본 문서는 과제에 대한 결과보고서입니다. 구체적인 프로그램 설계, 예외 상황과 테스트 케이스, 구현 결과에 대한 스크린샷 등을 확인하실 수 있습니다.

1. 서론

About

본 프로젝트는 유어슈 백엔드 지원 과제로서 수행한 간단한 블로그 만들기 프로젝트입니다.

Spring core, Spring boot, Spring mvc, JPA, Mysql, Slf4j, Junit 등을 사용하여 개발했습니다.

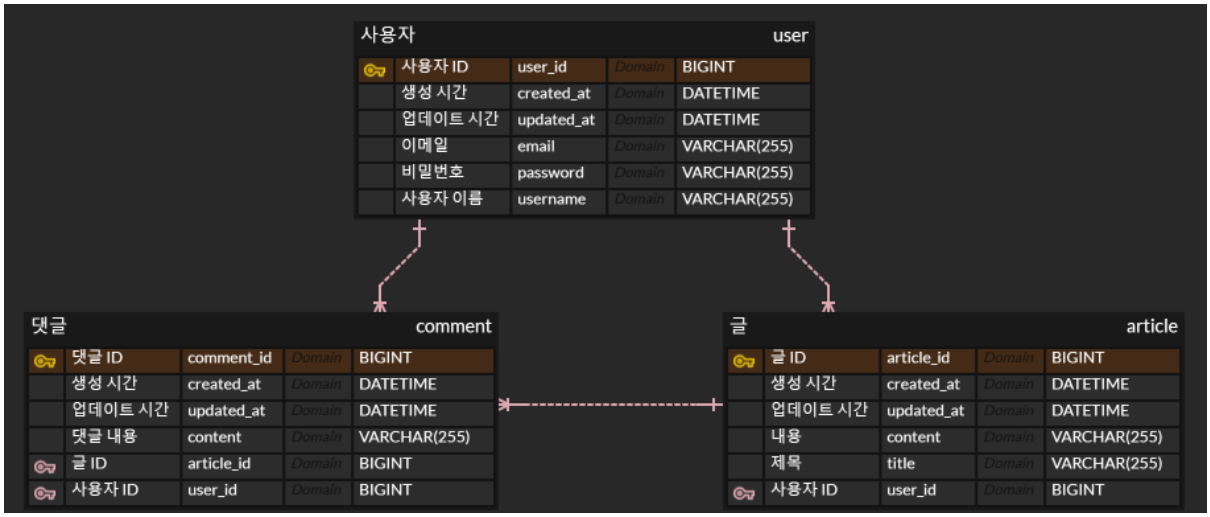
구현 기능

회원가입, 회원 삭제, 로그인, 로그아웃, 게시글 등록/수정/삭제, 댓글 등록/수정/삭제를 구현했습니다.

2. 설계

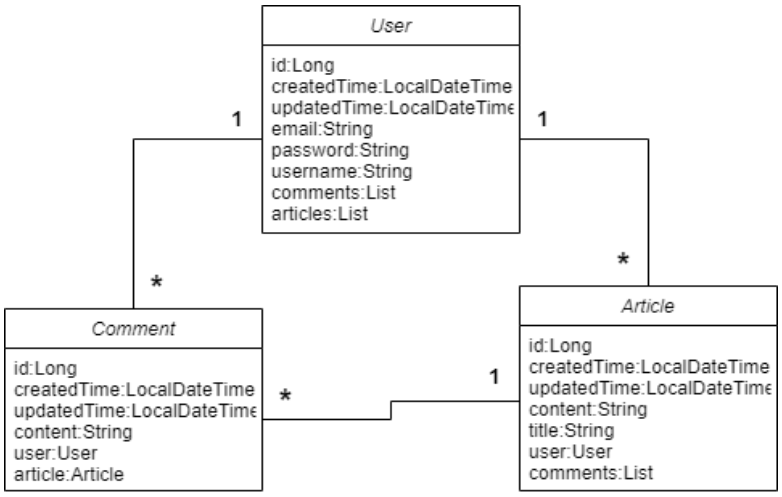
ERD

ERD는 아래와 같습니다.



엔티티 구조

엔티티 구조는 아래와 같습니다.



API 설계

API 설계는 아래와 같습니다.

/users

POST : 회원가입

/users/login

POST : 로그인

/users/logout

GET : 로그아웃

/users/{userId}

DELETE : 회원 탈퇴. 관련 게시물, 댓글 삭제

/posts

POST : 게시글 등록

/posts/{articleId}

PUT : 게시글 수정

DELETE : 게시글, 관련 댓글 삭제

/posts/{articleId}/comments

POST : 댓글 등록

/comments/{commentId}

PUT : 댓글 수정

DELETE : 댓글 삭제

요청/응답 정의

요청/응답 메시지 바디에 작성해야 하는 json은 아래와 같이 정의했습니다. 기본적으로 제공된 명세와 동일하지만, **로그인 구현에 따라 등록/삭제/수정 요청에서 email과 password 필드를 제외했습니다**.

회원가입

Request	Response

{
"email" : "email@urssu.com",
"password" : "password",
"username" : "username"
}
	{
	"email" : "email@urssu.com",
	"username" : "username"
	}

로그인

Request	Response

{
"email" : "email@urssu.com",
"password" : "password",
}
	{
	"email" : "email@urssu.com",
	"username" : "username"
	}

로그아웃

Request	Response
x	x

회원 삭제

Request	Response
x	x

게시글 등록

Request	Response

{
"title" : "title",
"content" : "content",
}
	{
	"articleId": 1,
	"email": "email@urssu.com"
	"title" : "title",
	"content" : "content",
	}

게시글 수정

Request	Response

{
"title" : "title",
"content" : "content",
}
	{
	"articleId": 1,
	"email": "email@urssu.com"
	"title" : "title",
	"content" : "content",
	}

게시글 삭제

Request	Response
x	x

댓글 등록

Request	Response

{
"content" : "content",
}
	{
	"commentId": 1,
	"email": "email@urssu.com"
	"content" : "content",
	}

댓글 수정

Request	Response

{
"content" : "content",
}
	{
	"commentId": 1,
	"email": "email@urssu.com"
	"content" : "content",
	}

댓글 삭제

Request	Response
x	x

DTO

컨트롤러에서는 아래와 같이 파라미터와 반환값에 사용할 DTO를 정의하여 사용했습니다. 입력에 대한 DTO에는 spring validation을 적용했습니다.

회원가입 → 파라미터(UserSignupRequestDto), 반환값(UserResponseDto)
로그인 → 파라미터(UserLoginRequestDto), 반환값(UserResponseDto)
게시글 등록, 수정 → 파라미터(ArticleRequestDto), 반환값(ArticleResponseDto)
댓글 등록, 수정 → 파라미터(CommentRequestDto), 반환값(CommentResponseDto)
예외 응답 → 반환값(ExceptionResponseDto)

로그인 설계

로그인은 아래의 방식으로 설계했습니다.

로그인 관련 정의
로그인 하지 않은 경우(세션이 존재하지 않는 경우)에는 필터를 사용하여 /users(회원가입)와 /users/login(로그인)에만 접근이 가능하도록 했습니다.

로그인 상태라면 게시글, 댓글 등록/수정/삭제에 별도의 인증이 필요하지 않는 것으로 하고, 이에 따라 명세의 리소스 등록/수정/삭제 요청에서 email, password 필드는 제외하는 것으로 정의했습니다.

로그인하면 세션이 생성되고, 세션 attribute에 id, email, username을 저장하여 이를 응답 작성, 권한 확인에 사용하도록 했습니다.

권한 확인
로그인 중이어도 자신의 소유가 아닌 리소스에 접근해서는 안 되므로, 해당 리소스 소유자의 id(db에서 확인)와 세션의 id(로그인 중인 사용자)를 비교하여 권한을 확인하도록 했습니다.

예외

아래의 예외들을 사용하여 에러 응답을 전송하도록 했습니다. 예외는 @RestControllerAdvice, @ExceptionHandler를 사용해 처리했습니다.

