JavaScript

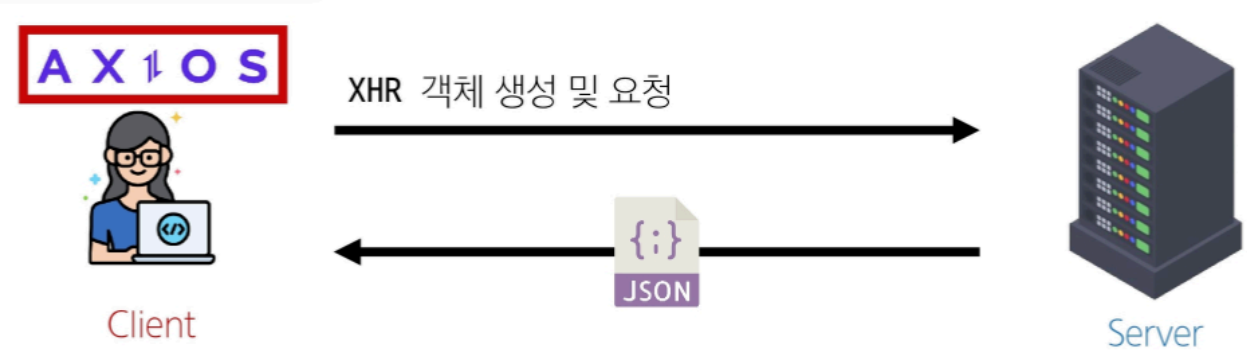
Ajax with Django

Ajax와 서버

Ajax (Asynchronous JavaScript and XML)

비동기적인 웹 애플리케이션 개발에 사용하는 기술

Ajax를 활용한 클라이언트 서버 간 동작



- XHR 객체 생성 및 요청 → 응답 데이터 생성 → JSON 데이터 응답→ Promise 객체 데이터를 활용해 DOM 조작 (웹 페이지의 일부분 만을 다시 로딩)

Ajax with follow

비동기 팔로우 구현

사전 준비

1. M:N까지 진행한 Django 프로젝트 준비
2. 가상 환경 생성, 활성화 및 패키지 설치

Ajax 적용

1. 프로필 페이지에 axios CDN 작성

```
<!-- accounts/profile.html -->

    <script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
    <script>
    </script>
</body>
</html>
```

2. form요소 선택을 위해 id 속성 지정 및 선택
   - action과 method 속성은 삭제
   - 요청은 axios로 대체되기 때문

```
<!-- accounts/profile.html -->
<form id="follow-form">
    ...
</form>
```

```
<!-- accounts/profile.html-->
const formTag = document.querySelector('#follow-form')
```

3. form 요소에 이벤트 핸들러 할당
   - submit 이벤트의 기본 동작 취소하기

```
<!-- accounts/profile.html -->
formTag.addEventListener('submit', function (event) {
    event.preventDefault()
})
```

4. axios 요청 코드 작성
   1. url 작성에 필요한 user pk는 어떻게 작성할까?
   2. csrftoken은 어떻게 보내야 할까?

```
<!-- accounts/profile.html -->
formTag.addEventListener('submit', function (event) {
    event.preventDefault()
    axios({
        method: 'post',
        url: `/accounts/${}/follow/`,
    })
})
```

5. url에 작성할 user pk 가져오기 (HTML ⇒ JavaScript)

```html
<!-- accounts/profile.html -->
<form id="follow-form" data-user-id="{{ person.pk }}">
    ...
</form>
```

```html
<!-- accounts/profile.html -->
formTag.addEventListener('submit', function (event) {
    event.preventDefault()
    const userId = event.currentTarget.dataset.userId
    const userId this.dataset.userId
    const userId = formTag.dataset.userId
```

'data-*' 속성

- 사용자 지정 데이터 특성을 만들어 임의의 데이터를 HTML과 DOM 사이에서 교환할 수 있는 방법

data-* 사용 예시

```html
<div data-my-id="my-data"></div>

<script>
    const myId = event.target.dataset.myId
</script>
```

- 모든 사용자 지정 데이터는 JavaScript에서 dataset 속성을 통해 접근
- 주의사항
    1. 대소문자 여부에 상관 없이 'xml' 문자로 시작 불가
    2. 세미콜론 포함 불가
    3. 대문자 포함 불가

6. 요청 url 작성 마무리

```html
<!-- accounts/profile.html -->
formTag.addEventListener('submit', function (event) {
    event.preventDefault()

    const userId = event.currentTarget.dataset.userId
    axios({
        method: 'post',
        url: `/accounts/${userId}/follow/`,
    })
})
```

