

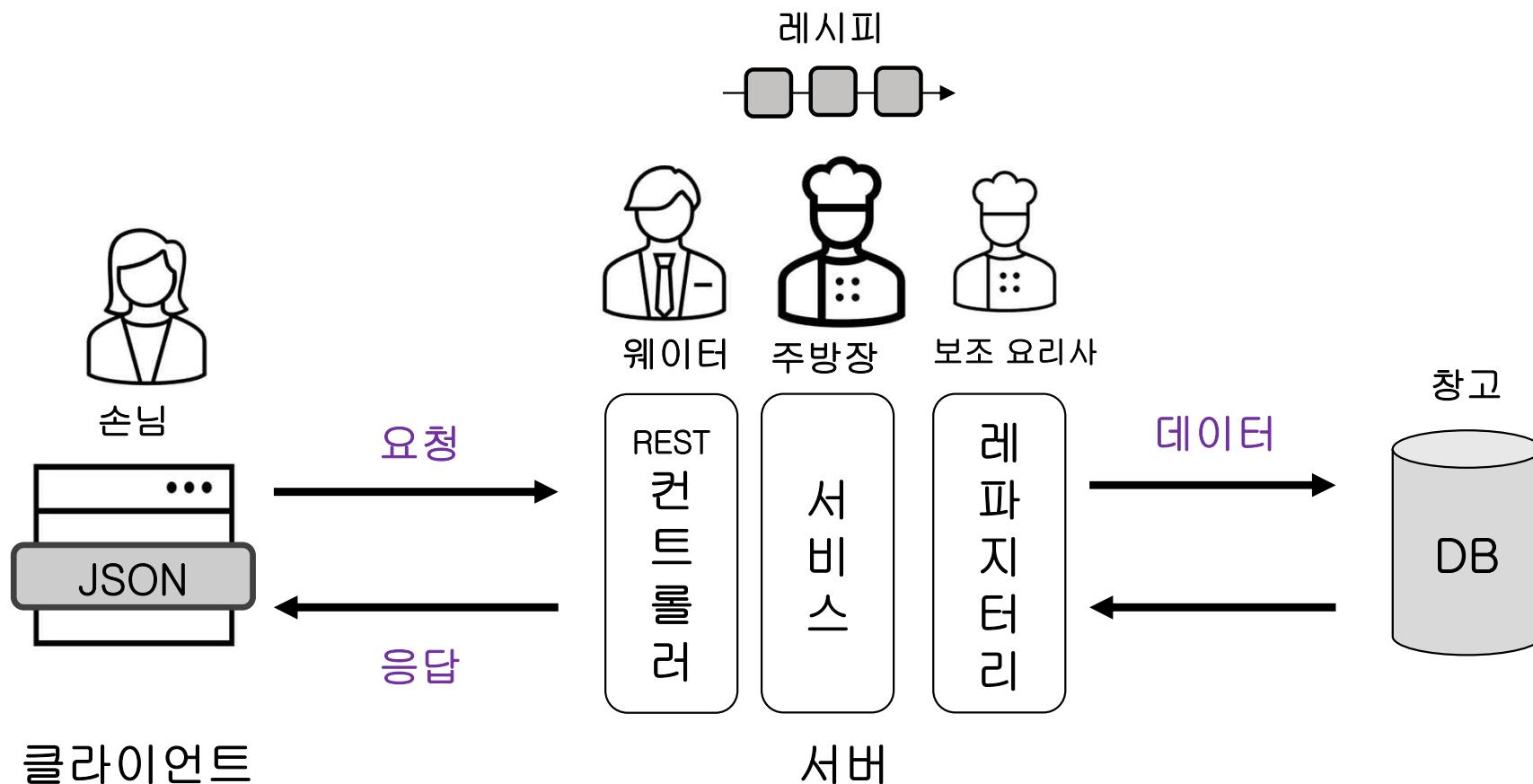
12장 서비스와 트랜잭션

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

서비스와 트랜잭션의 개념

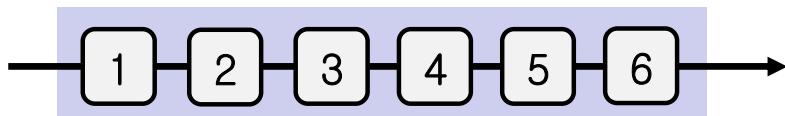
- 서비스(Service)

