django_with_axios

01. like (get)

- axios 추가

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

<script src="https://kit.fontawesome.com/caed28bd65.js" crossorigin="anonymous"></script>
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
{% bootstrap_css %}
<title>Document</title>
```

- templates
  - 좋아요 버튼의 구조를 바꿔보자.

```
<!-- articles/_article.html -->

<!-- 기존 코드 -->
<div class="card mb-3">
  ...
    <p class="card-text">
      <a href="{% url 'articles:like' article.pk %}">
        {% if user in article.like_users.all %}
          <i class="fas fa-heart fa-lg" style="color: crimson;"></i>
        {% else %}
          <i class="fas fa-heart fa-lg" style="color: black;"></i>
        {% endif %}
      </a>
...
```

```
<!-- 변경 코드 -->

<div class="card mb-3">
  <div class="card-header">
    작성자 : <a class="card-link" href="{% url 'accounts:profile' article.user %}">{{ article.user }}</a>
  </div>
  <div class="card-body">
    <h5 class="card-title">글 제목 : {{ article.title }}</h5>
    <p class="card-text">{{ article.pk }}</p>
    <hr>
    <p class="card-text">
      {% if user in article.like_users.all %}
        <i class="fas fa-heart fa-lg like-button" style="color: crimson;" data-id="{{ article.pk }}"></i>
      {% else %}
        <i class="fas fa-heart fa-lg like-button" style="color: black;" data-id="{{ article.pk }}"></i>
      {% endif %}
```

  - <a> 태그 제거
  - 하트 아이콘 태그(`<i>`) 에 다음과 같은 속성을 추가
    - `like-button` class : 선택하기 위해
    - `data-id`: 데이터 처리시에 해당 element 가 어떤 article 의 버튼인지 JS가 구분할 수 있도록 data-id attribute를 지정 (JS 는 `data-id` 를 통해 개별적인 오브젝트에 접근 가능하다)

Template 수정

- `views.py` 랑 `index.html` 에서 작업을 진행한다.
- axios 요청 보내보고, response `console.log` 로 찍어보면서 필요 데이터를 찾는다.

▼ `event.target` 찍어보고 `event.target.classList` 활용해서 class 를 추가/삭제 한다.

```
likeButtons
> NodeList(5) [i.fas.fa-heart.like-button, i.fas.fa-heart.like-button, i.fas.fa-heart.like-button, i.far.fa-heart.like-button, i

해당 노드 리스트 안에서 dataset -> id 보여주기
dataset: DOMStringMap
id: "5"
```

```
// index.html

...
<script>
  // 1. 각 게시글 별로 좋아요 버튼이 있으니 All(이 한줄 적고 바로 콘솔로 가서 찍어보자!)
  const likeButtons = document.querySelectorAll('.like-button')

  // 2. forEach 사용해서 각각의 좋아요(button)를 클릭했을 때
  likeButtons.forEach(button => {
    button.addEventListener('click', function (event) {
      console.log(event) // 디버깅용 -> event.target.classList, event.target.dataset.id 같이 보기
      })
    })
</script>
{% endblock %}
```

- 발생한 event 객체의 target안에 있는 요소들을 확인해봤다.
  - classList: 내가 클릭한 태그(i태그)에 적용된 **class 목록! (3개)**
  - dataset.id: `<i>` 태그 안에 작성한 `data-id` 속성의 값. 해당 게시글 번호

- 이 정보들을 최대한 활용하여 나머지 로직을 완성해보자.
- `response` 라는 응답 객체 내에 어떤 응답 데이터가 왔는지 확인해보면 `index.html` 페이지 전체를 확인할 수 있다.