7. 문서상 input hidden 타입으로 존재하는 csrf token 데이터를 이제는 axios로 전송해야 함
    - csrf값을 가진 input 요소를 직접 선택 후 axios에 작성하기
    - Django csrf 문서

```
<!-- accounts/profile.html -->
const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value
formTag.addEventListener('submit', function (event) {
    event.preventDefault()
    const userId = event.currentTarget.dataset.userId
    axios({
        method: "post",
        url: `/accounts/${userId}/follow/`,
        headers: {'X-CSRFToken': csrftoken,},
    })
})
```

- 팔로우 버튼을 토글하기 위해서는 현재 팔로우 상태인지 언팔로우 상태인지에 대한 상태 확인이 필요

▶ Django의 view 함수에서 팔로우 여부를 파악할 수 있는 변수를 추가로 생성해 JSON 타입으로 응답하기

8. 팔로우 상태 여부를 JavaScript에게 전달할 데이터 작성

- 응답은 더 이상 HTML 문서가 아닌 JSON 데이터로 응답하도록 변경

```
#accounts/views.py
from django.http import JsonResponse

@login_required
def follow(request, user_pk):
    User = get_user_model()
    person = User.objects.get(pk=user_pk)
    if person != request.user:
        if person.followers.filter(pk=request.user.pk).exists():
            person.followers.remove(request.user)
            is_followed = False
        else:
            person.followers.add(request.user)
            is_followed = True
        context = {
            'is_followed': is_followed,
        }
        return JsonResponse(context)
    return redirect('accounts:profile', person.username)
```

9. 팔로우 요청 후 Django 서버로부터 받은 응답 데이터 확인하기

```
<!-- accounts/profile.html -->

formTag.addEventListener('submit', function (event) {
    event.preventDefault()
    const userId = event.currentTarget.dataset.userId
    axios({
        method : 'post',
        url: `/accounts/${userId}/follow/`,
        headers: {'X-CSRFToken': csrftoken,},
    })
    .then((response) => {
        console.log(response)
        console.log(response.data)
    })
```

```
})
```

10. 응답 데이터 is_followed에 따라 팔로우 버튼 조작하기

```
<!-- accounts/profile.html -->

axios({
    method : 'post',
    url: `/accounts/${userId}/follow/`,
    headers: {'X-CSRFToken': csrftoken,},
})
.then((response) => {
    const isFollowed = response.data.is_followed
    const followBtn = document.querySelector('input[type=submit]')
    if (isFollowed === true) {
        followBtn.value = 'Unfollow'
    } else {
        followBtn.value = 'Follow'
    }
})
```

11. 개발자도구 - 네트워크 탭에서 클라이언트와 서버간 XHR 객체를 주고 받는 것을 확인하기

12. 팔로잉 수와 팔로워 수 비동기 적용

- 해당 요소를 선택할 수 있도록 span 태그와 id 속성 작성

```
<!-- accounts/profile.html -->
<div>
    팔로잉 : <span id="followings-count">{{ person.followings.all|length }}</span>
    팔로워 : <span id="followers-count">{{ person.followers.all|length }}</span>
</div>
```

- 각 span 태그를 선택

```
<!-- accounts/profile.html -->
.then((response) => {
    const followingsCountTag = document.querySelector('#followings-count')
    const followersCountTag = document.querySelector('#followers-count')
})
```

- Django view 함수에서 팔로워, 팔로잉 인원 수 연산을 진행하여 결과를 응답 데이터로 전달

```python
#accounts/views.py
@login_required
def follow(request, user_pk):
    ...
        context = {
            'is_followed': is_followed,
            'followings_count': person.followings.count(),
            'followers_count': person.followers.count(),
        }
        return JsonResponse(context)
    return redirect('accounts:profile', person.username)
```

- 응답 데이터를 받아 각 태그의 인원수 값 변경에 활용

```javascript
<!-- accounts/profile.html -->
.then((response) => {
    const followingsCountTag = document.querySelector('#followings-count')
    const followersCountTag = document.querySelector('#followers-count')

    followingsCountTag.textContent = response.data.followings_count
    followersCountTag.textContent = response.data.followers_count
})
```

Ajax with likes

비동기 좋아요 구현

- 전반적인 Ajax 적용은 팔로우 구현 과정과 모두 동일
- 단, 팔로우와 달리 좋아요 버튼은 한 페이지에 여러 개가 존재
- 버블링 사용하여 공통 조상 하나에 이벤트 할당
- 한 요소에 이벤트가 발생하면, 이 요소에 할당된 핸들러가 동작하고, 이어서 부모 요소의 핸들러가 동작하는 현상
- 가장 최상단의 조상 요소(document)를 만날 때까지 이 과정이 반복되면서 요소 각각에 할당된 핸들러가 동작

Ajax 적용

1. 모든 좋아요 form 요소를 포함하는 최상위 요소 작성

```javascript
<!-- articles/index.html -->
<article class="article-container">
    {% for article in articles %}
    ...
    {% endfor %}
</article>
```

1. 최상위 요소 선택
2. 이벤트 핸들러 할당
3. 하위 요소들의 submit 이벤트를 감지하고 submit 기본 이벤트를 취소

```
<!-- articles/index.html -->
const articleContainer = document.querySelector('.article-container')
articleContainer.addEventListener('submit', function (event) {
    event.preventDefault()
})
```

2. axios 코드 작성

▶ url 작성에 필요한 article pk는 어떻게 작성해야 할까?

