

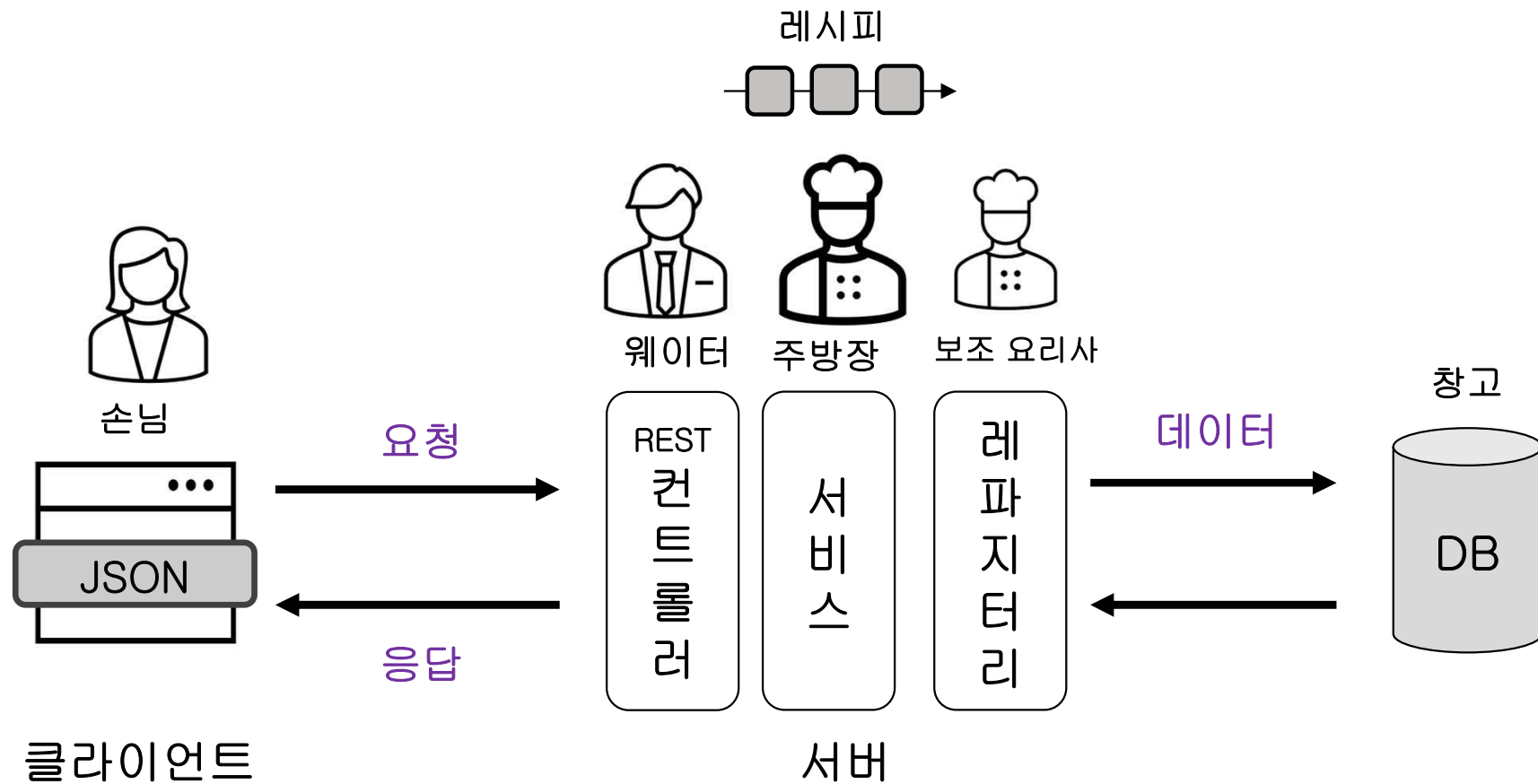
12장 서비스와 트랜잭션

출처: 코딩 자율학습 스프링부트3 자바 백엔드 개발 입문, 홍팩, 길벗, 2023

서비스와 트랜잭션의 개념

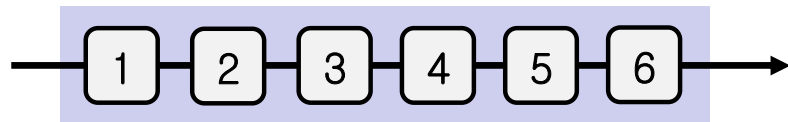
- 서비스(Service)

- 컨트롤러와 리파지터리 사이에 위치하는 계층
- 서버의 핵심 기능(비즈니스 로직)을 처리하는 순서를 총칭



트랜잭션 (transaction)

- 일반적으로 서비스의 업무 처리는 트랜잭션 단위로 진행됨
- 트랜잭션은 업무 처리의 최소 단위
- 트랜잭션은 전부 성공해야 하는 일련의 과정
- 예: 식당 예약으로 본 트랜잭션

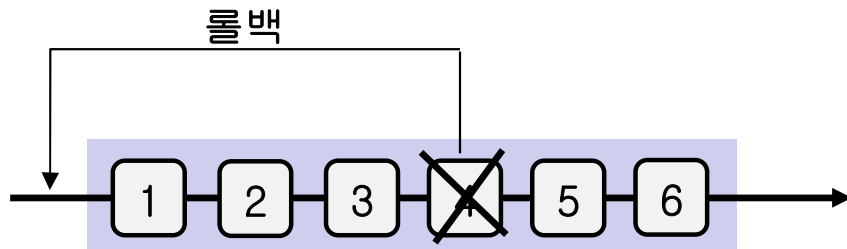


식당 예약

1. 시간 예약
2. 테이블 지정
3. 메뉴 선택
4. 결제
5. 영수증 발행
6. 예약 완료

- 롤백(rollback)

- 트랜잭션 내 작업이 실패하면, 트랜잭션의 진행 초기 단계로 되돌리는 것



식당 예약

1. 시간 예약
2. 테이블 지정
3. 메뉴 선택
4. 결제
5. 영수증 발행
6. 예약 완료

트랜잭션(transaction)

- 11장에서 구현한 REST 컨트롤러는 컨트롤러와 서비스의 역할을 동시에 수행
- → 12장에서는 컨트롤러와 리파지토리 사이에 서비스 계층을 두어 역할을 분업

```
public class ArticleApiController {  
    // PATCH  
    @PatchMapping("/api/articles/{id}")  
    public ResponseEntity<Article> update(@PathVariable Long id, @RequestBody ArticleForm dto){  
        // 1. DTO를 엔티티로 변환하기  
        Article article = dto.toEntity();  
        log.info("id: {}, article: {}", id, article.toString());  
        // 2. 타킷 조회하기  
        Article target = articleRepository.findById(id).orElse( other: null);  
        // 3. 잘 못된 요청 처리하기  
        if(target == null || id != article.getId()) {  
            // 400, 잘 못된 요청 응답  
            log.info("잘 못된 요청! id: {}, article: {}", id, article.toString());  
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);  
        }  
        // 4. 업데이트 및 정상 응답(200) 하기  
        target.patch(article);  
        Article updated = articleRepository.save(target);  
        return ResponseEntity.status(HttpStatus.OK).body(updated);  
    }  
}
```