```
// index.html
...
<script>
  const likeButtons = document.querySelectorAll('.like-button')
  likeButtons.forEach(button => {
    button.addEventListener('click', function (event) {
      console.log(event)
      //1. event.target.dataset.id의 value는 data-id의 값이 들어있어 variable routing으로 쓰임
      const articleId = event.target.dataset.id

      //2. 해당 상세 게시글의 좋아요 요청을 보낸다.
      axios.get(`/articles/${articleId}/like/`)

        //3. 응답을 확인한다.
        .then(response => {
          console.log(response)
          }
        })
        .catch(error => console.log(error))
      })
</script>
{% endblock %}
```

- 여기서 문제가 한가지 있음. 응답 결과가 `index.html` 임. 이건 like 함수의 실행이 끝나고 index 페이지로 redirect 시켰기 대문이다.
- 우리는 '좋아요' 기능만 구현 할 것이기 때문에 HTML 문서 전체를 반환할 필요가 없음. 이는 불필요하므로 수정이 필요하다.

View 수정

- **버튼 활성화를 위해 필요한 불리언 값을 생성**

💡 좋아요 버튼을 선택하고 NodeList의 요소를 `forEach` 로 꺼내면서 각각의 버튼에 이벤트를 생성한다. 해당 버튼의 `articleId` 를 찾고 좋아요 요청을 보냈을 때, view에서 보내준 boolean 값에 따라 클래스를 지우거나 추가한다.

- 기존 redirect되서 응답 답은 `index.html` 이 아닌 JSON의 형태로 응답 결과를 반환 받을 것!
- 좋아요를 **취소하는** 경우 liked 변수는 **False**로
- 좋아요를 **하는 경우** liked 변수는 **True**로 설정되게 한다.

```
# articles/views.py

from django.http import JsonResponse

@login_required
def like(request, article_pk):
    article = get_object_or_404(Article, pk=article_pk)
    user = request.user

    if article.like_users.filter(pk=user.pk).exists():
        article.like_users.remove(user)
        liked = False
    else:
        article.like_users.add(user)
        liked = True

    context = { 'liked': liked, }
    # Json 객체로 응답
    return JsonResponse(context)
```

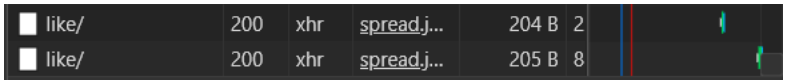
- script

```
// articles/index.html

<script>
  const likeButtons = document.querySelectorAll('.like-button')
  likeButtons.forEach(button => {
    button.addEventListener('click', function(event) {
      const articleId = event.target.dataset.id
        axios.get(`/articles/${articleId}/like/`)
          .then(response => {
            if (response.data.liked) {
              event.target.style.color = 'crimson'
            } else {
              event.target.style.color = 'black'
            }
          })
          .catch(error => console.log(error))
    })
  })
</script>
```

간단한 페이지 전환 다시 테스트!

1. 첫번째
   - 스크롤을 내릴 수 있을 만큼의 글을 미리 작성한다.
   - 가장 하단에 작성된 글에서 좋아요를 클릭한다.
   - **이제 페이지는 상단으로 올라가지 않는다! 즉, 페이지의 전체의 전환이 발생하지 않는다!!!!**
2. 두번째
   - 인덱스 페이지에서 좋아요를 누르기 전에 콘솔의 NetWork 탭을 띄어놓는다.

- 처음에 페이지를 새로고침하여 어떤 데이터들을 우리 서버(django)로부터 받아오는지 확인해보자
- 이번에는 좋아요 버튼을 클릭한다. 아까처럼 동일한 데이터를 다시 받는 것이 아니라 **좋아요 로직과 관련된 데이터만 받아온다!**



1.1. like count

```
# articles/views.py

@login_required
def like(request, article_pk):
    article = get_object_or_404(Article, pk=article_pk)
    user = request.user

    if article.like_users.filter(pk=user.pk).exists():
        article.like_users.remove(user)
        liked = False
    else:
        article.like_users.add(user)
        liked = True

    context = {
        'liked': liked,
        'count': article.like_users.count(),
    }
    return JsonResponse(context)
```

Template 수정

span 태그의 id 는 article의 id를 들고 있어야 원하는 게시글의 count 값에 변화를 줄 수 있다.

- 중요! 지금 이 로직은 JS axios 요청을 통해 처리하고 있는 것이지 context 에 변수를 넘겨 처리하는게 아님!
- 아래와 같은 로직을 작성하는 이유는 사용자가 버튼을 눌렀을 때마다 증가하는 count 값을 context가 아닌 JS로 동적으로 변환시키기 위함임.