- 각 좋아요 form에 article.pk를 부여 후 HTML의 article.pk 값을 JavaScript에서 참조하기

```
<!-- articles.index.html-->
<form data-article-id="{{ article.pk }}">
    ...
</form>
```

```
// articles.index.html
const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value

articleContainer.addEventListener('submit', function (event) {
    event.preventDefault()

    const articleId = event.target.dataset.article.Id

    axios({
        method: 'post',
        url: `/articles/${articleId}/likes/`,
        headers: {'X-CSRFToken': csrftoken,},
    })
})
```

3. 좋아요 버튼 토글 기능 추가

- 좋아요 버튼을 토글하기 위해서는 현재 사용자가 좋아요를 누른 상태인지 아닌지에 대한 상태 확인 필요
- Django의 view 함수에서 좋아요 여부를 파악할 수 있는 변수 추가 생성
- JSON 타입으로 응답하기

1. 좋아요 상태 여부를 JavaScript에게 전달할 데이터 작성 및 JSON 데이터 응답

```
#articles/views.py
from django.http import JsonResponse

@login_required
def likes(request, article_pk):
    article = Article.objects.get(pk=article_pk)
    if request.user in article.like_users.all():
```

```
        article.like_users.remove(request.user)
        is_likedn = False
    else:
        article.like_users.add(request.user)
        is_liked = True
    context = {
        'is_liked': is_liked,
    }
    return JsonResponse(context)
```

2. 응답 데이터 is_liked를 받아 is_liked 변수에 할당

```
// articles/index.html

axios({
    method: 'post',
    url: `/articles/${articleId}/likes/`,
    headers: {'X-CSRFToken': csrftoken},
})
    .then((response) => {
    console.log(response)
    const isLiked = response.data.is_liked
})
    .catch((error) => {
    console.log(error)
})
```

3. isLiked에 따라 좋아요 버튼을 토글하기

- 그런데 어떤 게시글의 좋아요 버튼을 선택했는지 구분 필요
- 문자와 article의 pk값을 혼합하여 각 버튼에 id 속성값을 설정

```
<!-- articles/index.html -->
{% if request.user in article.like_users.all %}
    <input type="submit" value="좋아요 취소" id="like-{{ article.pk }}">
{% else %}
    <input type="submit" value="좋아요" id="like-{{ article.pk }}">
{% endif %}
```

- 각 좋아요 버튼을 선택 후 isLiked에 따라 버튼을 토글

```
// articles/index.html

axios({
    method: 'post',
    url: `/articles/${articleId}/likes/`,
    headers: {'X-CSRFToken': csrftoken},
})
    .then((response) => {
    console.log(response)
    console.log(response.data)
    const isLiked = response.data.is_liked
```

```
    const likeBtn = document.querySelector(`#like-${articleId}`)
    if (isLiked === true) {
        likeBtn.value = '좋아요 취소'
    } else {
        likeBtn.value = '좋아요'
    }
})
    .catch((error) => {
    console.log(error)
})
```

버블링을 활용하지 않은 경우

1. querySelectorAll()을 사용해 전체 좋아요 버튼을 선택
   - querySelectorAll() 선택을 위한 class 적용

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

{% for article in articles %}
    ...
    <form class="like-forms" data-article-id="{{ article.pk }}">
        {% csrf_token %}
        {% if request.user in article.like_users.all %}
        {% comment %} 17. 각 좋아요 버튼을 구별할 수 있는 id 속성 추가 {% endcomment %}
        <input type="submit" value="좋아요 취소" id="like-{{ article.pk }}">
        {% else %}
        <input type="submit" value="좋아요" id="like-{{ article.pk }}">
        {% endif %}
    </form>
    <hr>
{% endfor %}
```

2. forEach()를 사용해 전체 좋아요 버튼을 순회하면서 진행

```
const formTags = document.querySelectorAll('.like-forms')
const csrftoken = document.querySelector('[name=csrfmiddlewaretoken]').value

formTags.forEach((formTag)=> {
    formTag.addEventListener('submit', function (event) {
        event.preventDefault()

        const articleID = formTag.dataset.articleID

        axios({
            method: 'post',
            url: `/articles/${articleId}/likes/`,
            headers: {'X-CSRFToken': csrftoken},
        })
            .then((response) => {
            const isLiked = response.data.is_liked
            const likeBtn = document.querySelector(`#like-${articleId}`)
            if (isLiked === true) {
                likeBtn.value = '좋아요 취소'
```

```
            } else {
                likeBtn.value = '좋아요'
            }
        })
    })
})
```