서비스의 역할:
레파지토리에 데이터를
가져오도록 명령하기

서비스 계층 만들기

- REST 컨트롤러 (api/ArticleApiController.java)에 서비스 계층을 추가해서 컨트롤러, 서비스, 리파지터리의 역할을 분업
- com.example.firstproject에 service 패키지를 생성



- com.example.firstproject.service 패키지에 ArticleService 클래스를 생성
 - @Service : 스프링 부트가 해당 클래스를 서비스로 인식해서, 서비스 객체를 생성

```
package com.example.firstproject.service;

import org.springframework.stereotype.Service;

@Service
public class ArticleService {

}
```

서비스 계층 만들기

- ArticleApiController 클래스
 - 모든 코드를 주석 처리
 - 서비스 객체를 주입

api/ArticleApiController.java

```
@Slf4j
@RestController
public class ArticleApiController {

    @Autowired
    private ArticleService articleService;

}
```

서비스 계층 만들기

- ArticleService에 게시글 리파지터리 객체 주입

service/ArticleService.java

```
@Service
public class ArticleService {

    @Autowired
    private ArticleRepository articleRepository;
}
```

모든 게시물 조회 요청 개선하기

api/ArticleApiController.java

```
public class ArticleApiController {  
  
    @Autowired  
    private ArticleService articleService;  
  
    // GET  
    @GetMapping("/api/articles")  
    public List<Article> index() {  
        return articleService.index();  
    }  
}
```

service/ArticleService.java

```
@Service  
public class ArticleService {  
  
    @Autowired  
    private ArticleRepository articleRepository;  
  
    public List<Article> index() {  
        return articleRepository.findAll();  
    }  
}
```


모든 게시물 조회 요청 개선하기

- 서버 재구동
- Talend API Tester
 - 메서드: GET
 - URL: <http://localhost:8080/api/articles>

The screenshot shows the Talend API Tester interface. At the top left, there is a 'DRAFT' label. On the right, there is a 'Save as' button. The main configuration area has a 'METHOD' dropdown set to 'GET' and a 'SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]' field containing 'http://localhost:8080/api/articles'. Both the 'METHOD' dropdown and the URL field are highlighted with red boxes. To the right of the URL field is a 'Send' button, also highlighted with a red box. Below the URL field, there is a 'QUERY PARAMETERS' section. At the bottom, there are tabs for 'HEADERS' and 'BODY'. The 'HEADERS' tab is active, showing a 'Form' dropdown and buttons for '+ Add header' and 'Add authorization'. The 'BODY' tab is also visible. A message at the bottom right states: 'XHR does not allow payloads for GET request.'

모든 게시물 조회 요청 개선하기

- 응답 결과 확인

The screenshot shows a web browser's developer tools interface. At the top, a green bar displays the status code **200**. Below this, the interface is split into two panels: **HEADERS** and **BODY**.

The **HEADERS** panel on the left shows the following details:

- Content-Type:** application/json;charset=UTF-8
- Transfer-Encoding:** chunked
- Date:** Tue, 02 Dec 2025 08:10:40 GMT
- Keep-Alive:** timeout=60
- Connection:** keep-alive

Below the headers, there is a link that says **▶ COMPLETE REQUEST HEADERS**.

The **BODY** panel on the right shows a JSON array of three objects, formatted in a "pretty" view. The JSON data is as follows:

```
[
  {
    "id": 1,
    "title": "가가가가",
    "content": "1111"
  },
  {
    "id": 2,
    "title": "나나나나",
    "content": "2222"
  },
  {
    "id": 3,
    "title": "다다다다",
    "content": "3333"
  }
]
```

단일 게시글 조회 요청 개선하기

api/ArticleApiController.java

```
@Slf4j
@RestController
public class ArticleApiController {

    (중략)

    @GetMapping("/api/articles/{id}")
    public Article show(@PathVariable Long id){
        return articleService.show(id);
    }
}
```

service/ArticleService.java

```
@Service
public class ArticleService {

    (중략)

    public Article show(Long id) {
        return articleRepository.findById(id).orElse( other: null);
    }
}
```

단일 게시물 조회 요청 개선하기

- 서버 재구동
- Talend API Tester
 - 메서드: GET
 - URL: <http://localhost:8080/api/articles/1>

The screenshot shows the Talend API Tester interface. At the top left, there is a 'DRAFT' label. On the right, there is a 'Save as' button with a dropdown arrow. The main configuration area has a 'METHOD' dropdown set to 'GET' and a 'SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]' field containing 'http://localhost:8080/api/articles/1'. Both the 'METHOD' dropdown and the URL field are highlighted with red boxes. To the right of the URL field is a 'Send' button, also highlighted with a red box. Below the URL field, there is a 'QUERY PARAMETERS' section. At the bottom, there are tabs for 'HEADERS' and 'BODY'. The 'HEADERS' tab is active, showing '+ Add header' and 'Add authorization' buttons. The 'BODY' tab is inactive, showing a message: 'XHR does not allow payloads for GET request.'

- 응답 결과 확인

The screenshot shows the 'Response' section of the Talend API Tester interface. At the top, there is a green bar with the status code '200'. Below this, there are tabs for 'HEADERS' and 'BODY'. The 'HEADERS' tab is active, showing a list of headers: 'Content-Type: application/json; charset=UTF-8', 'Transfer-Encoding: chunked', 'Date: Tue, 02 Dec 2025 08:53:37 GMT', 'Keep-Alive: timeout=60', and 'Connection: keep-alive'. The 'BODY' tab is inactive, showing a JSON response: '{ id: 1, title: "가가가가", content: "1111" }'. The 'BODY' tab is highlighted with a red box.

게시글 생성 요청 개선하기

- ArticleApiController 클래스에 create() 메소드

```
@PostMapping("/api/articles")
public ResponseEntity<Article> create(@RequestBody ArticleForm dto){
    Article created = articleService.create(dto);
    return (created != null) ? good 응답: bad 응답;
}
```



api/ArticleApiController.java

```
// POST
@PostMapping("/api/articles")
public ResponseEntity<Article> create(@RequestBody ArticleForm dto){
    Article created = articleService.create(dto);
    return (created != null) ?
        ResponseEntity.status(HttpStatus.OK).body(created):
        ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
}
```