사용자 정의 예외.
public abstract class BlogException extends RuntimeException // 상위 타입

public class BlogResourceNotFoundException extends BlogException // 리소스가 존재하지 않음
public class BlogUserNotFoundException extends BlogException // *사용자가 존재하지 않는 경우*
public class BlogNotAuthorizedException extends BlogException // 리소스 접근 권한 없음

spring, jpa 예외.
Spring validation 위반 예외 : MethodArgumentNotValidException
db 제약 조건 위반 예외 : DataIntegrityViolationException (DataAccessException의 하위 타입.)

3. 구현

작업 순서

구현 시에는 아래와 같은 작업 순서로 개발했습니다.

1. ERD, 엔티티 구조 작성.

2. API 설계.

3. 프로그램 설계. (예외 상황, 예외, 요청/응답, 로그인, DTO 등)

4. 프로젝트 생성, 깃허브 연결.

5. 코드 작성.
→ 엔티티(연관관계, 영속성 전이, jpa 제약 조건, 검증 등 고려) 코드 작성.
→ repository, service, controller를 기능 정의, 코드 작성, 예외 처리, 테스트 코드 작성의 순서로 개발.
→ 필터로 로그아웃 상태 접근 처리

6. 결과보고서 작성

패키지 구성

프로젝트의 핵심 코드는 src.main.java.springproject.urssublog에 있고, 해당 위치의 패키지 구성은 아래와 같습니다.

controller : 각 도메인 별 controller 클래스가 있습니다.
service : 각 도메인 별 service 클래스가 있습니다.
repository : 각 도메인 별 repository 클래스가 있습니다.
domain : 각 도메일 별 엔티티 클래스가 있습니다.
dto : 요청/응답 처리 시에 사용하는 DTO 클래스가 있습니다.
exception : 사용자 정의 예외와, 예외 처리 클래스가 있습니다.
filter : 필터 관련 클래스들이 있습니다.

예외 상황

고려된 예외 상황은 아래와 같습니다. 설계 시에 정의한 예외를 던지도록 했고, 이렇게 던져진 예외는 @RestControllerAdvice, @ExceptionHandler를 사용해 공통적으로 처리했습니다.

회원
1. 회원가입 정보(email, username)가 겹치는 경우.
→ DataIntegrityViolationException

2. 회원가입 시 spring validation 검증(@NotBlank, @Size, @Pattern)에 위배된 경우.
→ MethodArgumentNotValidException

3. 로그인 시 회원 정보가 없는 경우.
→ BlogUserNotFoundException

4. 로그인 시 spring validation 검증(@NotBlank, @Size, @Pattern)에 위배된 경우.
→ MethodArgumentNotValidException

5. 회원 탈퇴(삭제) 시 해당 리소스가 로그인 중인 회원의 리소스가 아닌 경우,
→ BlogNotAuthorizedException

게시글
1. 게시글 등록 시 spring validation 검증(@NotBlank, @Size)에 위배된 경우.
→ MethodArgumentNotValidException

2. 게시글 수정 시 spring validation 검증(@NotBlank, @Size)에 위배된 경우.
→ MethodArgumentNotValidException

3. 게시글 수정 시 해당 리소스가 로그인 중인 회원의 리소스가 아닌 경우,
해당 id의 게시글이 존재하지 않는 경우.
→ BlogNotAuthorizedException

4. 게시글 삭제 시 해당 리소스가 로그인 중인 회원의 리소스가 아닌 경우,
해당 id의 게시글이 존재하지 않는 경우.
→ BlogNotAuthorizedException

댓글
1. 댓글 등록 시 해당 리소스(게시글)가 로그인 중인 회원의 리소스가 아닌 경우,
해당 id의 게시글이 존재하지 않는 경우.
→ BlogNotAuthorizedException

2. 댓글 등록 시 spring validation 검증(@NotBlank, @Size)에 위배된 경우.
→ MethodArgumentNotValidException

3. 댓글 수정 시 해당 리소스가 로그인 중인 회원의 리소스가 아닌 경우,
해당 id의 댓글이 존재하지 않는 경우.
→ BlogNotAuthorizedException

4. 댓글 수정 시 spring validation 검증(@NotBlank, @Size)에 위배된 경우.
→ MethodArgumentNotValidException

5. 댓글 삭제 시 해당 리소스(게시글)가 로그인 중인 회원의 리소스가 아닌 경우,
해당 id의 댓글이 존재하지 않는 경우.
→ BlogNotAuthorizedException

테스트 케이스

정상적인 경우와 예외 상황에 대해 아래와 같이 테스트를 수행했습니다.

repository, service, controller 순으로 각각 테스트를 작성해서 겹치는 경우가 좀 있긴 한데, 각 계층에서 검

증해야 한다고 판단한 부분을 검증했습니다.

repository

회원가입 → 정상적인 경우 / 회원 예외 상황 1
회원 삭제 → 정상적인 경우 (관련 게시글, 댓글도 삭제 확인.)

게시글 등록 → 정상적인 경우
게시글 수정 → 정상적인 경우
게시글 삭제 → 정상적인 경우

댓글 등록 → 정상적인 경우
댓글 수정 → 정상적인 경우
댓글 삭제 → 정상적인 경우

service

비밀번호 암호화

회원가입 → 정상적인 경우
로그인 → 정상적인 경우 / 회원 예외 상황 3
회원 삭제 → 정상적인 경우 / 회원 예외 상황 5

게시글 등록 → 정상적인 경우
게시글 수정 → 정상적인 경우
게시글 삭제 → 정상적인 경우

댓글 등록 → 정상적인 경우
댓글 수정 → 정상적인 경우
댓글 삭제 → 정상적인 경우

controller

/users POST 회원가입 → 정상적인 경우 / 회원 예외 상황 1, 2
/users/login POST 로그인 → 정상적인 경우 / 회원 예외 상황 3, 4
/users/logout GET 로그아웃 → 정상적인 경우
/users/{userId} DELETE 회원 탈퇴(삭제) → 정상적인 경우 / 회원 예외 상황 5

/posts POST 게시물 등록 → 정상적인 경우 / 게시글 예외 상황 1
/posts/{articleId} PUT 게시물 수정 → 정상적인 경우 / 게시글 예외 상황 2, 3
/posts/{articleId} DELETE 게시물 삭제 → 정상적인 경우 / 게시글 예외 상황 4

/posts/{articleId}/comments POST 댓글 작성 → 정상적인 경우 / 댓글 예외 상황 1, 2
/comments/{commentId} → 정상적인 경우 / 댓글 예외 상황 3, 4
/comments/{commentId} → 정상적인 경우 / 댓글 예외 상황 5

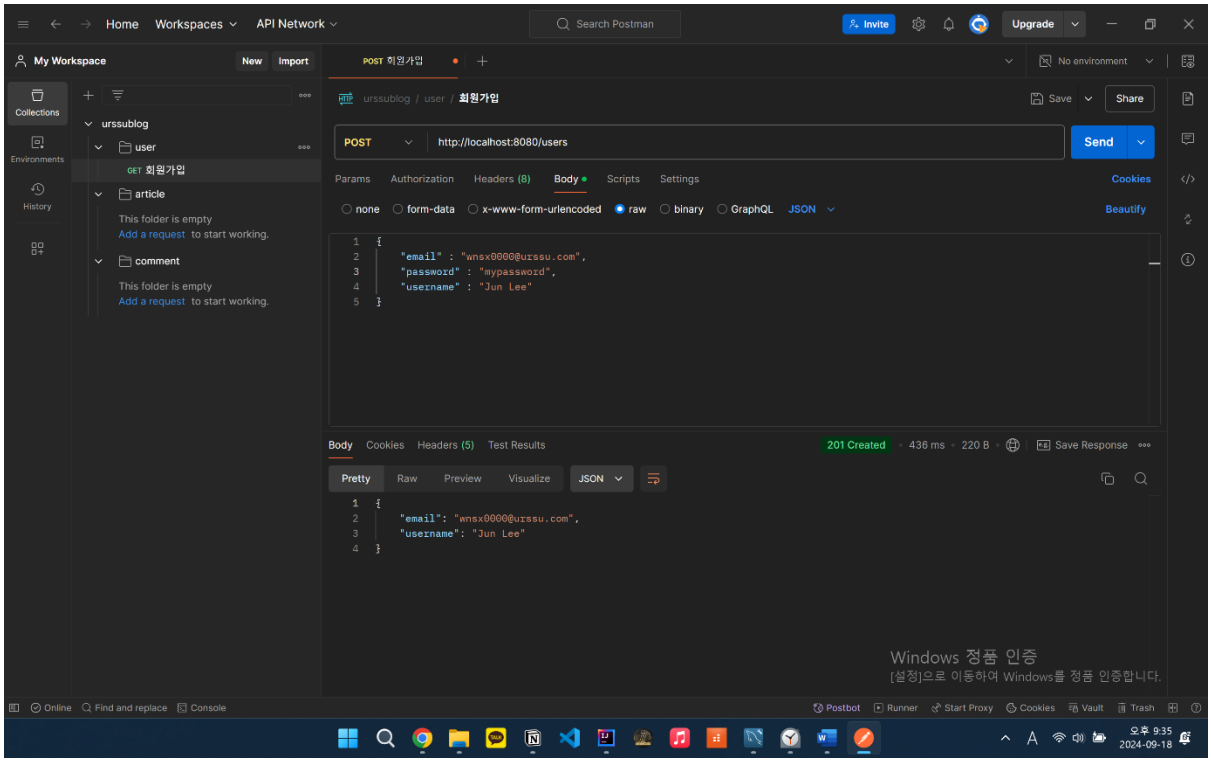
필터
로그인 상태가 아닌 경우.

