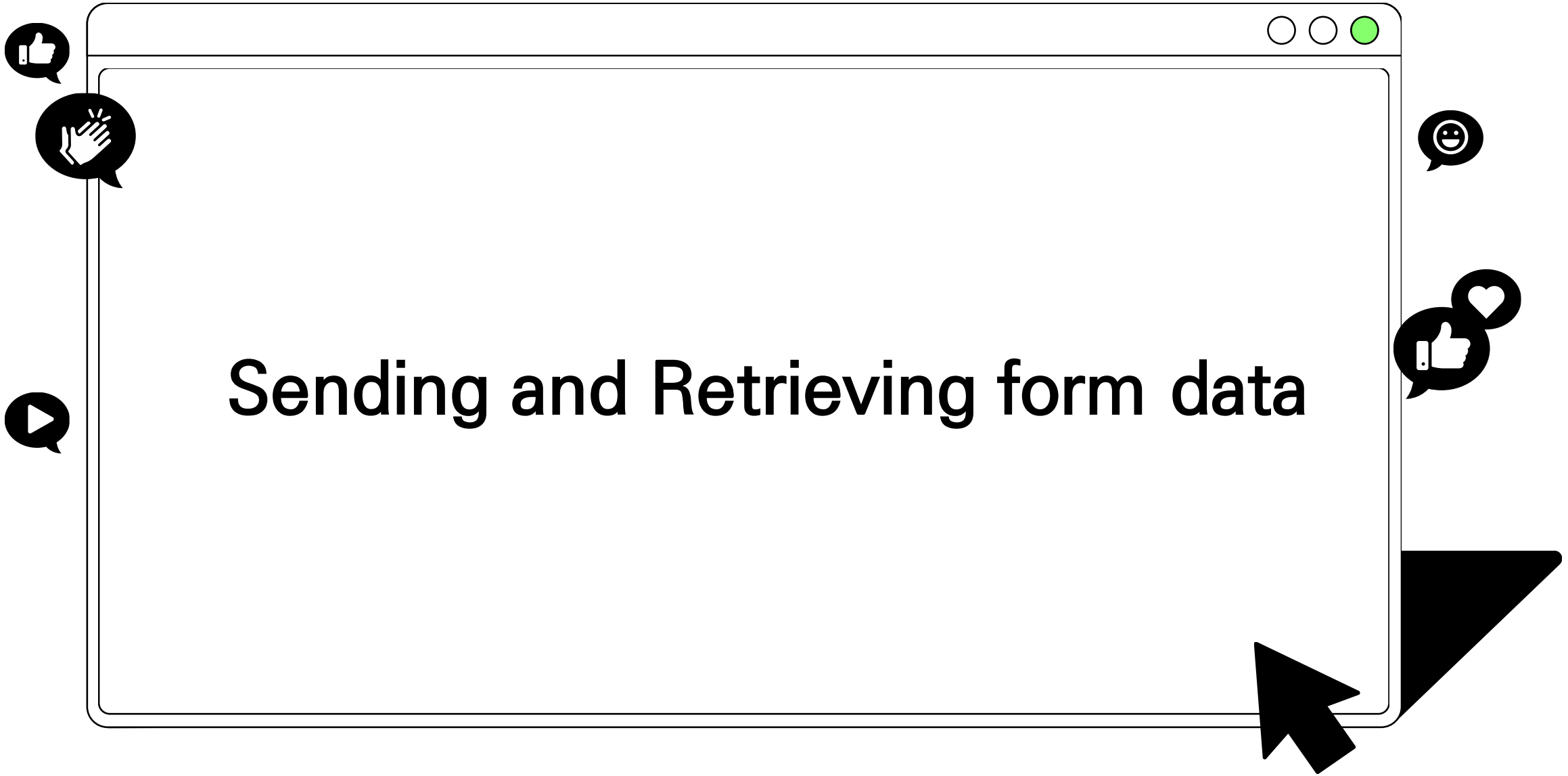
HAPPY HACKING

[참고] BASE_DIR

```python
# settings.py

BASE_DIR = Path(__file__).resolve().parent.parent
```