게시글 생성 요청 개선하기

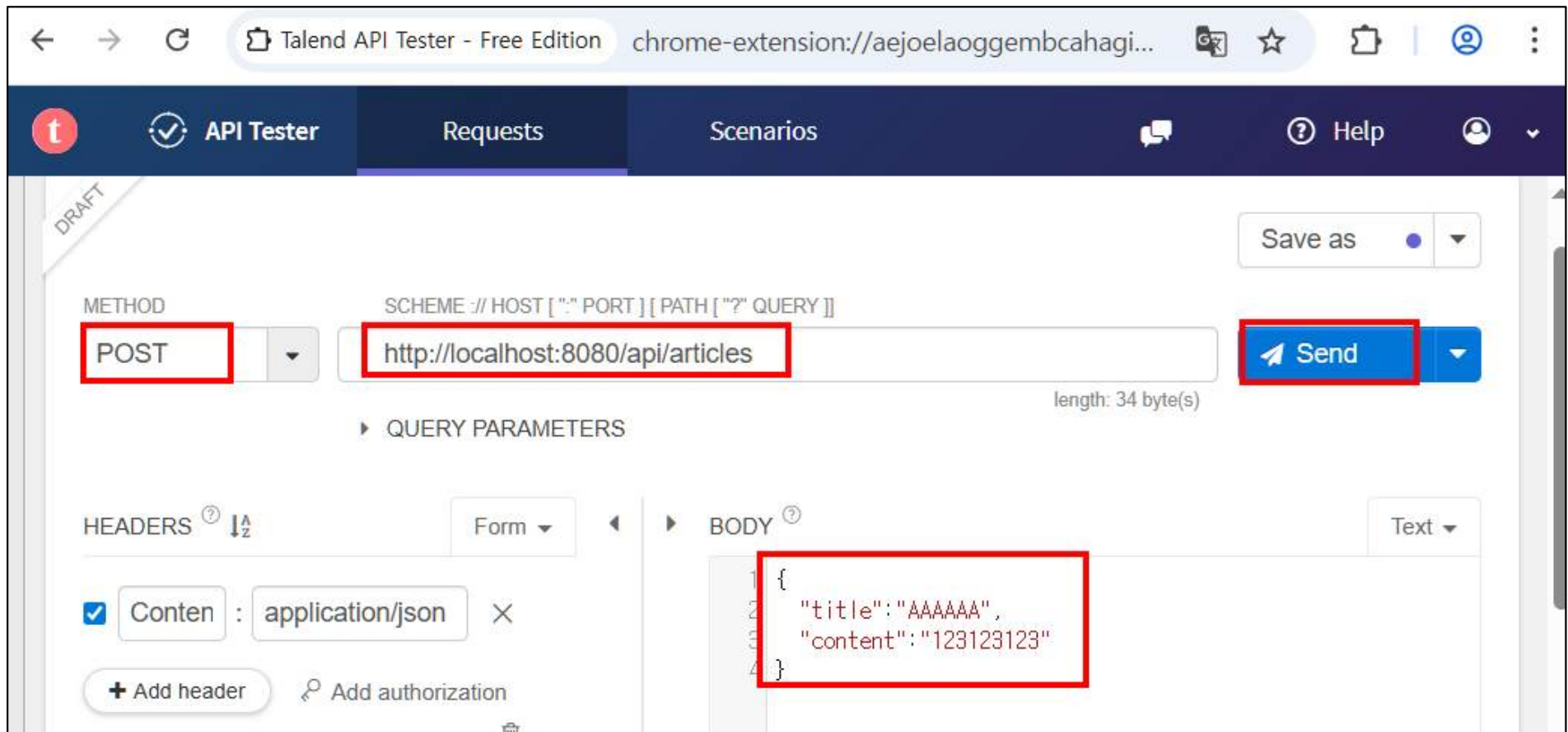
- ArticleService에 create() 메소드 추가

service/ArticleService.java

```
public Article create(ArticleForm dto) {  
    Article article = dto.toEntity();  
    return articleRepository.save(article);  
}
```

게시글 생성 요청 개선하기

- 서버 재실행
- Talend API Tester에서 게시물 생성 요청
 - 메서드: POST
 - URL: <http://localhost:8080/api/articles>
 - BODY에 생성할 데이터를 JSON 형식으로 입력



게시글 생성 요청 개선하기

- 응답 결과 확인

Response Cache Detected - Elapsed Time: 415ms

200

HEADERS [?]
Content-Type: application/json;charset=UTF-8
Transfer-Encodi... chunked
Date: Tue, 25 Nov 2025 13:16:05 GMT
Keep-Alive: timeout=60
Connection: keep-alive
▶ COMPLETE REQUEST HEADERS

BODY [?]

```
{
  id: 4,
  title: "AAAAAA",
  content: "123123123"
}
```


length: 47 bytes

- <https://localhost:8080/articles> 접속

localhost:8080/articles

Navbar

Id	Title	Content
1	가가가가	1111
2	나나나나	2222
3	다다다다	3333
4	AAAAAA	123123123

[New Article](#)

© CloudStudying | [Privacy](#) | [Terms](#)

게시글 생성 요청 개선하기

METHOD **POST** SCHEME // HOST [":" PORT] [PATH ["?" QUERY]] **Send** length: 34 byte(s)
http://localhost:8080/api/articles
Send request (Alt + Enter)

QUERY PARAMETERS

HEADERS [?] **Content-Type** : application/json **Form** **Body** [?] Text

+ Add header Add authorization

```
1 {  
2   "id": 1,  
3   "title": "abcabc",  
4   "content": "123123"  
5 }
```

Response

200

HEADERS [?] pretty **Body** [?]

Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 02 Dec 2025 09:39:43 GMT
Keep-Alive: timeout=60
Connection: keep-alive

```
{  
  id: 1,  
  title: "abcabc",  
  content: "123123"  
}
```

게시글 생성 요청 개선하기

- ArticleService의 create() 수정
 - article 객체에 id가 존재하면, null 반환

service/ArticleService.java

```
public Article create(ArticleForm dto) {  
    Article article = dto.toEntity();  
    if(article.getId() != null){  
        return null;  
    }  
    return articleRepository.save(article);  
}
```

게시글 생성 요청 개선하기

- 서버 재실행
- Talend API Tester에서 게시물 생성 요청

The screenshot shows the Talend API Tester interface for configuring a POST request. The **METHOD** is set to **POST**. The **URL** is **http://localhost:8080/api/articles**. The **Content-Type** header is set to **application/json**. The **BODY** is a JSON object:

```
{
  "id": 1,
  "title": "abcabc",
  "content": "123123"
}
```

. The **Send** button is highlighted with a red box, and a tooltip indicates **Send request (Alt + Enter)**.

- 응답 결과 확인

The screenshot shows the **Response** section of the Talend API Tester. The status code is **400**. The **HEADERS** section shows: **Content-Length: 0 byte**, **Date: Tue, 02 Dec 2025 09:50:57 GMT**, and **Connection: close**. The **BODY** section displays **No Content**.