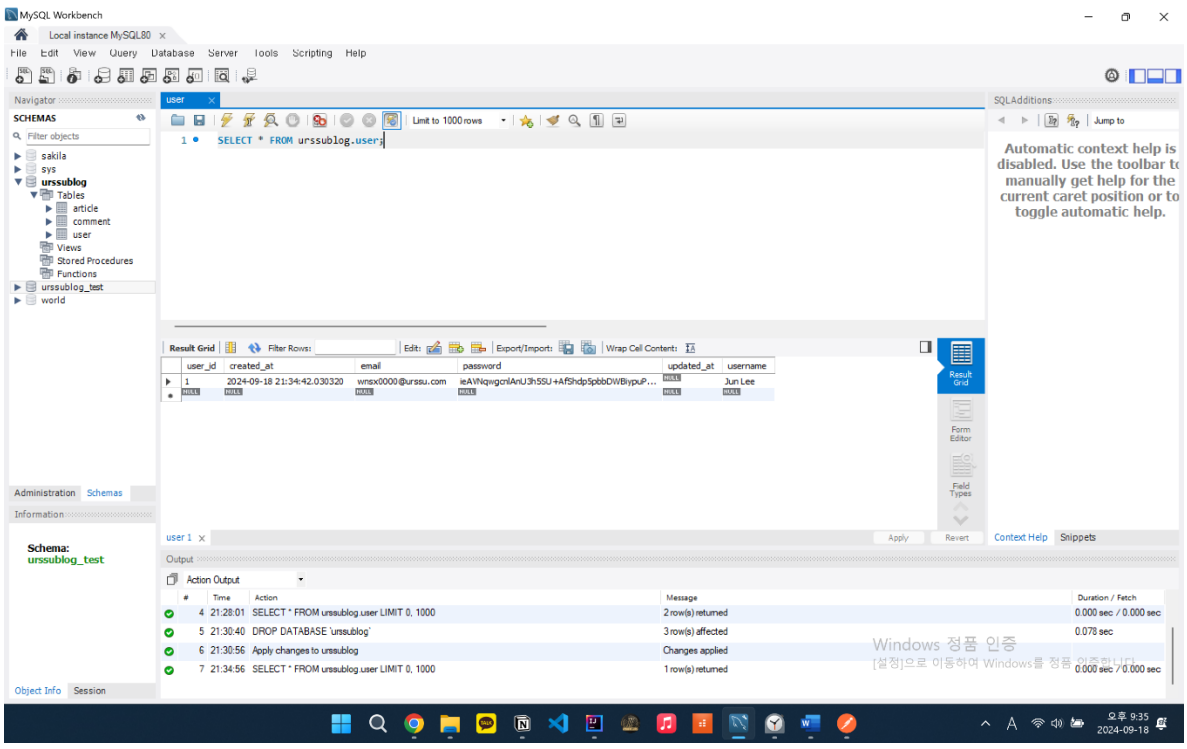
4. 결과

Postman과 Mysql workbench로 확인할 수 있는 구현 결과의 스크린샷을 정리했습니다.

