



파이썬/장고 웹서비스 개발 완벽 가이드 with 리액트 (3/E)

JS 없는 htmx 웹 프론트엔드 UI 개발

당신의 파이썬/장고 페이스메이커가 되겠습니다.

Tag 모델/뷰/템플릿 기본 구성

Tag 모델/뷰/템플릿 기본 구성

blog/models.py

```
class Tag(models.Model):
    name = models.CharField(max_length=100,
                           unique=True)
```

```
def __str__(self) -> str:
    return self.name
```

```
class Meta:
    ordering = ["-pk"]
```

blog/forms.py

```
class TagForm(forms.ModelForm):
    class Meta:
        model = Tag
        fields = ["name"]
```

blog/views.py

```
def tag_list(request):
    tag_qs = Tag.objects.all()

    query = request.GET.get("query", "")
    if query:
        tag_qs = tag_qs.filter(name__icontains=query)

    return render(request, "blog/tag_list.html", {
        "tag_list": tag_qs,
    })
```

```
def tag_new(request):
    if request.method == "GET":
        form = TagForm()
    else:
        form = TagForm(data=request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "태그를 저장했습니다.")
            return redirect("blog:tag_list")
```

```
template_name = "blog/tag_form.html"
```

```
return render(request, template_name, {
    "form": form,
})
```

blog/urls.py

```
urlpatterns += [
    path("tags/", views.tag_list, name="tag_list"),
    path("tags/new/", views.tag_new, name="tag_new"),
]
```

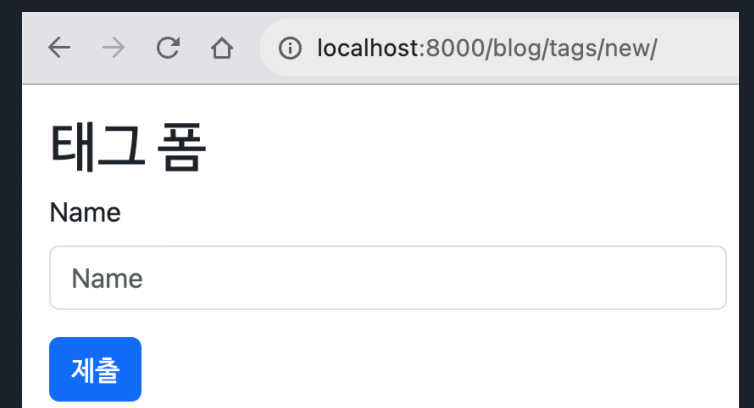
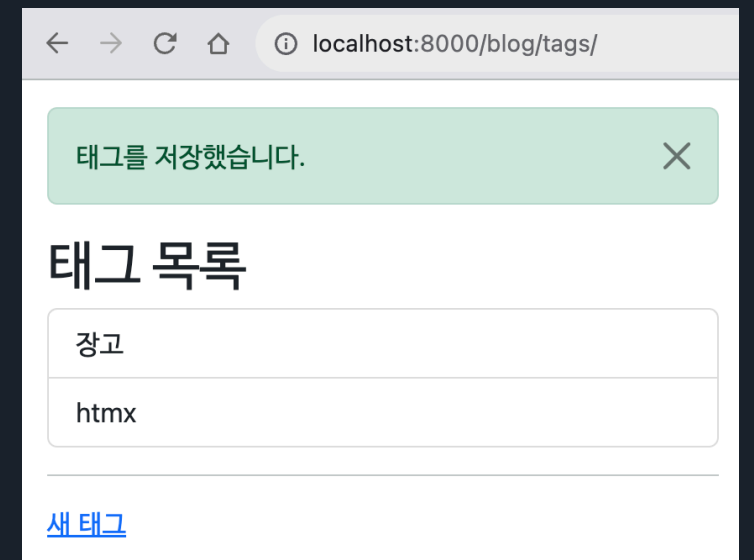
```
{# blog/templates/blog/base.html #}
{% load django_bootstrap5 %}
<!doctype html>
<html lang="ko">
<head>
    <meta charset="UTF-8" />
    <title>{% block title %}{% endblock %}</title>
    {% bootstrap_css %}
    {% bootstrap_javascript %}
    <script src="https://unpkg.com/htmx.org@1.9.10/dist/htmx.min.js"></script>
</head>
<body>
    <main style="width: 400px; margin: 1em;">
        {% bootstrap_messages %}
        {% block content %}{% endblock %}
    </main>
    {% block extra-script %}{% endblock %}
</body>
</html>
```

```
{# blog/templates/blog/_tag_list.html : HTML 레이아웃없이 콘텐츠로만 구성 #}
<div class="list-group">
{% for tag in tag_list %}
    <div class="list-group-item">{{ tag.name }}</div>
{% endfor %}
</div>
```

```
{# blog/templates/blog/tag_list.html #}
{% extends "blog/base.html" %}
{% block title %}태그 목록{% endblock %}
{% block content %}
    <h2>태그 목록</h2>
    {% include "blog/_tag_list.html" %}
    <hr />
    <a href="{% url 'blog:tag_new' %}">새 태그</a>
{% endblock %}
```

```
{# blog/templates/blog/_tag_form.html : HTML 레이아웃없이 콘텐츠로만 구성 #}
{% load django_bootstrap5 %}
<form action="{% url 'blog:tag_new' %}" method="post" autocomplete="off"
    novalidate>
    {% csrf_token %}
    {% bootstrap_form form %}
    <input type="submit" class="btn btn-primary" />
</form>
```

```
{# blog/templates/blog/tag_form.html #}
{% extends "blog/base.html" %}
{% block title %}태그 폼{% endblock %}
{% block content %}
    <h2>태그 폼</h2>
    {% include "blog/_tag_form.html" %}
{% endblock %}
```



<https://gist.github.com/allieus/4f20e7ca6c4596f19694bed31fb44a89>

htmx 간단 예시

click to load 예시

vanilla js 버전

```
{# blog/templates/blog/test.html #}
```

```
<!doctype html>
<html lang="ko">
<head>
<meta charset="UTF-8"/>
<title>vanilla js</title>
<style>
/* show 클래스가 적용되면, 애니메이션과 함께 투명도(opacity)가 1로 변경 */
#tag-list-container { transition: opacity 300ms ease-in-out; opacity: 0; }
#tag-list-container.show { opacity: 1; }
</style>
</head>
<body>
```

```
<button id="refresh-button">새로고침</button>
```

```
<div id="tag-list-container"></div>
```

```
<script>
const buttonEl = document.querySelector("#refresh-button");
const containerEl = document.querySelector("#tag-list-container");
```

```
buttonEl.addEventListener('click', async () => {
  try {
    const response = await fetch('/blog/tags/')
    const htmlText = await response.text();
```

```
    // 응답의 전체 HTML을 적용하기
    containerEl.innerHTML = htmlText;
```

```
    // 응답에서 body 요소만 적용하기
    // const parser = new DOMParser();
    // const document = parser.parseFromString(html, "text/html");
    // containerEl.innerHTML = document.body.innerHTML;
```

```
    // show 클래스 적용하여, 투명도 애니메이션 적용
    containerEl.classList.add('show');
  } catch (error) {
    console.error("Error:", error);
  }
});
</script>
```

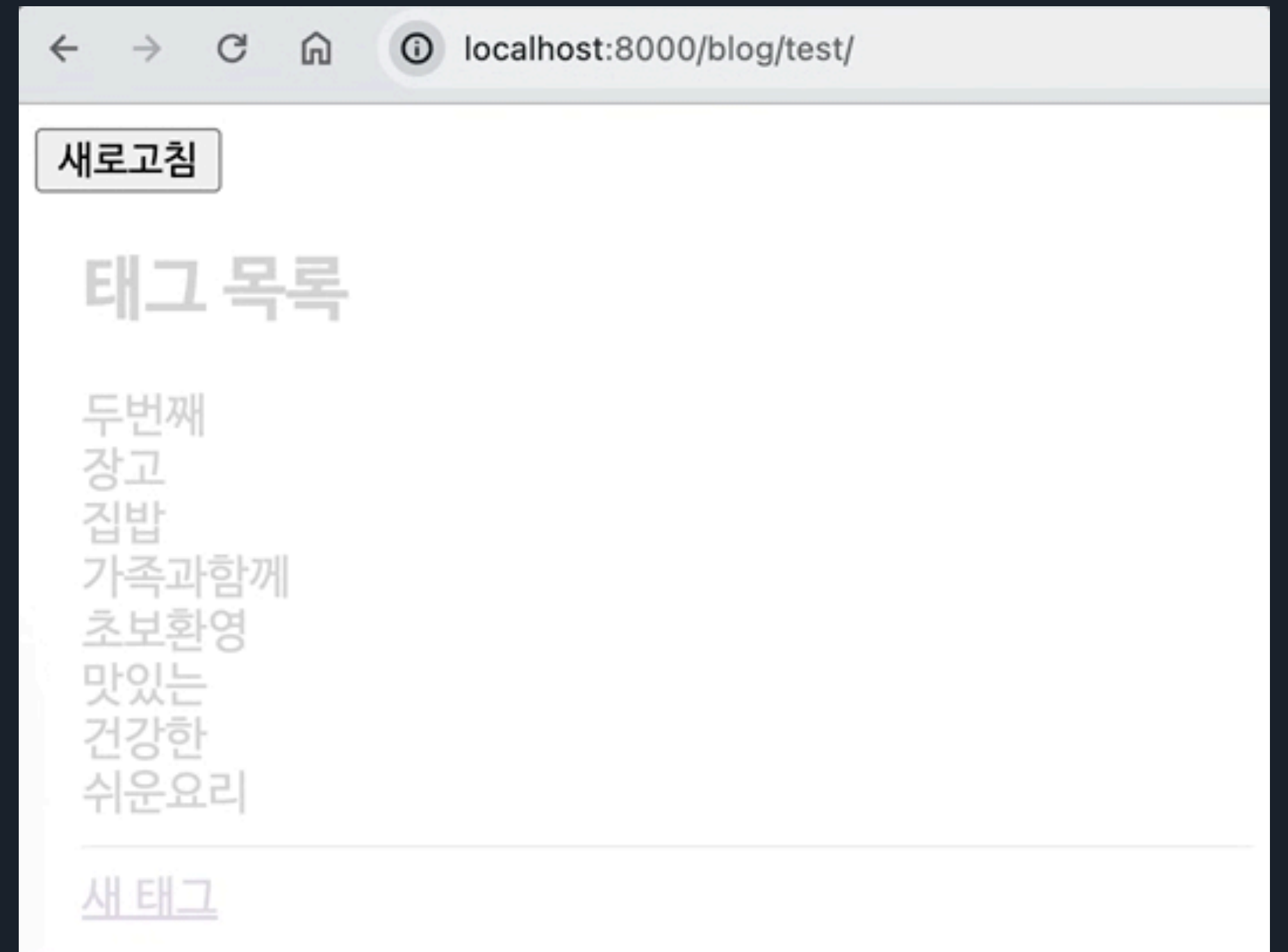
```
</body>
</html>
```

```
# blog/views.py
```

```
def test(request):
    return render(request, "blog/test.html")
```

```
# blog/urls.py
```

```
path("test/", views.test, name="test"),
```



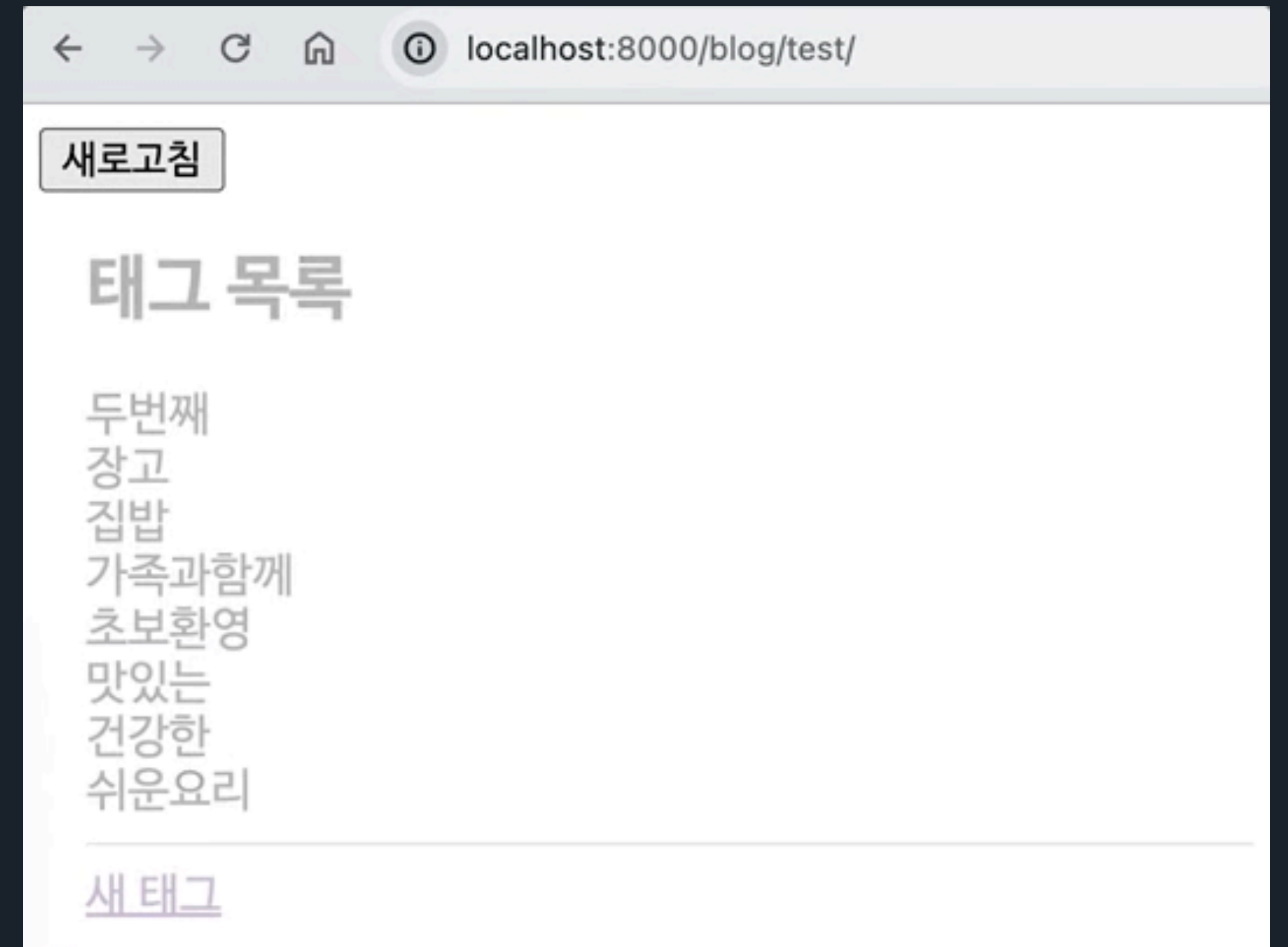
실습 소스코드 : <https://gist.github.com/allieus/0d214828663b66575299a405fcfc7e68>