- settings.py에서 특정 경로를 절대 경로로 편하게 작성할 수 있도록 Django에서 미리 지정해둔 경로 값
- “객체 지향 파일 시스템 경로”
  - 운영체제별로 파일 경로 표기법이 다르기 때문에 어떤 운영체제에서 실행되더라도 각 운영체제 표기법에 맞게 해석될 수 있도록 하기 위해 사용
  - 자세한 내용은 공식문서에서 확인하기
  - https://docs.python.org/ko/3.9/library/pathlib.html#module-pathlib

HAPPY HACKING



Sending and Retrieving form data

HAPPY HACKING

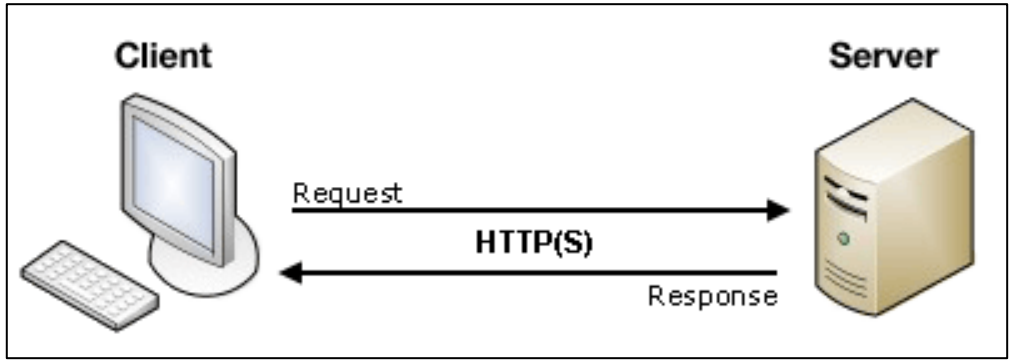
Sending and Retrieving form data

- “데이터를 보내고 가져오기”
- HTML form element를 통해 사용자와 애플리케이션 간의 상호작용 이해하기

Sending and Retrieving form data

HAPPY HACKING

Client & Server architecture

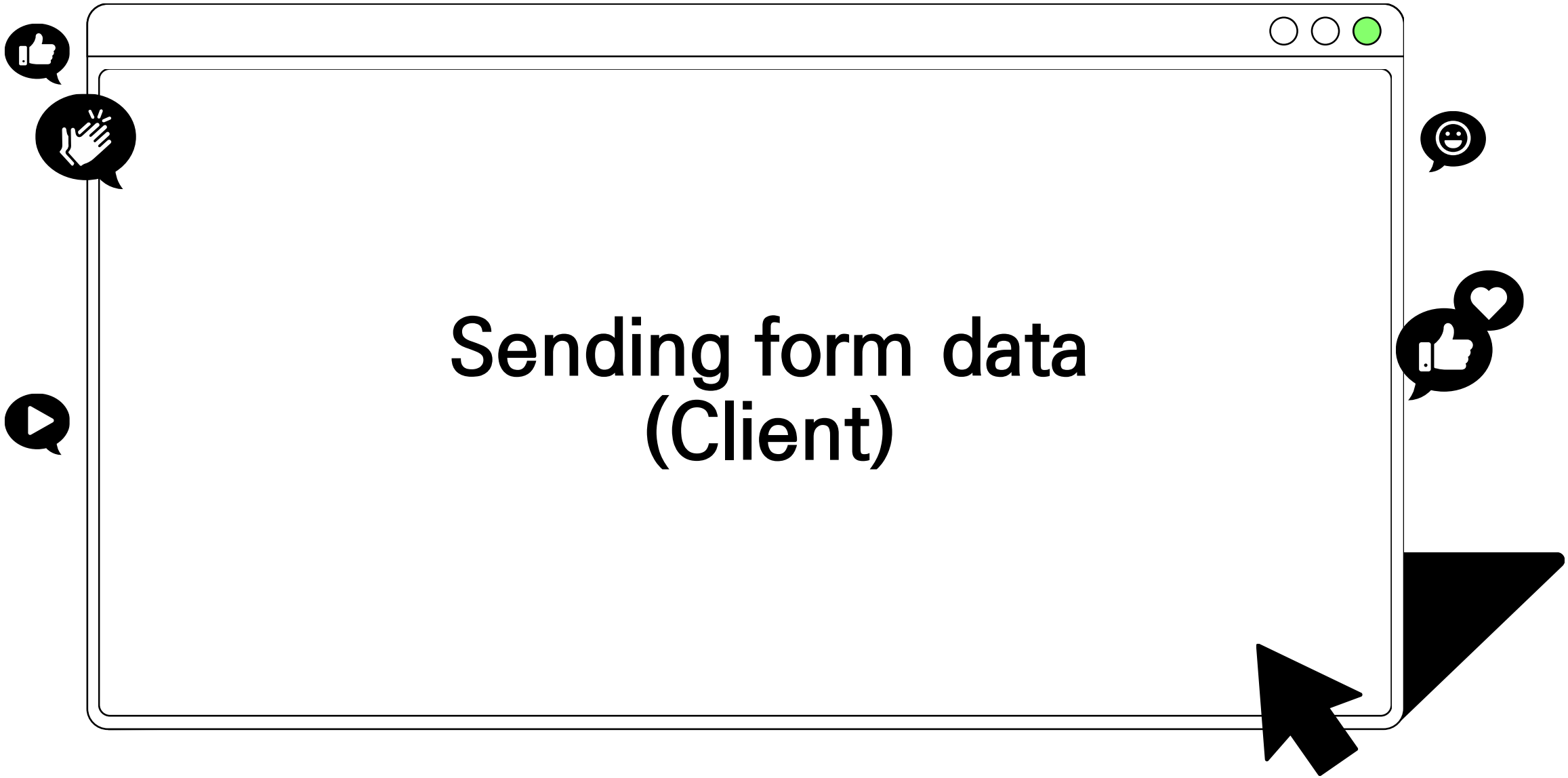


서버-클라이언트 구조 (출처 – MDN)

- 웹은 다음과 같이 가장 기본적으로 클라이언트-서버 아키텍처를 사용
  - 클라이언트(일반적으로 웹 브라우저)가 서버에 요청을 보내고,
    서버는 클라이언트의 요청에 응답
- 클라이언트 측에서 HTML form은 HTTP 요청을 서버에 보내는 가장 편리한 방법
- 이를 통해 사용자는 HTTP 요청에서 전달할 정보를 제공할 수 있음

<!-- JavaScript Bundle with Popper --> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.bundle.min.js" integrity="sha384-u1OknCvxWvY5kfmNBILK2hRnQC3Pr17a+RTT6rIHI7NnikvbZlHgTP