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



트랜잭션(transaction)

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

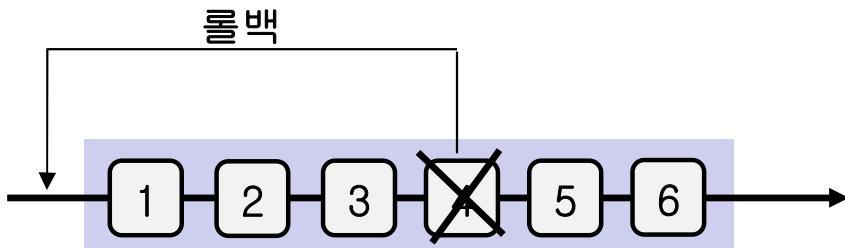


식당 예약

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. 타겟 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:
리파지토리에 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 게시글 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 게시글 조회 요청 개선하기

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



모든 게시글 조회 요청 개선하기

- 응답 결과 확인

The screenshot shows a REST API response with a status of 200 OK. The response includes headers and a JSON body.

HEADERS

- 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

BODY

```
[{"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(null);  
    }  
}
```

단일 게시글 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, there's a 'DRAFT' indicator, a 'Save as' dropdown, and a 'Send' button which is also highlighted with a red box. Below that, the 'METHOD' is set to 'GET' and the 'SCHEME :// HOST [":" PORT] [PATH ["?" QUERY]]' field contains 'http://localhost:8080/api/articles/1'. A red box highlights this URL field. The 'BODY' section shows a note: 'XHR does not allow payloads for GET request.' At the bottom, there are 'HEADERS' and 'BODY' tabs, and buttons for '+ Add header' and 'Add authorization'.

- 응답 결과 확인

The screenshot shows the 'Response' tab in Talend API Tester. The status code is '200'. The 'HEADERS' section lists:

- Content-Type: application/json; charset=UTF-8
- Transfer-Encoding: chunked
- Date: Tue, 02 Dec 2025 08:53:37 GMT
- Keep-Alive: timeout=60
- Connection: keep-alive

The 'BODY' section shows a JSON response:

```
{  
  "id": 1,  
  "title": "가가가가",  
  "content": "1111"  
}
```

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

The screenshot shows the Talend API Tester interface. The 'API Tester' tab is selected. In the 'METHOD' dropdown, 'POST' is chosen. The 'SCHEME // HOST [":" PORT] [PATH ["?" QUERY]]' field contains 'http://localhost:8080/api/articles'. The 'Send' button is highlighted with a red box. In the 'BODY' section, there is a JSON object:

```
{  
  "title": "AAAAAAA",  
  "content": "123123123"  
}
```

게시글 생성 요청 개선하기

- 응답 결과 확인

Response

Cache Detected - Elapsed Time: 415ms

200

HEADERS pretty

Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 25 Nov 2025 13:16:05 GMT
Keep-Alive: timeout=60
Connection: keep-alive

BODY pretty

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

length: 47 bytes

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

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

New Article

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

게시글 생성 요청 개선하기



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

- METHOD:** POST (highlighted with a red box)
- SCHEME // HOST [":" PORT] [PATH ["?" QUERY]]:** http://localhost:8080/api/articles (highlighted with a red box)
- Send** button (highlighted with a red box)
- QUERY PARAMETERS:** None
- HEADERS:** Content-Type: application/json (checked and highlighted with a red box)
- BODY:** A JSON object with the following content:

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

Response (highlighted with a red box):

200

HEADERS:

- 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

BODY:

```
{  
  "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



- 응답 결과 확인

The screenshot shows the 'Response' tab 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 the message '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. 타겟 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)
- SCHEME // HOST [":" PORT] [PATH "?" QUERY]:** http://localhost:8080/api/articles/1 (highlighted with a red box)
- Send button:** A blue button with a white arrow icon labeled "Send" (highlighted with a red box).
- QUERY PARAMETERS:** None listed.
- HEADERS:** Content-Type: application/json (selected, highlighted with a red box). Other options include Form and Text.
- BODY:** Text tab selected. The JSON payload is: { "id": 1, "content": "5678" } (highlighted with a red box).

- 응답 결과 확인

The screenshot shows the REST client displaying the response to the update request:

- Response:** 200
- HEADERS:** Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Date: Tue, 02 Dec 2025 10:29:48 GMT -1s
Keep-Alive: timeout=60
Connection: keep-alive
- BODY:** The response body is a JSON object: { "id": 1, "title": "가가가가", "content": "5678" } (highlighted with a red box). The "pretty" dropdown is set to "pretty".

게시글 삭제 요청 개선하기

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