```
<!-- articles/_article.html -->

<div class="card mb-3">
  ...
    <p class="card-text">
      {% if user in article.like_users.all %}
        <i class="fas fa-heart fa-lg like-button" style="color: crimson;" data-id="{{ article.pk }}"></i>
      {% else %}
        <i class="fas fa-heart fa-lg like-button" style="color: black;" data-id="{{ article.pk }}"></i>
      {% endif %}
      <span id="like-count-{{ article.pk }}">{{ article.like_users.count }}</span> 명이 이 글을 좋아합니다.
    ...
</div>
```

- span 태그와 id 추가

  여기서 id는 article의 id를 들고있어야 원하는 article의 count 값에 변화를 줄 수 있다.

  ```
  // articles/index.html

  <script>
    const likeButtons = document.querySelectorAll('.like-button')
    likeButtons.forEach(button => {
  ```

```
    button.addEventListener('click', function(event) {
      const articleId = event.target.dataset.id
      axios.get(`/articles/${articleId}/like/`)
      .then( response => {
        // console.log(response)
        // 위에서 정의한 span 태그의 id값에 사용자가 좋아요를 누를 때마다 response.data.count의 값을 갱신시킴
        document.querySelector(`#like-count-${articleId}`).innerText = response.data.count
        if (response.data.liked) {
        ...
  })
</script>
```

---

02. like - POST

- 좋아요도 결국은 데이터에 조작을 가하는 행위이기 때문에 GET이 아닌 POST 요청으로 처리해야 한다.
- 먼저 다음과 같이 post 요청으로 바꿔보자. post 요청은 반드시 url 마지막에 `/`가 있어야한다.

```
// index.html

...
axios.post(`/articles/${articleId}/like/`)
```

- 제대로 동작하지 않음을 알 수 있다. ajax 요청은 아래 두가지 방법으로 디버깅을 진행한다.
  - 크롬 콘솔 : 요청이 제대로 갔는지, 오류는 없는지. 왔는데 처리를 못한 것인지. (`403 Forbidden` 확인)
  - 서버 로그 : 요청이 들어왔는지, 오류는 없는지. (`CSRF token missing or incorrect` 확인)

```
Forbidden (CSRF token missing or incorrect.): /articles/5/like/
```

- CSRF token 이 존재하지 않는다. 우리는 form을 사용해서 요청을 보내는 것이 아니기 때문에 `{% csrf_token %}` 을 적어줄 수도 없는데 어떻게 하면 좋을까?
- cookie를 활용하면 된다. `axios` **는 csrftoken을 쿠키에 담아서 편하게 서버로 전송할 수 있게 지원한다.**
- csrftoken 은 `개발자도구 - Application - Cookies` 에서 확인할 수 있다.

▼ **csrftoken - Django**

https://docs.djangoproject.com/en/2.2/ref/csrf/#ajax

AJAX

While the above method can be used for AJAX POST requests, it has some inconveniences: you have to remember to pass the CSRF token in as POST data with every POST request. For this reason, there is an alternative method: on each XMLHttpRequest, set a custom **X-CSRFToken** header (as specified by the **CSRF_HEADER_NAME** setting) to the value of the CSRF token. This is often easier because many JavaScript frameworks provide hooks that allow headers to be set on every request.

If you're using AngularJS 1.1.3 and newer, it's sufficient to configure the **`$http`** provider with the cookie and header names:

```
$httpProvider.defaults.xsrfCookieName = 'csrftoken';
$httpProvider.defaults.xsrfHeaderName = 'X-CSRFToken';
```

```
// response.config 에서 확인 가능
// csrftoken을 Header에 담을 때 X-CSRFToken 으로 보내야함(위의 공식문서 참고)
// articles/index.html (이걸로 진행)

<script>
  const likeButtons = document.querySelectorAll('.like-button')
  likeButtons.forEach(button => {
    button.addEventListener('click', function(event) {
      const articleId = event.target.dataset.id
      axios.defaults.xsrfCookieName = 'csrftoken'
      axios.defaults.xsrfHeaderName = 'X-CSRFToken'
```

```
// 2번째 방법 (참고만)
// 두번째 인자 {} 는 post 요청으로 데이터를 생성할 때 필요한 데이터를 넣는 곳.
// 지금은 필요없기 때문에 비어 놓는다.