HAPPY HACKING



Sending form data (Client)

HAPPY HACKING

HTML 〈form〉 element

- 데이터가 전송되는 방법을 정의
- 웹에서 사용자 정보를 입력하는 여러 방식(text, button, submit 등)을 제공하고, **사용자로부터 할당된 데이터를 서버로 전송**하는 역할을 담당
- “데이터를 어디(action)로 어떤 방식(method)으로 보낼지”
- 핵심 속성
  - action
  - method

HAPPY HACKING

HTML form's attributes

1. action

- 입력 데이터가 전송될 URL을 지정
- 데이터를 어디로 보낼 것인지 지정하는 것이며 이 값은 반드시 유효한 URL이어야 함
- 만약 이 속성을 지정하지 않으면 데이터는 현재 form이 있는 페이지의 URL로 보내짐

2. method

- 데이터를 어떻게 보낼 것인지 정의
- 입력 데이터의 HTTP request methods를 지정
- HTML form 데이터는 오직 2가지 방법으로만 전송 할 수 있는데
  바로 GET 방식과 POST 방식

Sending form data (Client)

HAPPY HACKING

```python
# urls.py

urlpatterns = [
    ...,
    path('throw/', views.throw),
]
```

```python
# articles/views.py

def throw(request):
    return render(request, 'throw.html')
```

```html
<!-- articles/templates/throw.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>Throw</h1>
  <form action="#" method="#">
  </form>
{% endblock content %}
```

