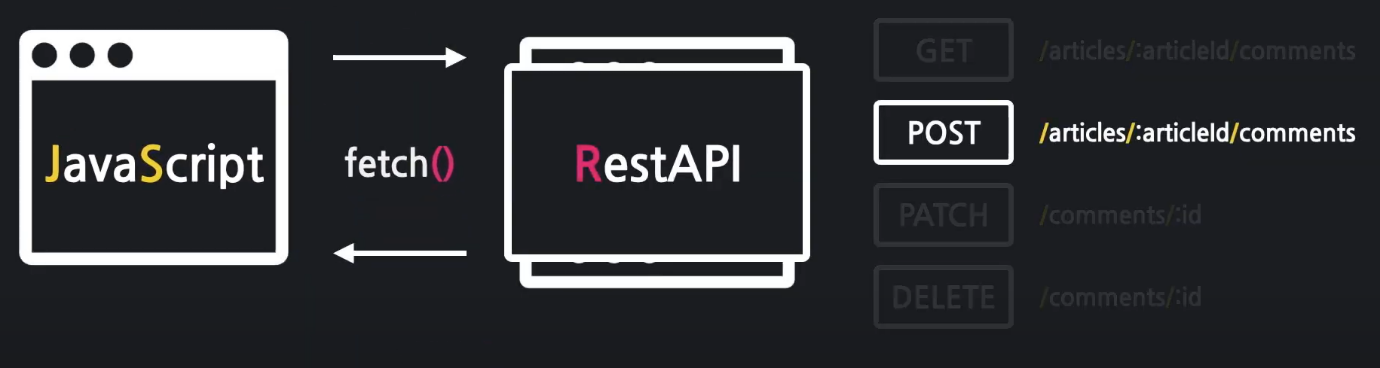
댓글 등록,수정 with 자바스크립트(JS)





**댓글 생성 뷰 페이지**

**\_new.mustache**

* 부트스트랩 card이용.
* 히든 인풋 :  
  댓글은 게시글에 포함되어야하니까 article\_id의 값을 hidden로 가지고 있어야한다.

<div class="container">

<div class="card m-2" id="comments-new">

<div class="card-body">

<!-- 댓글 작성 폼-->

<form>

<!-- 닉네임 입력 -->

<div class="mb-3">

<label class="form-label">닉네임</label>

<input type="text" class="form-control form-control-sm" id="new-comment-nickname">

</div>

<!-- 댓글 본문 입력 -->

<div class="mb-3">

<label class="form-label">댓글 내용</label>

<textarea type="text" class="form-control form-control-sm" rows="3" id="new-comment-body"></textarea>

</div>

<!-- 히든 인풋 -->

<!-- 댓글은 게시글에 포함되어야하니까 article\_id의 값을 hidden로 가지고 있어야한다. -->

{{#article}}

<input type="hidden" id="new-comment-article-id" value="{{id}}">

{{/article}}

<!-- 전송 버튼 -->

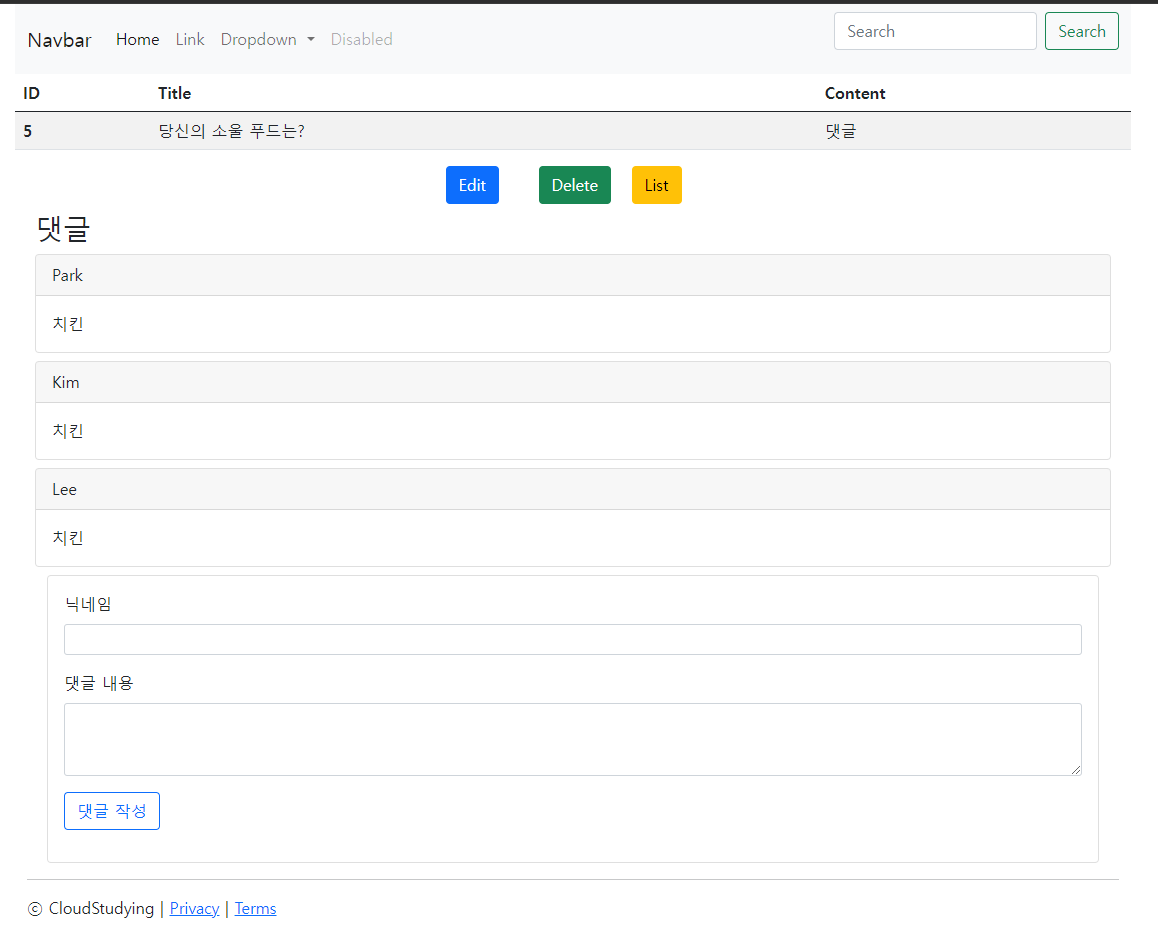
<button type="button" class="btn btn-outline-primary btn-m" id="comment-create-btn">댓글 작성</button>

</form>

</div>

</div>

</div>



**버튼 요소 선택**

\_new.mustache안에 script 열어서  
javascript코드 작성.