HTMX CDN 페이지 : <https://cdnjs.com/libraries/htmx>

click to load 예시

HTMX 버전

```
{# blog/templates/blog/test.html #}  
  
<!doctype html>  
<html lang="ko">  
<head>  
<meta charset="UTF-8" />  
<title>htmx</title>  
  
{# https://cdnjs.com/libraries/htmx #}  
<script src="https://unpkg.com/htmx.org@1.9.10/dist/htmx.min.js"></script>  
  
</head>  
<body>  
  
<button hx-get="/blog/tags/"  
        hx-trigger="click"  
        hx-target="#tag-list-container"  
        hx-swap="innerHTML transition:true">  
    새로고침  
</button>  
  
<div id="tag-list-container"></div>  
  
</body>  
</html>
```



https://gist.github.com/allieus/0d214828663b66575299a405fcfc7e68#file-click_to_load_2_htmx-html

htmx 라이브러리

45KB 작은 용량. HTML 마크업만으로 서버로 요청을 보내고 웹페이지 DOM 갱신 지원

```
<script src="https://unpkg.com/htmx.org"></script> <!-- 최신버전 CDN -->
<!-- <script src="https://unpkg.com/htmx.org@1.9.10/dist/htmx.min.js"></script> -->
```

```
<button hx-get="/blog/tags/"
        hx-trigger="click"
        hx-target="#tag-list-container"
        hx-swap="innerHTML transition:true">
```

새로고침

```
</button>
```

hx-get

- 지정 주소로 GET 요청을 보냅니다. hx-post, hx-put, hx-patch, hx-delete 등이 지원됩니다.
- 공식문서 : <https://htmx.org/attributes/hx-get/>

hx-trigger

- HTTP 요청을 실행하는 조건
- changed (값 변경), click (클릭), keyup (키 입력), revealed (화면에 보여질 때), load (로딩 시), mouseenter (마우스가 올라갔을 때)
- once (1회만 실행), delay:딜레이시간, from:CSS셀렉터 (지정 요소의 이벤트를 처리), every 주기 (대시보드에 적합) 등
- 공식문서 : <https://htmx.org/attributes/hx-trigger/>

hx-target

- HTTP 응답을 반영할 대상 요소
- this (hx-target이 지정된 요소), <CSS셀렉터>, closest <CSS셀렉터> (CSS셀렉터와 일치하는 가장 가까운 조상 요소), find <CSS셀렉터>, next, next <CSS셀렉터>, previous, previous <CSS셀렉터>
- 공식문서 : <https://htmx.org/attributes/hx-target/>

hx-swap

- HTTP 응답을 대상 요소에 적용하는 방법
- innerHTML (디폴트, 타겟 내부 대체), outerHTML (타겟 전체 대체), beforebegin (요소 이전에 추가), afterbegin (첫 자식 이전에 추가), beforeend (마지막 자식 다음에 추가), afterend (요소 이후에 추가), delete (응답과 상관없이 타겟 삭제), none
- 공식문서 : <https://htmx.org/attributes/hx-swap/>

hx-swap 예시

```
<button hx-get="/blog/tags/?page=2"
        hx-trigger="click"
        hx-target="#tag-list-container"
        hx-swap="innerHTML">
    2페이지 로딩
</button>
```



{# 2페이지 요청에 대한 응답 HTML #}

```
<div>태그 #4</div>
<div>태그 #5</div>
<div>태그 #6</div>
```

```
# htmx 요청 여부에 따른 템플릿 변경
is_htmx: bool = request.META.get("HTTP_HX_REQUEST") == "true"
if is_htmx:
    template_name = "blog/_tag_list.html" # 레이아웃없이 콘텐츠만 응답하기
else:
    template_name = "blog/tag_list.html" # 전체 레이아웃 응답
```

```
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
</div>
```

{# hx-swap="innerHTML" #}

```
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
    <div>태그 #4</div>
    <div>태그 #5</div>
    <div>태그 #6</div>
</div>
```

{# hx-swap="beforebegin" #}

```
<div>태그 #4</div>
<div>태그 #5</div>
<div>태그 #6</div>
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
</div>
```

{# hx-swap="afterbegin" #}

```
<div id="tag-list-container">
    <div>태그 #4</div>
    <div>태그 #5</div>
    <div>태그 #6</div>
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
</div>
```

{# hx-swap="delete" #}

```
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
</div>
```

{# hx-swap="outerHTML" #}

```
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
</div>

<div>태그 #4</div>
<div>태그 #5</div>
<div>태그 #6</div>
```

{# hx-swap="beforeend" : 현 구성에 적합 #}

```
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
    <div>태그 #4</div>
    <div>태그 #5</div>
    <div>태그 #6</div>
</div>
```

{# hx-swap="afterend" #}

```
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
</div>
<div>태그 #4</div>
<div>태그 #5</div>
<div>태그 #6</div>
```

{# hx-swap="none" #}

```
<div id="tag-list-container">
    <div>태그 #1</div>
    <div>태그 #2</div>
    <div>태그 #3</div>
</div>
```


웹페이지 새로고침없이,
목록만 새로고침

웹페이지 새로고침

```
{# blog/templates/blog/tag_list.html #}

{% extends "blog/base.html" %}

{% block title %}태그 목록{% endblock %}

{% block content %}
    <h2>태그 목록</h2>

    <a href="">페이지 새로고침</a>
    <button onclick="window.location.reload();">페이지 새로고침</button>

    {% include "blog/_tag_list.html" %}
    <hr/>
    <a href="{% url 'blog:tag_new' %}">새 태그</a>
{% endblock %}
```



컨텐츠만 새로고침

```
{# blog/templates/blog/tag_list.html #}

생략

<h2>태그 목록</h2>
<button class="btn btn-primary my-3"
        hx-get="{% url 'blog:tag_list' %}"
        hx-target="#tag-list-container"
        hx-swap="innerHTML transition:true">
    새로고침
</button>
<div id="tag-list-container">
    {% include "blog/_tag_list.html" %}
</div>
```



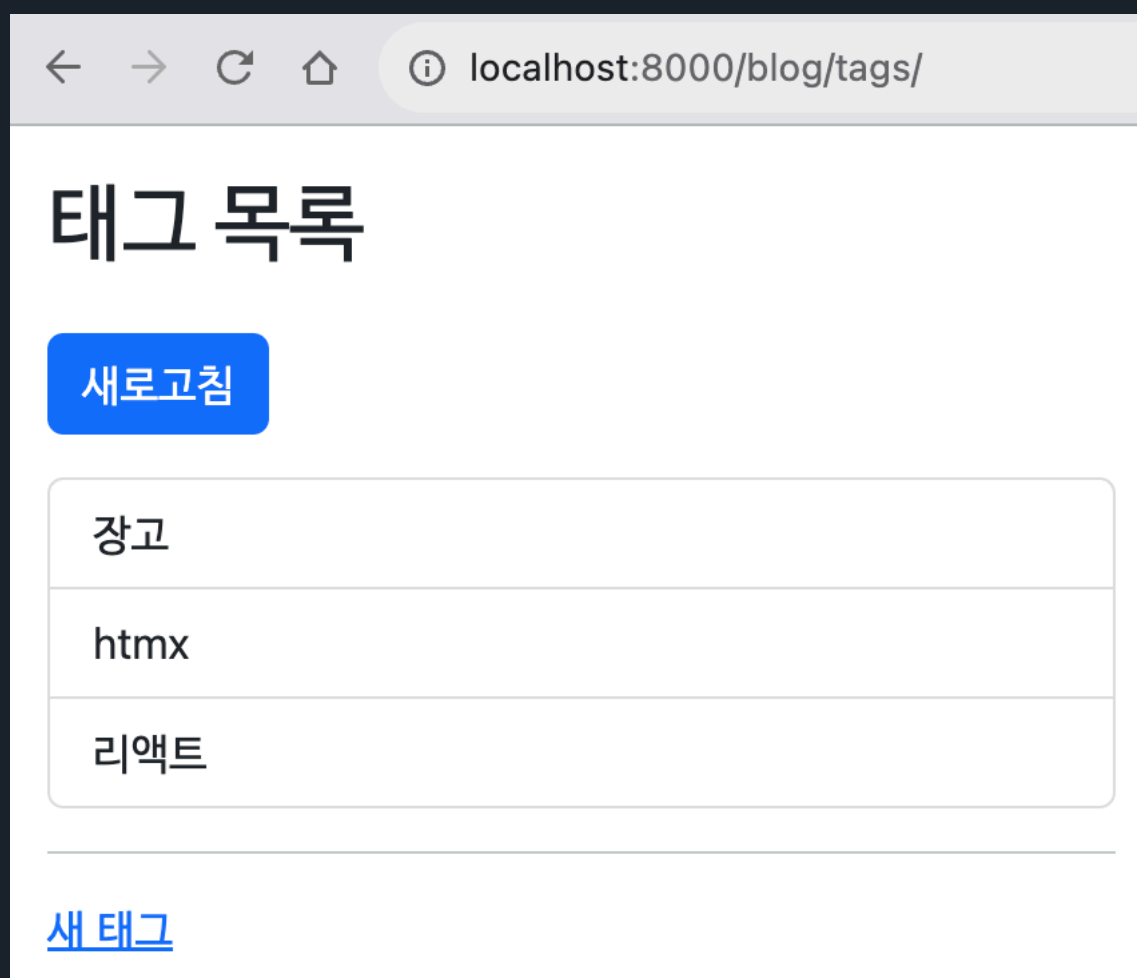
```
# blog/views.py

def tag_list(request):
    tag_qs = Tag.objects.all()

    query = request.GET.get("query", "")
    if query:
        tag_qs = tag_qs.filter(name__icontains=query)

    # htmx 요청 여부에 따른 템플릿 변경
    is_htmx: bool = request.META.get("HTTP_HX_REQUEST") == "true"
    if is_htmx:
        template_name = "blog/_tag_list.html" # 레이아웃없이 컨텐츠만 응답하기
    else:
        template_name = "blog/tag_list.html"  # 전체 레이아웃 응답

    return render(request, template_name, {
        "tag_list": tag_qs,
    })
```



```
# blog/views.py

def tag_list(request):
    tag_qs = Tag.objects.all()

    query = request.GET.get("query", "")
    if query:
        tag_qs = tag_qs.filter(name__icontains=query)

    # htmx 요청 여부에 따른 템플릿 변경
    if request.htmx: # django-htmx 라이브러리 활용
        template_name = "blog/_tag_list.html" # 컨텐츠 응답
    else:
        template_name = "blog/tag_list.html"  # 전체 페이지 응답

    return render(request, template_name, {
        "tag_list": tag_qs,
    })
```

django-htmx 라이브러리

- htmx 요청 헤더 식별을 도와주는 `request.htmx` 속성 (`HtmxDetails` 타입) 지원
- htmx 응답 헤더 생성을 도와주는 다양한 `HttpResponse` 클래스 지원
- 설치
 - `python -m pip install "django-htmx~=1.17.0"`
 - `settings.INSTALLED_APPS`에 `"django_htmx"` 추가
 - `settings.MIDDLEWARE`에 `"django_htmx.middleware.HtmxMiddleware"` 추가 => 요청 헤더를 분석해서 `request.htmx` 속성으로 제공
 - htmx 자바스크립트 라이브러리 추가
 - cdn 버전 : <https://cdnjs.com/libraries/htmx> 페이지에서 URL 참고 (개발 시에 보다 빠른 적용)
 - 다운로드 : <https://unpkg.com/browse/htmx.org@1.9.10/dist/> 페이지에서 다운로드 후에 `static` 경로에 추가 (실서비스에 적합)

아래 조건문들은 동일합니다.

```
if request.htmx: # django-htmx 활용
```

```
if request.headers.get("HX-Request") == "true":
```

```
if request.META.get("HTTP_HX_REQUEST") == "true":
```

```
def __bool__(self) -> bool:
    return self._get_header_value("HX-Request") == "true"
```

https://github.com/adamchainz/django-htmx/blob/main/src/django_htmx/middleware.py#L62

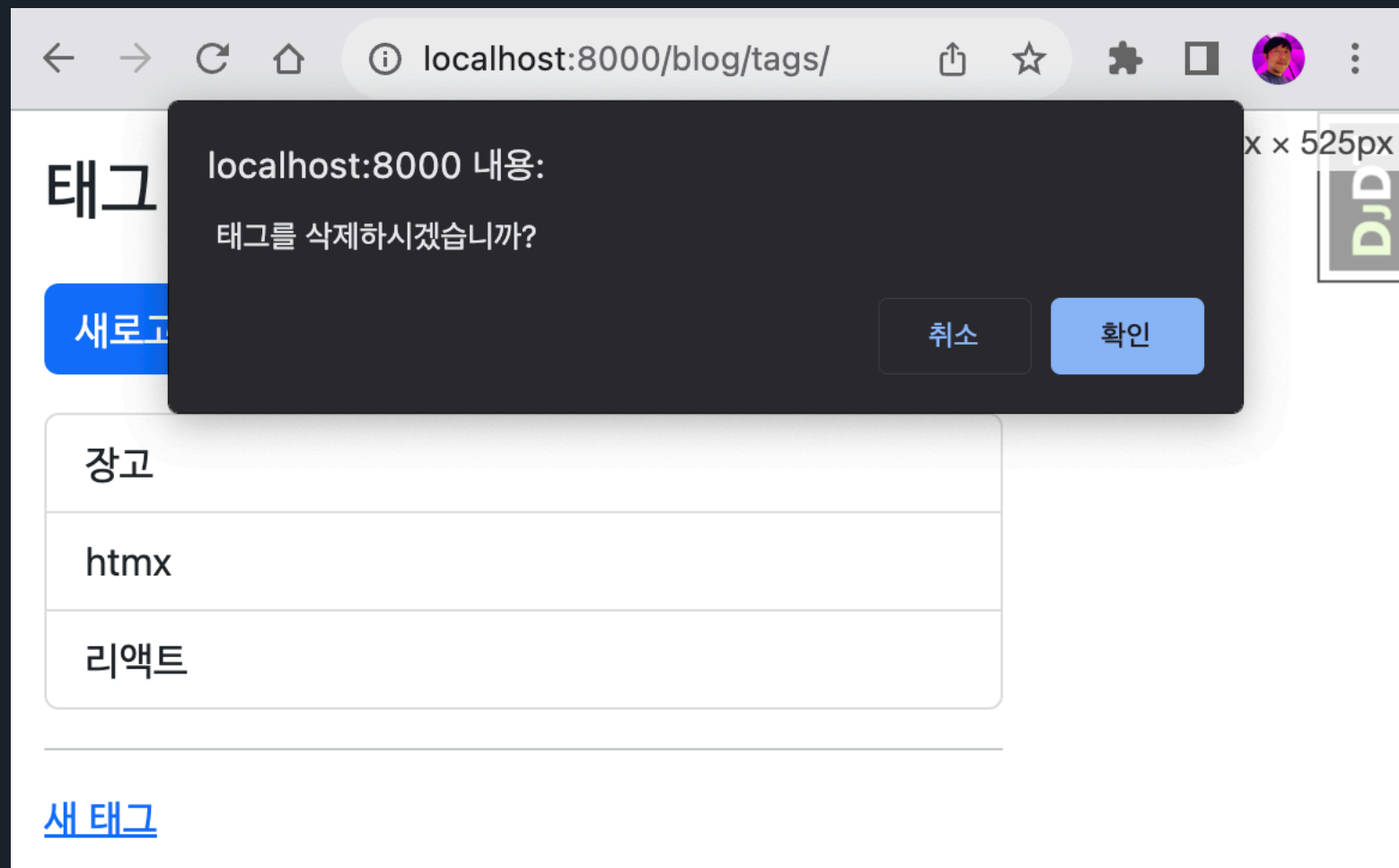
django-htmx 설치문서 : <https://django-htmx.readthedocs.io/en/latest/installation.html>

django-htmx 소스코드 : https://github.com/adamchainz/django-htmx/blob/main/src/django_htmx/middleware.py#L50

django-htmx 응답 클래스 : https://github.com/adamchainz/django-htmx/blob/main/src/django_htmx/http.py