HAPPY HACKING

HTML 〈input〉 element

- 사용자로부터 데이터를 입력 받기 위해 사용
- “type” 속성에 따라 동작 방식이 달라진다.
  - input 요소의 동작 방식은 type 특성에 따라 현격히 달라지므로
    각각의 type은 별도로 MDN 문서에서 참고하여 사용하도록 함
  - type을 지정하지 않은 경우, 기본값은 ”text”
- 핵심 속성
  - name

HAPPY HACKING

HTML input's attribute

- **name**
  - form을 통해 데이터를 제출(submit)했을 때 name 속성에 설정된 값을 서버로 전송하고, 서버는 name 속성에 설정된 값을 통해 사용자가 입력한 데이터 값에 접근할 수 있음
  - 주요 용도는 GET/POST 방식으로 서버에 전달하는 파라미터(name은 key, value는 value)로 매핑하는 것
    - GET 방식에서는 URL형식으로 데이터를 전달

```
'?key=value&key=value/'
```

HAPPY HACKING

HTML 〈input〉 element 작성

```
<!-- articles/templates/throw.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>Throw</h1>
  <form action="#" method="#">
    <label for="message">Throw</label>
    <input type="text" id="message" name="message">
    <input type="submit">
  </form>
{% endblock %}
```

HAPPY HACKING

HTTP request methods (1/2)

- HTTP
  - HTML 문서와 같은 리소스(데이터, 자원)들을 가져올 수 있도록 해주는 프로토콜(규칙, 규약)
- 웹에서 이루어지는 모든 데이터 교환의 기초
- HTTP는 주어진 리소스가 수행 할 원하는 작업을 나타내는 request methods를 정의

HTTP request methods (2/2)

- 자원에 대한 행위(수행하고자 하는 동작)을 정의
- 주어진 리소스(자원)에 수행하길 원하는 행동을 나타냄
- HTTP Method 예시
  - GET, POST, PUT, DELETE
- GET이 아닌 다른 method는 추후 다시 알아볼 예정

HAPPY HACKING

GET

- 서버로부터 정보를 조회하는 데 사용
  - 즉, 서버에게 리소스를 요청하기 위해 사용
- 데이터를 가져올 때만 사용해야 함
- 데이터를 서버로 전송할 때 Query String Parameters를 통해 전송
  - 데이터는 URL에 포함되어 서버로 보내짐

HAPPY HACKING

GET 메서드 작성

- GET과 get 모두 대소문자 관계없이 동일하게 동작하지만
  명시적 표현을 위해 대문자 사용을 권장
- 데이터를 입력 후 submit 버튼을 누르고 URL의 변화를 확인한다.

```
<!-- throw.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>Throw</h1>
  <form action="#" method="GET">
    <label for="message">Throw</label>
    <input type="text" id="message" name="message">
    <input type="submit">
  </form>
{% endblock %}
```

HAPPY HACKING

Query String Parameters (1/2)

- 사용자가 입력 데이터를 전달하는 방법 중 하나로,
  url 주소에 데이터를 파라미터를 통해 넘기는 것
- 이러한 문자열은 앰퍼샌드(&)로 연결된 key=value 쌍으로 구성되며
  기본 URL과 물음표(?) 로 구분됨
  - 예시
    - http://host:port/path**?key=value&key=value**
- Query String이라고도 함