버튼을 클릭했을 때 댓글이 작성될 수 있도록 연결할 수 있게 자바스크립트를 이용해보자.

* document.querySelector를 이용 :  
  해당 요소를 선택해서 가져옴.

<script>

{

// 댓글 작성 버튼 변수화(id가 comment-create-btn인 대상)

const commentCreateBtn = document.querySelector("#comment-create-btn");

}

</script>

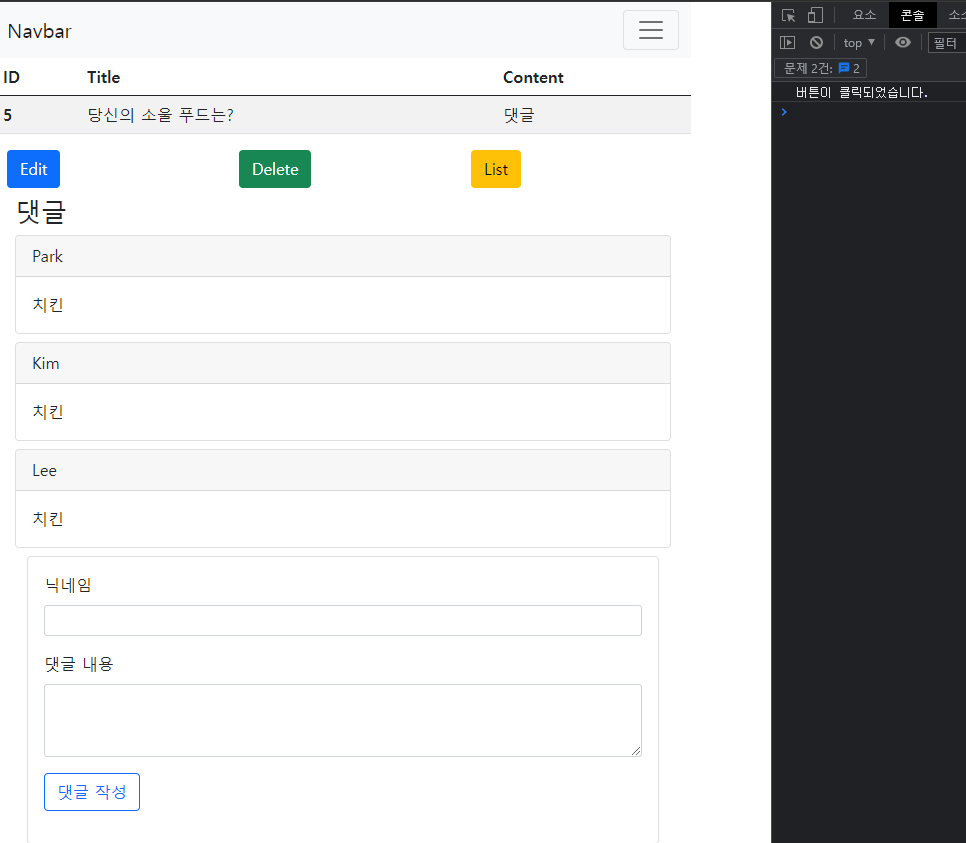
**클릭 이벤트 감지**

// 버튼 클릭 이벤트 감지!

commentCreateBtn.addEventListener("click", function() {

console.log("버튼이 클릭되었습니다.");

});



**새 댓글 JS 객체 생성**

* document.querySelector("id").value :  
  선택 요소의 값 가져오기

댓글에 닉네임과 댓글 내용을 객체로 만들어서 console로 출력해보기.

// 버튼 클릭 이벤트 감지!