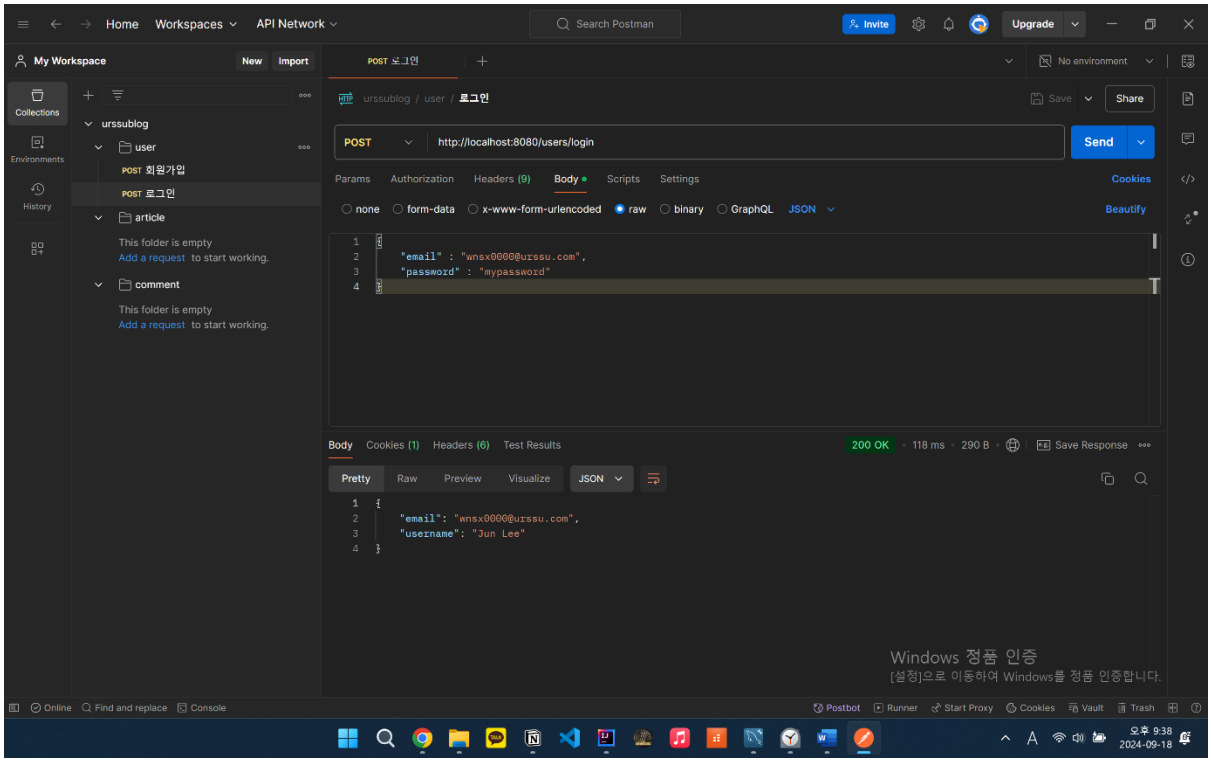
정상적인 경우

회원가입

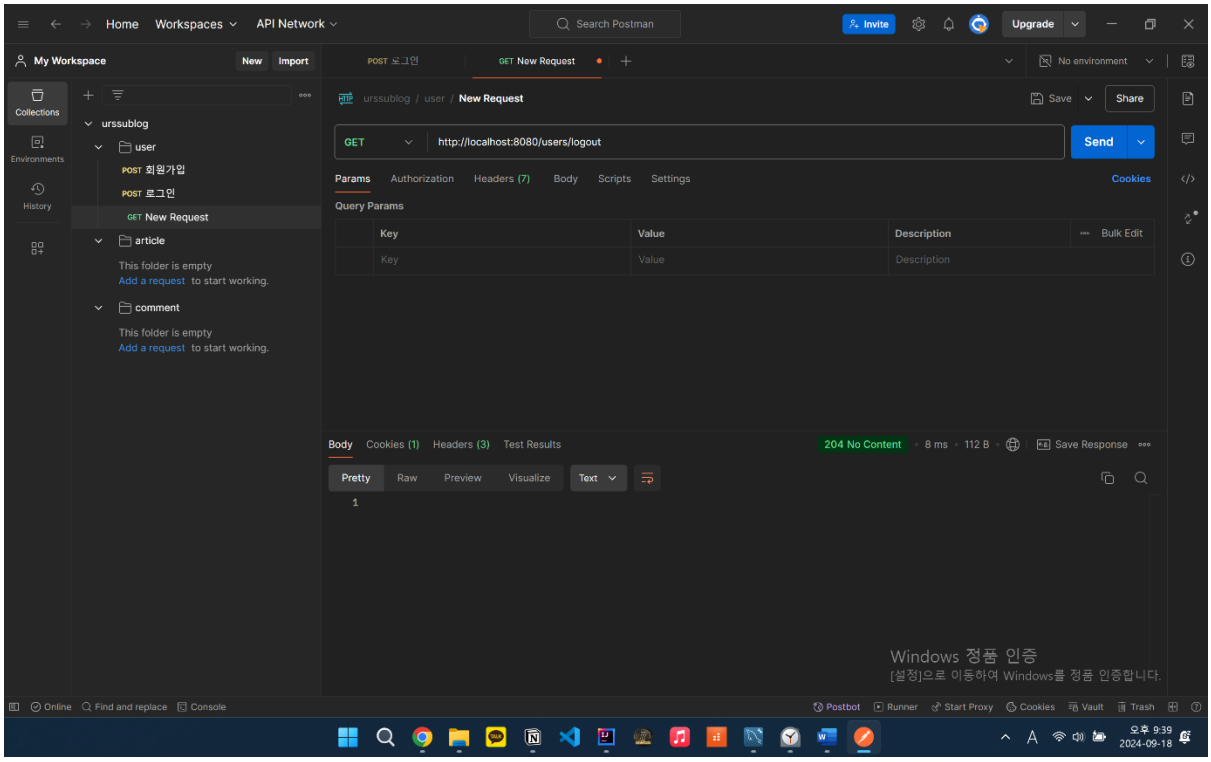




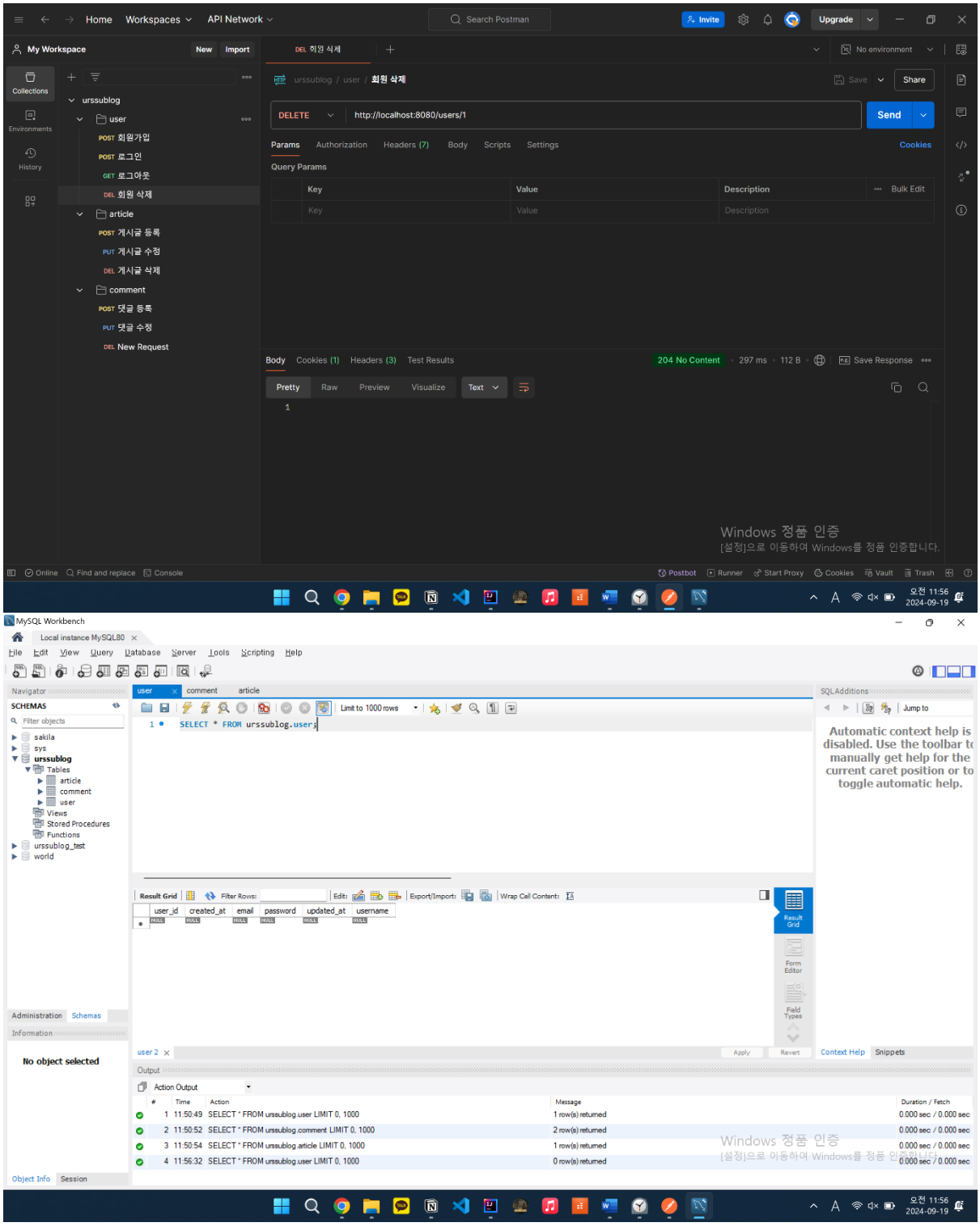
로그인


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST 로그인
No environment
Collections
Environments
History
urssublog
user
POST 회원가입
POST 로그인
article
This folder is empty
Add a request to start working.
comment
This folder is empty
Add a request to start working.
urssublog / user / 로그인
Save
Share
POST
http://localhost:8080/users/login
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"email" : "wnsx0000@urssu.com",
"password" : "mypassword"
}
Body
Cookies (1)
Headers (6)
Test Results
200 OK
118 ms
290 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"email": "wnsx0000@urssu.com",
"username": "Jun Lee"
}
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
오후 9:38
2024-09-18

