유비샘 2차 세미나

2026-02-10

- https://code.visualstudio.com/docs
  - 위 docs 에는 여러 언어들의 대한 설명이 많으므로 참조하면 좋습니다.

컨트롤러 없이 DB 직접 접근하기

- 우선 예전 만들어두었던 Hello.java 프로젝트를 바탕으로 합니다.

  .\mvnw spring-boot:run (윈도우버전) 으로 프로젝트 실행 시

```
2026-02-10T16:13:21.850+09:00  INFO 3704 --- [example1] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : To
mcat started on port 8080 (http) with context path '/'
2026-02-10T16:13:21.882+09:00  INFO 3704 --- [example1] [           main] c.ubisam.example1.Example1Application    : St
arted Example1Application in 7.353 seconds (process running for 7.801)
```

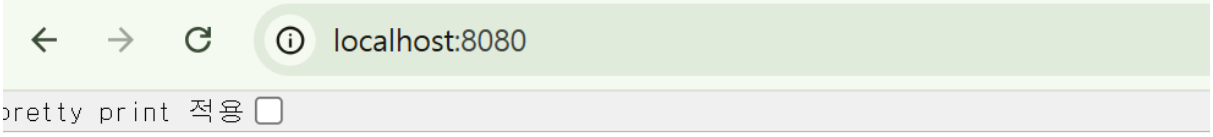
  아래와 같이

  Tomcat started on port 8080 이라는 문구와

  Started ExampleApplication 문구가

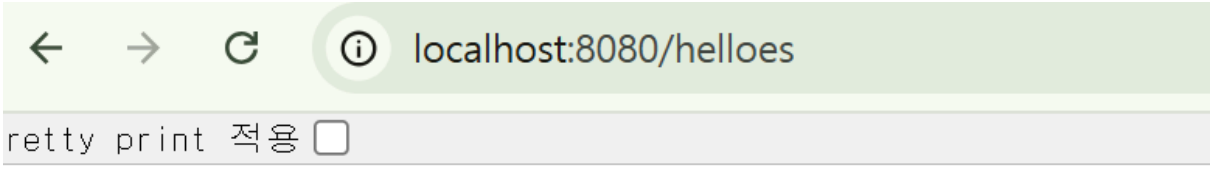
  뜨면 성공입니다.

- 그 후 localhost:8080 주소로 접속 시 아래 사진과 같이 뜨면 성공입니다.




```
{
  "_links" : {
    "helloes" : {
      "href" : "http://localhost:8080/helloes{?page,size,sort*}",
      "templated" : true
    },
    "profile" : {
      "href" : "http://localhost:8080/profile"
    }
  }
}
```

- CRUD 중 **“Search”** Operation 입니다.
    - 위 localhost:8080/helloes 라는 주소로 접속 시 아래 사진 처럼 뜨면 성공입니다

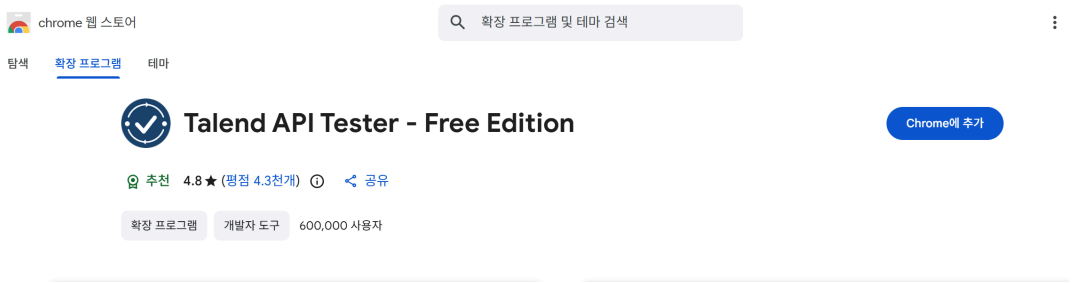



```
"_embedded" : {
  "helloes" : [ ]
},
"_links" : {
  "self" : {
    "href" : "http://localhost:8080/helloes?page=0&size=20"
  },
  "profile" : {
    "href" : "http://localhost:8080/profile/helloes"
  },
  "search" : {
    "href" : "http://localhost:8080/helloes/search"
  }
},
"page" : {
  "size" : 20,
  "totalElements" : 0,
  "totalPages" : 0,
  "number" : 0
}
```