commentCreateBtn.addEventListener("click", function() {

// 새 댓글 객체 생성

const comment = {

nickname: document.querySelector("#new-comment-nickname").value,

body: document.querySelector("#new-comment-body").value,

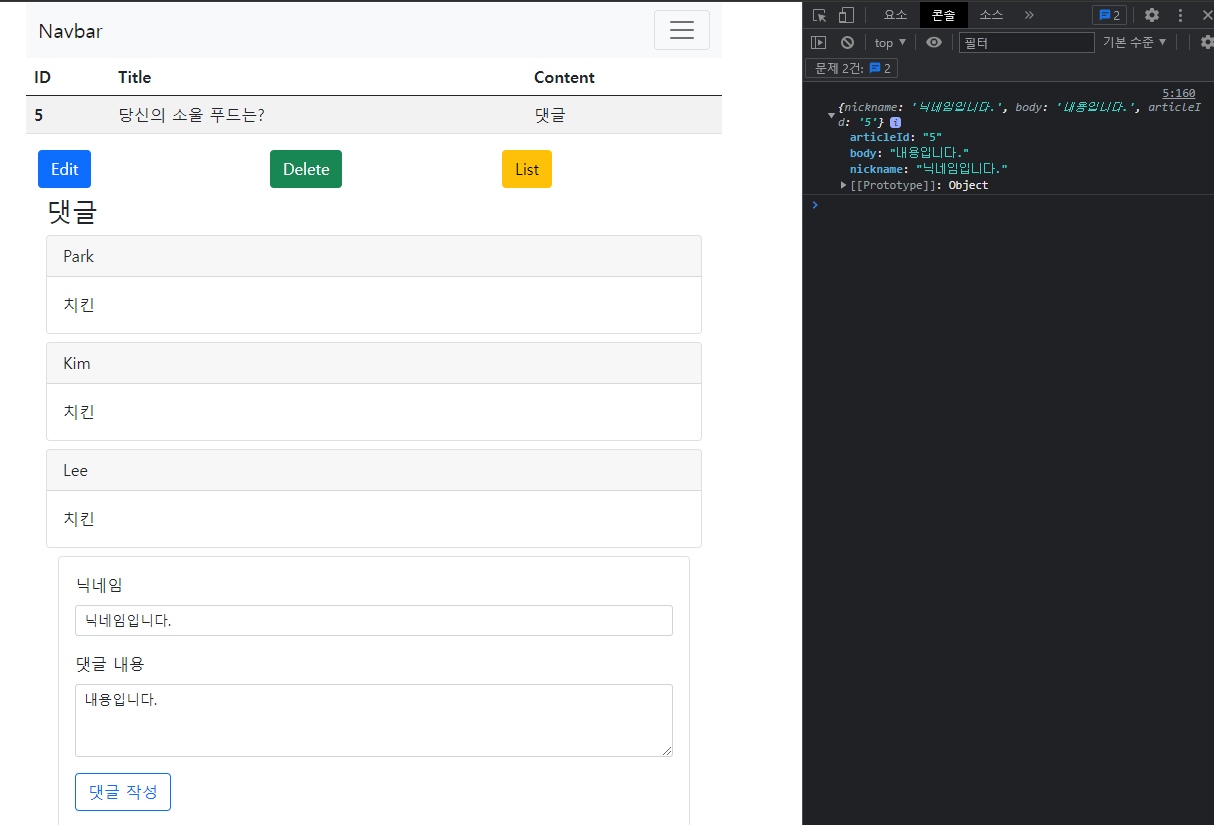
articleId: document.querySelector("#new-comment-article-id").value

};

// 댓글 객체 출력

console.log(comment);

});



**fetch( )**

fetch() 메소드는 JavaScript에서 HTTP 요청을 보내기 위해 사용되는 API입니다. 이 메소드를 사용하여 서버로부터 데이터를 가져올 수 있습니다.

fetch() 메소드는 다음과 같은 구문을 가지고 있습니다:

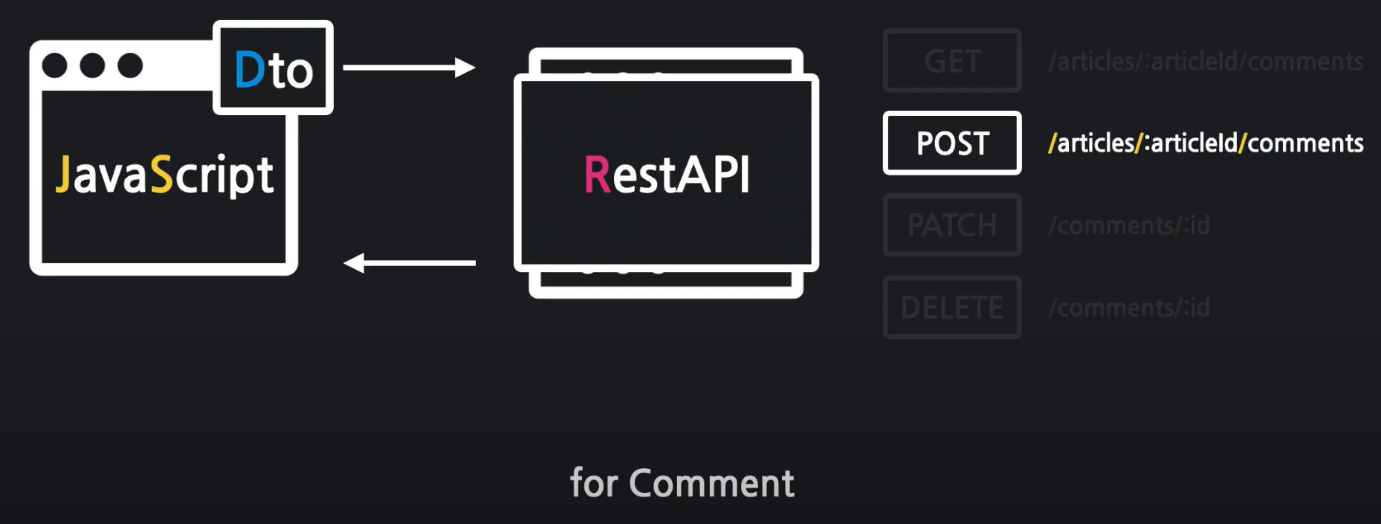
fetch(url, options)