게시글 수정 요청 개선하기

- ArticleApiController의 update() 메소드

```
// PATCH
@PatchMapping("/api/articles/{id}")
public ResponseEntity<Article> update(@PathVariable Long id, @RequestBody ArticleForm dto){
    Article updated = articleService.update(id, dto);
    return (updated != null)?
        ResponseEntity.status(HttpStatus.OK).body(updated):
        ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
}
```

* 컨트롤러는 서비스에 무슨 지시를 하고, 무슨 데이터를 받아 오는지만 알면 됨
실제 작업은 모두 서비스에 맡김

게시글 수정 요청 개선하기

- ArticleService에 update() 추가

```
@Slf4j  
@Service
```

```
public class ArticleService {
```

(중략)

```
public Article update(Long id, ArticleForm dto) {  
    // 1. DTO를 엔티티로 변환하기  
    Article article = dto.toEntity();  
    log.info("id: {}, article: {}", id, article.toString());  
    // 2. 타킷 조회하기  
    Article target = articleRepository.findById(id).orElse( other: null);  
    // 3. 잘 못된 요청 처리하기  
    if(target == null || id != article.getId()) {  
        // 400, 잘 못된 요청 응답  
        log.info("잘 못된 요청! id: {}, article: {}", id, article.toString());  
        return null;  
    }  
    // 4. 업데이트 및 정상 응답(200) 하기  
    target.patch(article);  
    Article updated = articleRepository.save(target);  
    return updated;  
}
```

```
}
```

게시글 수정 요청 개선하기

- 서버 재실행
- 데이터의 일부만 수정 요청

The screenshot shows a REST client interface with the following details:

- METHOD:** PATCH (highlighted with a red box)
- URL:** http://localhost:8080/api/articles/1 (highlighted with a red box)
- Send Button:** A blue button with a paper plane icon and the text "Send" (highlighted with a red box).
- HEADERS:** A section with a checkbox for "Content-Type" set to "application/json".
- BODY:** A section with a JSON object:

```
{  "id": 1,  "content": "5678"}
```

 (highlighted with a red box).

- 응답 결과 확인

The screenshot shows the response of a PATCH request in a REST client interface:

- Response Status:** 200 (highlighted with a green bar).
- HEADERS:** A list of headers including "Content-Type: application/json; charset=UTF-8", "Transfer-Encoding: chunked", "Date: Tue, 02 Dec 2025 10:29:48 GMT", "Keep-Alive: timeout=60", and "Connection: keep-alive".
- BODY:** A JSON object:

```
{  id: 1,  title: "가가가가",  content: "5678"}
```

 (highlighted with a red box).

게시글 삭제 요청 개선하기

- ArticleApiController의 update() 메소드
 - HttpStatus.NO_CONTENT
 - ✓ 상태 코드: 204
 - ✓ 서버가 요청을 성공적으로 처리했지만, 응답 본문을 반환하지 않음

```
// DELETE
@DeleteMapping("/api/articles/{id}")
public ResponseEntity<Article> delete(@PathVariable Long id){
    Article deleted = articleService.delete(id);
    return (deleted != null)?
        ResponseEntity.status(HttpStatus.NO_CONTENT).build():
        ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
}
```

게시글 삭제 요청 개선하기

- ArticleService에 delete() 추가

```
public Article delete(Long id) {  
    // 1. 대상 찾기  
    Article target = articleRepository.findById(id).orElse( other: null);  
    // 2. 잘못된 요청 처리하기  
    if(target == null){  
        return null;  
    }  
    // 3. 대상 삭제하기  
    articleRepository.delete(target);  
    return target;  
}
```

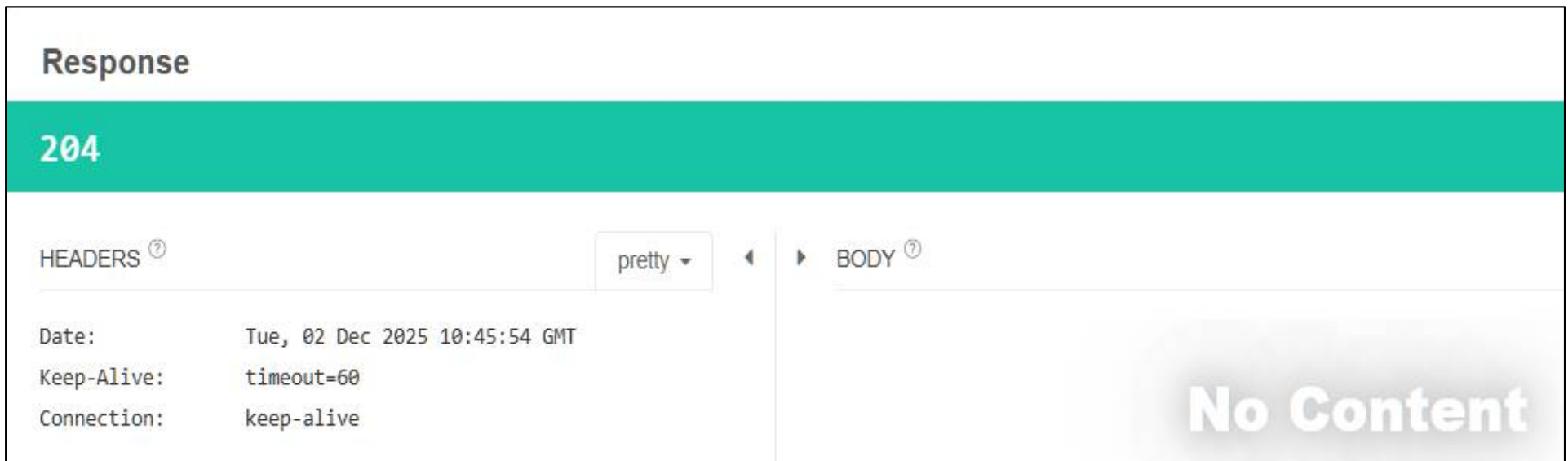

게시글 삭제 요청 개선하기

- 서버 재실행
- DELETE 요청 보내기



A screenshot of a REST client interface. The top left corner has a 'DRAFT' label. On the right, there is a 'Save as' button with a dropdown arrow. The main area shows a request configuration: the 'METHOD' dropdown is set to 'DELETE' (highlighted with a red box), and the 'URL' field contains 'http://localhost:8080/api/articles/1' (also highlighted with a red box). Below the URL, there is a 'QUERY PARAMETERS' section with a right-pointing arrow. On the right side, there is a 'Send' button with a paper plane icon (highlighted with a red box) and a dropdown arrow. Below the URL field, it says 'length: 36 byte(s)'.

- 응답 확인



A screenshot of a REST client interface showing the response details. The top section is titled 'Response'. Below it, a green bar displays the status code '204'. Underneath, there are two tabs: 'HEADERS' (selected) and 'BODY'. The 'HEADERS' tab shows a list of headers: 'Date: Tue, 02 Dec 2025 10:45:54 GMT', 'Keep-Alive: timeout=60', and 'Connection: keep-alive'. The 'BODY' tab is currently empty and displays a large, light gray box with the text 'No Content'.