특정 태그만 삭제 요청

태그 클릭하여, 삭제 요청



```
{# blog/templates/blog/_tag_list.html #}

<div class="list-group">
  {% for tag in tag_list %}
    <div class="list-group-item"
      style="cursor: pointer;"
      hx-trigger="click"
      hx-confirm="{{ tag.name }} 태그를 삭제하시겠습니까?"
      hx-delete="{{ url 'blog:tag_delete' tag.pk }}"
      hx-swap="delete">
      {{ tag.name }}
    </div>
  {% endfor %}
</div>
```

```
{# blog/templates/blog/base.html #}
```

```
{# htmx 요청 시에 X-CSRFToken 헤더 자동 전송 #}
<body hx-headers='{ "X-CSRFToken": "{{ csrf_token }}" }'>
```

```
# blog/views.py
```

```
@require_http_methods(["DELETE"])
def tag_delete(request, pk):
    get_object_or_404(Tag, pk=pk).delete()
    return HttpResponse("") # hx-swap="delete"에 의해 요소가 삭제될 것이기에 빈 응답을 합니다.
```

```
# blog/urls.py
```

```
urlpatterns += [
    path("tags/<int:pk>/delete/", views.tag_delete, name="tag_delete"),
]
```

새 태그 생성 요청

← → ↺ ⌵ ⓘ localhost:8000/blog/tags/

태그 목록

태그를 저장했습니다. ✕

Name

새로고침

장고

htmx

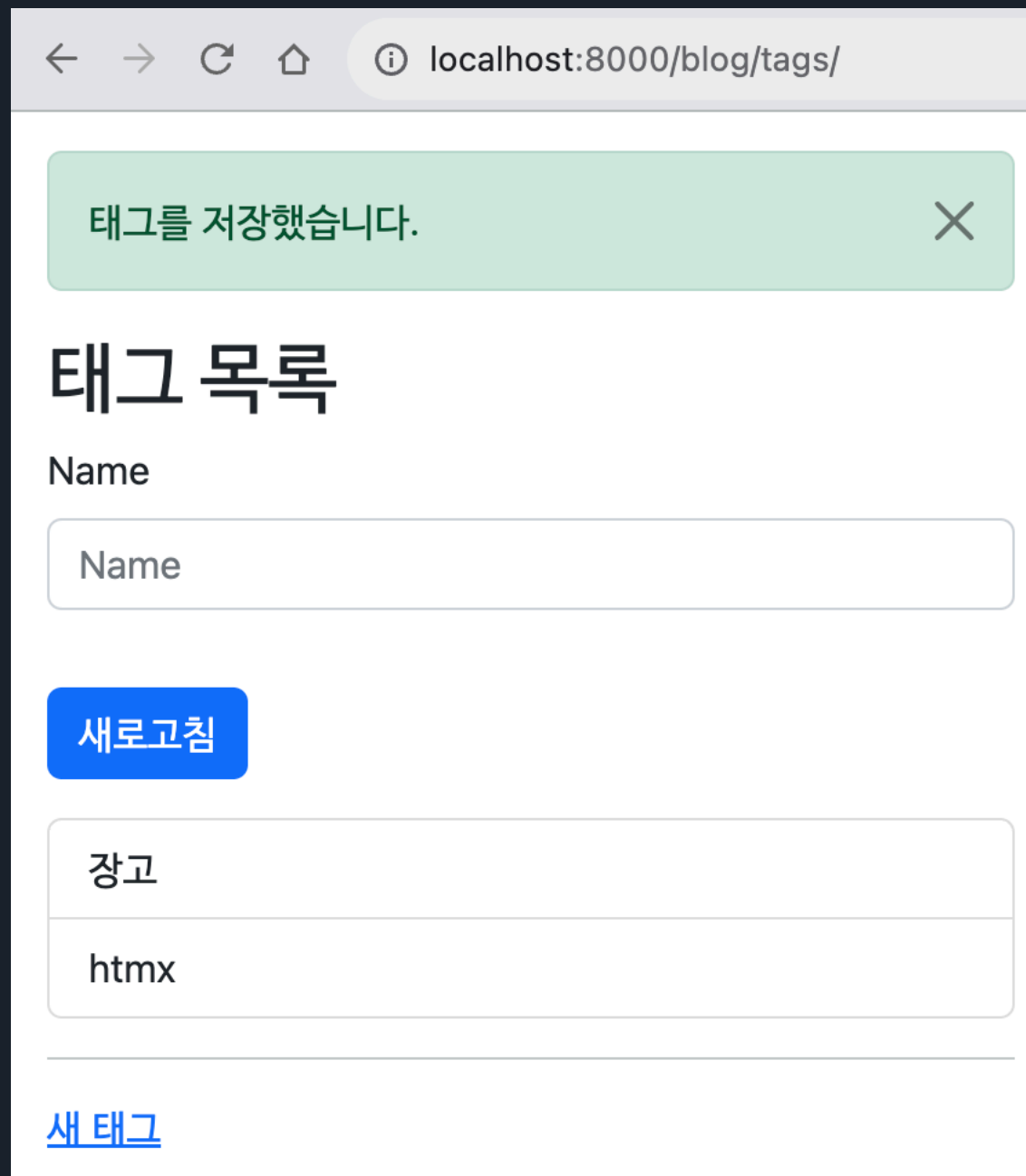
리액트

자바스크립트

[새 태그](#)

태그 목록에 태그 생성 폼 생성

htmx를 통해 폼은 로딩했지만, 폼 전송은 htmx가 아니라 form 전송을 통한 처리



```
{# blog/templates/blog/tag_list.html #}

{% extends "blog/base.html" %}

{% block title %}태그 목록{% endblock %}

{% block content %}
<h2>태그 목록</h2>

{# #1) 장고 기본 기능 만으로 태그 생성 폼을 노출시키기 #}
{#   tag_list 뷰에서 TagForm 인스턴스 생성도 필요 #}
<form action="{% url 'blog:tag_new' %}"
      method="post" novalidate>
    {% csrf_token %}
    {% bootstrap_form tag_form %}
</form>

{# #2) 웹페이지가 로딩되면 태그 생성폼도 로딩 #}
<div hx-get="{% url 'blog:tag_new' %}"
      hx-trigger="load">
</div>

<button class="btn btn-primary my-3"
        hx-get="{% url 'blog:tag_list' %}"
        hx-target="#tag-list-container"
        hx-swap="innerHTML transition:true">
    새로고침
</button>

<div id="tag-list-container">
    {% include "blog/_tag_list.html" %}
</div>

<hr />
<a href="{% url 'blog:tag_new' %}">새 태그</a>
{% endblock %}
```

```
# blog/views.py

def tag_new(request):
    if request.method == "GET":
        form = TagForm()
    else:
        form = TagForm(data=request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "태그를 저장했습니다.")
            # htmx 요청여부에 상관없이 redirect 응답
            return redirect("tag_list")

# htmx 요청 여부에 따른 템플릿 변경
if request.htmx:
    template_name = "blog/_tag_form.html"
else:
    template_name = "blog/tag_form.html"

return render(request, template_name, {
    "form": form,
})

{# blog/templates/blog/_tag_form.html #}
{% load django_bootstrap5 %}

<form action="{% url 'blog:tag_new' %}" method="post" autocomplete="off" novalidate>
    {% csrf_token %}
    {% bootstrap_form form %}

    {# 전송 버튼없이 태그 입력하고 Enter키 입력해서 Submit받겠습니다. #}
    {# <input type="submit" class="btn btn-primary"/> #}

</form>
```

htmx를 활용한 form 처리

페이지 전환없이 처리되었으나, 별도로 목록 새로고침이 필요

← → ↻ ⬆ ⓘ localhost:8000/blog/tags/

태그 목록

태그를 저장했습니다. ✕

Name

새로고침

| |
|------|
| 장고 |
| htmx |

[새 태그](#)

```
{% blog/templates/blog/_tag_form.html %}
{% load django_bootstrap5 %}

{% htmx 응답에서 messages를 보여주기 위함 %}
{% bootstrap_messages %}

{% comment %}<form action="{% url 'blog:tag_new' %}" method="post" autocomplete="off" novalidate>{% endcomment %}
<form hx-post="{% url 'blog:tag_new' %}" hx-trigger="submit once" hx-swap="outerHTML" autocomplete="off" novalidate>
  {% csrf_token %}
  {% bootstrap_form form %}
  <input type="submit" class="btn btn-primary" />
</form>
```

```
# blog/views.py

def tag_new(request):
    if request.method == "GET":
        form = TagForm()
    else:
        form = TagForm(data=request.POST)
        if form.is_valid():
            form.save()
            messages.success(request, "태그를 저장했습니다.")
            if request.htmx:
                form = TagForm() # 빈 폼 보여주기
                return render(request, "blog/_tag_form.html", {
                    "form": form,
                })
            else:
                return redirect("tag_list")

# htmx 요청 여부에 따른 템플릿 변경
if request.htmx:
    template_name = "blog/_tag_form.html"
else:
    template_name = "blog/tag_form.html"

return render(request, template_name, {
    "form": form,
})
```

form 응답에서 새로고침 이벤트 발생

HX-Trigger 헤더를 활용해서, 클라이언트 단에 이벤트를 발생시킬 수 있습니다.

← → ↺ ⬆ ⓘ localhost:8000/blog/tags/

태그 목록

태그를 저장했습니다. ✕

Name

Name

새로고침

장고

htmx

리액트

자바스크립트

[새 태그](#)

```
# blog/views.py

from django_htmx.http import trigger_client_event

def tag_new(request):
    if request.method == "GET":
        form = TagForm()
    else:
        form = TagForm(data=request.POST)
        if form.is_valid():
            form.save()
            if request.htmx:
                form = TagForm() # 빈 폼 보여주기
                response = render(request, "blog/_tag_form.html", {
                    "form": form,
                })

                # 페이지 새로고침 요청
                # - True로 지정하시면 "True"로 전달됨에 유의
                # response["HX-Refresh"] = "true"
                # django-htmx를 활용한 HX-Refresh 헤더 설정.
                # response = HttpResponseRedirectRefresh()

                # htmx 웹페이지의 body 요소에 커스텀 이벤트 refres-tag-list 발생.
                # response["HX-Trigger"] = "refresh-tag-list"
                # django-htmx를 활용한 HX-Trigger 헤더 설정.
                response = trigger_client_event(response, "refresh-tag-list")

            return response
        else:
            return redirect("tag_list")

# 생략
```

```
{# blog/templates/blog/tag_list.html #}

{% extends "blog/base.html" %}

{% block title %}태그 목록{% endblock %}

{% block content %}
    <h2>태그 목록</h2>

    <div hx-get="{% url 'blog:tag_new' %}"
        hx-trigger="load">

    </div>

    <button class="btn btn-primary my-3"
        hx-get="{% url 'blog:tag_list' %}"
        hx-target="#tag-list-container"
        hx-trigger="click, refresh-tag-list from:body"
        hx-swap="innerHTML transition:true">
        새로고침
    </button>

    <div id="tag-list-container">
        {% include "blog/_tag_list.html" %}
    </div>

    <hr />
    <a href="{% url 'blog:tag_new' %}">새 태그</a>
{% endblock %}
```

HTMX 지원 헤더 : https://htmx.org/reference/#response_headers

HttpResponseClientRefresh 소스코드 : https://github.com/adamchainz/django-htmx/blob/main/src/django_htmx/http.py#L37

trigger_client_event 소스코드 : https://github.com/adamchainz/django-htmx/blob/main/src/django_htmx/http.py#L106

브라우저 캐시 무효화

HTMX에서 브라우저 캐시 무효화 시키기

```
<button class="btn btn-primary my-3"
  hx-get="요청주소"
  hx-target="#tag-list-container"
  hx-swap="innerHTML transition:true"
  hx-vals="js:{_:new Date().getTime()}"
>새로고침
</button>
```

```
<script>
document.body.addEventListener('htmx:configRequest', function (event) {
  const htmxElement = event.detail.elt;
  const isGetRequest = htmxElement.hasAttribute('hx-get');
  if ( isGetRequest && htmxElement.hasAttribute('hx-get-with-timestamp') ) {
    const paramName = htmxElement.getAttribute('hx-get-with-timestamp') || '_';
    event.detail.parameters[paramName] = new Date().getTime();
  }
});
</script>
```

무한 스크롤

클래스 기반 뷰로 변경하고, 페이징 처리

← → ↺ ⬆ ⓘ localhost:8000/blog/tags/

태그 목록

Name

새로고침

| |
|--------|
| 장고 |
| htmx |
| 리액트 |
| 자바스크립트 |
| 파이썬 |

« 1 »

[새 태그](#)

```
# blog/views.py

from django.views.generic import ListView

class TagListView(ListView):
    queryset = Tag.objects.all()
    paginate_by = 10 # 1페이지에 10개씩 보여주기

    def get_queryset(self):
        qs = super().get_queryset()
        query = self.request.GET.get("query", "")
        if query:
            qs = qs.filter(name__icontains=query)
        return qs

    def get_template_names(self):
        if self.request.htmx:
            return ["blog/_tag_list.html"]
        else:
            return ["blog/tag_list.html"]

tag_list = TagListView.as_view()
```

```
{# blog/templates/blog/tag_list.html #}
{% extends "blog/base.html" %}

{% load django_bootstrap5 %}

{% block title %}태그 목록{% endblock %}

{% block content %}
    <h2>태그 목록</h2>

    <div hx-get="{% url 'blog:tag_new' %}"
        hx-trigger="load">

        <button class="btn btn-primary my-3"
            hx-get="{% url 'blog:tag_list' %}?page={{ request.GET.page|default:1 }}"
            hx-target="#tag-list-container"
            hx-trigger="click, refresh-tag-list from:body"
            hx-swap="innerHTML transition:true">
            새로고침
        </button>

        <div id="tag-list-container">
            {% include "blog/_tag_list.html" %}
        </div>

        <div class="my-3">
            {% bootstrap_pagination page_obj %}
        </div>

    <hr />
    <a href="{% url 'blog:tag_new' %}">새 태그</a>
{% endblock %}
```