```
// DELETE
@GetMapping("/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



- 응답 확인

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

- Response:** 204 (highlighted with a green bar)
- HEADERS:** Date: Tue, 02 Dec 2025 10:45:54 GMT, Keep-Alive: timeout=60, Connection: keep-alive
- BODY:** No Content (displayed in large white text)

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 를 엔티티 를 으로 변환하기  
    // 2. 엔티티 를 으로 DB에 저장하기  
    // 3. 강제 예외 발생시키기  
    // 4. 결과 값 반환하기  
}
```

트랜잭션 맛보기

- 1. dto 뮤음을 엔티티 뮤음으로 변환하기

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

service/ArticleService.java

```
public List<Article> createArticles(List<ArticleForm> dtos) {  
    // 1. dto 뮤음을 엔티티 뮤음으로 변환하기  
    List<Article> articleList = dtos.stream() Stream<ArticleForm>  
        .map( ArticleForm dto -> dto.toEntity() ) Stream<Article>  
        .collect(Collectors.toList());
```

// 2. 엔티티 뮤음을 DB에 저장하기

// 3. 강제 예외 발생시키기

// 4. 결과 값 반환하기

}

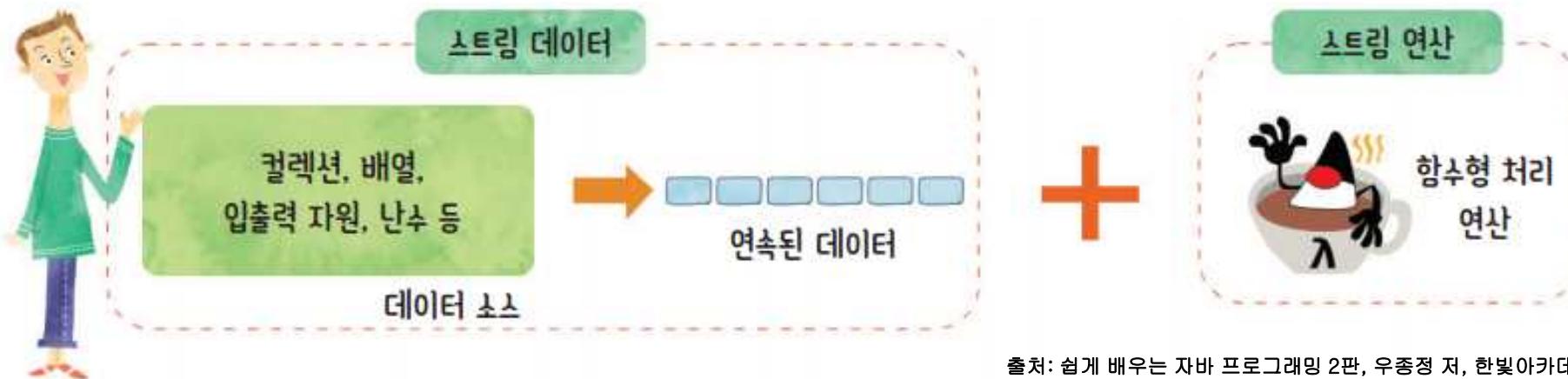
```
List<Article> articleList = new ArrayList<>();  
for(int i=0; i<dtos.size(); i++){  
    ArticleForm dto = dtos.get(i);  
    Article entity = dto.toEntity();  
    articleList.add(entity);  
}
```



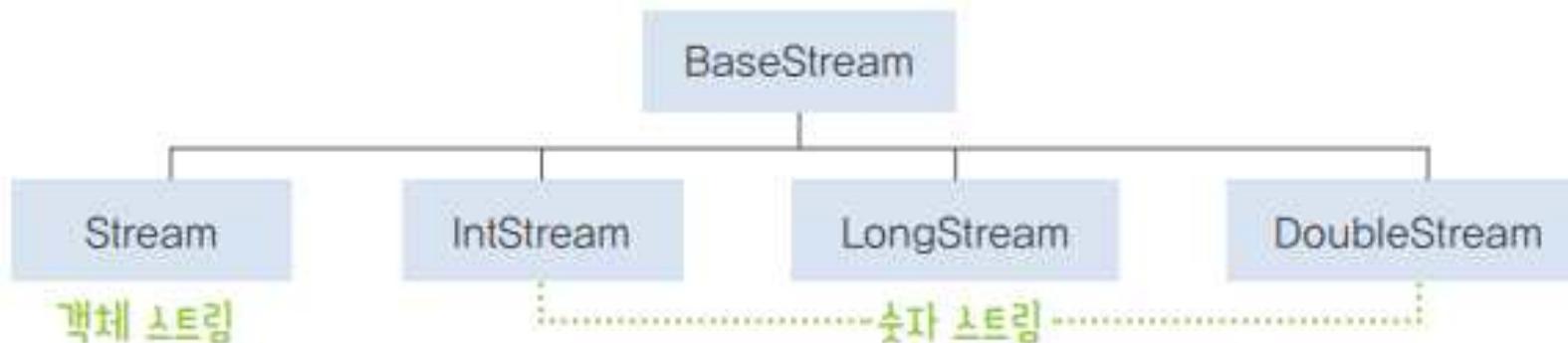
Java의 Stream

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



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, peek() - 중간 연산