트랜잭션 맛보기

- 트랜잭션이 반드시 성공해야 할 일련의 과정임을 확인
- 시나리오
 - 게시판에 데이터 3개를 한꺼번에 생성 요청하기
 - 데이터를 DB에 저장하는 과정에서 의도적으로 오류 발생시키기
 - 어떻게 롤백되는지 확인하기
- 요청
 - 전송방식: POST
 - URL: <http://localhost:8080/api/transaction-test>

트랜잭션 맛보기

- ArticleApiController에 transactionTest() 추가

api/ArticleApiController.java

```
@PostMapping("api/transaction-test") //여러 게시글 생성 요청 접수
public ResponseEntity<List<Article>> transactionTest
    (@RequestBody List<ArticleForm> dtos){
    List<Article> createdList = articleService.createArticles(dtos);
    return (createdList != null) ?
        ResponseEntity.status(HttpStatus.OK).body(createdList) :
        ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
}
```

트랜잭션 맛보기

- ArticleService에 createArticles() 추가

service/ArticlesService.java

```
public List<Article> createArticles(List<ArticleForm> dtos) {  
    // 1. dto 목록을 엔티티 목록으로 변환하기  
    // 2. 엔티티 목록을 DB에 저장하기  
    // 3. 강제 예외 발생시키기  
    // 4. 결과 값 반환하기  
}
```

트랜잭션 맛보기

- 1. dto 묶음을 엔티티 묶음으로 변환하기

- 1) dtos를 스트림화
- 2) map()으로 dto가 하나하나 올 때마다 dto.toEntity()를 수행해 매핑
- 3) 이렇게 매핑한 것을 리스트로 묶음
- 4) 최종 결과를 articleList에 저장

service/ArticleService.java

```
public List<Article> createArticles(List<ArticleForm> dtos) {  
    // 1. dto 묶음을 엔티티 묶음으로 변환하기  
    List<Article> articleList = dtos.stream() Stream<ArticleForm>  
        .map( ArticleForm dto -> dto.toEntity()) Stream<Article>  
        .collect(Collectors.toList());  
    // 2. 엔티티 묶음을 DB에 저장하기  
    // 3. 강제 예외 발생시키기  
    // 4. 결과 값 반환하기  
}
```

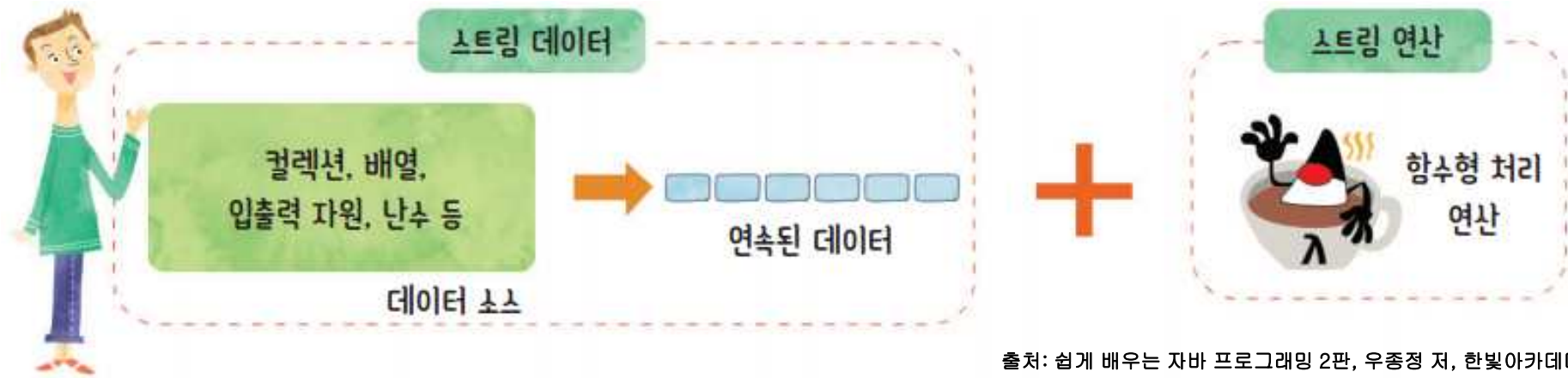
↑

```
List<Article> articleList = new ArrayList<>();  
for(int i=0; i<dtos.size(); i++){  
    ArticleForm dto = dtos.get(i);  
    Article entity = dto.toEntity();  
    articleList.add(entity);  
}
```

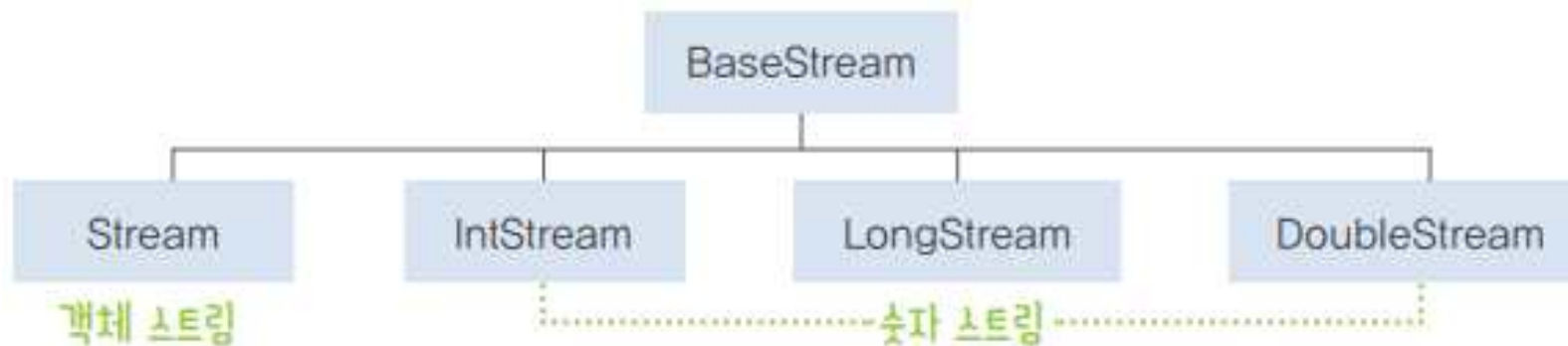
Java의 Stream

출처: 쉽게 배우는 자바 프로그래밍 2판, 우종정 저, 한빛아카데미

- 리스트와 같은 자료구조에 저장된 요소를 하나씩 순회하면서 처리
- 스트림은 스트림 데이터와 스트림 연산의 개념을 모두 포함



- 스트림의 종류



출처: 쉽게 배우는 자바 프로그래밍 2판, 우종정 저, 한빛아카데미

Java의 Stream

- 스트림 연산

- 필터링

- ✓ 입력된 스트림 원소에서 일부 원소를 걸러내는 중간 연산
 - ✓ filter(), distinct(), limit(), skip()

- 매핑

- ✓ 입력 스트림을 다른 종류의 스트림으로 변경
 - ✓ map(), flatMap(), mapToObj(), mapToInt(), ...

- 정렬

- ✓ 입력된 스트림 원소 전체를 정렬하는 중간 연산
 - ✓ sorted()

- 루핑

- ✓ 입력된 스트림의 전체 원소를 반복하는 연산
 - ✓ forEach() - 최종 연산, peak() - 중간 연산

- 단순 집계 연산

- ✓ count(), sum(), average(), max(), min(), ...