태그 개수 늘리기

장고 커스텀 관리 명령으로 구현해봅시다.

```
# blog/management/commands/load_blog_tags.py

import requests # pip install requests
from django.core.management import BaseCommand

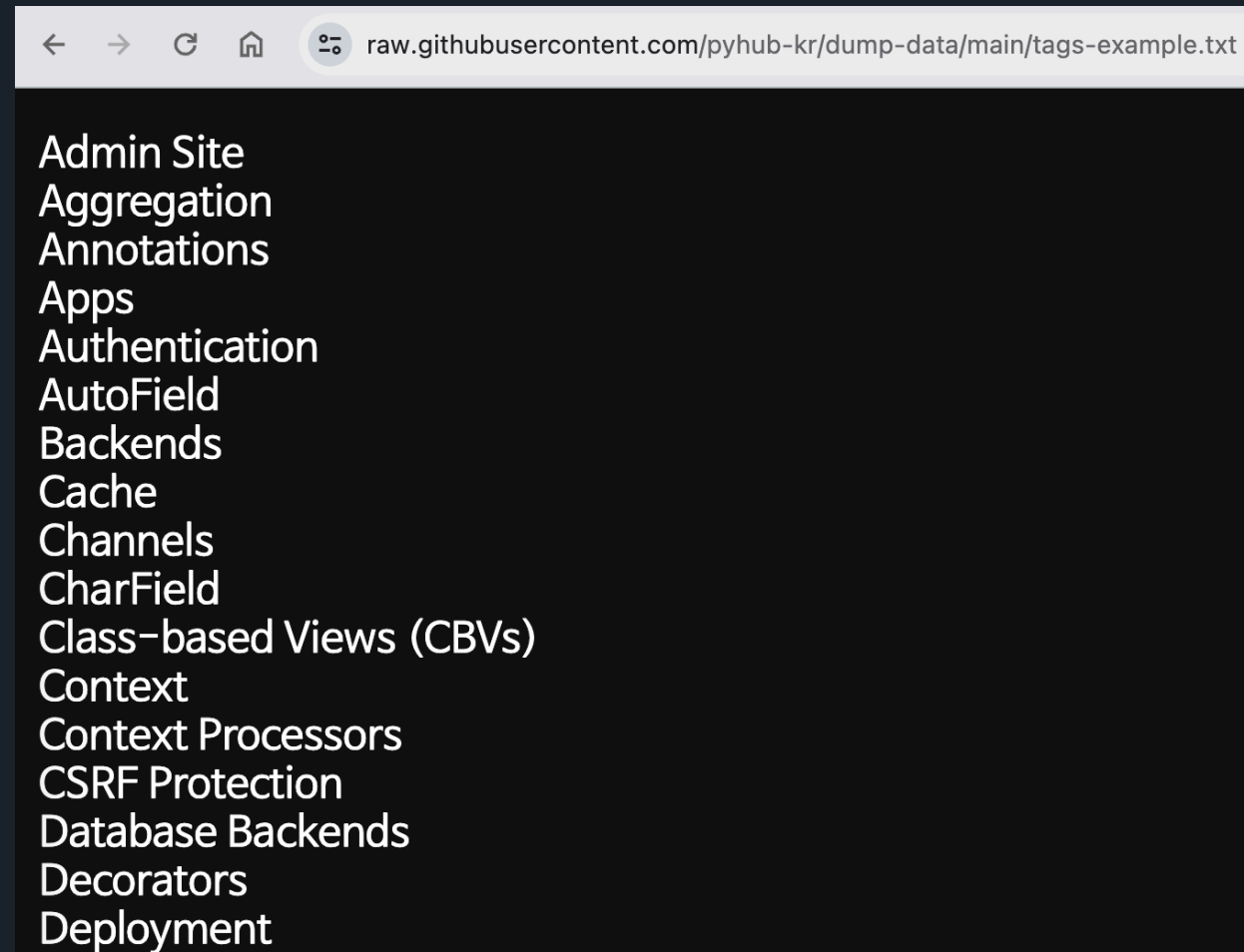
from blog.models import Tag

class Command(BaseCommand):
    def handle(self, *args, **options):
        txt_url = "https://raw.githubusercontent.com/pyhub-kr/dump-data/main/tags-example.txt"

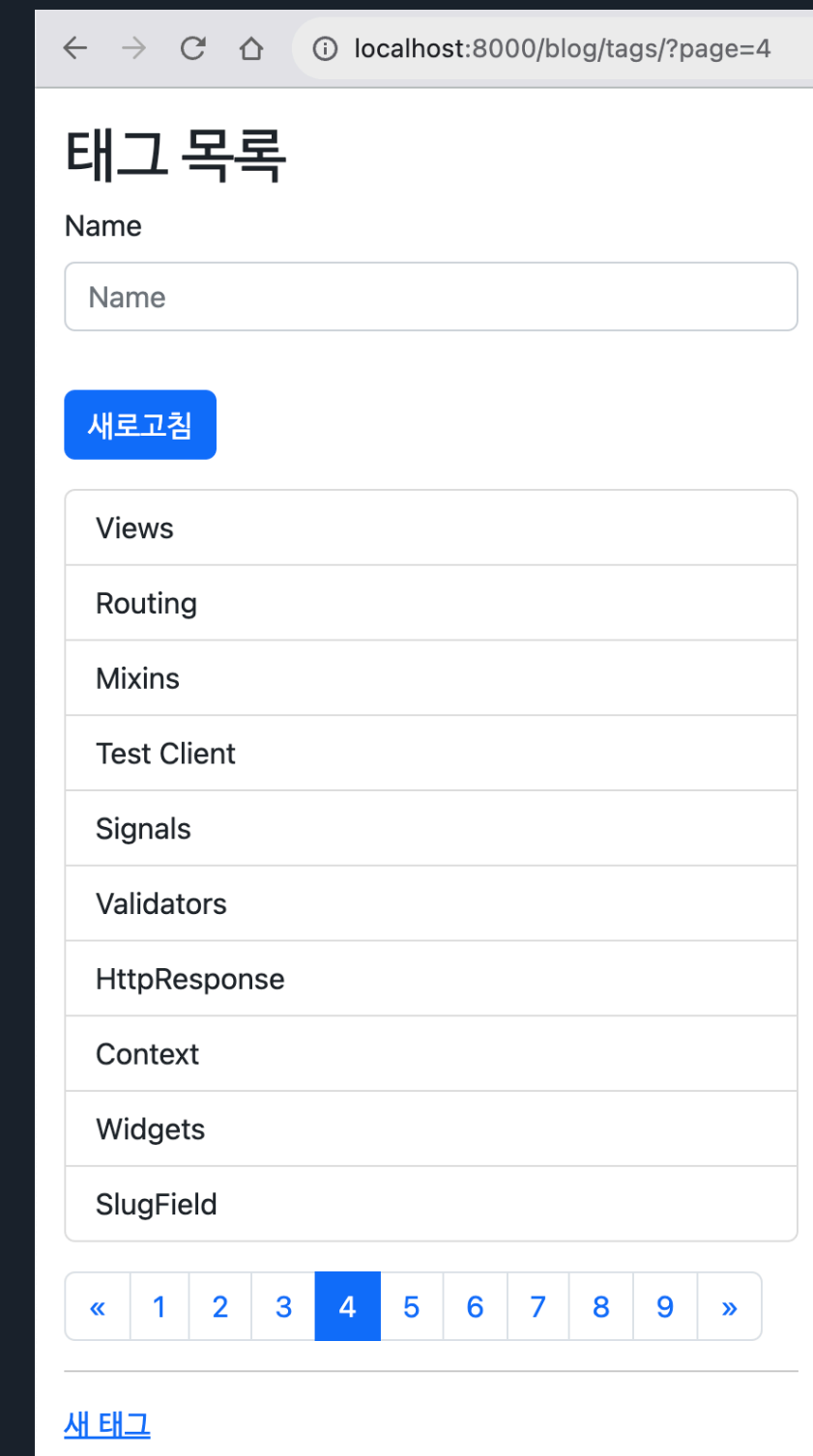
        txt = requests.get(txt_url).text
        tag_set = {line.strip() for line in txt.splitlines()}

        existed_tag_set = set(Tag.objects.all().values_list("name", flat=True))
        making_tag_set = tag_set - existed_tag_set

        tag_list = [Tag(name=tag_name) for tag_name in making_tag_set]
        created_tag_list = Tag.objects.bulk_create(tag_list)
        self.stdout.write(f"{len(created_tag_list)} tags created")
```

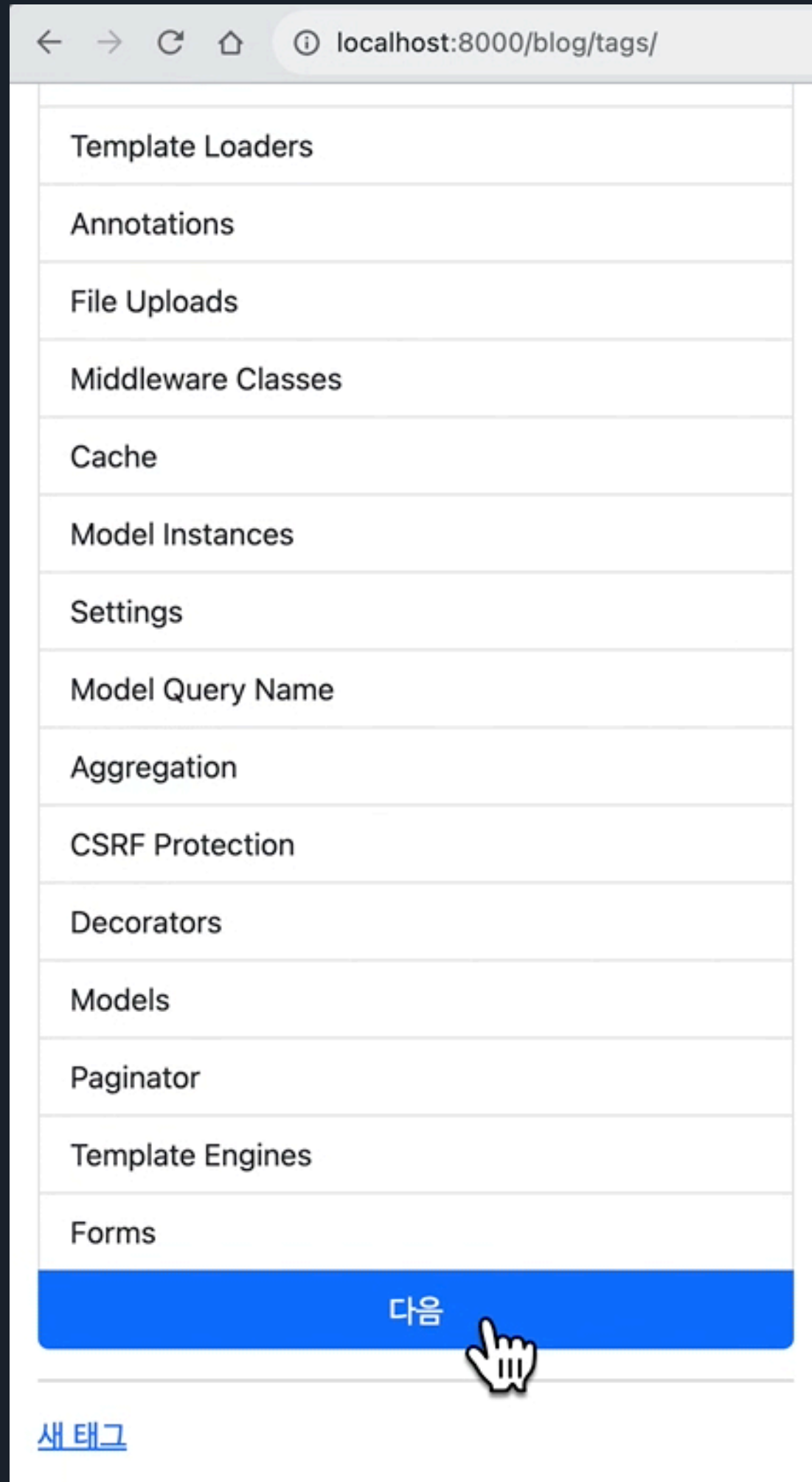


```
> python manage.py load_blog_tags
78 tags created
```



Click to load 패턴

다음 버튼을 클릭하면, 다음 페이지를 뒤에 추가합니다.



```
{# blog/templates/blog/tag_list.html #}
{% extends "blog/base.html" %}
```

```
{% load django_bootstrap5 %}
```

```
{% block title %}태그 목록{% endblock %}
```

```
{% block content %}
<h2>태그 목록</h2>
```

```

<div hx-get="{% url "blog:tag_new" %}"
      hx-trigger="load">
</div>
```

```

<button class="btn btn-primary my-3"
        hx-get="{% url 'blog:tag_list' %}?page={{ request.GET.page|default:1 }}"
        hx-target="#tag-list-container .list-group"
        hx-trigger="click, refresh-tag-list from:body"
        hx-swap="innerHTML transition:true">
    새로고침
</button>
```

```

<div id="tag-list-container">
    {# _tag_list.html에서는 자식 요소들만 다루도록, #}
    {# .list-group을 부모 템플릿으로 옮깁니다. #}
    <div class="list-group">
        {% include "blog/_tag_list.html" %}
    </div>
</div>
```

```

{% comment %}
<div class="my-3">
    {% bootstrap_pagination page_obj %}
</div>
{% endcomment %}
```

```

<hr />
<a href="{% url 'blog:tag_new' %}">새 태그</a>
{% endblock %}
```

```
{# blog/templates/blog/_tag_list.html #}
```

```
{# <div class="list-group"> #}
```

```

    {% for tag in tag_list %}
        <div class="list-group-item"
            style="cursor: pointer;"
            hx-delete="{% url 'blog:tag_delete' tag.pk %}"
            hx-trigger="click"
            hx-confirm="{{ tag.name }} 태그를 삭제하시겠습니까?"
            hx-swap="delete">
            {{ tag.name }}
        </div>
```

```

    {# 현 페이지의 마지막 태그에서 다음 페이지를 로딩토록 버튼을 추가 #}
```

```

    {# - 이 버튼 요소가 다음 페이지 내용으로 변경될 것입니다. #}
```

```

    {% if forloop.last and page_obj.has_next %}
        <div class="list-group-item list-group-item-action active text-center"
            style="cursor: pointer;"
            hx-get="{% url 'blog:tag_list' %}?page={{ page_obj.next_page_number }}"
            hx-trigger="click once"
            hx-swap="outerHTML transition:true">
            다음
        </div>
    {% endif %}
    {% endfor %}
```

```
{# </div> #}
```