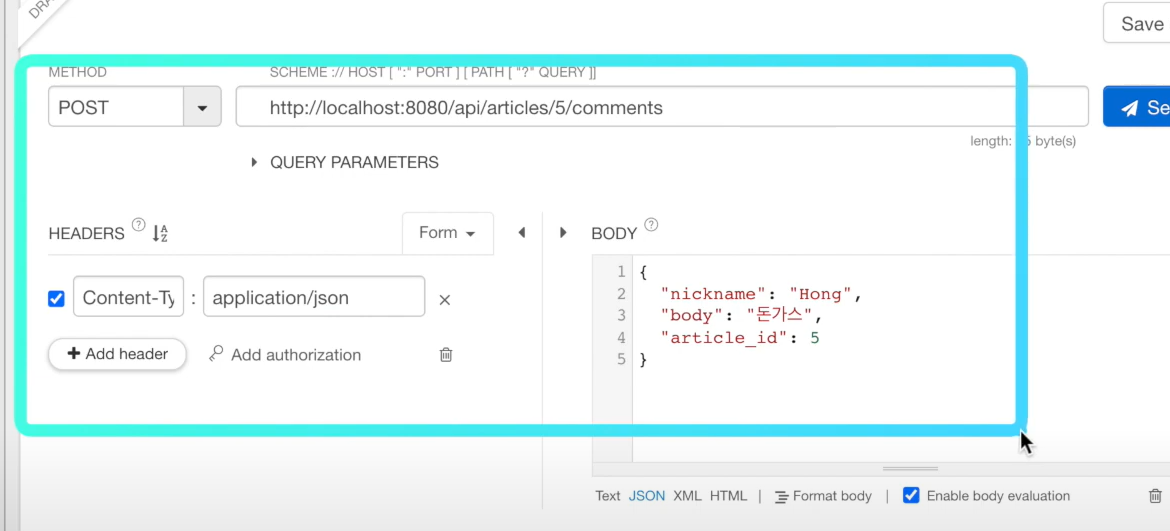
여기서 url은 가져올 데이터가 위치한 URL을 나타내고, options는 요청을 구성하는 다양한 옵션들을 포함하는 객체입니다. options 객체는 생략 가능하며, 기본값은 GET 요청을 전송하고, CORS 제한을 허용하는 것입니다.

**options 매개변수**

* method: HTTP 요청 메소드를 지정합니다. 기본값은 GET입니다. 다른 유효한 값으로는 POST, PUT, DELETE 등이 있습니다.
* headers: HTTP 요청 헤더를 지정합니다. 객체 형태로 전달되며, 각 헤더는 속성-값 쌍으로 표시됩니다.
* body: HTTP 요청 바디를 지정합니다. 문자열, FormData, Blob 등의 유형이 될 수 있습니다.
* mode: 요청을 보낼 때 CORS (Cross-Origin Resource Sharing)를 처리하는 방법을 지정합니다. 기본값은 "cors"입니다. 다른 유효한 값으로는 "no-cors", "same-origin", "navigate" 등이 있습니다.
* credentials: 요청에 포함될 자격 증명을 지정합니다. 기본값은 "same-origin"입니다. 다른 유효한 값으로는 "include", "omit"이 있습니다.
* cache: 요청 결과를 캐시할지 여부를 지정합니다. 기본값은 "default"입니다. 다른 유효한 값으로는 "no-store", "reload", "no-cache", "force-cache", "only-if-cached"가 있습니다.
* redirect: 서버가 리디렉션을 지시할 경우 처리 방법을 지정합니다. 기본값은 "follow"입니다. 다른 유효한 값으로는 "manual", "error"가 있습니다.
* referrer: Referer 헤더에 사용될 URL을 지정합니다.
* referrerPolicy: Referer 헤더 처리 정책을 지정합니다. 기본값은 "no-referrer-when-downgrade"입니다. 다른 유효한 값으로는 "no-referrer", "origin", "origin-when-cross-origin", "unsafe-url"이 있습니다.

**JS로 REST API 호출(fetch( ))**



Taclend API로 요청을 보내듯이  
JS로도 요청을 보낼 수 있다.  


// fetch() - Talend API요청을 JavaScript로 보내준다

const url = "/api/articles/" + comment.articleId + "/comments";

fetch(url, {

method: "post", // POST 요청

body: JSON.stringify(comment), // comment JS 객체를 JSON으로 변경하여 보냄

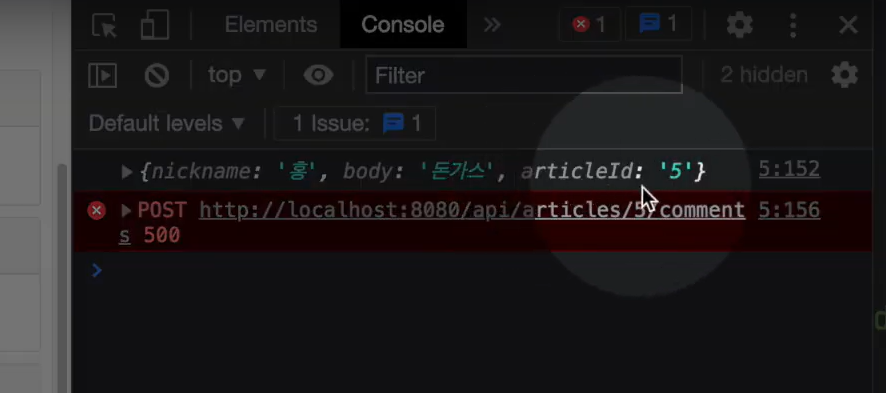
headers: {

"Content-Type": "application/json"

}

});

**500에러**



public class CommentDto {

private Long id;

@JsonProperty("article\_id")