- 컬렉터 연산

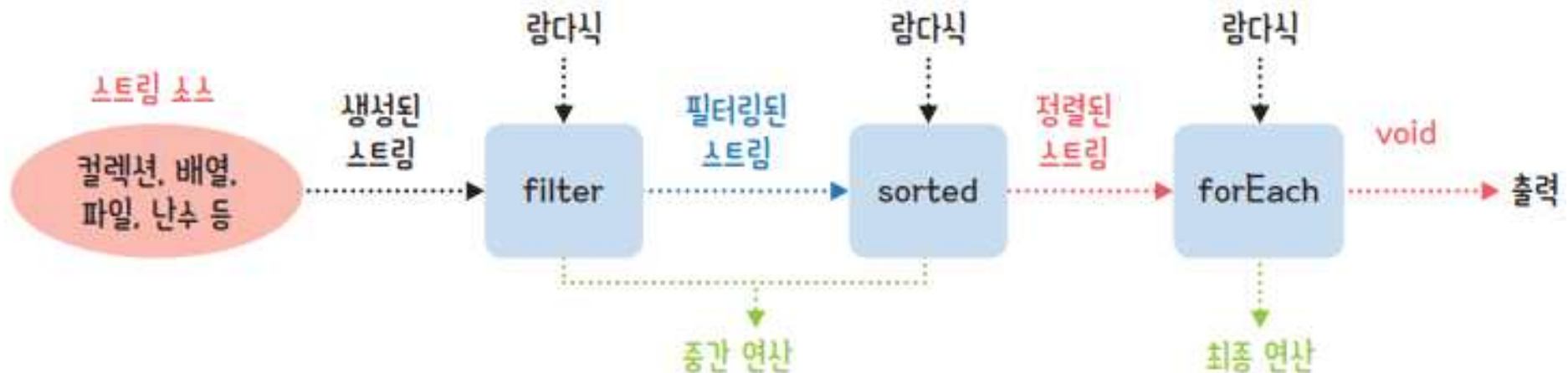
- ✓ 연산의 실행결과를 컬렉션에 수집할 수 있도록 함

- ...

Java의 Stream

● 스트림 파이프라인

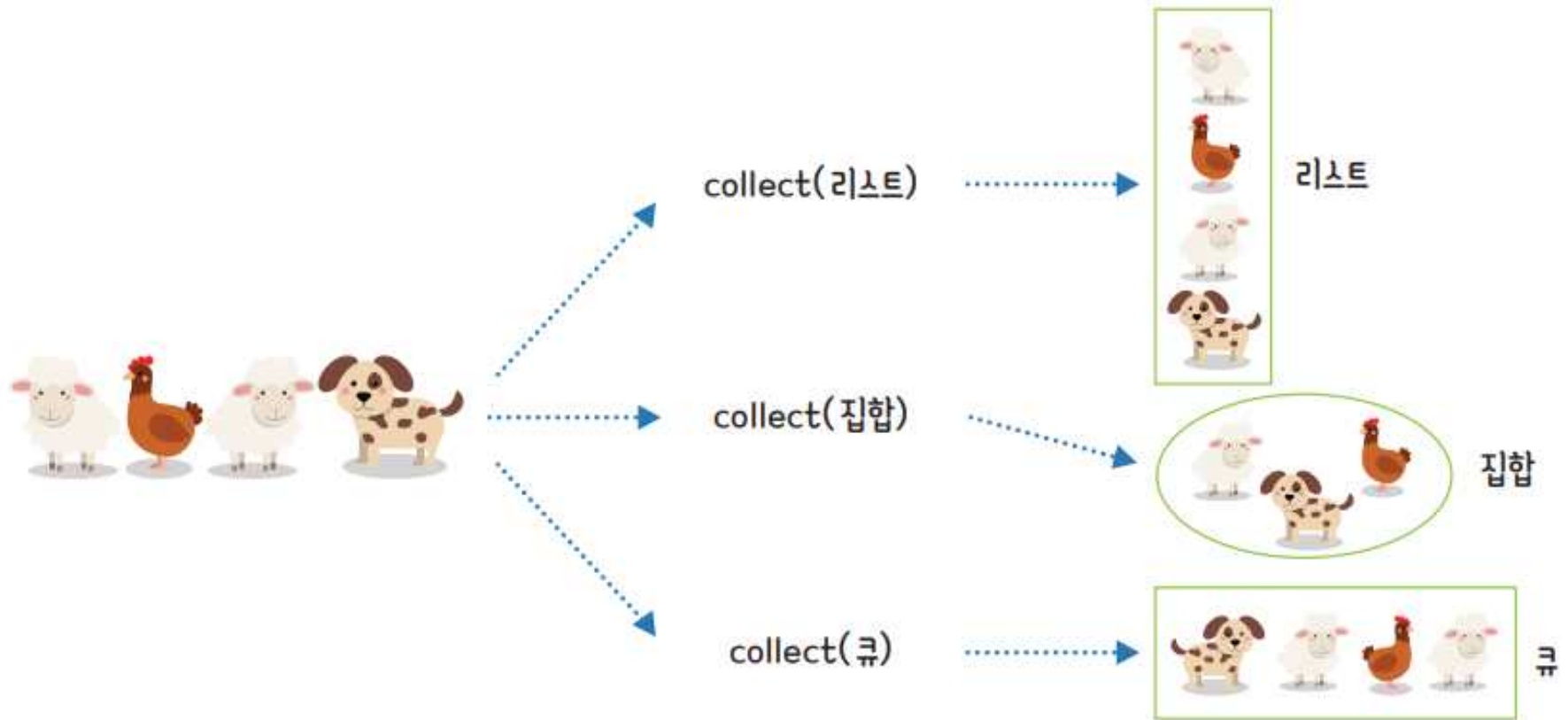
- 스트림 연산의 결과가 Stream 이면, 연속적으로 스트림 연산을 호출할 수 있음.
- 스트림 연산의 연속 호출은 여러 개의 스트림이 연결되어 스트림 파이프라인을 형성



출처: 쉽게 배우는 자바 프로그래밍 2판, 우종정 저, 한빛아카데미

Java의 Stream

- 컬렉터 연산
 - 컬렉터는 원소를 어떤 컬렉션에 수집할 것인지 결정



출처: 쉽게 배우는 자바 프로그래밍 2판, 우종정 저, 한빛아카데미

트랜잭션 맛보기

- 2. 엔티티 묶음을 DB에 저장하기

- articleList를 스트림화
- article이 하나씩 올 때마다 articleRepository를 통해 DB에 저장

service/ArticleService.java

```
public List<Article> createArticles(List<ArticleForm> dtos) {  
    // 1. dto 묶음을 엔티티 묶음으로 변환하기  
    List<Article> articleList = dtos.stream() Stream<ArticleForm>  
        .map( ArticleForm dto -> dto.toEntity()) Stream<Article>  
        .collect(Collectors.toList());  
  
    // 2. 엔티티 묶음을 DB에 저장하기  
    articleList.stream()  
        .forEach( Article article -> articleRepository.save(article));  
  
    // 3. 강제 예외 발생시키기  
    // 4. 결과 값 반환하기  
}
```

↑

```
for(int i=0; i<articleList.size(); i++){  
    Article article = articleList.get(i);  
    articleRepository.save(article);  
}
```

트랜잭션 맛보기

• 3. 강제로 예외 발생시키기

- Optional.orElseThrow()
 - ✓ 값이 존재하면 그 값을 반환, 값이 존재하지 않으면 전달 값으로 보낸 예외 발생
- IllegalArgumentException
 - ✓ 전달 값이 없거나 유효하지 않은 경우