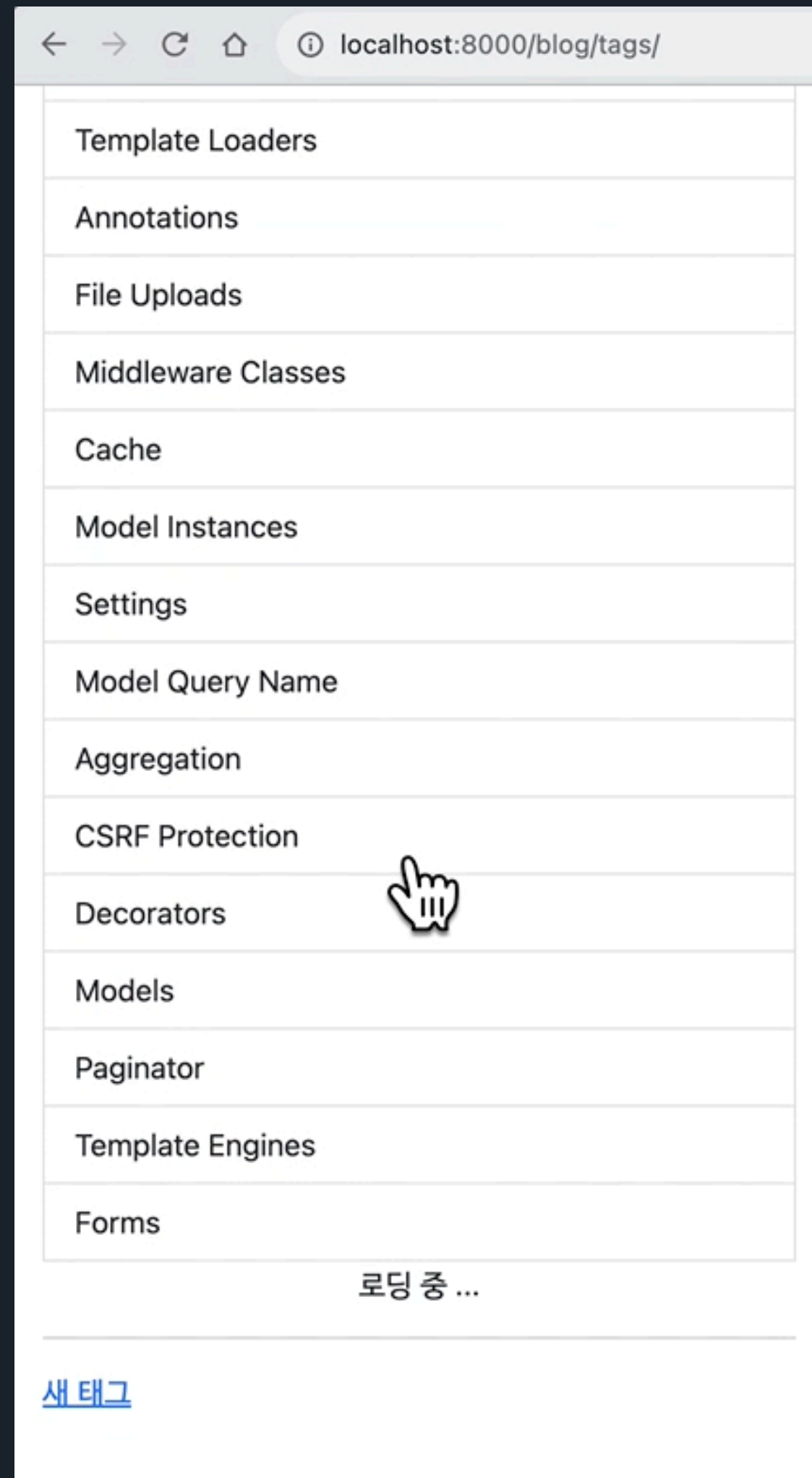
실습 소스코드 : <https://gist.github.com/allieus/6624f02ba6b2e2427a435246beb49af8>

HTMX, click to load 예시 : <https://htmx.org/examples/click-to-load/>

© All rights reserved by 파이썬 사랑방

무한 스크롤

마지막 항목이 보여질 때 자동으로 다음 페이지를 뒤에 추가합니다.



```
{# blog/templates/blog/_tag_list.html #}

{% for tag in tag_list %}
    <div class="list-group-item"
        style="cursor: pointer;"
        hx-delete="{% url 'blog:tag_delete' tag.pk %}"
        hx-trigger="click"
        hx-confirm="{% tag.name %} 태그를 삭제하시겠습니까?"
        hx-swap="delete">
        {% tag.name %}
    </div>

    {# 화면에 노출되면, 즉시 다음 페이지를 로딩하여, 요소를 대체 #}
    {% if forloop.last and page_obj.has_next %}
        <div {# class="list-group-item list-group-item-action active text-center" #}
            class="text-center"
            {# style="cursor: pointer;" #}
            hx-get="{% url 'blog:tag_list' %}?page={% page_obj.next_page_number %}"
            {# hx-trigger="click once" #}
            hx-trigger="revealed once"
            hx-swap="outerHTML transition:true">
            {# 다음 #}
            로딩 중 ...
        </div>
    {% endif %}
{% endfor %}
```

태그 inplace 편집

← → ↻ 🏠 ⓘ localhost:8000/blog/tags/

Name

장고

저장 취소

htmx X

Name

리액트 수정할려다가 ...

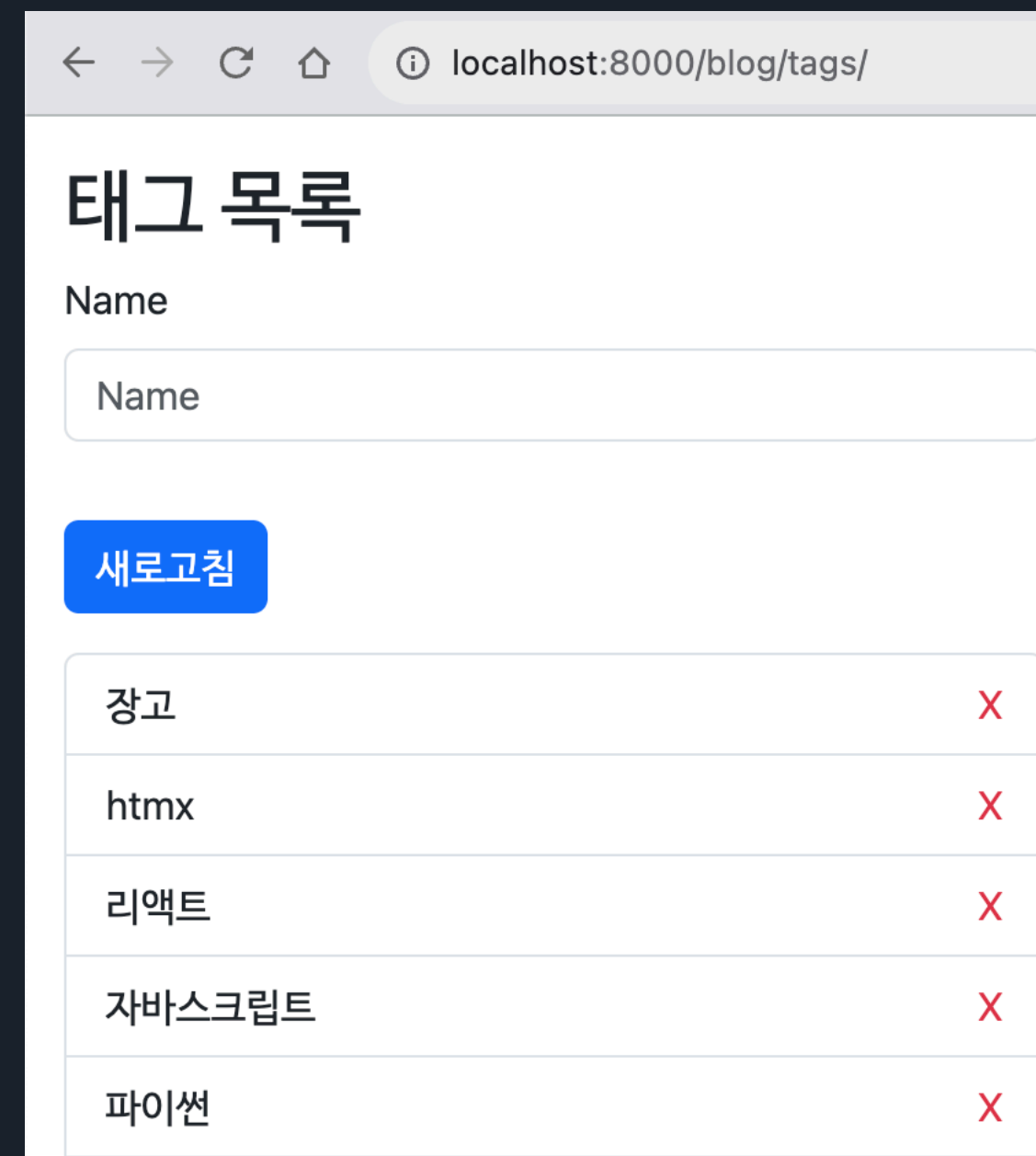
저장 취소

자바스크립트 X

파이썬 X

리스트 스타일 변경

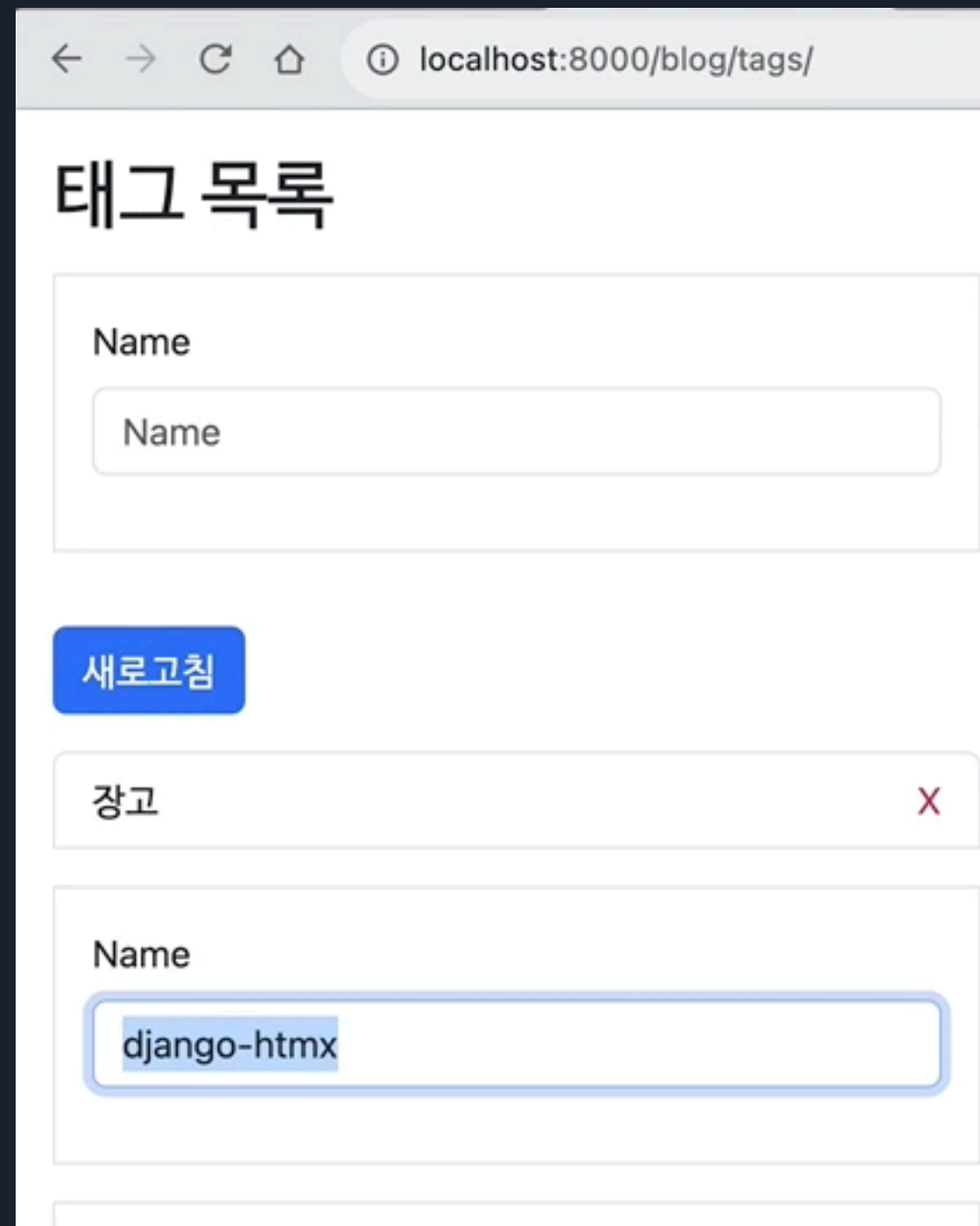
list-group-item 요소에 수정폼 전환을 적용하기 위함



```
{# blog/templates/blog/_tag_list.html #}

{% for tag in tag_list %}
  <div class="list-group-item d-flex justify-content-between align-items-center"
    {% comment %}
      style="cursor: pointer;"
      hx-delete="{% url 'blog:tag_delete' tag.pk %}"
      hx-trigger="click"
      hx-confirm="{{ tag.name }}" 태그를 삭제하시겠습니까?"
      hx-swap="delete"
    {% endcomment %}
  >
    <div>{{ tag.name }}</div>
    <span class="text-danger" style="cursor: pointer;"
      hx-delete="{% url 'blog:tag_delete' tag.pk %}"
      hx-trigger="click"
      hx-target="closest .list-group-item"
      hx-swap="delete"
      hx-confirm="{{ tag.name }}" 태그를 삭제하시겠습니까?">
      X
    </span>
  </div>
{% if forloop.last and page_obj.has_next %}
  <div class="text-center"
    hx-get="{% url 'blog:tag_list' %}?page={{ page_obj.next_page_number }}"
    hx-trigger="revealed"
    hx-swap="outerHTML transition:true">
    {# 다음 #}
    로딩 중 ...
  </div>
{% endif %}
{% endfor %}
```

inplace 편집 구현



```
{# blog/templates/blog/_tag_list.html #}

{% for tag in tag_list %}
  <div class="list-group-item d-flex justify-content-between align-items-center">
    <div style="cursor: pointer;"
      hx-get="{% url 'blog:tag_edit' tag.pk %}"
      hx-trigger="click"
      hx-target="closest .list-group-item"
      hx-swap="outerHTML">
      {{ tag.name }}
    </div>
    <span class="text-danger" style="cursor: pointer;"
      hx-delete="{% url 'blog:tag_delete' tag.pk %}"
      hx-trigger="click"
      hx-target="closest .list-group-item"
      hx-swap="delete"
      hx-confirm="{{ tag.name }} 태그를 삭제하시겠습니까?">
      X
    </span>
  </div>
  {% if forloop.last and page_obj.has_next %}
    <div class="text-center"
      hx-get="{% url 'blog:tag_list' %}?page={{ page_obj.next_page_number }}"
      hx-trigger="revealed"
      hx-swap="outerHTML transition:true">
      로딩 중 ...
    </div>
  {% endif %}
{% endfor %}
```

```
# blog/views.py

def tag_new(request, pk=None):
    instance = get_object_or_404(Tag, pk=pk) if pk else None

    if request.method == "GET":
        form = TagForm(instance=instance)
    else:
        form = TagForm(data=request.POST, instance=instance)

    # 생략
```

```
def tag_edit(request, pk):
    return tag_new(request, pk)
```

```
# blog/urls.py
```

```
urlpatterns += [
    path("tags/<int:pk>/edit/", views.tag_edit, name="tag_edit"),
]
```

```
{# blog/templates/blog/_tag_form.html #}

{% load django_bootstrap5 %}

{% bootstrap_messages %}

{# form 요청 주소와 같은 주소로 post 요청을 받습니다. #}
<form hx-post="{{ request.get_full_path }}"
      class="border my-3 p-3"
      hx-trigger="submit once"
      hx-swap="outerHTML"
      autocomplete="off" novalidate>
  {% csrf_token %}
  {% bootstrap_form form %}
</form>
```