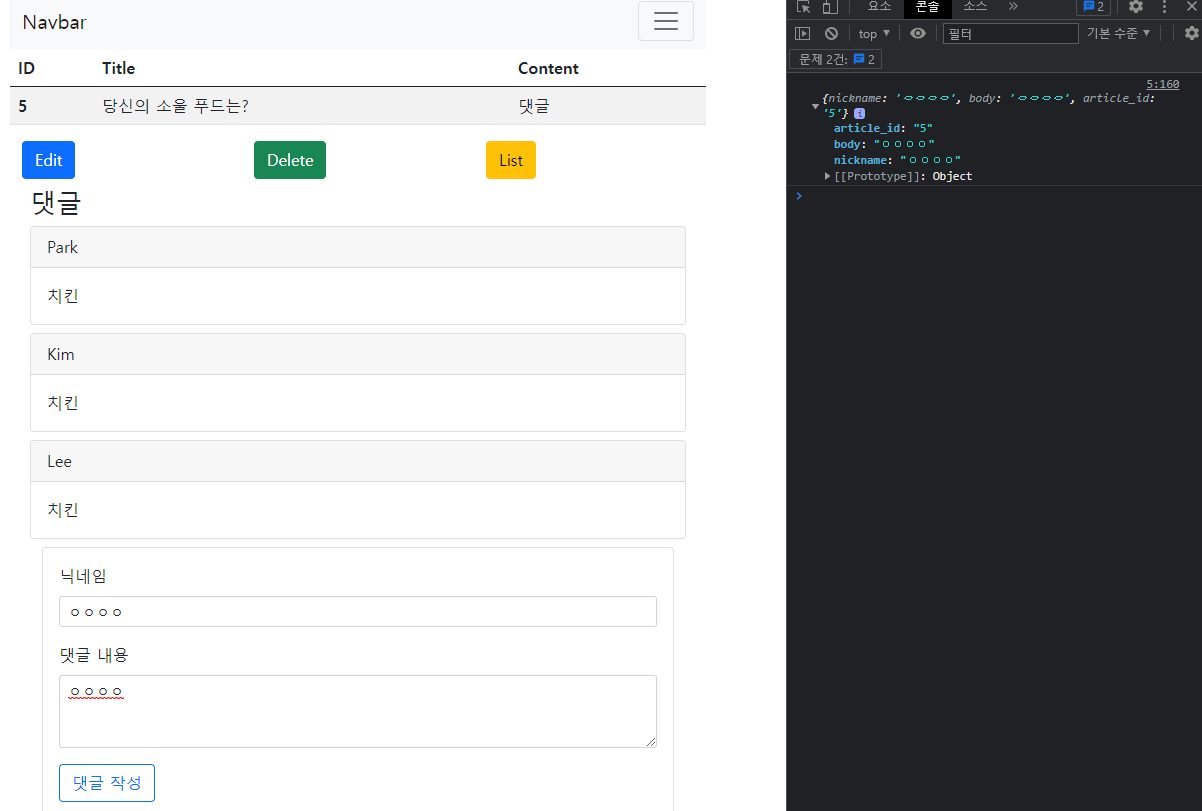
private Long articleId;

...

}

CommentDto를 확인해보면  
articleId를 JSON으로 받아올때 article\_id로 받아오기로 했기 때문에 에러가 발생한다.

articleId -> article\_id 변경하자~!

  
✨ 성공 ✨

**fetch( ) 응답 처리**

fetch 응답이 끝나고 나서 (then) 다음에 할 처리추가.

* 정상응답과 실패할시에 나타낼 메시지 추가.
* 페이지 리로드

fetch(url, {

method: "post", // POST 요청

body: JSON.stringify(comment), // comment JS 객체를 JSON으로 변경하여 보냄

headers: {

"Content-Type": "application/json"

}

// 응답 후 다음 처리 추가

}).then(response => {

// http 응답 코드에 따른 메시지 출력

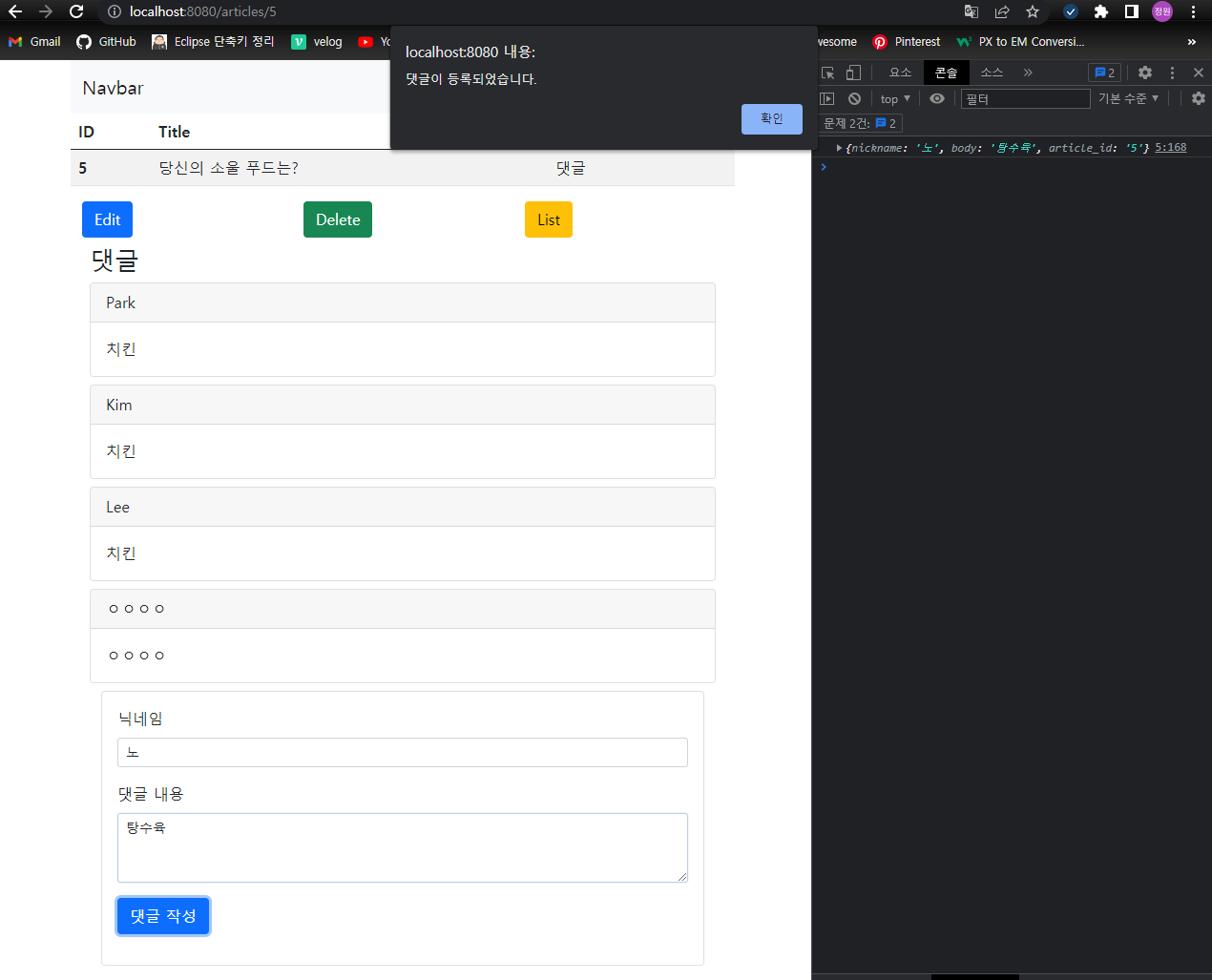
const msg = (response.ok) ? "댓글이 등록되었습니다." : "댓글 등록 실패!!";

alert(msg);