로그아웃

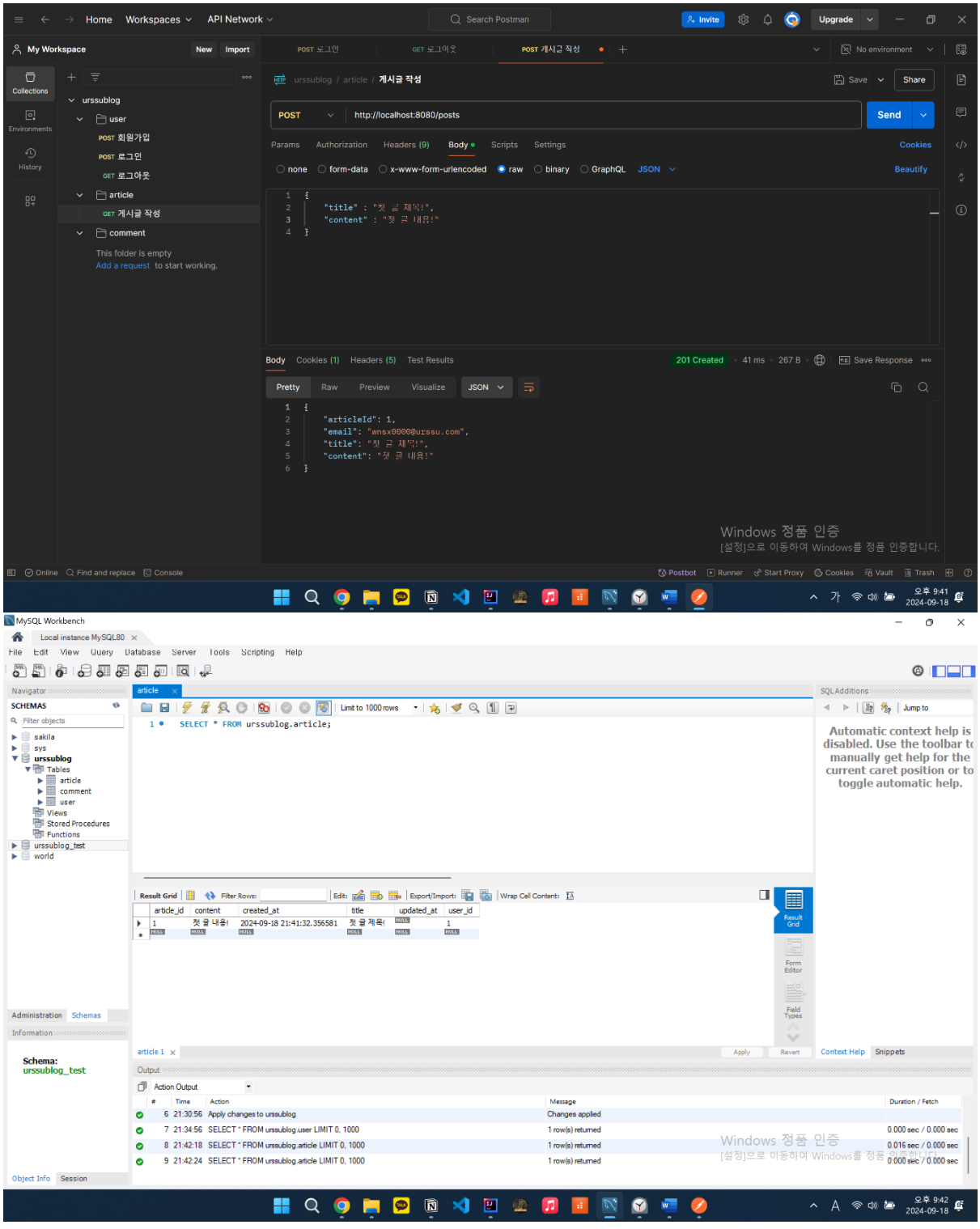

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST 로그인
GET New Request
No environment
Collections
Environments
History
urssublog
user
POST 회원가입
POST 로그인
GET New Request
article
This folder is empty
Add a request to start working.
comment
This folder is empty
Add a request to start working.
urssublog / user / New Request
Save
Share
GET
http://localhost:8080/users/logout
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies (1)
Headers (3)
Test Results
204 No Content
8 ms
112 B
Save Response
Pretty
Raw
Preview
Visualize
Text
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
오후 9:39
2024-09-18

회원 삭제


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
DEL 회원 삭제
No environment
Collections
Environments
History
urssublog / user / 회원 삭제
Save
Share
urssublog
user
POST 회원가입
POST 로그인
GET 로그아웃
DEL 회원 삭제
article
POST 게시글 등록
PUT 게시글 수정
DEL 게시글 삭제
comment
POST 댓글 등록
PUT 댓글 수정
DEL New Request
DELETE
http://localhost:8080/users/1
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies (1)
Headers (3)
Test Results
204 No Content
297 ms
112 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
오전 11:56
2024-09-19

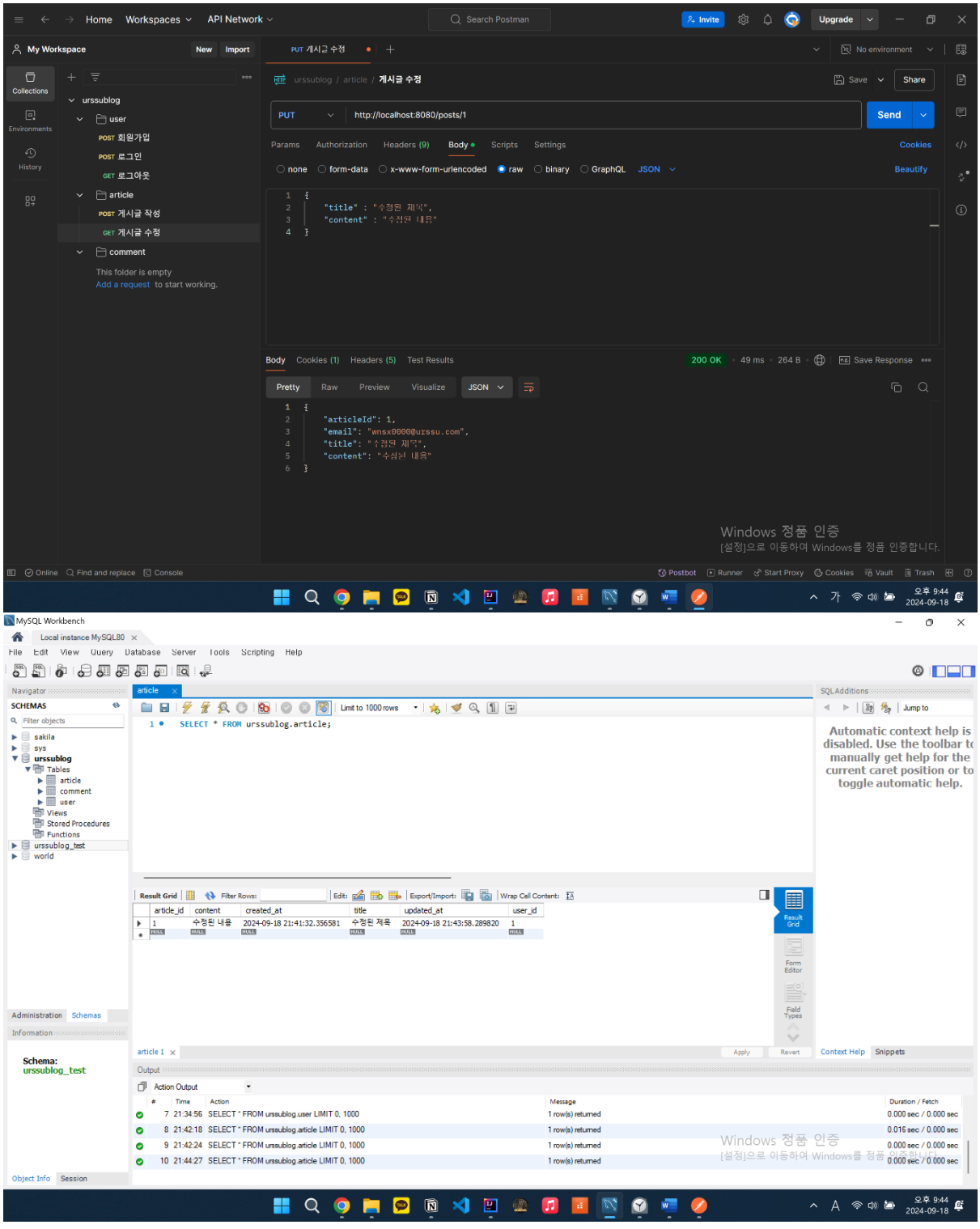
MySQL Workbench
Local instance MySQL80
File
Edit
View
Query
Database
Server
Tools
Scripting
Help
Navigator
SCHEMAS
Filter objects
sakila
sys
urssublog
Tables
article
comment
user
Views
Stored Procedures
Functions
urssublog_test
world
Administration
Schemas
Information
No object selected
Object Info
Session
user
comment
article
Limit to 1000 rows
1 SELECT * FROM urssublog.user;
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
user_id
created_at
email
password
updated_at
username
Form Editor
Field Types
user 2
Apply
Revert
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
1 11:50:49 SELECT * FROM urssublog.user LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
2 11:50:52 SELECT * FROM urssublog.comment LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
3 11:50:54 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
4 11:56:32 SELECT * FROM urssublog.user LIMIT 0, 1000 0 row(s) returned 0.000 sec / 0.000 sec
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
오전 11:56
2024-09-19