inplace 편집 취소 구현

태그 목록

Name

Name

저장

취소

새로고침

파이썬사랑방이요

X

Name

파이썬사랑방 ♥

저장

취소

```
# blog/views.py
# 취소 시에 각 태그 list-group-item HTML으로 대체
def tag_list_item(request, pk):
    tag = get_object_or_404(Tag, pk=pk)
    return render(request, "blog/_tag_list_item.html", {"tag": tag})

# blog/urls.py
urlpatterns += [
    path("tags/item/<int:pk>/", views.tag_list_item, name="tag_list_item"),
]

{# blog/templates/blog/_tag_list_item.html #}

{# _tag_list.html 내역을 그대로 옮김 #}

<div class="list-group-item d-flex justify-content-between align-items-center">
  <div style="cursor: pointer;"
    hx-get="{% url 'blog:tag_edit' tag.pk %}"
    hx-trigger="click"
    hx-target="closest .list-group-item"
    hx-swap="outerHTML">
    {{ tag.name }}
  </div>
  <span class="text-danger"
    style="cursor: pointer;"
    hx-trigger="click"
    hx-confirm="{{ tag.name }} 태그를 삭제하시겠습니까?"
    hx-delete="{% url 'blog:tag_delete' tag.pk %}"
    hx-target="closest .list-group-item"
    hx-swap="delete">
    X
  </span>
</div>
```

```
{# blog/templates/blog/_tag_list.html #}

{% for tag in tag_list %}
  {% include "blog/_tag_list_item.html" %}
  {% if forloop.last and page_obj.has_next %}
    <div class="text-center"
      hx-get="{% url 'blog:tag_list' %}?page={{ page_obj.next_page_number }}"
      hx-trigger="revealed"
      hx-swap="outerHTML transition:true">
      로딩 중 ...
    </div>
  {% endif %}
{% endfor %}
```

```
# blog/views.py

from django.shortcuts import resolve_url

def tag_new(request, pk=None):
    if pk:
        instance = get_object_or_404(Tag, pk=pk)
        tag_list_item_url = reverse("blog:tag_list_item", args=[pk])
    else:
        instance = None
        tag_list_item_url = None

    if request.method == "GET":
        form = TagForm(instance=instance)
    else:
        form = TagForm(data=request.POST, instance=instance)
        if form.is_valid():
            instance = form.save()
            messages.success(request, "태그를 저장했습니다.")
            if request.htmx:
                # 수정 저장 후에, 태그 내역으로 응답합니다.
                if tag_list_item_url:
                    return redirect(tag_list_item_url)
                else:
                    form = TagForm()
                    response = render(request, "blog/_tag_form.html", {
                        "form": form,
                    })
                    return trigger_client_event(response, "refresh-tag-list")
            else:
                return redirect("tag_list")

    if request.htmx:
        template_name = "blog/_tag_form.html"
    else:
        template_name = "blog/tag_form.html"

    return render(request, template_name, {
        "form": form,
        # 취소 요청 시에도, 태그 내역으로 응답합니다.
        "cancel_url": tag_list_item_url,
    })

{# blog/templates/blog/_tag_form.html #}
{% load django_bootstrap5 %}
{% bootstrap_messages %}
<form hx-post="{{ request.get_full_path }}" hx-trigger="submit once" hx-swap="outerHTML"
  autocomplete="off" novalidate class="border my-3 p-3">
  {% csrf_token %}
  {% bootstrap_form form %}
  <button class="btn btn-primary btn-sm">저장</button>
  <button class="btn btn-warning btn-sm"
    hx-get="{{ cancel_url|default:request.get_full_path }}"
    hx-target="closest form"
    hx-swap="outerHTML">
    취소
  </button>

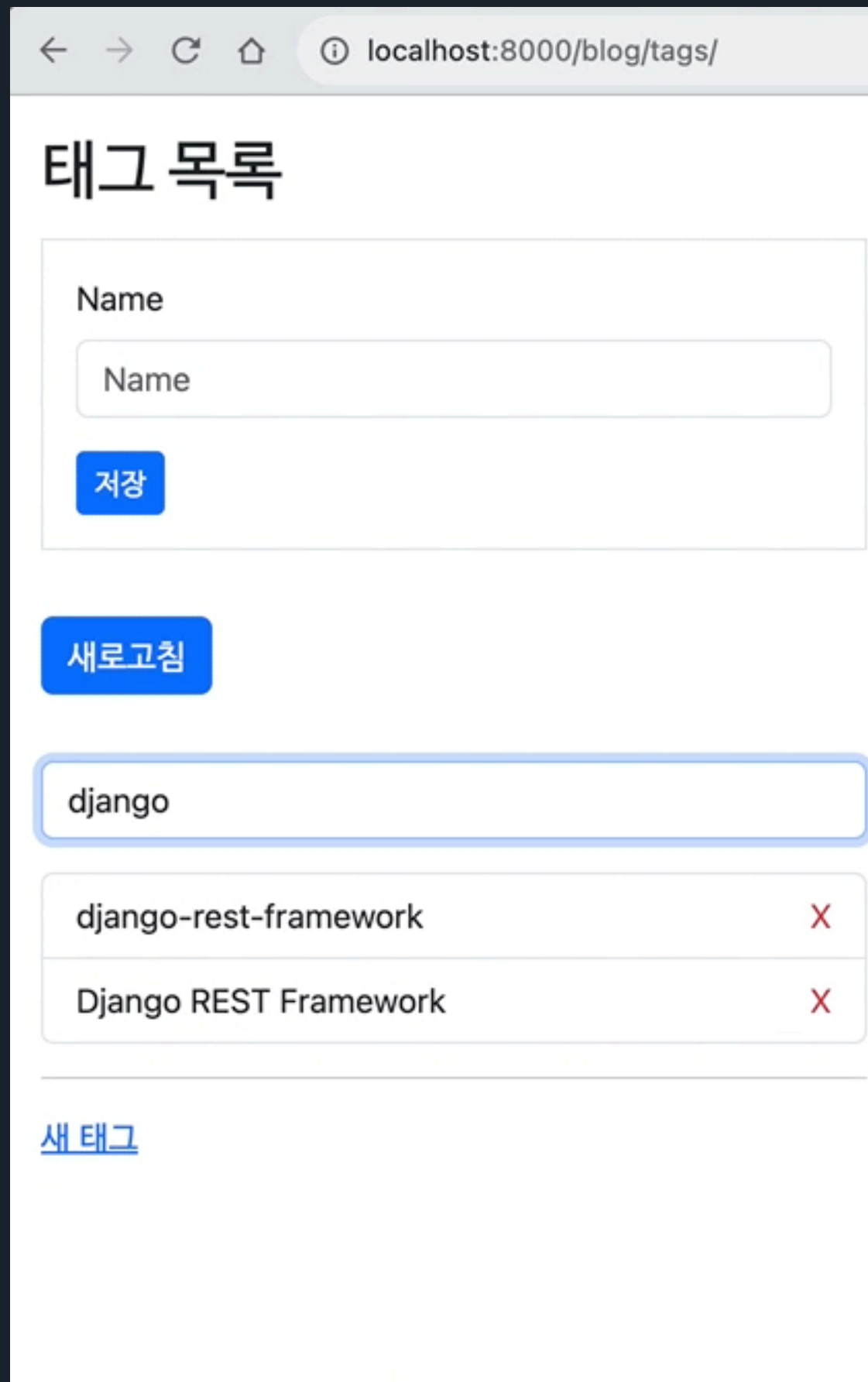
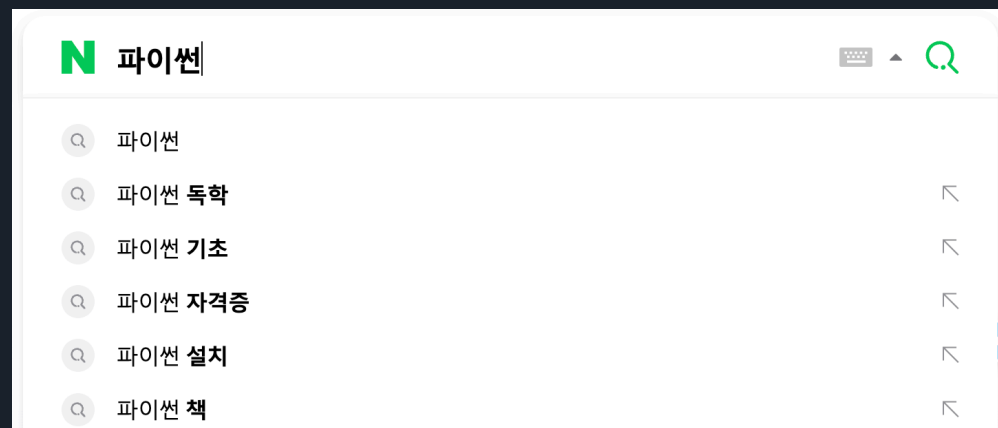
</form>
```

인생은 짧아요. 파이썬/장고를 쓰세요. :-)

© All rights reserved by 파이썬 사랑방

inplace 검색

inplace 검색



```
# blog/templates/blog/tag_list.html
```

```
{# 생략 #}
```

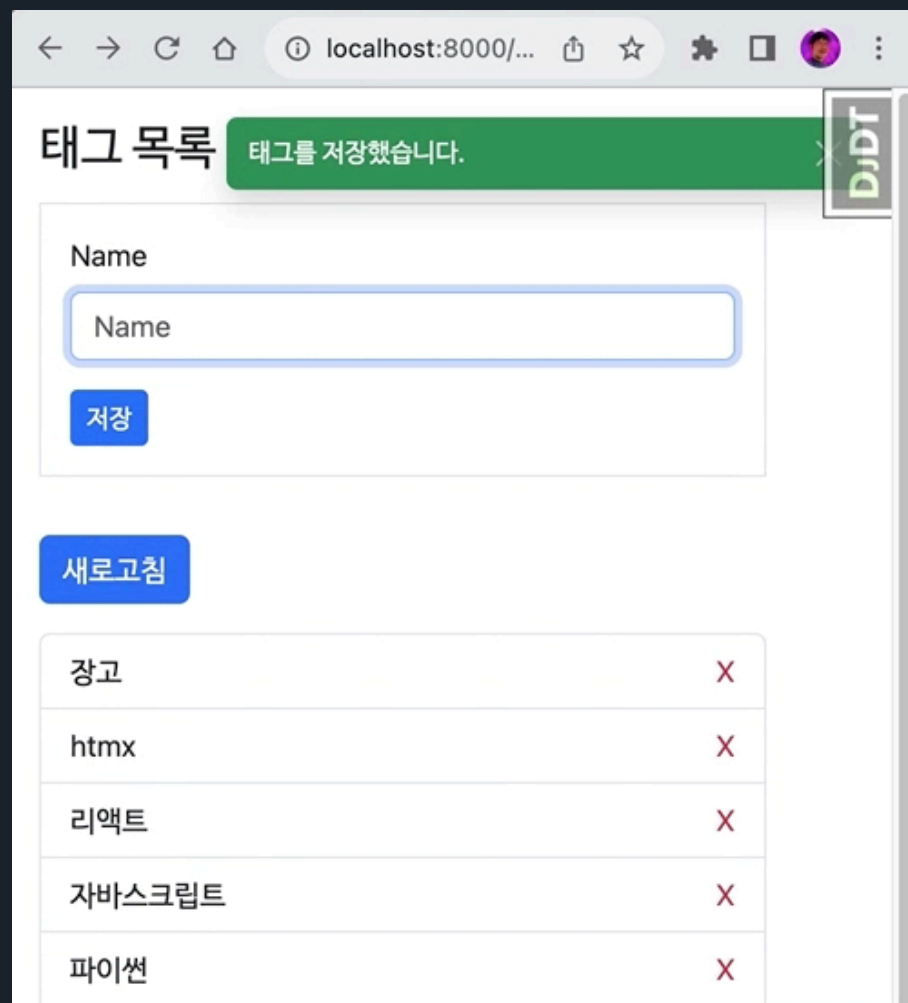
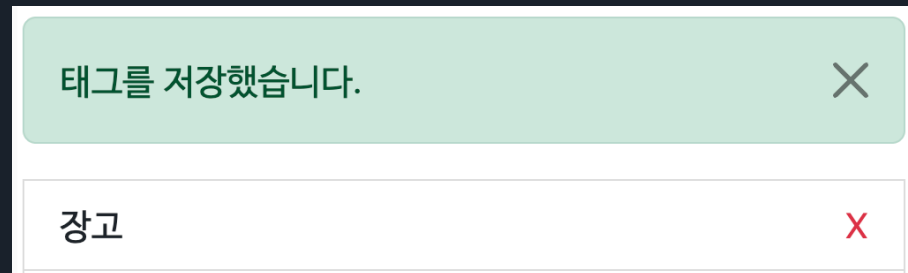
```
<input type="text" name="query" class="form-control my-3" placeholder="검색어를 입력해주세요."
      hx-get="{% url 'blog:tag_list' %}"
      hx-trigger="keyup[target.value.length === 0 || target.value.length >= 2] changed delay:400ms"
      hx-target="#tag-list-container .list-group" />
```

```
<div id="tag-list-container">
  <div class="list-group">
    {% include "blog/_tag_list.html" %}
  </div>
</div>
```

```
{# 생략 #}
```

Toast 스타일의 messages

토스트 메시지 보여주기



```
{# blog/templates/blog/base.html #}
{# 생략 #}
<main style="width: 400px; margin: 1em;">
    {# bootstrap_messages #}
    {% include "_messages_as_toast.html" %}
{# 생략 #}
```

```
{# blog/templates/blog/_tag_form.html #}
{% load django_bootstrap5 %}
```

```
{# bootstrap_messages #}
{% include "_messages_as_toast.html" %}
```

```
{# 생략 #}
```

```
{# blog/templates/blog/_tag_list_item.html #}
{# load django_bootstrap5 #}
```

```
{# bootstrap_messages #}
{% include "_messages_as_toast.html" %}
```

```
{# 생략 #}
```

```
<div class="list-group"> flex
  ><div class="toast-container position-fixed top-0 end-0 p-3">...</div>
  ><script class="htmx-settling">...</script>
  ><div class="toast-container position-fixed top-0 end-0 p-3">...</div>
  ><script class="htmx-settling">...</script>
  ><div class="toast-container position-fixed top-0 end-0 p-3">...</div>
  ><script class="htmx-settling">...</script>
  ><div class="toast-container position-fixed top-0 end-0 p-3">...</div>
  ><script class="htmx-settling">...</script>
```

```
{# core/templates/_messages_as_toast.html #}
```

```
{% if messages %}
  <div class="toast-container position-fixed top-0 end-0 p-3">
    {% for message in messages %}
      {# 생략 #}
    {% endfor %}
  </div>

  <script>
    (function () {
      const toast_container = document.currentScript.previousElementSibling;
      // 매 HTMX 응답에서도 토스트 메시지 노출이 되도록 DOMContentLoaded 이벤트 제거
      // document.addEventListener("DOMContentLoaded", function () {
      toast_container.querySelectorAll(".toast").forEach(function (el) {
        const toast = new bootstrap.Toast(el);
        toast.show();
      });
      // });
    })();
  </script>
{% endif %}
```