// 현재 페이지 새로고침

window.location.reload();

});



**댓글 수정 with 자바스크립트**



\_list.mustache에 수정 버튼 추가.  
수정 버튼 클릭 시 MODAL에서 수정할 수 있도록  
부트스트랩 MODAL 이용.  


**MODAL 추가**

\_list.mustache에  
수정 버튼트리거와 modal 추가.

<!-- Button trigger modal -->

<button type="button"

class="btn btn-sm btn-outline-success m-2 "

data-bs-toggle="modal"

data-bs-target="#comment-edit-modal">수정</button>

<!-- Modal -->

<div class="modal fade" id="comment-edit-modal" tabindex="-1">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">댓글 수정</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

...

</div>

</div>

</div>

</div>

**수정 폼 삽입**

MODAL 안에 댓글 수정 form 만들기

<div class="modal-body">

<!-- 댓글 수정 폼-->

<form>

<!-- 닉네임 입력 -->

<div class="mb-3">

<label class="form-label">닉네임</label>

<input type="text" class="form-control form-control-sm" id="edit-comment-nickname">

</div>

<!-- 댓글 본문 입력 -->

<div class="mb-3">

<label class="form-label">댓글 내용</label>

<textarea type="text" class="form-control form-control-sm" rows="3" id="edit-comment-body"></textarea>

</div>

<!-- 히든 인풋 -->

<input type="hidden" id="edit-comment-id">

<input type="hidden" id="edit-comment-article-id">

<!-- 전송 버튼 -->

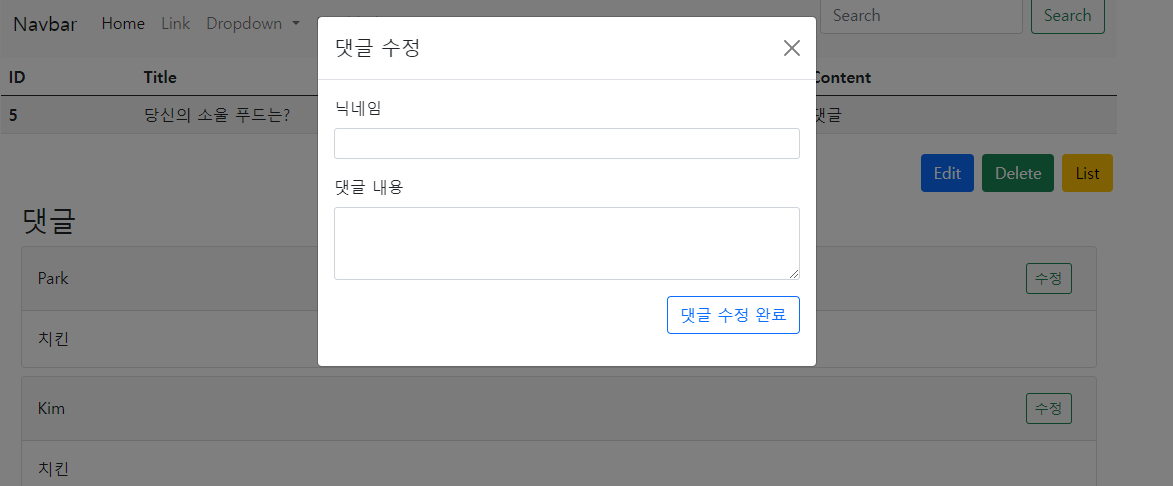
<div class="d-flex justify-content-end">

<button type="button" class="btn btn-outline-primary btn-m" id="comment-update-btn">댓글 수정 완료</button>

</div>

</form>

</div>



**트리거 데이터 전달**

* 수정버튼 클릭 시 전달할 데이터를 추가.

<!-- Button trigger modal -->

<button type="button"

class="btn btn-sm btn-outline-success m-2 "

data-bs-toggle="modal"

data-bs-target="#comment-edit-modal"

data-bs-id="{{id}}"

data-bs-nickname="{{nickname}}"

data-bs-body="{{body}}"

data-bs-article-id="{{articleId}}"

>수정</button>

* 위의 데이터 모달에서 가져올 수 있도록 script코드 추가

1. 이벤트를 걸 모달 요소 선택
2. 선택한 요소에 이벤트 걸기
3. 트리거 버튼 선택 :  
   모달을 열기위해 눌렀던 수정 버튼.
4. 수정 버튼에 들어있는 데이터 가져와서 변수화
5. 모달 폼 요소를 선택하고 value를 수정 버튼에서 가져온 데이터로 변경.