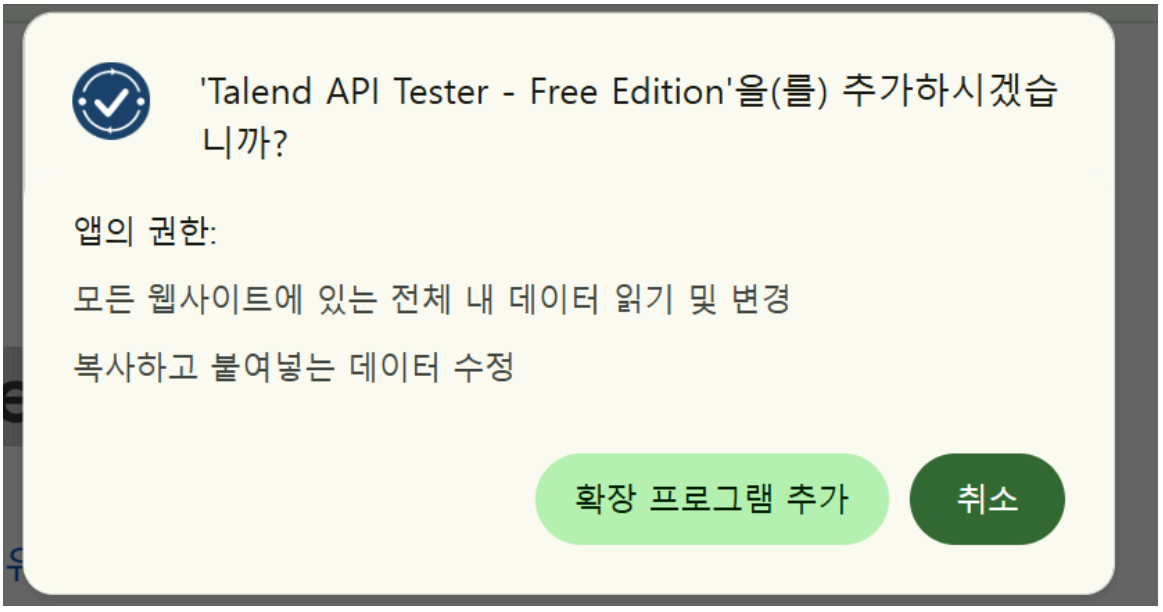

CRUD 확인 (Talend 사용)

- Talend 다운로드
    - https://chromewebstore.google.com/detail/talend-api-tester-free-ed/aejoelaoggembcahagimdiliamlcdmfm?utm_source=ext_app_menu