커스텀 이벤트로 토스트 메시지 보여주기

단일 toast-container에 토스트 메시지를 누적합니다.

```
/* core/static/core/toast-messages.js */
(function () {
  function getColorClass(tag) {
    return {
      "info": "text-white bg-primary",
      "success": "text-white bg-success",
      "warning": "text-dark bg-warning",
      "error": "text-white bg-danger",
      "debug": "text-white bg-secondary",
    }[tag] || "text-white bg-secondary";
  }

  function getButtonClass(tag) {
    return {
      "warning": "btn-close-dark",
      "debug": "btn-close-dark",
    }[tag] || "btn-close-white";
  }

  const html = `<div class="toast-container position-fixed top-0 end-0 p-3"></div>`
  document.body.insertAdjacentHTML('beforeend', html);

  /* body 요소에서 toast-message 이벤트를 처리합니다. */
  document.body.addEventListener("toast-message", function (e) {
    const {message, tag} = e.detail;
    const colorClass = getColorClass(tag);
    const buttonClass = getButtonClass(tag);

    const html = `
      <div class="toast align-items-center border-0 ${colorClass}" data-bs-autohide="true">
        <div class="d-flex">
          <div class="toast-body">${message}</div>
          <button type="button" class="btn-close me-2 m-auto ${buttonClass}" data-bs-dismiss="toast"></button>
        </div>
      </div>
    `;

    const container = document.querySelector(".toast-container");
    container.insertAdjacentHTML('afterbegin', html);

    const toastEl = container.querySelector(".toast:first-child");
    const toast = new bootstrap.Toast(toastEl);
    toast.show();
  });
})();
```

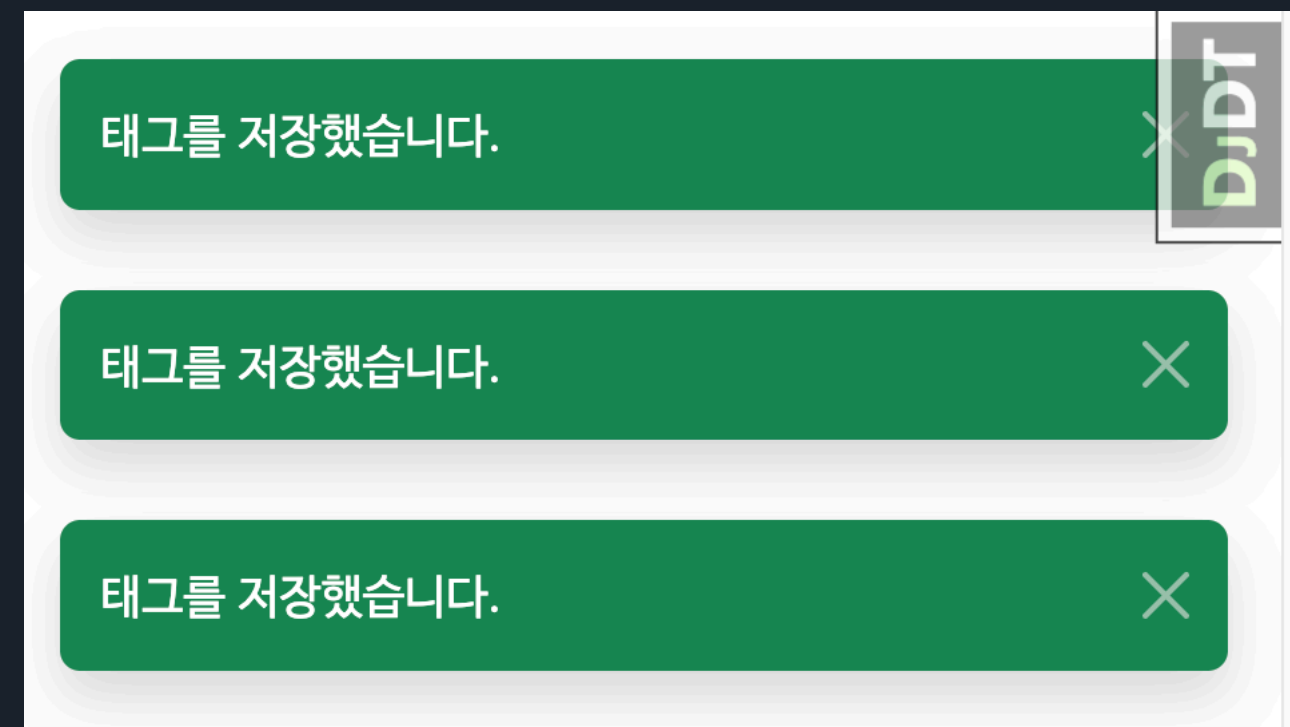
```
{# blog/templates/blog/base.html #}
{% load django_bootstrap5 static %}
{# 생략 #}
<main style="width: 400px; margin: 1em;">
  {# bootstrap_messages #}
{# 생략 #}
  {% bootstrap_javascript %}
  <script src="{% static 'core/toast-messages.js' %}"></script>
  {% include "blog/_messages_as_event.html" %}
  {% block extra-script %}{% endblock %}
</body>
</html>
```

```
{# core/templates/core/_messages_as_event.html #}

{% if messages %}
  {% for message in messages %}
    <script>
      (function () {
        const tag = "{{ message.level_tag }}";
        const message = "{{ message }}"
        /* body 요소에 toast-message 이벤트를 보냅니다. */
        hx.trigger(document.body, 'toast-message', {message: message, tag: tag});
      })();
    </script>
  {% endfor %}
{% endif %}
```

```
{# blog/templates/blog/base.html #}
{# blog/templates/blog/_tag_form.html #}
{# blog/templates/blog/_tag_list_item.html #}
```

```
{# 생략 #}
{# bootstrap_messages #}
{% include "core/_messages_as_event.html" %}
{# 생략 #}
```



toast-message.js 코드 : <https://gist.github.com/allieus/0e947e7c4e0fb1d414f65bdeb48efcde>

bootstrap5 공식문서 : <https://getbootstrap.kr/docs/5.0/components/toasts/#%EC%A4%91%EC%B2%A9>

© All rights reserved by 파이썬 사랑방

인생은 짧아요. 파이썬/장고를 쓰세요. :-)

login_required_hx 커스텀 장식자

htmx 요청을 받는 뷰에서 인증 여부를 확인한다면?

비로그인 상황에서 tag_new 뷰가 아닌 LoginView 응답을 화면에 대체됩니다. => 로그인 페이지로의 이동이 필요합니다.

```
# blog/views.py
```

```
@login_required
def tag_new(request, pk=None):
    ...
```

태그 목록

사용자 이름

사용자 이름

비밀번호

비밀번호

저장

새로그침

장고 X

사용자 이름

사용자 이름

비밀번호

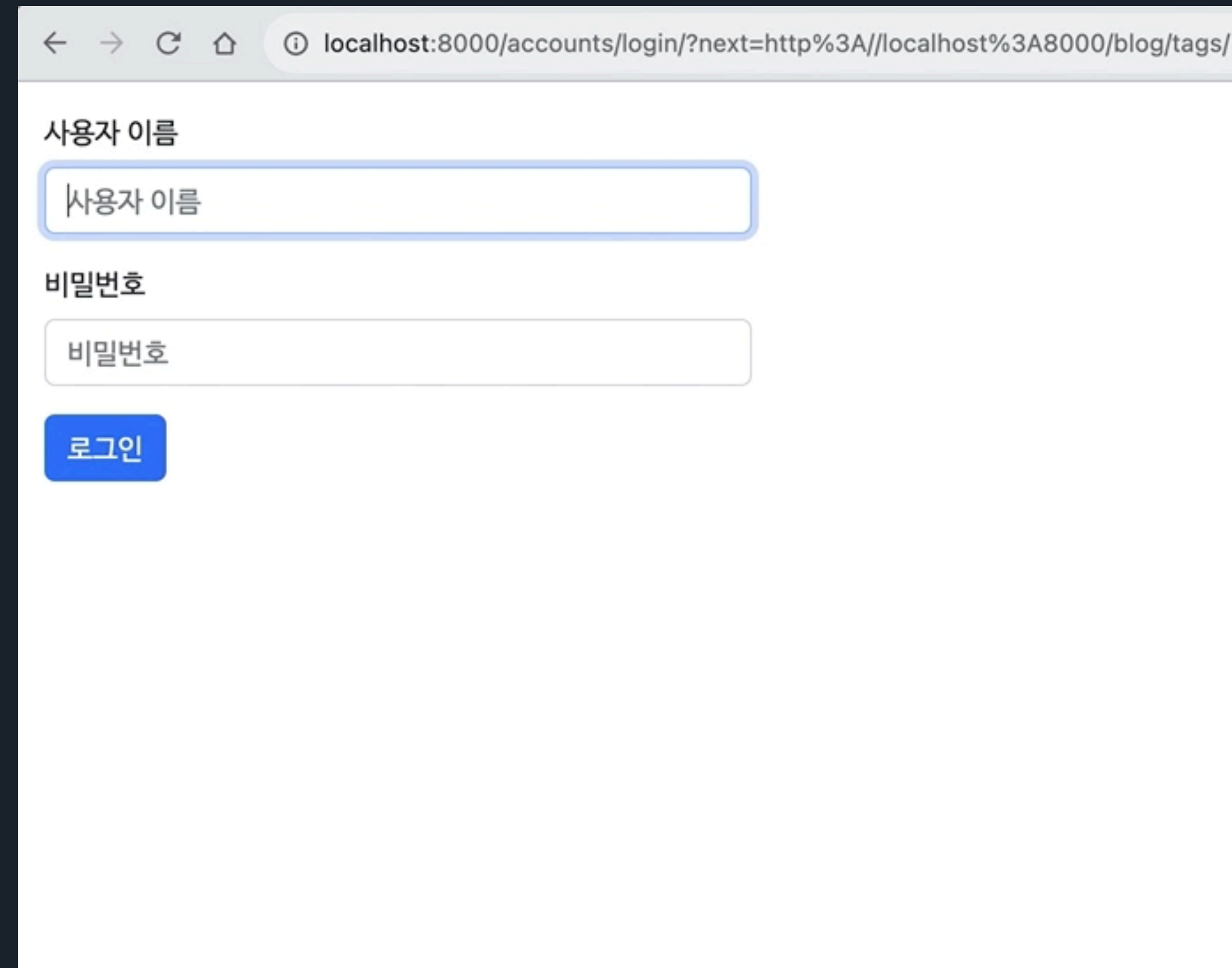
비밀번호

저장

리액트 X

login_required_hx 장식자 구현

htmx 요청에서는 HX-Redirect 헤더를 활용한 Redirect 응답을 합니다.



사용자 이름

비밀번호

로그인

```
# core/decorators.py

from functools import wraps

from django.contrib.auth import REDIRECT_FIELD_NAME
from django.contrib.auth.decorators import login_required as django_login_required
from django.http import HttpResponseRedirect
from django_htmx.http import HttpResponseRedirect

def login_required_hx(                                     # blog/views.py
    function=None,
    redirect_field_name=REDIRECT_FIELD_NAME,
    login_url=None,
):
    def decorator(view_func):
        @wraps(view_func)
        def wrapper(request, *args, **kwargs):
            decorated_view_func = django_login_required(
                function=view_func,
                redirect_field_name=redirect_field_name,
                login_url=login_url,
            )

            response = decorated_view_func(request, *args, **kwargs)

            if isinstance(response, HttpResponseRedirect):
                if request.htmx:
                    next_url = response.url
                    # TODO: next_url 문자열에서 next 인자만 현재 페이지 주소로 변경하기
                    return HttpResponseRedirect(next_url)

            return response

        return wrapper

    if function:
        return decorator(function)

    return decorator

# blog/views.py



from core.decorators import login_required_hx

@login_required_hx
def tag_new(request, pk=None):
    ...
```

HTMX 응답 대기 중임을 보여주기

HTMX 응답 대기 중임을 보여주기

hx-indicator 속성을 활용합니다.

| | |
|------------|--|
| 파이 |  |
| 파이썬 |  |
| 페이지 끝 입니다. | |

```
{# blog/templates/blog/tag_list.html #}

<div class="position-relative">
  <input type="text" name="query" class="form-control my-3"
    hx-get="{% url 'blog:tag_list' %}"
    hx-get-with-timestamp
    hx-trigger="keyup[target.value.length == 0 || target.value.length >= 2] changed delay:400ms"
    hx-target="#tag-list-container .list-group"
    hx-swap="innerHTML transition:true"
    hx-indicator="#tag-list-query-indicator"
  />

  {# indicator가 사용될 때에는 .htmx-request가 적용됩니다. #}
  <div id="tag-list-query-indicator" class="htmx-indicator">
    {# ref: https://getbootstrap.com/docs/5.3/components/spinners/#growing-spinner #}
    <div class="spinner-grow text-primary" style="width: 20px; height: 20px; position: absolute; top: 50%; margin-top: -10px; right: 10px;">
    </div>
  </div>
</div>

<!-- HTMX 기본 스타일 -->

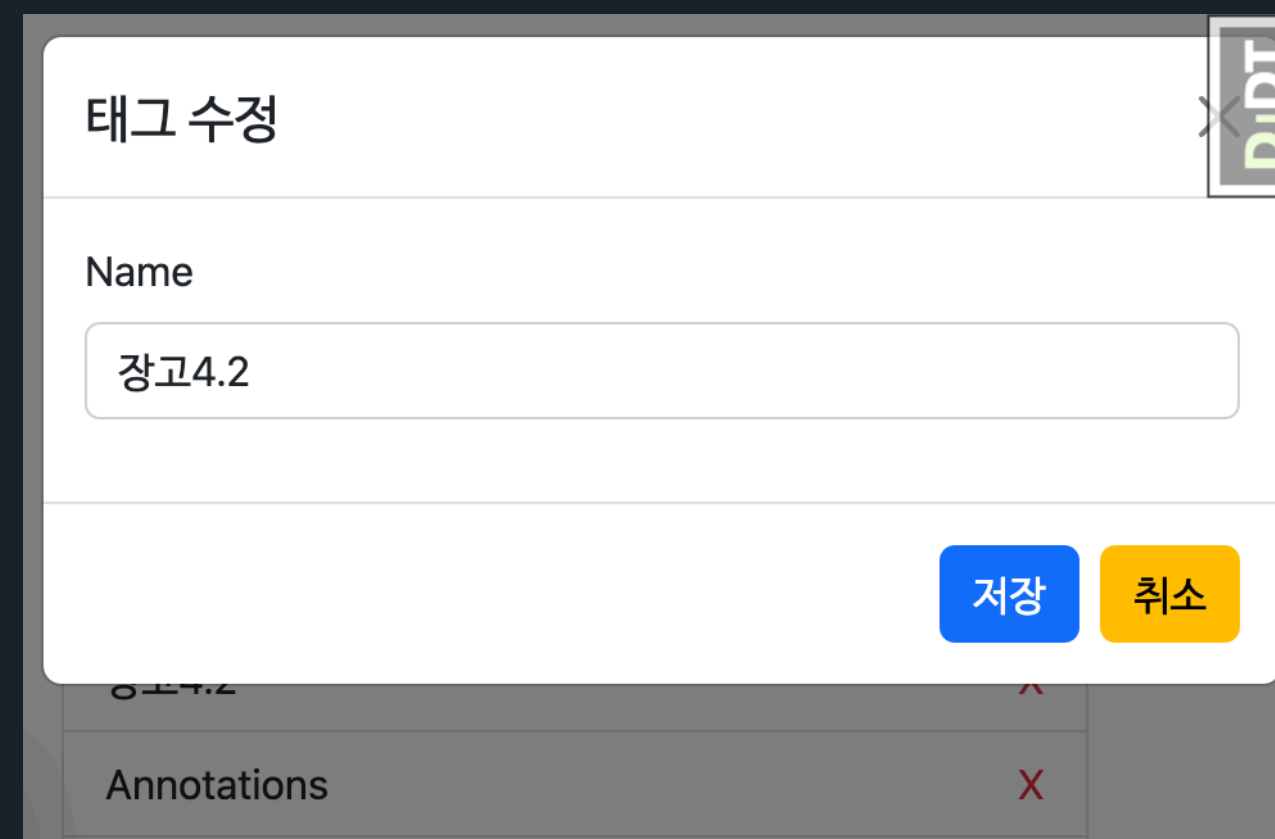
<style>
.htmx-indicator { opacity: 0; transition: opacity 200ms ease-in; }
.htmx-request .htmx-indicator { opacity: 1 }
.htmx-request.htmx-indicator { opacity: 1 }
</style>
```

실습 코드 : <https://gist.github.com/allieus/03f3e61af3164b8440f523663ddcd919>

HTMX hx-indicator 공식문서 : <https://htmx.org/attributes/hx-indicator/>

bootstrap5 spinner 공식문서 : <https://getbootstrap.com/docs/5.3/components/spinners/#growing-spinner>

Modal로 생성/수정폼 띄우기



태그 수정

Name

장고4.2

저장 취소

Annotations

django-template-partials 라이브러리

한 템플릿 파일 내에서 partial을 정의하고, partial을 재사용할 수 있습니다.