게시글 등록


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
POST 로그인
GET 로그아웃
POST 게시글 작성
No environment
Collections
Environments
History
urssublog
user
POST 회원가입
POST 로그인
GET 로그아웃
article
GET 게시글 작성
comment
This folder is empty
Add a request to start working.
urssublog / article / 게시글 작성
Save
Share
POST
http://localhost:8080/posts
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"title" : "첫 글 제목!",
"content" : "첫 글 내용!"
}
Body
Cookies (1)
Headers (5)
Test Results
201 Created
41 ms
267 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"articleId": 1,
"email": "wnsx0000@urssu.com",
"title": "첫 글 제목!",
"content": "첫 글 내용!"
}
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
오후 9:41
2024-09-18

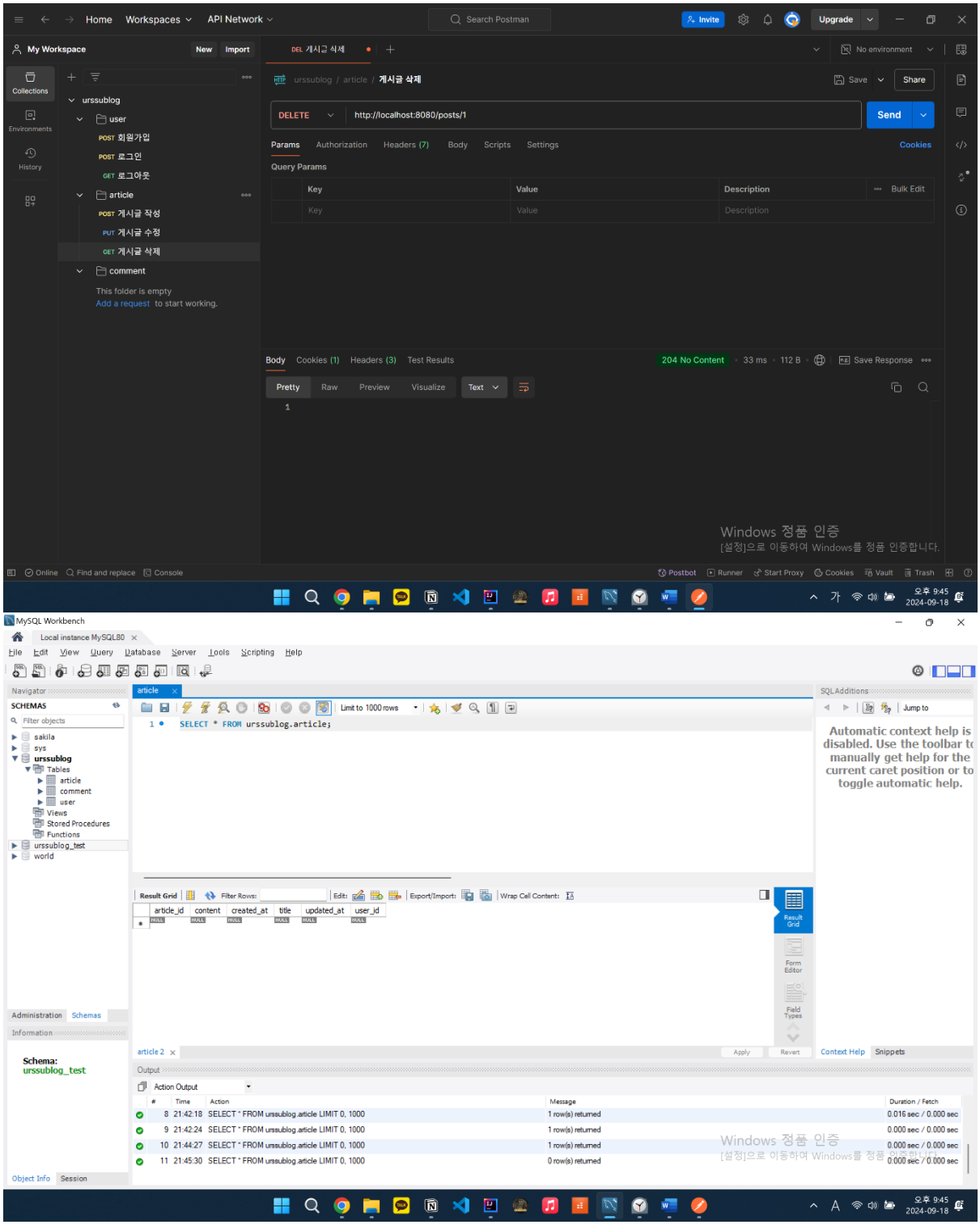
MySQL Workbench
Local instance MySQL80
File
Edit
View
Query
Database
Server
Tools
Scripting
Help
Navigator
SCHEMAS
Filter objects
sakila
sys
urssublog
Tables
article
comment
user
Views
Stored Procedures
Functions
urssublog_test
world
Administration
Schemas
Information
Schema:
urssublog_test
Object Info
Session
article
Limit to 1000 rows
SELECT * FROM urssublog.article;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
article_id
content
created_at
title
updated_at
user_id
1
첫 글 내용!
2024-09-18 21:41:32.356581
첫 글 제목!
1
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help
Snippets
article 1
Apply
Revert
Output
Action Output
Time Action Message Duration / Fetch
6 21:30:56 Apply changes to urssublog Changes applied
7 21:34:56 SELECT * FROM urssublog.user LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
8 21:42:18 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.016 sec / 0.000 sec
9 21:42:24 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
오후 9:42
2024-09-18

게시글 수정


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
PUT 게시글 수정
No environment
urssublog / article / 게시글 수정
Save
Share
Collections
Environments
History
urssublog
user
POST 회원가입
POST 로그인
GET 로그아웃
article
POST 게시글 작성
GET 게시글 수정
comment
This folder is empty
Add a request to start working.
PUT
http://localhost:8080/posts/1
Send
Params
Authorization
Headers (9)
Body
Scripts
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"title" : "수정된 제목",
"content" : "수정된 내용"
}
Body
Cookies (1)
Headers (5)
Test Results
200 OK
49 ms
264 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"articleId": 1,
"email": "wnsx0000@urssu.com",
"title": "수정된 제목",
"content": "수정된 내용"
}
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
오후 9:44
2024-09-18