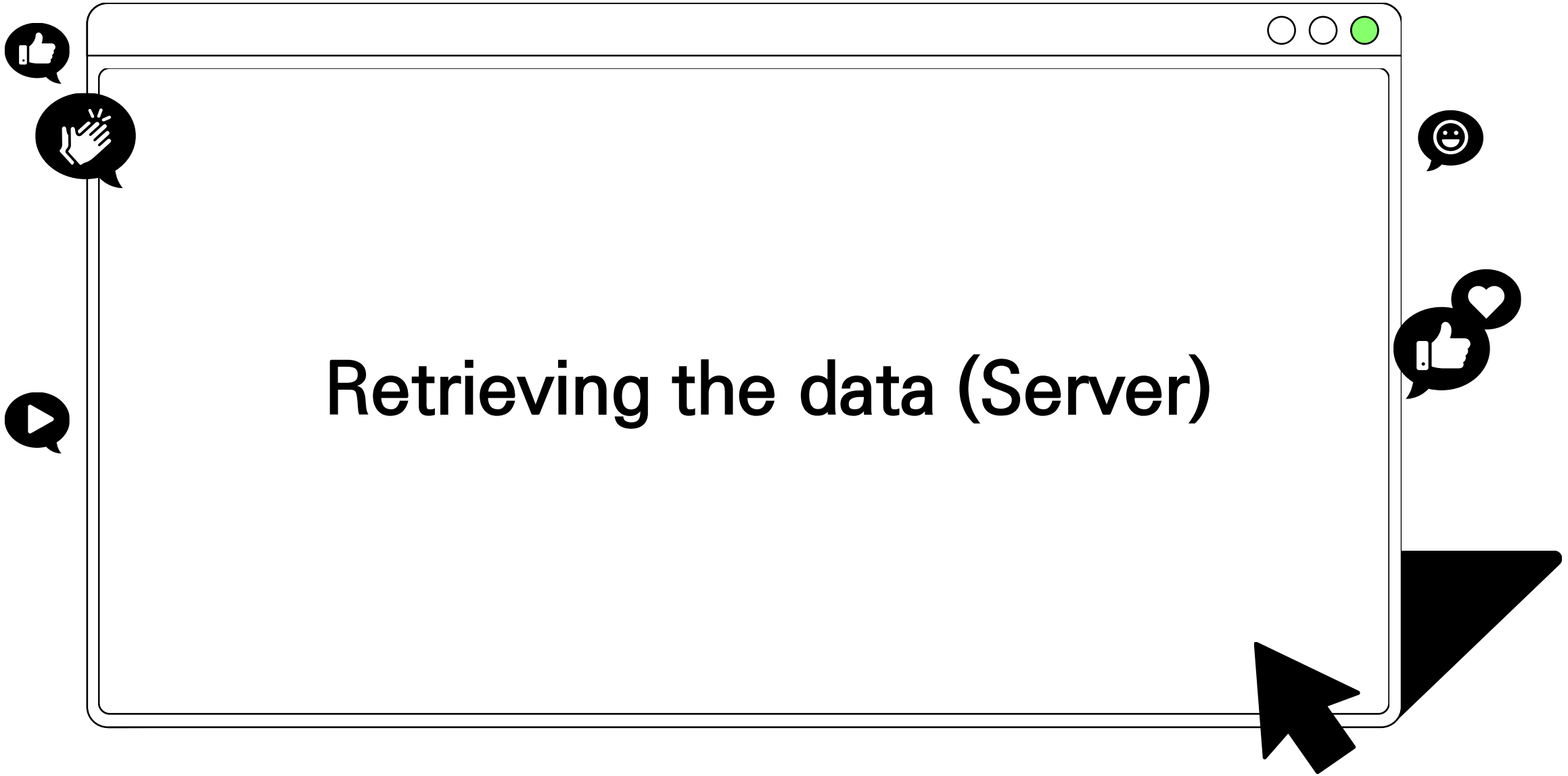
HAPPY HACKING

Query String Parameters (2/2)

- 정해진 주소 이후에 물음표를 쓰는 것으로 Query String이 시작함을 알림
- “key=value”로 필요한 파라미터의 값을 적음
  - “=“ 로 key와 value가 구분됨
- 파라미터가 여러 개일 경우 “&”를 붙여 여러 개의 파라미터를 넘길 수 있음

- 그런데 아직 어디로 보내야(action) 할 지 작성하지 않았다.

HAPPY HACKING



Retrieving the data (Server)

HAPPY HACKING

Retrieving the data (Server)

- “데이터 가져오기(검색하기)”
- 서버는 클라이언트로 받은 key-value 쌍의 목록과 같은 데이터를 받게 됨
- 이러한 목록에 접근하는 방법은 사용하는 특정 프레임워크에 따라 다름
- 우리는 Django 프레임워크에서 어떻게 데이터를 가져올 수 있을지 알아볼 것
  - throw가 보낸 데이터를 catch에서 가져오기

Retrieving the data (Server)