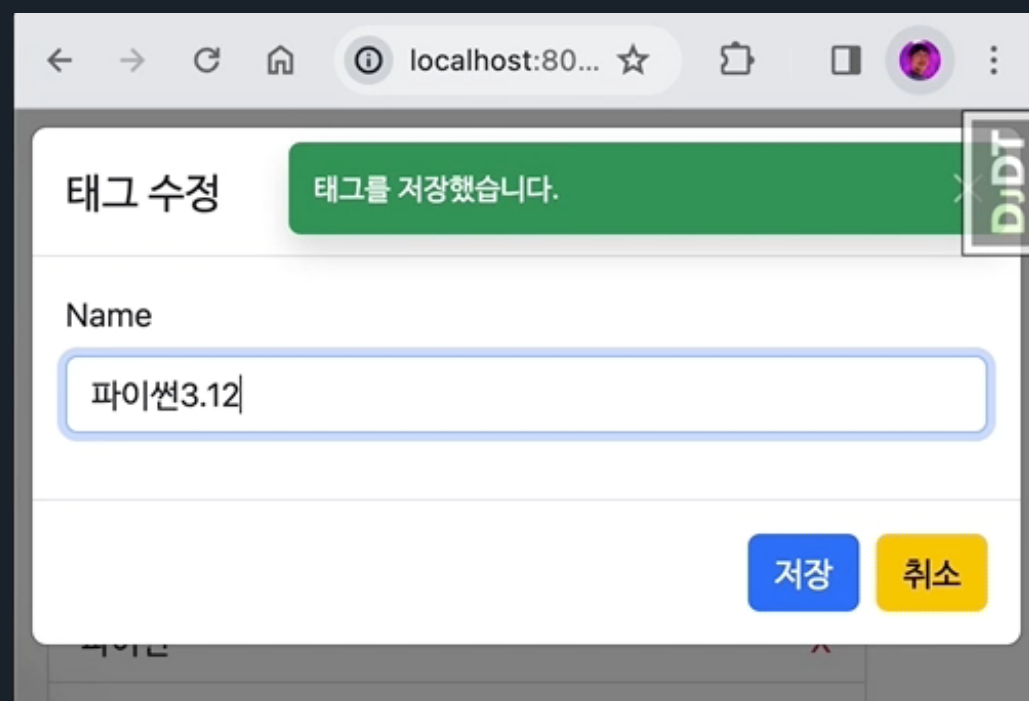
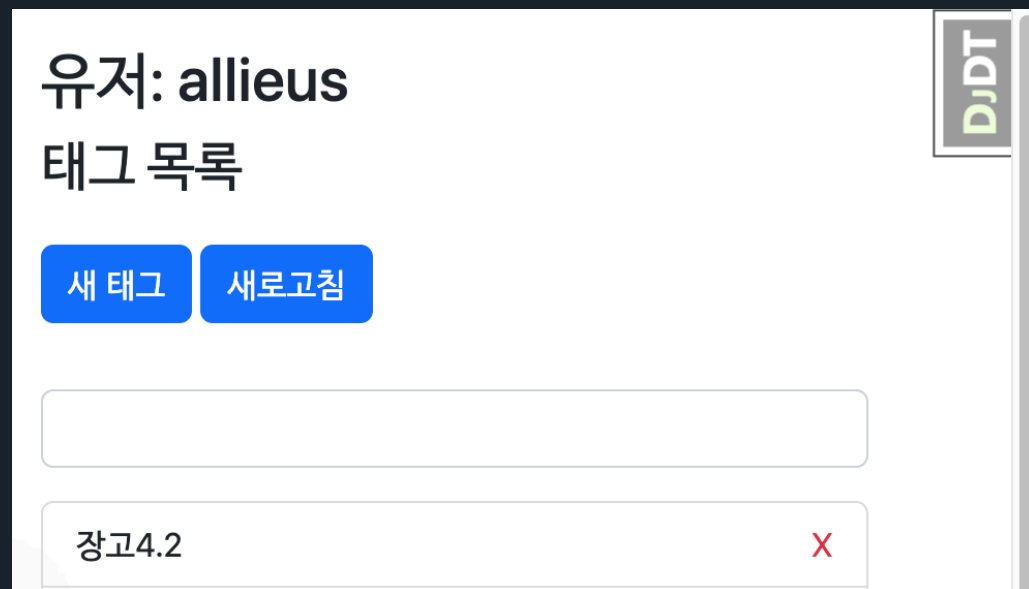
- `python -m pip install django-template-partials`
- `settings.INSTALLED_APPS`에 "template_partials" 추가

```
{% load partials %}

{# partial 정의 #}
{% partialdef tag-form %}
    <form>
        {% csrf_token %}
        {% bootstrap_form form %}
    </form>
{% endpartialdef %}

{% if request.method == "POST" %}
    {# partial 활용 #1 #}
    {% partial tag-form %}
{% else %}
    <div class="modal-body">
        {# partial 활용 #2 #}
        {% partial tag-form %}
    </div>
{% endif %}
```

Modal로 생성/수정 폼 띄우기



```
{# blog/templates/blog/base.html #}

    {# Modal이 위치할 요소를 먼저 정의합니다. #}
    <div id="modal-container"></div>
</body>
</html>
```

```
{# blog/templates/blog/tag_list.html #}

<div hx-get="{% url 'blog:tag_new' %}"
      hx-get-with-timestamp
      hx-trigger="load"></div>

<button class="btn btn-primary"
        hx-get="{% url 'blog:tag_new' %}"
        hx-get-with-timestamp
        hx-trigger="click"
        hx-target="#modal-container"
        hx-swap="innerHTML">
    새 태그
</button>
```

```
{# blog/templates/blog/_tag_list_item.html #}

{% include "core/_messages_as_event.html" %}

<div class="list-group-item d-flex justify-content-between align-items-center">
    <div
        style="cursor: pointer;"
        hx-get="{% url 'blog:tag_edit' tag.pk %}"
        hx-trigger="click"
        hx-target="closest .list-group-item"
        hx-target="#modal-container"
        hx-swap="outerHTML"
        hx-swap="innerHTML">

        >
        {{ tag.name }}
    </div>
    {# 삭제 버튼 생략 #}
</div>
```

```
# blog/views.py

@login_required_hx
def tag_new(request, pk=None):
    if pk:
        instance = get_object_or_404(Tag, pk=pk)
        tag_list_item_url = reverse("blog:tag_list_item", args=[pk])
    else:
        instance = None
        tag_list_item_url = None

    if request.method == "GET":
        form = TagForm(instance=instance)
    else:
        form = TagForm(data=request.POST, instance=instance)
        if form.is_valid():
            form.save()
            messages.success(request, "태그를 저장했습니다.")
            # 항상 HTML를 통한 요청. 메시지 노출을 위한 템플릿 지정
            response = render(request, "core/_messages_as_event.html")
            response = trigger_client_event(response, "refresh-tag-list")
            return response

            if request.htmx:
                if tag_list_item_url:
                    return redirect(tag_list_item_url)
                else:
                    form = TagForm()
                    response = render(request, "blog/_tag_form.html", {
                        "form": form,
                    })
                    response = trigger_client_event(response, "refresh-tag-list")
                    return response
            else:
                return redirect("blog:tag_list")

            if request.htmx:
                template_name = "blog/_tag_form.html"
            else:
                template_name = "blog/tag_form.html"

    return render(
        request,
        template_name,
        {
            "form": form,
            "cancel_url": tag_list_item_url,
        },
    )
```

다음 페이지에 계속 ...

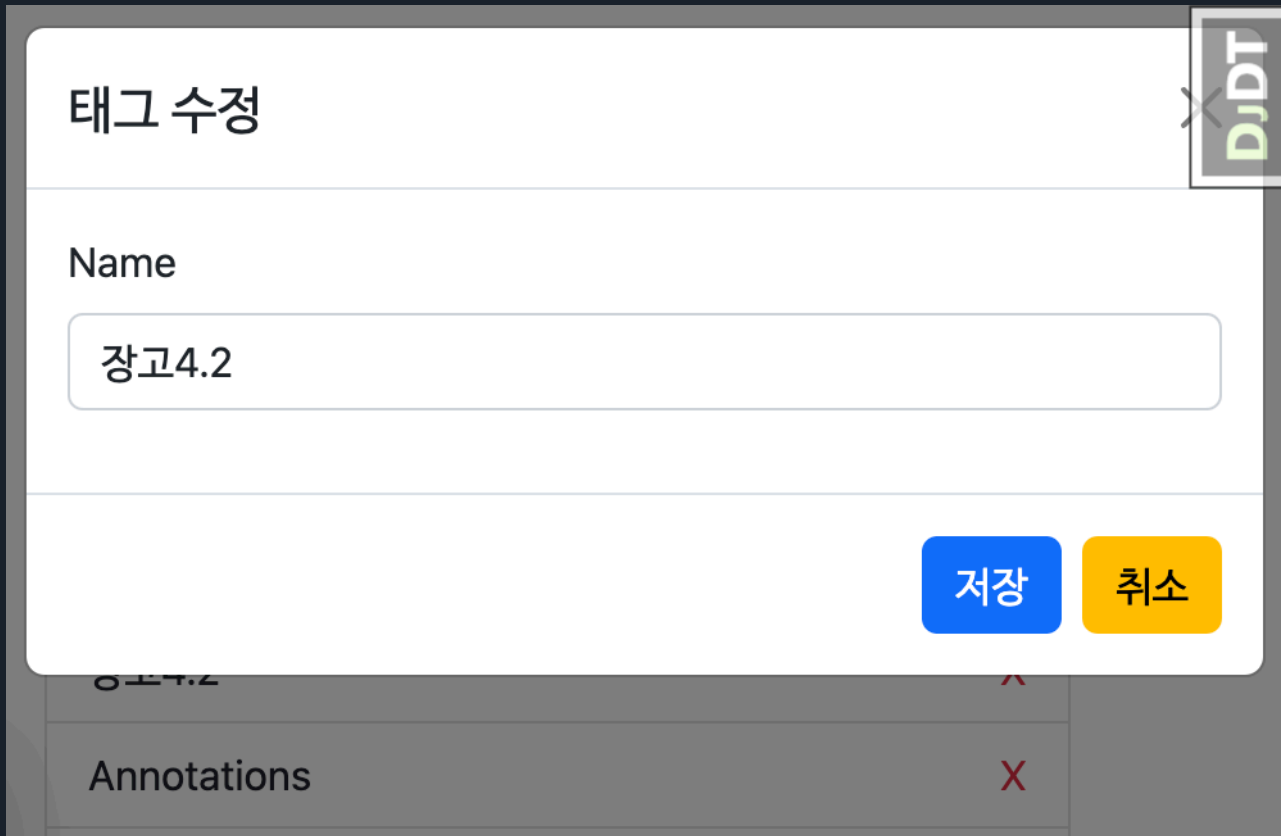

```
{# blog/templates/blog/_tag_form.html #}

{% load django_bootstrap5 partials %}

{% include "core/_messages_as_event.html" %}

{% partialdef tag-form %}
    <form class="border my-3 p-3"
        hx-post="{{ request.get_full_path }}"
        hx-trigger="submit once"
        hx-swap="outerHTML"
        autocomplete="off"
        novalidate>
        {% csrf_token %}
        {% bootstrap_form form %}

        <button class="btn btn-primary btn-sm">저장</button>
        <button class="btn btn-warning btn-sm"
            hx-get="{{ cancel_url|default:request.get_full_path }}"
            hx-target="closest form"
            hx-swap="outerHTML">
            취소
        </button>
    </form>
{% endpartialdef %}
```



```
{# "GET" 요청인 modal 요청. "POST" 요청은 modal 없이 폼 필드 응답만. #}
{% if request.method == "POST" %}
    {% partial tag-form %}
{% else %}
    {# https://getbootstrap.com/docs/5.3/components/modal/#examples #}
    <div class="modal fade" tabindex="-1">
        <div class="modal-dialog modal-dialog-scrollable">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title">
                        {% if not form.instance.pk %}태그 생성{% else %}태그 수정{% endif %}
                    </h5>
                    <button type="button" class="btn-close" data-bs-dismiss="modal"
                        aria-label="Close"></button>
                </div>
                <div class="modal-body">
                    {% partial tag-form %}
                </div>
                <div class="modal-footer">
                    <button type="submit" class="btn btn-primary">저장</button>
                    <button type="button" class="btn btn-warning" data-bs-dismiss="modal">취소</button>
                </div>
            </div>
        </div>
    </div>
</div>
<script>
    (function () {
        {# 위 modal 요소로 즉시 모달창을 띄웁니다. #}
        const modalEl = document.currentScript.previousElementSibling;
        const modal = new bootstrap.Modal(modalEl);
        modal.show();

        {# 저장 버튼을 클릭하면, form submit 이벤트를 발생시킵니다. #}
        modalEl.querySelector("button[type=submit]").onclick = () => {
            const formEl = modalEl.querySelector("form");
            htmx.trigger(formEl, "submit");
        };

        {# refresh-tag-list 이벤트를 받으면 모달창을 닫습니다. #}
        document.body.addEventListener("refresh-tag-list", function () {
            modal.hide();
        });
    })();
</script>
{% endif %}
```


인생은 짧아요.
파이썬/장고로 시간을 아끼세요.
당신의 페이스메이커가 되겠습니다.



me@pyhub.kr