Ajax를 이용한 댓글생성 (보충수업)

구분	Javascript
과목	

[스켈레톤 코드]

bo.zip

요구사항

1. comment_form 에 입력된 데이터는 aixo의 data에 전달
2. 댓글 생성 form 선택

   2-1. const commentCreateForm=document.querySelector('#comment-creat-form)
3. 새로운 formData 객체 생성

   3-1. const formData=new FormData(commentCreateForm)

4. view 함수는 댓글 생성후 상세페이지로 redirect 하지 않고
5. view 함수는 댓글이 생성된 게시글의 pk와 생성된 댓글의 pk를 json으로 반환한다.
6. 댓글 생성 요청 후, 댓글 생성 input tag에 작성된 내용은 초기화 되어야 한다.
7. 댓글 생성 요청 후, 브라우저 새로고침 없이 댓글 작성자와 생성된 댓글내용, 삭제버튼이 화면에 랜더링 되어야 한다.

스켈레톤 코드

articles/detail.html

```
<hr>
{% if request.user.is_authenticated %}
  <form action="{% url 'articles:comments_create' article.pk %}" method="POST">
    {% csrf_token %}
    {{ comment_form }}
    <input type="submit">
  </form>
{% else %}
  <a href="{% url 'accounts:login' %}">[댓글을 작성하려면 로그인하세요.]</a>
{% endif %}
```

우리는 요청을 보낼 때 form 태그의 action 그리고 method 속성을 이용하는 것이 아니라

axios로 대체를 할 것이다.

Axios란?

JavaScript에서 ajax 기술을 사용하기 위한 라이브러리다.

Ajax기술은 HTTP통신을 할 때 XHR 객체를 이용해 새로고침 없이 비동기적으로 서버와 통신하여 데이터를 주고받을 수 있는 기술을 말한다.

CDN은 스켈레톤 코드에 이미 적혀 있다.

따라서 form 태그와 Axios를 함께 사용해서 사용자가 폼을 제출했을 때 페이지 새로고침 없이 서버로 데이터를 전송 하는 것을 실습 해 보자.

- 먼저 form 태그를 수정해보자.

불필요해진 action과 method 속성은 삭제할 것이다. 우리는 axios로 대체할 것이기 때문이다.

```
{% if request.user.is_authenticated %}
  <form id="comment-create-form" data-article-id="{{article.pk}}">
    {% csrf_token %}
    {{ comment_form }}
    ...
```

- data-article-id 라고 속성명을 적어준 이유는 무엇일까?

[참고] data-* attributes

- 임의의 데이터를 HTML과 DOM사이에서 교환 할 수 있는 속성이다.

  사용 예시를 살펴보자.

```
<div data-my-id="my-data"></div>
<script>
  const myId = event.target.dataset.myId
</script>
```

- “data-” 속성은 “사용자 지정 데이터”라고 한다.

  이는 돔 조작을 위해서 html에서 제공해 주는 속성이다.
- HTML 요소에 data-my-id 라는 개별속성에 "my-data" 라는 값을 넣어 줬다.
  - 표준속성 (a, p, div)
- Javascript로 데이터 속성에 접근 하기 위해서는 아래와 같이 접근이 가능하다.

  ```
  const myId = event.target.dataset.myId;
  ```

https://developer.mozilla.org/ko/docs/Web/HTML/Global_attributes/data-*

- 예를 들어 `data-my-id` 라는 이름의 특성을 지정했다면,

  JavaScript에서는 `element.dataset.myId` 로 해당 속성을 접근할 수 있다.

즉, `dataset` 은 JavaScript에서 HTML 요소의 `data-*` 속성을 읽고 쓰기 위한 객체다.

- 그리고 axios로 동작시키기 위해서 해당 form요소를 선택해 준다.

스켈레톤 코드의 script태그 최 하단에 코드를 추가하자.

```
// 댓글 생성
const commentCreateForm = document.querySelector('#comment-create-form')
```

```
commentCreateForm.addEventListener('submit',function(event){
  event.preventDefault()


})
```

- form 요소에 이벤트 핸들러 작성 및 submit 이벤트 취소 했다.

  form 태그의 submit 이 작동하면 브라우저에서는 새로고침이 작동 되는 것이 기본 디폴트값 이다. 따라서 브라우저에서는 자동으로 작동하는 새로고침을 막기 위해 preventDefault() 함수를 추가했다.

- ~Form.addEventListener('submit', function (event) 의 의미는

  1. submit 이벤트가 발생하면,
  2. 서버로 댓글생성 요청을 보내는 콜백함수가 실행 되도록 하겠다.