<!-- 모달 이벤트 처리 -->

<script>

{

// 모달 요소 선택

const commentEditModal = document.querySelector("#comment-edit-modal");

// 모달 이벤트 감지

commentEditModal.addEventListener("show.bs.modal", function(e) {

// 트리거 버튼 선택

const triggerBtn = e.relatedTarget;

// 데이터 가져오기

const id = triggerBtn.getAttribute("data-bs-id");

const nickname = triggerBtn.getAttribute("data-bs-nickname");

const body = triggerBtn.getAttribute("data-bs-body");

const articleId = triggerBtn.getAttribute("data-bs-article-id");

// 데이터를 반영

document.querySelector("#edit-comment-nickname").value = nickname;

document.querySelector("#edit-comment-body").value = body;

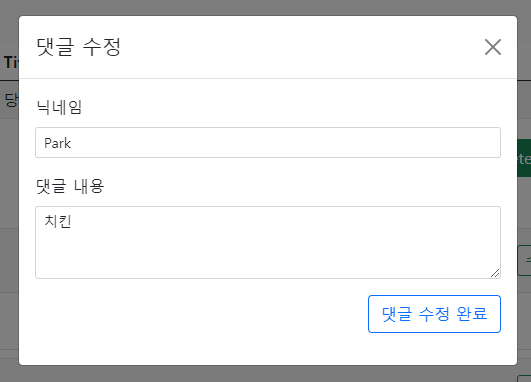
document.querySelector("#edit-comment-id").value = id;

document.querySelector("#edit-comment-article-id").value = articleId;

});

}

</script>



**relatedTarget**

relatedTarget는 JavaScript의 이벤트의 속성(property) 중 하나로, 이벤트의 보조 대상에 관한 정보를 제공합니다.

예를 들어, mouseover 이벤트는 마우스 포인터가 요소에 진입할 때 트리거됩니다. 이 경우, relatedTarget 속성은 일반적으로 마우스 포인터가 직전에 위치했던 요소인 이전 대상을 나타내게 됩니다.

**REST API 호출로 수정**

**부트스트랩에 modal에 데이터 전달하는 것을 이용해보자.**



\_list.mustache script에 수정 코드 추가.

댓글 수정 완료 버튼 클릭 시 수정된 데이터 보내고 반영하기.

{

// 수정 완료 버튼 가져오기

const commentUpdateBtn = document.querySelector("#comment-update-btn");

// 클릭 이벤트 감지 및 처리

commentUpdateBtn.addEventListener("click", function() {

// 수정 댓글 객체 생성

const comment = {

id: document.querySelector("#edit-comment-id").value,

nickname: document.querySelector("#edit-comment-nickname").value,

body: document.querySelector("#edit-comment-body").value,

article\_id: document.querySelector("#edit-comment-article-id").value

};

// 수정 REST API 호출 - fetch( ) 사용

const url = "/api/comments/" + comment.id;

fetch(url, {

method: "PATCH", // PATCH 요청

body: JSON.stringify(comment), // 수정된 댓글 객체를 JSON으로 전달

headers: { // body에 담겨진 데이터의 타입 알려주기

"Content-Type": "application/json"

}

}).then(response => {

// http 응답 코드에 따른 메시지 출력

const msg = (response.ok) ? "댓글이 수정 되었습니다." : "댓글 수정 실패!!";

alert(msg);

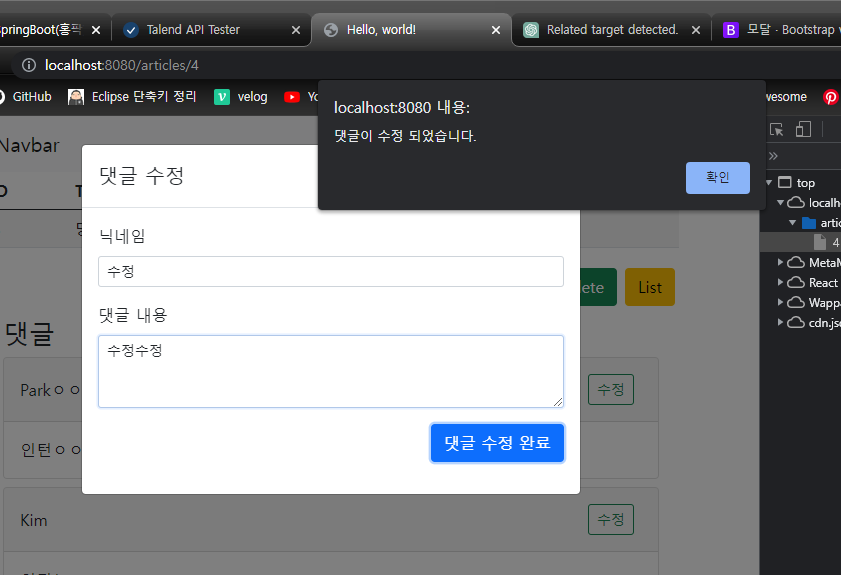
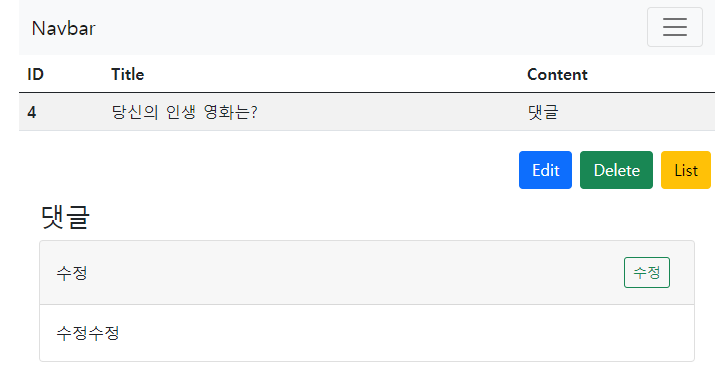
// 현재 페이지를 새로고침

window.location.reload();

});

});

}

  
✨ 수정 완료 ✨  


**전체코드**

**\_list.mustache**

<div id="comments-list">

{{#commentDtos}}

<div class="card m-2" id="comments-{{id}}">

<div class="card-header d-flex justify-content-between align-items-center">

{{nickname}}

<div>

<!-- Button trigger modal -->

<button type="button"

class="btn btn-sm btn-outline-success m-2 "

data-bs-toggle="modal"

data-bs-target="#comment-edit-modal"

data-bs-id="{{id}}"

data-bs-nickname="{{nickname}}"

data-bs-body="{{body}}"

data-bs-article-id="{{articleId}}"