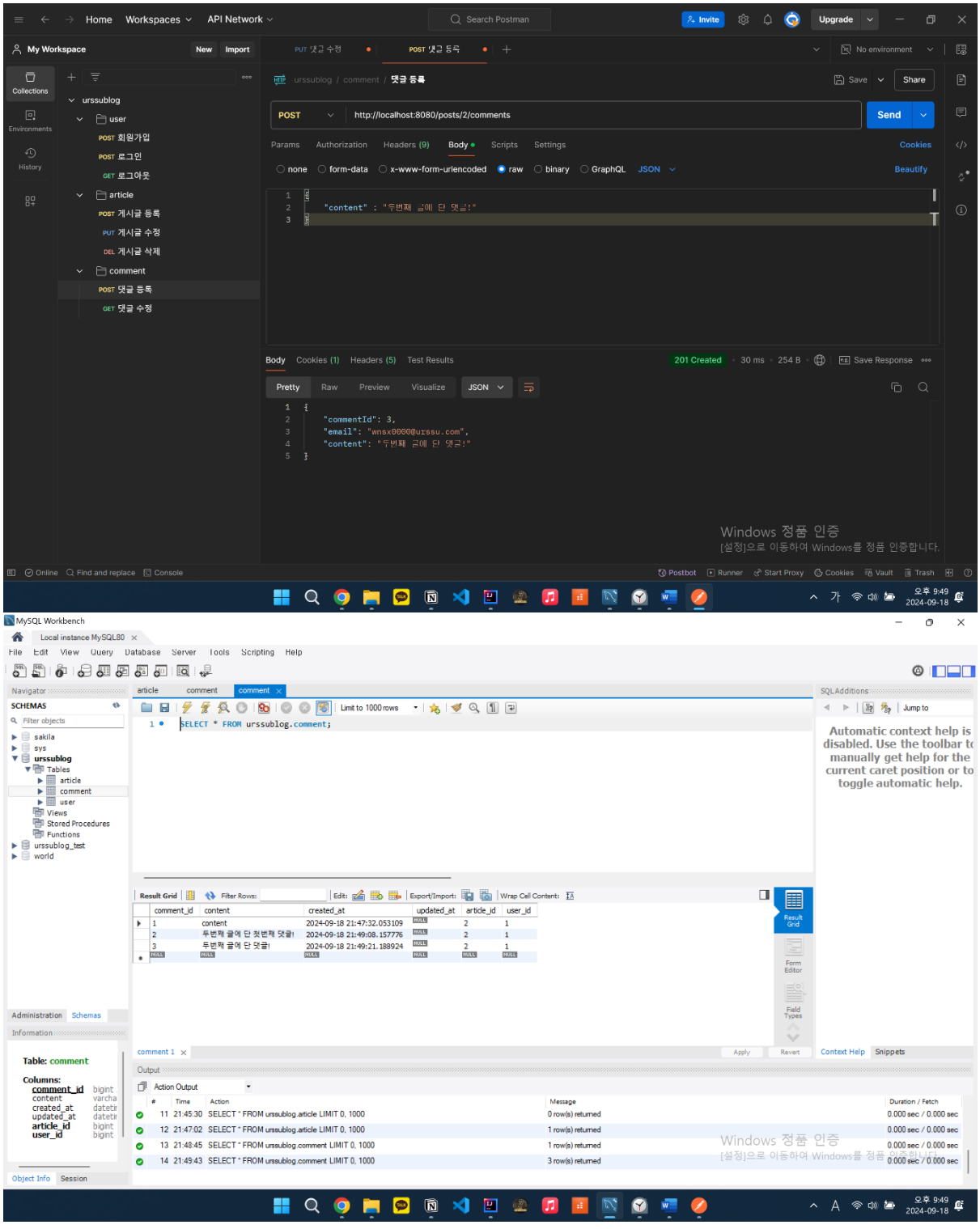
MySQL Workbench
Local instance MySQL80
File
Edit
View
Query
Database
Server
Tools
Scripting
Help
Navigator
SCHEMAS
Filter objects
sakila
sys
urssublog
Tables
article
comment
user
Views
Stored Procedures
Functions
urssublog_test
world
Administration
Schemas
Information
Schema:
urssublog_test
Object Info
Session
article
Limit to 1000 rows
SELECT * FROM urssublog.article;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
article_id
content
created_at
title
updated_at
user_id
1
수정된 내용
2024-09-18 21:41:32.356581
수정된 제목
2024-09-18 21:43:58.289820
1
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help
Snippets
article 1
Apply
Revert
Output
Action Output
Time
Action
Message
Duration / Fetch
7 21:34:56 SELECT * FROM urssublog.user LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
8 21:42:18 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.016 sec / 0.000 sec
9 21:42:24 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
10 21:44:27 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
오후 9:44
2024-09-18

게시글 삭제

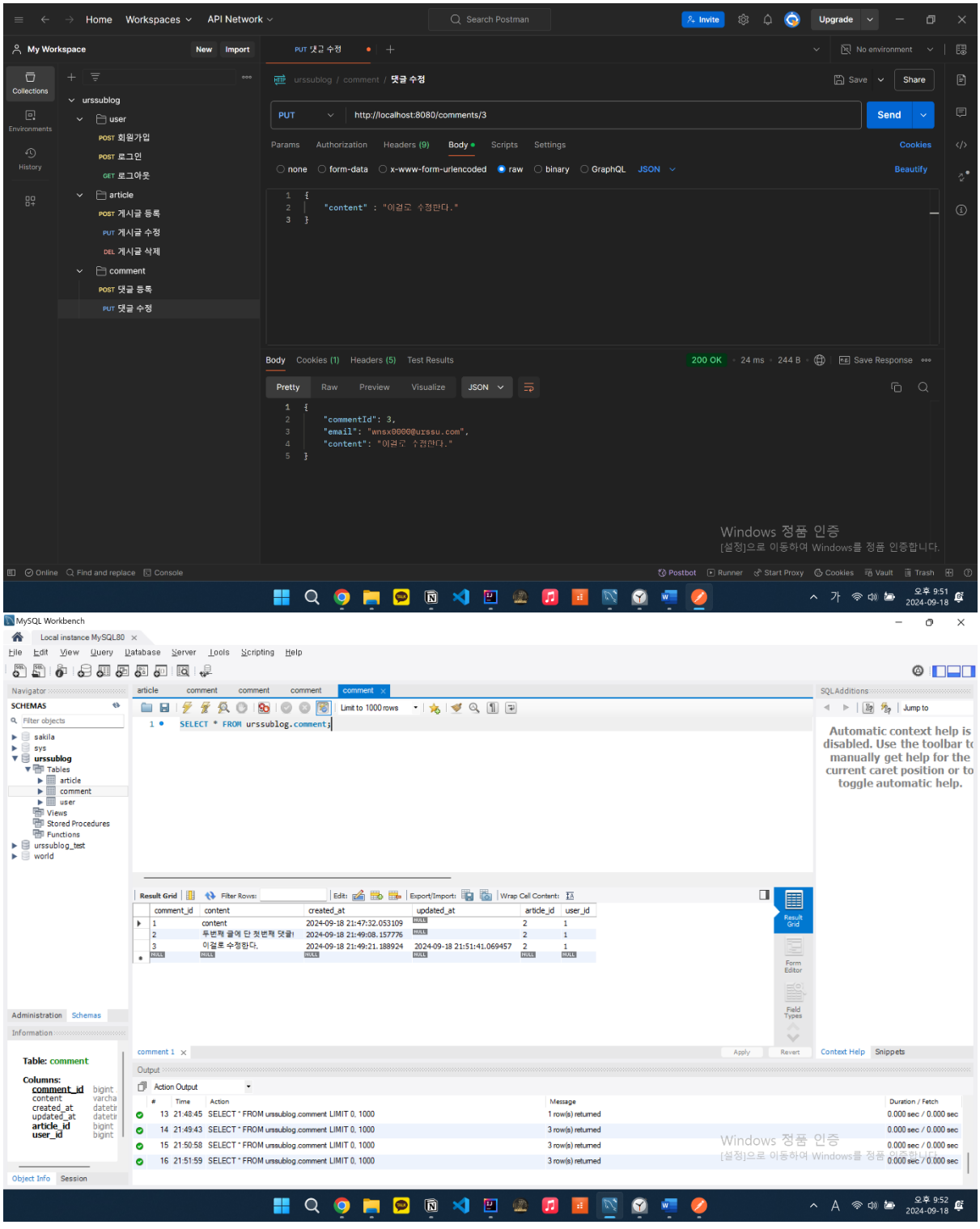

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
DEL 게시글 삭제
No environment
Collections
Environments
History
urssublog / article / 게시글 삭제
Save
Share
urssublog
user
POST 회원가입
POST 로그인
GET 로그아웃
article
POST 게시글 작성
PUT 게시글 수정
GET 게시글 삭제
comment
This folder is empty
Add a request to start working.
DELETE
http://localhost:8080/posts/1
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies (1)
Headers (3)
Test Results
204 No Content
33 ms
112 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
오후 9:45
2024-09-18

MySQL Workbench
Local instance MySQL80
File
Edit
View
Query
Database
Server
Tools
Scripting
Help
Navigator
SCHEMAS
Filter objects
sakila
sys
urssublog
Tables
article
comment
user
Views
Stored Procedures
Functions
urssublog_test
world
Administration
Schemas
Information
Schema:
urssublog_test
Object Info
Session
article
Limit to 1000 rows
1 SELECT * FROM urssublog.article;
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
article_id
content
created_at
title
updated_at
user_id
Result Grid
Form Editor
Field Types
article 2
Apply
Revert
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
8 21:42:18 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.016 sec / 0.000 sec
9 21:42:24 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
10 21:44:27 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
11 21:45:30 SELECT * FROM urssublog.article LIMIT 0, 1000 0 row(s) returned 0.000 sec / 0.000 sec
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
오후 9:45
2024-09-18