HAPPY HACKING

```python
# urls.py

urlpatterns = [
    ...,
    path('catch/', views.catch),
]
```

```python
# articles/views.py

def catch(request):
    pass
    return render(request, 'catch.html')
```

```html
<!-- articles/templates/catch.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>Catch</h1>
  <h2>여기서 데이터를 받았어!!</h2>
  <a href="/throw/">다시 던지러</a>
{% endblock %}
```

Retrieving the data (Server)

HAPPY HACKING

action 작성

- throw 페이지에서 form의 action 부분을 마저 작성하고 데이터를 보낸다.
- 실습 편의를 위해 index 페이지에 throw 하이퍼 링크를 작성한다.

```
<!-- articles/templates/throw.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>Throw</h1>
  <form action="/catch/" method="GET">
    <label for="message">Throw</label>
    <input type="text" id="message" name="message">
    <input type="submit">
  </form>

  <a href="/index/">뒤로</a>
{% endblock %}
```

```
<!-- articles/templates/index.html -->

{% extends 'base.html' %}

{% block content %}
  ...
  <a href="/throw/">throw</a>
{% endblock %}
```

Retrieving the data (Server)

HAPPY HACKING

데이터 가져오기

- catch 페이지가 잘 응답되어 출력됨을 확인
- 그런데 throw 페이지의 form이 보낸 데이터는 어디에 들어 있는걸까?
  - catch 페이지의 url 확인 `http://127.0.0.1:8000/catch/?message=데이터`
  - GET method로 보내고 있기 때문에 데이터를 서버로 전송할 때 Query String Parameters를 통해 전송
  - 즉, 데이터는 URL에 포함되어 서버로 보내짐
- 그러면 우리가 작성해야 하는 view 함수에서는 해당 데이터에 어떻게 접근 가능할까?

Retrieving the data (Server)

HAPPY HACKING

데이터 가져오기

- “모든 요청 데이터는 view 함수의 첫번째 인자 request에 들어있다.”
- request가 어떤 객체인지 확인해보기

Retrieving the data (Server)

HAPPY HACKING

request 객체 살펴보기 – 1

- print를 통해 살펴보기

```python
# articles/views.py

def catch(request):
    print(request)
    print(type(request))
    print(request.GET)
    print(request.GET.get('message'))
    return render(request, 'catch.html')
```

- 출력 결과

```
<WSGIRequest: GET '/catch/?message=%EB%8D%B0%EC%9D%B4%ED%84%B0%21'>
<class 'django.core.handlers.wsgi.WSGIRequest'>
<QueryDict: {'message': ['데이터']}>
'데이터'
```

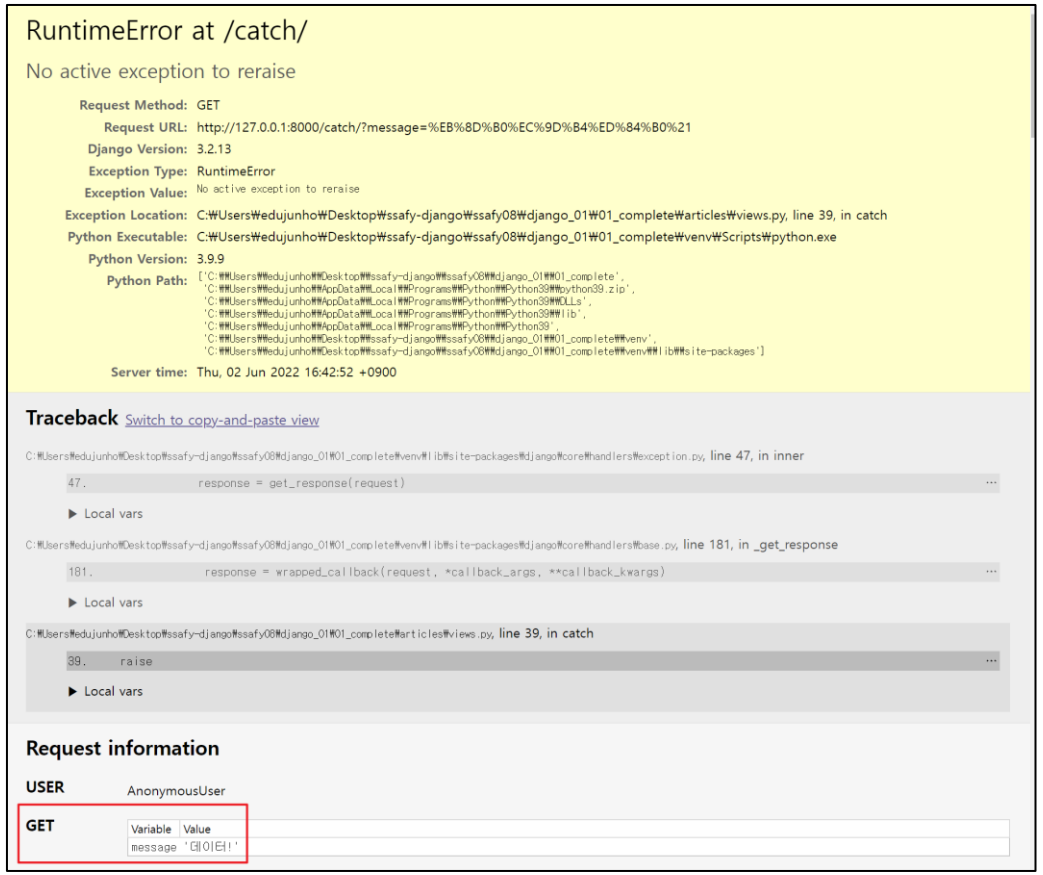
Retrieving the data (Server)