- 단순 집계 연산

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

- 컬렉터 연산

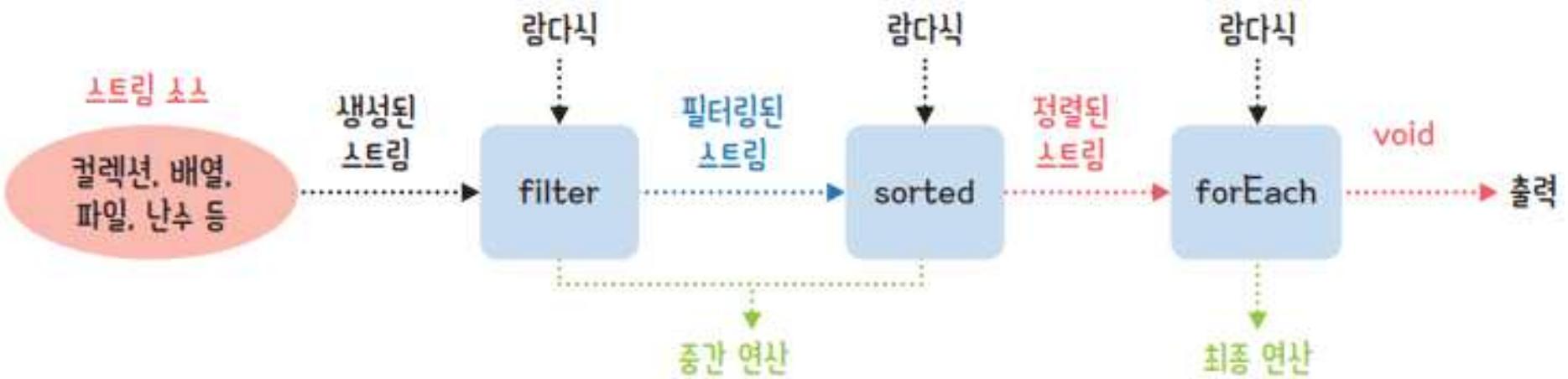
- ✓ 연산의 실행결과를 컬렉션에 수집할 수 있도록 함

- ...

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

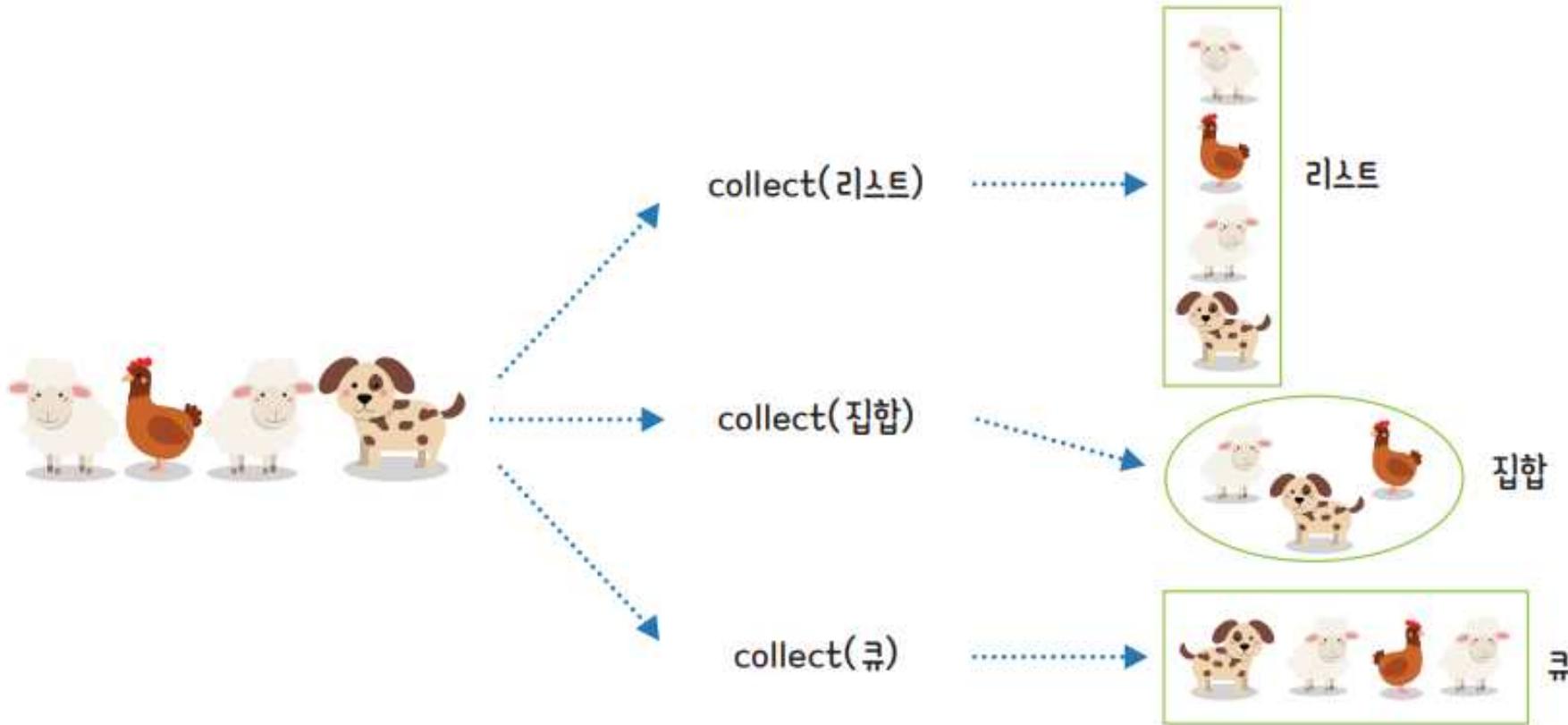


출처: 쉽게 배우는 자바 프로그래밍 2판, 우종정 저, 한빛아카데미

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

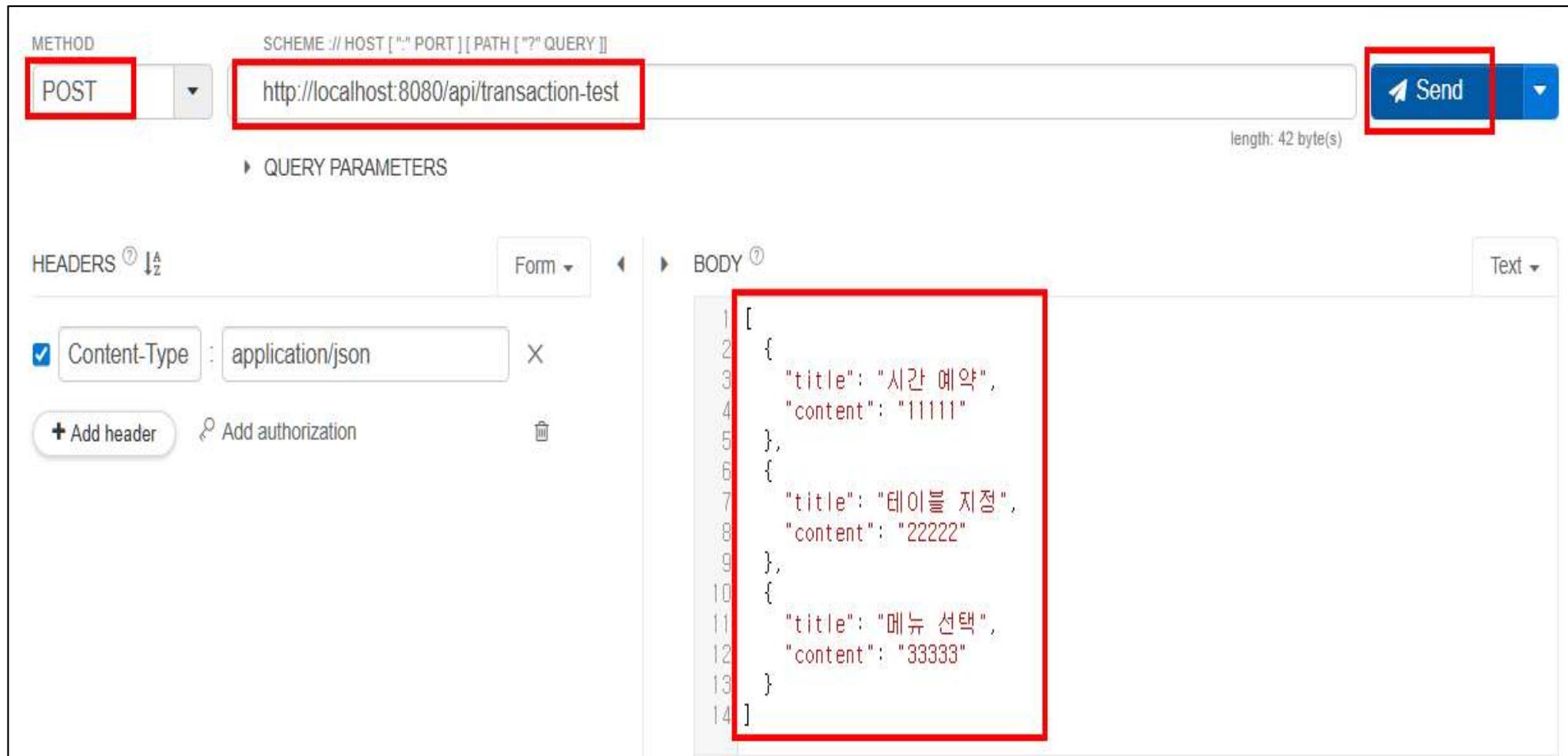
● 4. 결과 값 반환하기

service/ArticleService.java