>수정</button>

<!-- 댓글 삭제 버튼 -->

<button type="button"

class="btn btn-sm btn-outline-danger m-2 comment-delete-btn"

data-comment-id="{{id}}"

>삭제</button>

</div>

</div>

<div class="card-body">

{{body}}

</div>

</div>

{{/commentDtos}}

</div>

<!-- Modal -->

<div class="modal fade" id="comment-edit-modal" tabindex="-1">

<div class="modal-dialog">

<div class="modal-content">

<div class="modal-header">

<h5 class="modal-title" id="exampleModalLabel">댓글 수정</h5>

<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>

</div>

<div class="modal-body">

<!-- 댓글 수정 폼-->

<form>

<!-- 닉네임 입력 -->

<div class="mb-3">

<label class="form-label">닉네임</label>

<input type="text" class="form-control form-control-sm" id="edit-comment-nickname">

</div>

<!-- 댓글 본문 입력 -->

<div class="mb-3">

<label class="form-label">댓글 내용</label>

<textarea type="text" class="form-control form-control-sm" rows="3" id="edit-comment-body"></textarea>

</div>

<!-- 히든 인풋 -->

<input type="hidden" id="edit-comment-id">

<input type="hidden" id="edit-comment-article-id">

<!-- 전송 버튼 -->

<div class="d-flex justify-content-end">

<button type="button" class="btn btn-outline-primary btn-m" id="comment-update-btn">댓글 수정 완료</button>

</div>

</form>

</div>

</div>

</div>

</div>

<!-- 모달 이벤트 처리 -->

<script>

{

// 모달 요소 선택

const commentEditModal = document.querySelector("#comment-edit-modal");

// 모달 이벤트 감지

commentEditModal.addEventListener("show.bs.modal", function(e) {

// 트리거 버튼 선택

const triggerBtn = e.relatedTarget;

// 데이터 가져오기

const id = triggerBtn.getAttribute("data-bs-id");

const nickname = triggerBtn.getAttribute("data-bs-nickname");

const body = triggerBtn.getAttribute("data-bs-body");

const articleId = triggerBtn.getAttribute("data-bs-article-id");

console.log("id");

// 데이터를 반영

document.querySelector("#edit-comment-nickname").value = nickname;

document.querySelector("#edit-comment-body").value = body;

document.querySelector("#edit-comment-id").value = id;

document.querySelector("#edit-comment-article-id").value = articleId;

});

}

{

// 수정 완료 버튼 가져오기

const commentUpdateBtn = document.querySelector("#comment-update-btn");

// 클릭 이벤트 감지 및 처리

commentUpdateBtn.addEventListener("click", function() {

// 수정 댓글 객체 생성

const comment = {

id: document.querySelector("#edit-comment-id").value,

nickname: document.querySelector("#edit-comment-nickname").value,

body: document.querySelector("#edit-comment-body").value,

article\_id: document.querySelector("#edit-comment-article-id").value

};

// 수정 REST API 호출 - fetch( ) 사용

const url = "/api/comments/" + comment.id;

fetch(url, {

method: "PATCH", // PATCH 요청

body: JSON.stringify(comment), // 수정된 댓글 객체를 JSON으로 전달

headers: { // body에 담겨진 데이터의 타입 알려주기

"Content-Type": "application/json"

}

}).then(response => {

// http 응답 코드에 따른 메시지 출력

const msg = (response.ok) ? "댓글이 수정 되었습니다." : "댓글 수정 실패!!";

alert(msg);

// 현재 페이지를 새로고침

window.location.reload();

});

});

}

</script>

**\_new.mustache**

<div class="container">

<div class="card m-2" id="comments-new">

<div class="card-body">

<!-- 댓글 작성 폼-->

<form>

<!-- 닉네임 입력 -->

<div class="mb-3">

<label class="form-label">닉네임</label>

<input type="text" class="form-control form-control-sm" id="new-comment-nickname">

</div>

<!-- 댓글 본문 입력 -->

<div class="mb-3">

<label class="form-label">댓글 내용</label>

<textarea type="text" class="form-control form-control-sm" rows="3" id="new-comment-body"></textarea>

</div>

<!-- 히든 인풋 -->

<!-- 댓글은 게시글에 포함되어야하니까 article\_id의 값을 hidden로 가지고 있어야한다. -->

{{#article}}

<input type="hidden" id="new-comment-article-id" value="{{id}}">

{{/article}}

<!-- 전송 버튼 -->

<div class="d-flex justify-content-end">

<button type="button" class="btn btn-outline-primary btn-m" id="comment-create-btn">댓글 작성</button>

</div>

</form>

</div>

</div>

</div>

<script>

{

// 댓글 작성 버튼 변수화(id가 comment-create-btn인 대상)

const commentCreateBtn = document.querySelector("#comment-create-btn");

// 버튼 클릭 이벤트 감지!

commentCreateBtn.addEventListener("click", function() {

// 새 댓글 객체 생성

const comment = {

nickname: document.querySelector("#new-comment-nickname").value,

body: document.querySelector("#new-comment-body").value,

article\_id: document.querySelector("#new-comment-article-id").value

};

// 댓글 객체 출력

console.log(comment);

// fetch() - Talend API요청을 JavaScript로 보내준다

const url = "/api/articles/" + comment.article\_id + "/comments";

fetch(url, {

method: "post", // POST 요청

body: JSON.stringify(comment), // comment JS 객체를 JSON으로 변경하여 보냄

headers: {

"Content-Type": "application/json"

}

}).then(response => { // 응답 후 다음 처리 추가

// http 응답 코드에 따른 메시지 출력

const msg = (response.ok) ? "댓글이 등록되었습니다." : "댓글 등록 실패!!";

alert(msg);

// 현재 페이지 새로고침

window.location.reload();

});

});

}

</script>