HAPPY HACKING

request 객체 살펴보기 – 2

- 에러를 강제로 발생시켜 에러 페이지 하단에서 살펴보기

```python
# articles/views.py

def catch(request):
    raise
    return render(request, 'catch.html')
```



Retrieving the data (Server)

HAPPY HACKING

- catch 작성 마무리

```python
def catch(request):
    message = request.GET.get('message')
    context = {
        'message': message,
    }
    return render(request, 'catch.html', context)
```

```
<!-- articles/templates/catch.html -->

{% extends 'base.html' %}

{% block content %}
  <h1>Catch</h1>
  <h2>여기서 {{ message }}를 받았어!!</h2>
  <a href="/throw/">다시 던지러</a>
{% endblock %}
```

Retrieving the data (Server)

HAPPY HACKING

catch 작성 마무리

- 데이터를 보낸 후 결과 확인

Retrieving the data (Server)

HAPPY HACKING

Request and Response objects

- 요청과 응답 객체 흐름

1. 페이지가 요청되면 Django는 요청에 대한 메타데이터를 포함하는 HttpRequest object를 생성
2. 그리고 해당하는 적절한 view 함수를 로드하고 HttpRequest를 첫번째 인자로 전달
3. 마지막으로 view 함수는 HttpResponse object를 반환

HAPPY HACKING

Django URLs

HAPPY HACKING

Django URLs

- “Dispatcher(운행 관리원)로서의 URL 이해하기”
- 웹 어플리케이션은 URL을 통한 클라이언트의 요청에서부터 시작함

Trailing Slashes

- Django는 URL 끝에 /가(Trailing slash) 없다면 자동으로 붙여주는 것이 기본 설정
  - 그래서 모든 주소가 `/`로 끝나도록 구성 되어있음
  - 그러나 모든 프레임워크가 이렇게 동작하는 것은 아님
- Django의 url 설계 철학을 통해 먼저 살펴보면 다음과 같이 설명함

  “기술적인 측면에서, **foo.com/bar**와 **foo.com/bar/**는 서로 다른 URL이다.”
  - 검색 엔진 로봇이나 웹 트래픽 분석 도구에서는 그 둘을 서로 다른 페이지로 봄
  - 그래서 Django는 URL을 정규화하여 검색 엔진 로봇이 혼동하지 않게 해야 함

[참고] URL 정규화

- 정규 URL(=오리지널로 평가되어야 할 URL)을 명시하는 것
- 복수의 페이지에서 같은 콘텐츠가 존재하는 것을 방지하기 위함

- “Django에서는 trailing slash가 없는 요청에 대해 자동으로 slash를 추가하여 통합된 하나의 콘텐츠로 볼 수 있도록 한다.”