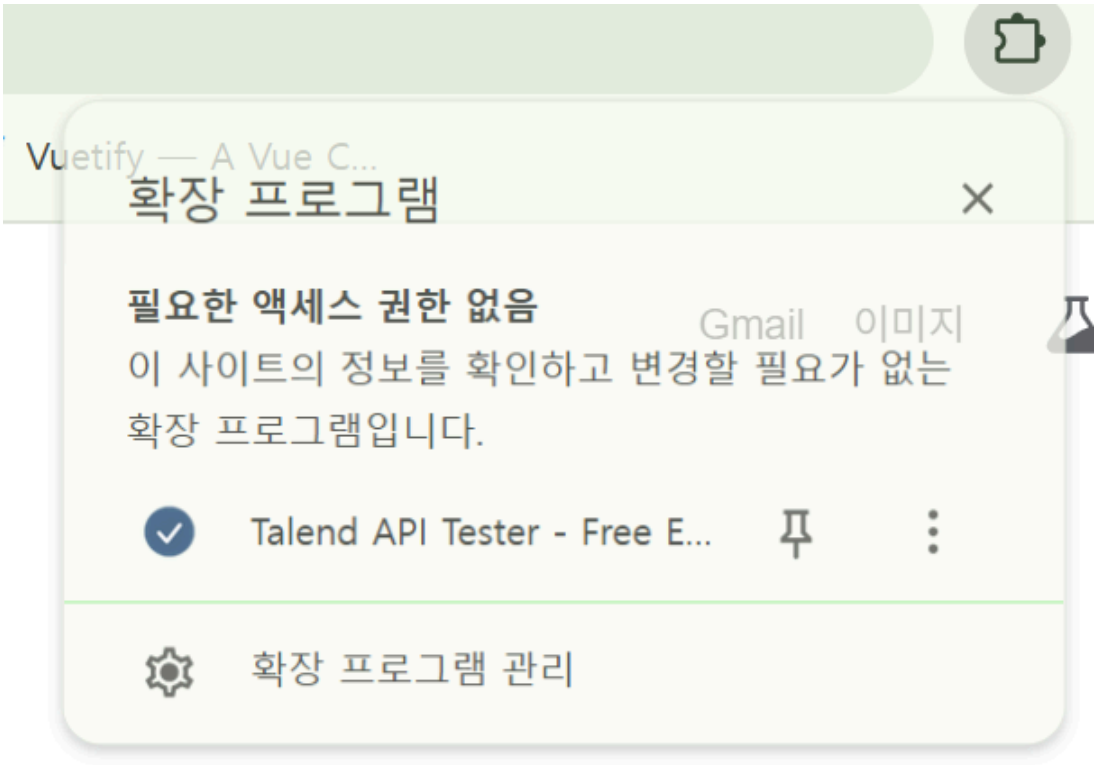
      위 주소로 가시면 **Talend API Tester - Free Edition** Chrome에 추가가 있습니다.

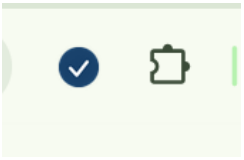


- 그 후 확장프로그램 추가 클릭하시면 추가가 됩니다.



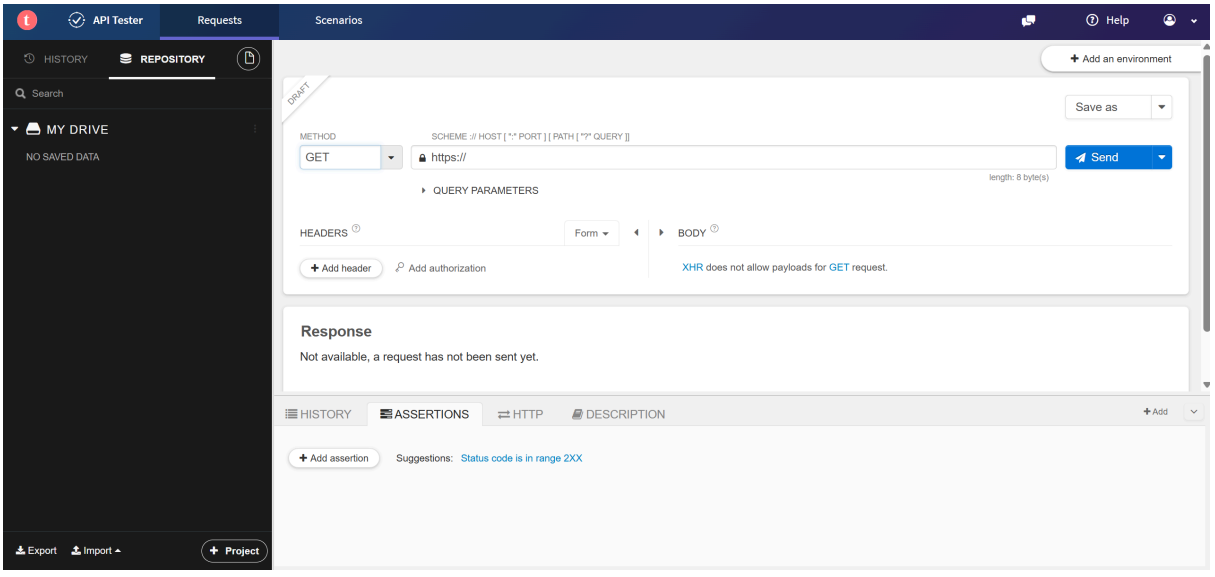
크롬 기준 주소창 옆에 확장 프로그램 아이콘을 클릭하시면 Talend API Tester가 있습니다.