```
public List<Article> createArticles(List<ArticleForm> dtos) {  
    // 1. dto 륙음을 엔티티 륙음으로 변환하기  
    List<Article> articleList = dtos.stream() Stream<ArticleForm>  
        .map( ArticleForm dto -> dto.toEntity()) Stream<Article>  
        .collect(Collectors.toList());  
    // 2. 엔티티 륙음을 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



트랜잭션 맛보기

- 응답 결과 확인

Response

500

HEADERS ?

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

pretty ▼

BODY ?

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

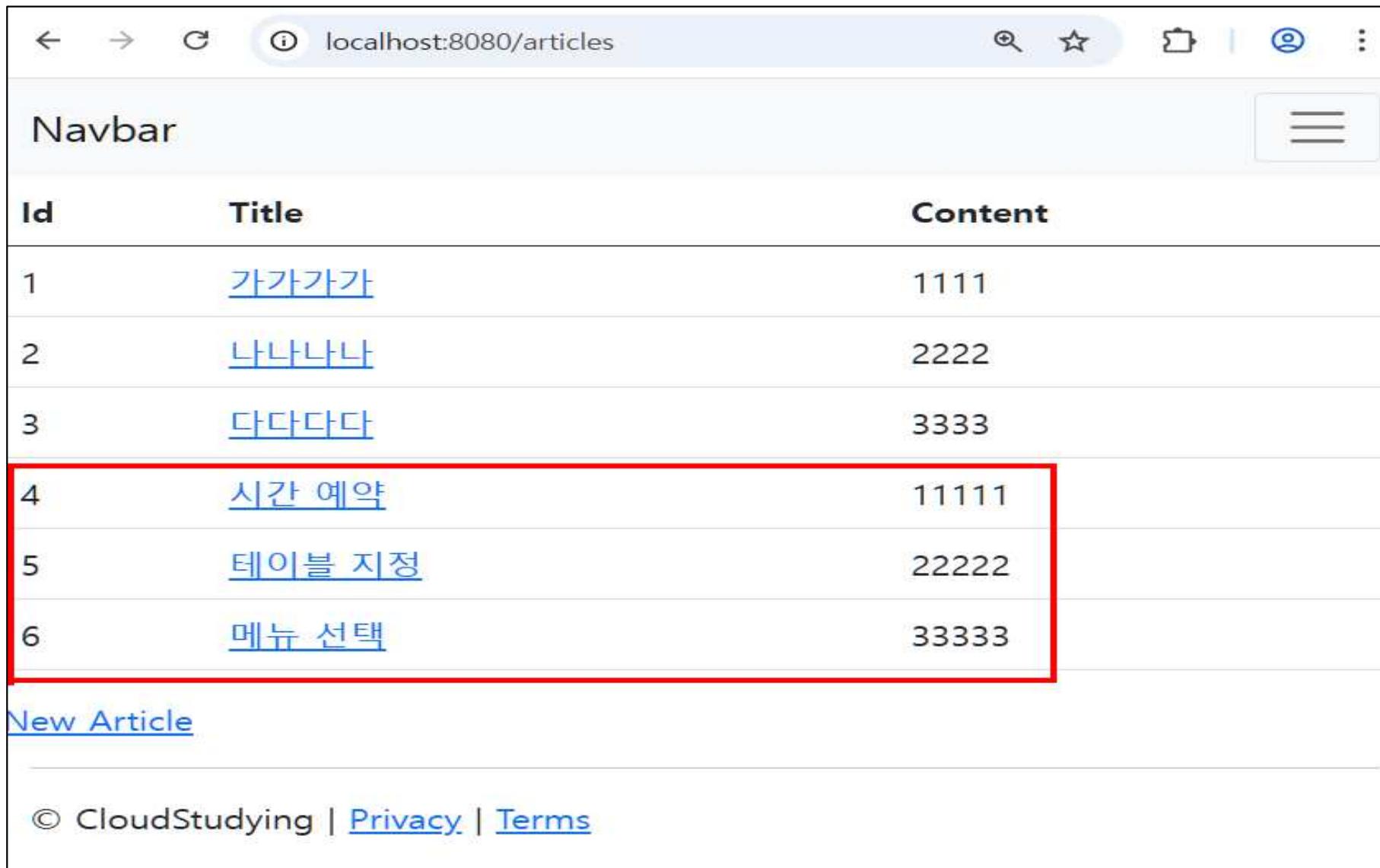
- 로그에서 에러 원인 탐색

```
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



A screenshot of a web browser displaying a table of articles at the URL <http://localhost:8080/articles>. The browser interface includes a back/forward button, refresh, search, and other standard icons. The page has a header "Navbar" and a table with columns "Id", "Title", and "Content". Rows 4, 5, and 6 are highlighted with a red border.

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 되돌아감

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

public class ArticleService {
    (중략)

    @Transactional
    public List<Article> createArticles(List<ArticleForm> dtos) {
        // 1. dto 를 엔티티 를 으로 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 를 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. The top bar has 'METHOD' set to 'POST', 'SCHEME // HOST [:: PORT] [PATH ? QUERY]' set to 'http://localhost:8080/api/transaction-test', and a 'Send' button. A red box highlights the 'POST' method, the URL, and the 'Send' button. Below the top bar, there's a section for 'QUERY PARAMETERS'. The 'HEADERS' section shows 'Content-Type: application/json' with a checked checkbox, and buttons for '+ Add header' and 'Add authorization'. The 'BODY' section is set to 'Text' and contains a JSON array with three objects:

```
[{"title": "시간 예약", "content": "11111"}, {"title": "테이블 지정", "content": "22222"}, {"title": "메뉴 선택", "content": "33333"}]
```

A red box highlights the entire JSON body area.

트랜잭션 맛보기

- 응답 결과 확인

Response

500

HEADERS ?

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

pretty ▼

BODY ?

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

▶ COMPLETE REQUEST HEADERS

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

localhost:8080/articles

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

[New Article](#)

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