- axios 요청 준비

```
const commentCreateForm = document.querySelector('#comment-create-form')
commentCreateForm.addEventListener('submit', function(event) {
  event.preventDefault()

  const articleId = event.target.dataset.articleId // 게시글아이디 가져오기
  const formData = new FormData(commentCreateForm) // 새로운 form data 객체생성
```

[참고] new 키워드로 객체생성

예를 들어 이런 HTML이 있다고 가정하고

```
<form id="commentCreateForm">
  <input name="username" value="alice">
  <textarea name="comment">Hello!</textarea>
</form>
```

JS에서 다음과 같이 작성하면

```
const commentCreateForm = document.getElementById('commentCreateForm');
const formData = new FormData(commentCreateForm);
```

이렇게 하면 formData 안에는 다음과 같은 데이터가 들어가게 된다.

"username": "alice"
"comment": "Hello!"

그리고

formData.get('username') 같은 식으로 값을 꺼내 사용이 가능하다.

- axios 요청보내기

```
axios({
      method: 'POST',
      url: `/articles/${articleId}/comments/`,
      headers: {
        'X-CSRFToken': csrf,
      },
      data: formData, // 폼 데이터도 같이 전송
    })              // 사용자가 입력한 내용도 같이 전송
```

- .then 구현
  - 요청에 대한 응답이 왔을 때 무엇을 해야 할까?
  - 요구사항에 있는대로 사용자가 댓글을 작성한 뒤, **성공적으로 서버에 저장되었을 때**

    클라이언트 화면에 해당 댓글을 **즉시 추가하는 코드를 작성해 보자.**
    - 댓글에 삭제 버튼을 추가 할 것이고

```
const deleteForm = document.createElement('form'); // DELETE 버튼을 만들기 위한 폼
deleteForm.classList.add('d-inline'); // 클래스 추가
deleteForm.id = 'delete-comment-form'; // id 속성 추가
deleteForm.dataset.articleId = articleId;
deleteForm.dataset.commentId = response.data.pk;
deleteForm.innerHTML = `<input type="submit" value="DELETE">`;
```

    - deleteForm.dataset.articleId = articleId;
      - `deleteForm.dataset.articleId`는 자바스크립트에서 설정하면,
      - HTML 태그에 자동으로 `data-article-id`라는 속성이 추가된다.

    - `response`: Axios로 서버에 요청을 보내고 받은 **전체 응답 객체**
    - `response.data`: Axios로 서버에 요청을 보내고 받은 응답의 body (보통 JSON)
    - `response.data.pk`: 그 데이터 안에 포함된 `pk` **라는 키(속성)의 값**
    - 즉, 응답받은 댓글의 pk가 된다.
    - 위 소스코드를 HTML로 표현 하자면 다음과 같다

```
<form class="d-inline" id="delete-comment-form" data-article-id="123" data-comment-id="42">
  <input type="submit" value="DELETE">
</form>
```

  .then 구문안에 이어서 작성하자.

```
    // 댓글 담을수 있는 요소 생성
    const li = document.createElement('li');
    li.textContent = `${response.data.username} - ${response.data.content}`;

    // 새 댓글을 추가후
    li.appendChild(deleteForm);
    commentContainer.appendChild(li);
```

```
        // 폼에 있는 댓글입력 값 지우기
        commentCreateForm.reset();
```

* 여기까지를 html로 보면 다음과 같다

```
<ul id="commentContainer">
  <li>
    최민호 - 이 영화 정말 재밌어요!
    <form class="d-inline" id="delete-comment-form" data-article-id="123" data-comment-id="42">
      <input type="submit" value="DELETE">
    </form>
  </li>
  <li>
    최민호2 - 나도 재밌게 보았어요!
    <form class="d-inline" id="delete-comment-form" data-article-id="123" data-comment-id="42">
      <input type="submit" value="DELETE">
    </form>
  </li>
</ul>
```

- 마지막으로 catch구문도 작성하자.

```
.catch(function(error) {
      console.error(error);
    });
```

- 완성된 폼은 다음과 같다

```
  // 댓글 생성
  const commentCreateForm = document.querySelector('#comment-create-form')
  commentCreateForm.addEventListener('submit', function(event) {
    event.preventDefault()
    const articleId = event.target.dataset.articleId // 게시글아이디 가져오기
    const formData = new FormData(commentCreateForm)
    axios({
      method: 'POST',
      url: `/articles/${articleId}/comments/`,
      headers: {
        'X-CSRFToken': csrf,
      },
      data: formData, // 폼 데이터도 같이 전송
    }).then(function(response) {
      // 삭제버튼 포함시키기
      const deleteForm = document.createElement('form');
      deleteForm.classList.add('d-inline');
      deleteForm.id = 'delete-comment-form';
      deleteForm.dataset.articleId = articleId;
      deleteForm.dataset.commentId = response.data.pk;
      deleteForm.innerHTML = `<input type="submit" value="DELETE">`;

      // 댓글 담을수 있는 요소 생성
      const li = document.createElement('li');
```

```
        li.textContent = `${response.data.username} - ${response.data.content}`;
        // 새 댓글을 추가후
        li.appendChild(deleteForm);
        commentContainer.appendChild(li);
        // 폼에 있는 댓글입력 값 지우기
        commentCreateForm.reset();
      }).catch(function(error) {
        console.error(error);
      });
    });



  </script>
{% endblock content %}
```