• 4. 결과 값 반환하기

service/ArticleService.java

```
public List<Article> createArticles(List<ArticleForm> dtos) {  
    // 1. dto 묶음을 엔티티 묶음으로 변환하기  
    List<Article> articleList = dtos.stream() Stream<ArticleForm>  
        .map( ArticleForm dto -> dto.toEntity()) Stream<Article>  
        .collect(Collectors.toList());  
  
    // 2. 엔티티 묶음을 DB에 저장하기  
    articleList.stream()  
        .forEach( Article article -> articleRepository.save(article));  
  
    // 3. 강제 예외 발생시키기  
    articleRepository.findById(-1L)  
        .orElseThrow(() -> new IllegalArgumentException(("결재 실패!")));  
  
    // 4. 결과 값 반환하기  
    return articleList;  
}
```

트랜잭션 맛보기

- 서버 재실행
- Talend API Tester

The screenshot shows the Talend API Tester interface with the following configuration:

- METHOD:** POST
- SCHEME :** // **HOST :** ["."] **PORT :** **PATH :** ["?"] **QUERY :**
`http://localhost:8080/api/transaction-test`
- length:** 42 byte(s)
- Send** button
- QUERY PARAMETERS:** (collapsed)
- HEADERS:**
 - ☒ **Content-Type :** application/json
 - + Add header** **Add authorization**
- Form** (selected) **Body** (selected)
- Body:**

```
1 [
2   {
3     "title": "시간 예약",
4     "content": "11111"
5   },
6   {
7     "title": "테이블 지정",
8     "content": "22222"
9   },
10  {
11    "title": "메뉴 선택",
12    "content": "33333"
13  }
14 ]
```

트랜잭션 맛보기

- 응답 결과 확인

Response

500

HEADERS ⓘ

pretty ▾ ◀ ▶

BODY ⓘ

Content-Type: application/json;charset=UTF-8

Transfer-Encoding: chunked

Date: Tue, 02 Dec 2025 16:32:08 GMT

Connection: close

▶ COMPLETE REQUEST HEADERS

{

timestamp: "2025-12-02T16:32:08.109+00:00",

status: 500,

error: "Internal Server Error",

path: ↗ "/api/transaction-test"

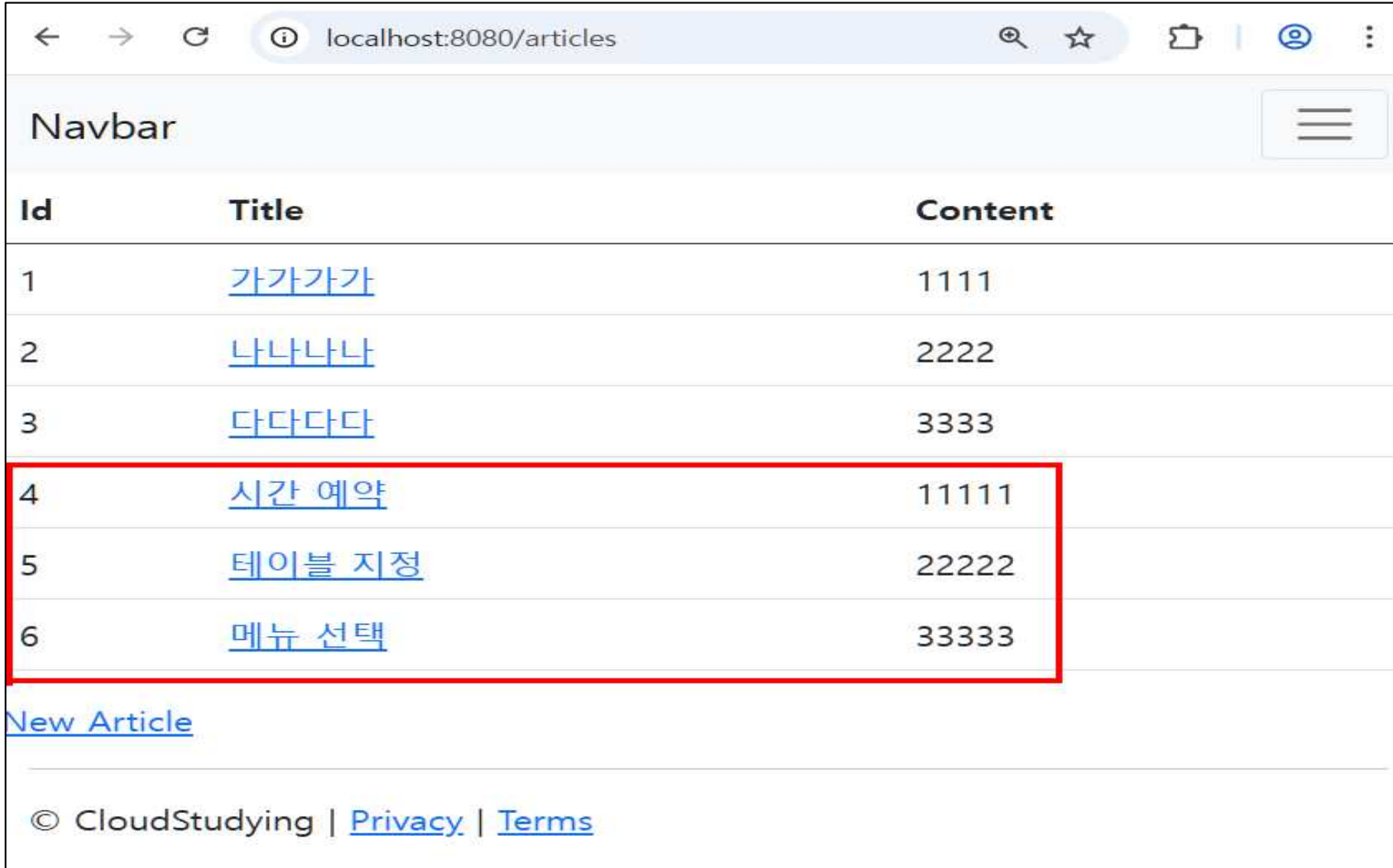
}

- 로그에서 에러 원인 탐색

```
java.lang.IllegalArgumentException Create breakpoint : 결재 실패!  
at com.example.firstproject.service.ArticleService.lambda$createArticles$2(ArticleService.java:81) ~[
```

트랜잭션 맛보기

- <http://localhost:8080/articles> 접속



Id	Title	Content
1	가가가가	1111
2	나나나나	2222
3	다다다다	3333
4	시간 예약	11111
5	테이블 지정	22222
6	메뉴 선택	33333

[New Article](#)

© CloudStudying | [Privacy](#) | [Terms](#)

트랜잭션 맛보기

● 트랜잭션으로 묶기

- 서비스의 메서드에 @Transactional 어노테이션 설정하여 해당 메서드를 하나의 트랜잭션으로 묶음
- 메서드가 중간에 실패하면 롤백을 통해 이전 상태로 되돌아 감

```
(중략)
import org.springframework.transaction.annotation.Transactional;

public class ArticleService {
    (중략)
    @Transactional
    public List<Article> createArticles(List<ArticleForm> dtos) {
        // 1. dto 묶음을 엔티티 묶음으로 변환하기
        log.info(dtos.toString());
        List<Article> articleList = dtos.stream() Stream<ArticleForm>
            .map( ArticleForm dto -> dto.toEntity()) Stream<Article>
            .collect(Collectors.toList());

        log.info(articleList.toString());
        // 2. 엔티티 묶음을 DB에 저장하기
        articleList.stream()
            .forEach( Article article -> articleRepository.save(article));
        // 3. 강제 예외 발생시키기
        articleRepository.findById(-1L)
            .orElseThrow(() -> new IllegalArgumentException(("결재 실패!")));
        // 4. 결과 값 반환하기
        return articleList;
    }
}
```

트랜잭션 맛보기

- 서버 재실행
- Talend API Tester

The screenshot shows the Talend API Tester interface. At the top, the METHOD is set to POST, and the URL is http://localhost:8080/api/transaction-test. The Send button is visible. Below the URL, the QUERY PARAMETERS section is empty. The HEADERS section shows Content-Type: application/json. The BODY section is set to Text and contains a JSON array with three objects, each with title and content fields. The entire body content is highlighted with a red box.