axios.post(`/articles/${articleId}/like/`, {}, {
  headers: {
    'X-CSRFToken': csrftoken,
  }
})
```

- 잘 동작하는지 확인해보자

2.1. `is_ajax` 분기

View 수정

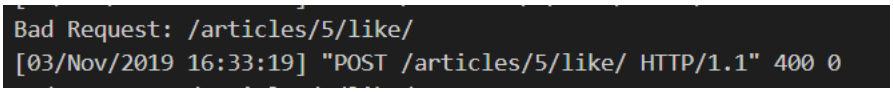
- view 함수 내에서 ajax 요청은 JsonResponse로 응답하고 웹 요청은 render 혹은 redirect 등으로 표현
- 하지만 **이제 우리의 좋아요 함수는 Json 객체만 리턴 하기 때문에 render 혹은 redirect를 사용할 필요가 없다.**
- 요청이 ajax 일 경우에만 좋아요 로직을 수행하게 하고 그렇지 않으면 400 에러를 보여주자.

```
# articles/views.py

from django.http import JsonResponse, HttpResponseBadRequest

@login_required
def like(request, article_pk):
    if request.is_ajax():
        article = get_object_or_404(Article, pk=article_pk)
        if article.like_users.filter(pk=request.user.pk).exists():
            article.like_users.remove(request.user)
            liked = False
        else:
```

```
            article.like_users.add(request.user)
            liked = True
        context = {'liked': liked, 'count': article.like_users.count(), }
        return JsonResponse(context)
    else:
        return HttpResponseBadRequest
```

요청 보낸 후 에러 확인

- 이 상태에서 요청을 보내보자. 우리는 ajax 요청이라고 생각했지만, django 는 이 요청이 ajax 요청인지 모른다.
- 그래서 `is_ajax()` 분기를 통과하지 못해 `Bad request` 라는 메시지를 받게 된다.



- **`XHR` 요청을 확인하는 것은 요청 정보 내에 `HTTP_X_REQUESTED_WITH` header 에 `XMLHttpRequest` 객체 값을 가지고 있어야 한다.**
- axios 문서를 확인하고 다음과 같이 입력한다. [global axios defaults]

▼ 스택 오버 플로우

https://stackoverflow.com/questions/41163800/how-does-one-set-global-headers-in-axios

```
// articles/index.html

<script>
  const likeButtons = document.querySelectorAll('.like-button')
  likeButtons.forEach(button => {
    button.addEventListener('click', function(event) {
      console.log(event)
      const articleId = event.target.dataset.id
      axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest'
      ...
```

- 새로 고침을 한번 하고 다시 한번 요청을 보내보면 이제는 정상적으로 작동하는 것을 확인할 수 있다.

- 완성코드

```
// articles/index.html

<script>
  const likeButtons = document.querySelectorAll('.like-button')
  likeButtons.forEach(button => {
    button.addEventListener('click', function(event) {
      const articleId = event.target.dataset.id
      axios.defaults.headers.common['X-Requested-With'] = 'XMLHttpRequest'
      axios.defaults.xsrfCookieName = 'csrftoken'
      axios.defaults.xsrfHeaderName = 'X-CSRFToken'
        axios.post(`/articles/${articleId}/like/`)
          .then(response => {
            document.querySelector(`#like-count-${articleId}`).innerText = response.data.count
            if (response.data.liked) {
              event.target.style.color = 'crimson'
            } else {
              event.target.style.color = 'black'
            }
          })
          .catch(error => console.log(error))
    })
```

```
  })
</script>
```

- 비회원이 좋아요 버튼 누를 경우 undefined 출력 해결

```
{% if user.is_authenticated %}
  axios.post(`/articles/${articleId}/like/`)
    .then(response => {
      document.querySelector(`#like-count-${articleId}`).innerText = response.data.count
      if (response.data.liked) {
        event.target.style.color = 'crimson'
      } else {
        event.target.style.color = 'black'
      }
    })
    .catch(error => console.log(error))
{% else %}
  alert('좋아요 기능을 사용하려면 로그인을 해주세요.')
{% endif %}
```