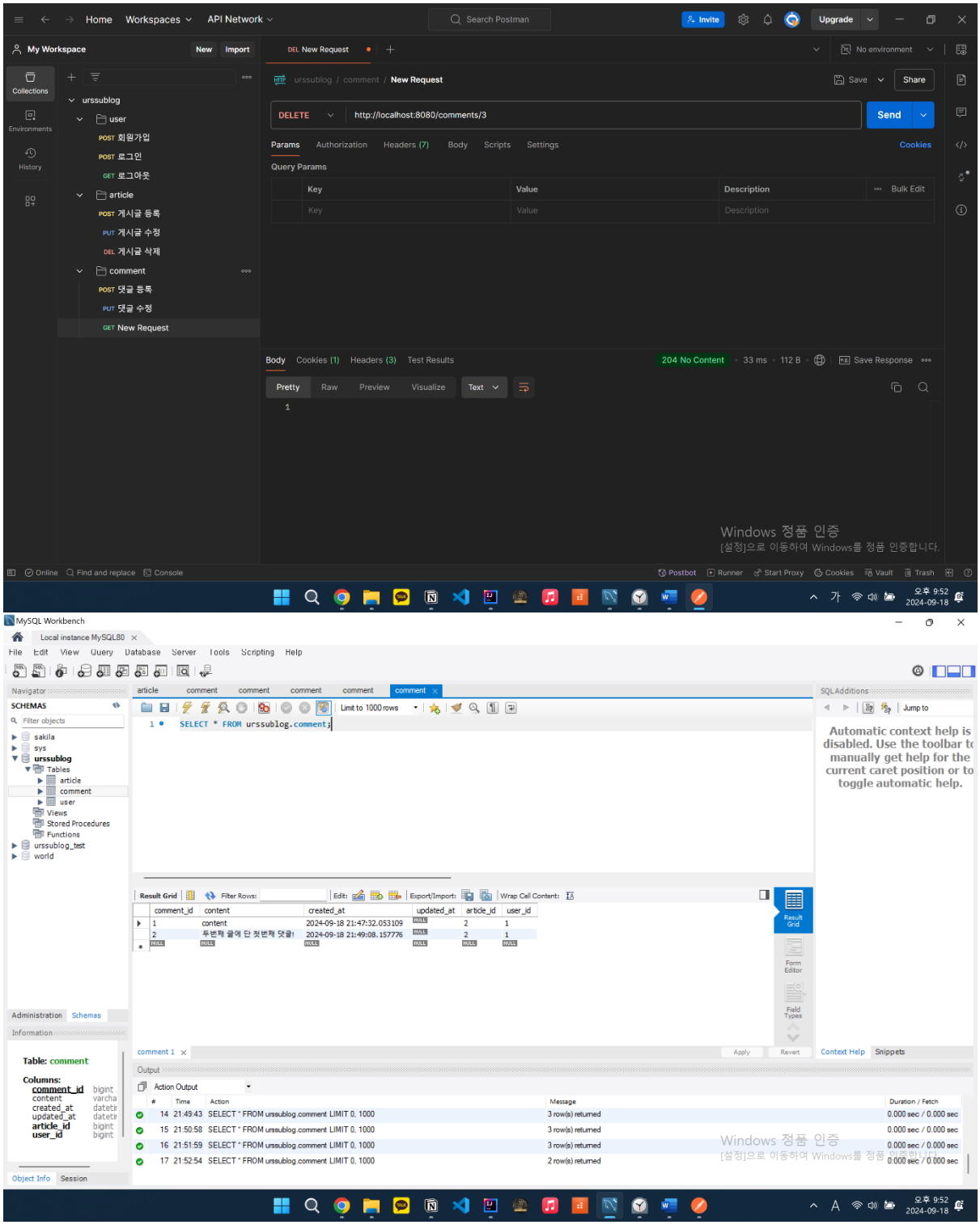
댓글 등록


POST
http://localhost:8080/posts/2/comments
Send
"content" : "두번째 글에 단 댓글!"
201 Created
"commentId": 3,
"email": "wnsx0000@urssu.com",
"content": "두번째 글에 단 댓글!"
SELECT * FROM urssublog.comment;
comment_id content created_at updated_at article_id user_id
1 content 2024-09-18 21:47:32.053109 2 1
2 두번째 글에 단 첫번째 댓글! 2024-09-18 21:49:08.157776 2 1
3 두번째 글에 단 댓글! 2024-09-18 21:49:21.188924 2 1
Table: comment
Columns:
comment_id
content
created_at
updated_at
article_id
user_id
11 21:45:30 SELECT * FROM urssublog.article LIMIT 0, 1000 0 row(s) returned
12 21:47:02 SELECT * FROM urssublog.article LIMIT 0, 1000 1 row(s) returned
13 21:48:45 SELECT * FROM urssublog.comment LIMIT 0, 1000 1 row(s) returned
14 21:49:43 SELECT * FROM urssublog.comment LIMIT 0, 1000 3 row(s) returned

댓글 수정


PUT 댓글 수정
urssublog / comment / 댓글 수정
PUT
http://localhost:8080/comments/3
Send
{
"content" : "이걸로 수정한다."
}
200 OK
{
"commentId": 3,
"email": "wnsx0000@urssu.com",
"content": "이걸로 수정한다."
}
SELECT * FROM urssublog.comment;
comment_id content created_at updated_at article_id user_id
1 content 2024-09-18 21:47:32.053109 2 1
2 두번째 글에 단 첫번째 댓글! 2024-09-18 21:49:08.157776 2 1
3 이걸로 수정한다. 2024-09-18 21:49:21.188924 2024-09-18 21:51:41.069457 2 1
Table: comment
Columns:
comment_id bigint
content varcha
created_at datetir
updated_at datetir
article_id bigint
user_id bigint
13 21:48:45 SELECT * FROM urssublog.comment LIMIT 0, 1000 1 row(s) returned 0.000 sec / 0.000 sec
14 21:49:43 SELECT * FROM urssublog.comment LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
15 21:50:58 SELECT * FROM urssublog.comment LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
16 21:51:59 SELECT * FROM urssublog.comment LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec

댓글 삭제


Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Collections
Environments
History
urssublog
user
POST 회원가입
POST 로그인
GET 로그아웃
article
POST 게시글 등록
PUT 게시글 수정
DEL 게시글 삭제
comment
POST 댓글 등록
PUT 댓글 수정
GET New Request
DEL New Request
No environment
urssublog / comment / New Request
Save
Share
DELETE
http://localhost:8080/comments/3
Send
Params
Authorization
Headers (7)
Body
Scripts
Settings
Cookies
Query Params
Key
Value
Description
Bulk Edit
Body
Cookies (1)
Headers (3)
Test Results
204 No Content
33 ms
112 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
오후 9:52
2024-09-18

MySQL Workbench
Local instance MySQL80
File
Edit
View
Query
Database
Server
Tools
Scripting
Help
Navigator
SCHEMAS
Filter objects
sakila
sys
urssublog
Tables
article
comment
user
Views
Stored Procedures
Functions
urssublog_test
world
Administration
Schemas
Information
Table: comment
Columns:
comment_id bigint
content varcha
created_at datetir
updated_at datetir
article_id bigint
user_id bigint
Object Info
Session
article
comment
comment
comment
comment
comment
Limit to 1000 rows
1 SELECT * FROM urssublog.comment;
Result Grid
Filter Rows:
Edit:
Export/Import:
Wrap Cell Content:
comment_id
content
created_at
updated_at
article_id
user_id
1
content
2024-09-18 21:47:32.053109
2
1
2
두번째 글에 단 첫번째 댓글!
2024-09-18 21:49:08.157776
2
1
Result Grid
Form Editor
Field Types
comment 1
Apply
Revert
SQLAdditions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help
Snippets
Output
Action Output
Time Action Message Duration / Fetch
14 21:49:43 SELECT * FROM urssublog.comment LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
15 21:50:58 SELECT * FROM urssublog.comment LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
16 21:51:59 SELECT * FROM urssublog.comment LIMIT 0, 1000 3 row(s) returned 0.000 sec / 0.000 sec
17 21:52:54 SELECT * FROM urssublog.comment LIMIT 0, 1000 2 row(s) returned 0.000 sec / 0.000 sec
Windows 정품 인증
[설정]으로 이동하여 Windows를 정품 인증합니다.
오후 9:52
2024-09-18

예외 상황

예외 상황 전부를 찍기에는 너무 많아서 대표로 하나만 찍었습니다.

게시글 수정 시에 잘못된 content(빈 content)를 입력한 경우.