METHOD: POST

SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]

http://localhost:8080/api/transaction-test

length: 42 byte(s)

Send

QUERY PARAMETERS

HEADERS ②

Form

Content-Type : application/json

+ Add header Add authorization

BODY ②

```
1 [
2   {
3     "title": "시간 예약",
4     "content": "11111"
5   },
6   {
7     "title": "테이블 지정",
8     "content": "22222"
9   },
10  {
11    "title": "메뉴 선택",
12    "content": "33333"
13  }
14 ]
```


트랜잭션 맛보기

- 응답 결과 확인

Response

500

HEADERS ⓘ

pretty ▾

Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 02 Dec 2025 16:32:08 GMT
Connection: close

▶ COMPLETE REQUEST HEADERS

BODY ⓘ

▾

```
{  
  timestamp: "2025-12-02T16:32:08.109+00:00",  
  status: 500,  
  error: "Internal Server Error",  
  path: ↗ "/api/transaction-test"  
}
```

트랜잭션 맛보기

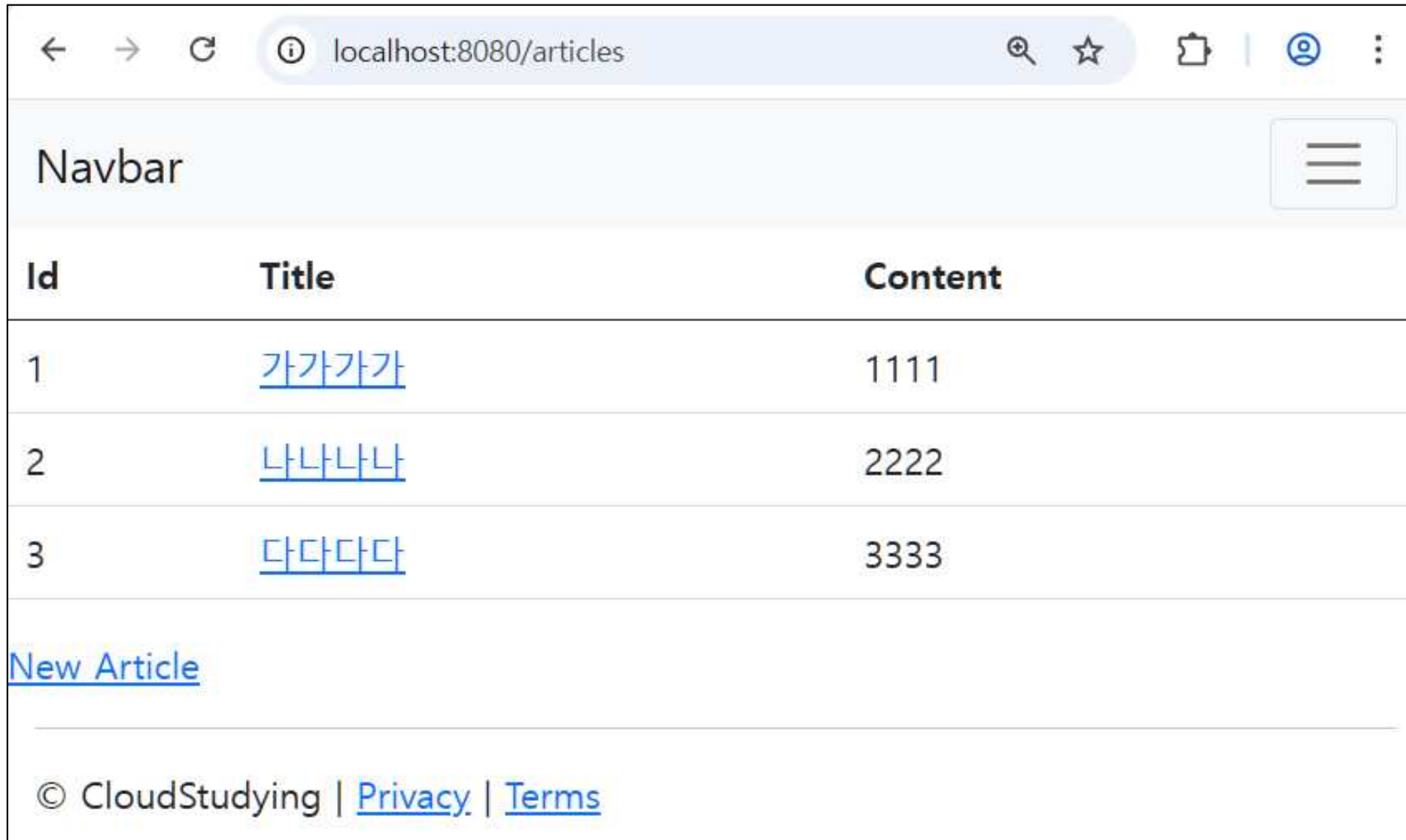
● 로그에서 에러 원인 탐색

```
2025-12-03T01:56:14.246+09:00 DEBUG 1248 --- [nio-8080-exec-4] org.hibernate.SQL :
insert
into
    article
    (content,title,id)
values
    (?, ?, default)
2025-12-03T01:56:14.249+09:00 TRACE 1248 --- [nio-8080-exec-4] org.hibernate.orm.jdbc.bind : binding par
2025-12-03T01:56:14.252+09:00 TRACE 1248 --- [nio-8080-exec-4] org.hibernate.orm.jdbc.bind : binding par
2025-12-03T01:56:14.273+09:00 DEBUG 1248 --- [nio-8080-exec-4] org.hibernate.SQL :
select
    a1_0.id,
    a1_0.content,
    a1_0.title
from
    article a1_0
where
    a1_0.id=?
2025-12-03T01:56:14.274+09:00 TRACE 1248 --- [nio-8080-exec-4] org.hibernate.orm.jdbc.bind : binding par
2025-12-03T01:56:14.283+09:00 ERROR 1248 --- [nio-8080-exec-4] o.a.c.c.C.[.].[/].[dispatcherServlet] : Servlet.ser

java.lang.IllegalArgumentException Create breakpoint 결재 실패!
    at com.example.firstproject.service.ArticleService.lambda$createArticles$2(ArticleService.java:83) ~[main/:na]
```

트랜잭션 맛보기

- <http://localhost:8080/articles> 접속



Id	Title	Content
1	가가가가	1111
2	나나나나	2222
3	다다다다	3333

[New Article](#)

© CloudStudying | [Privacy](#) | [Terms](#)