- 해당 핀 아이콘 클릭 시
  왼쪽에 Talend가 추가됩니다.

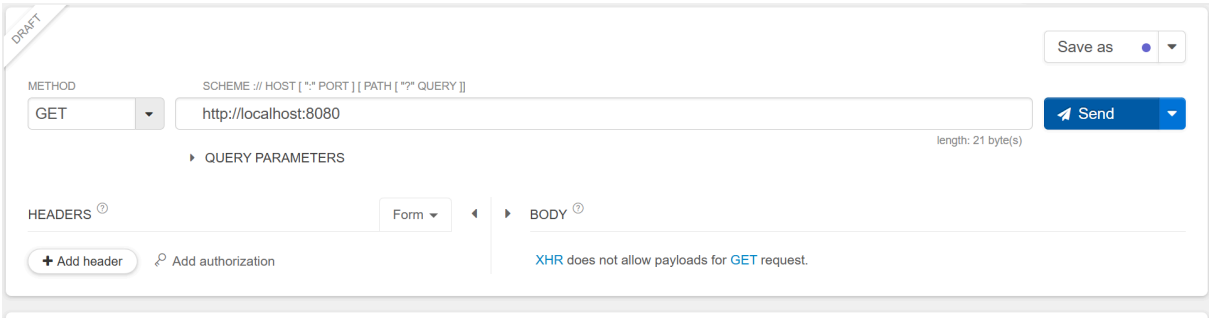
- 처음 Talend 다운로드 후 Use Talend API Tester - Free Edition을 누르시면 아래와 같은 화면이 나옵니다.



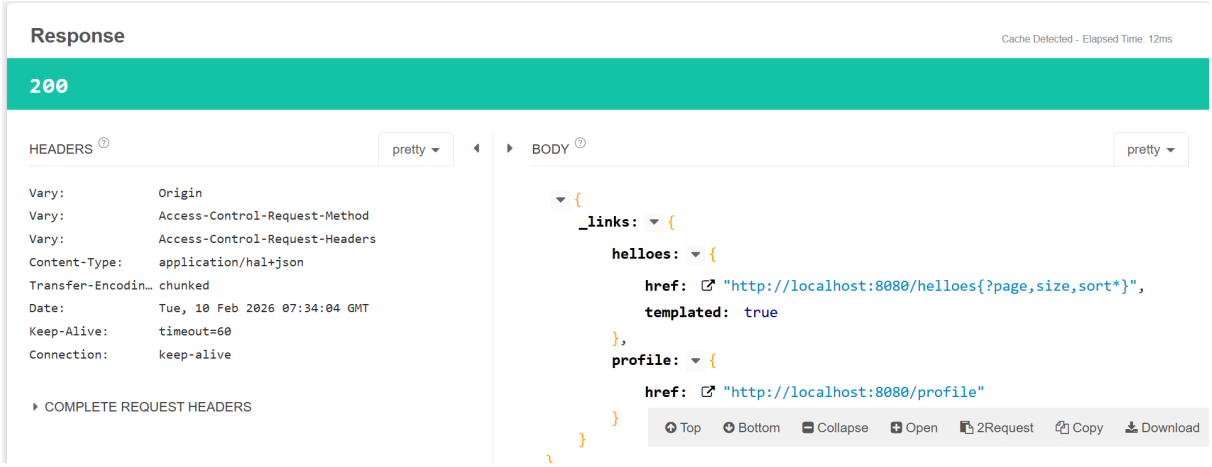
GET - Read (조회)

- 서버가 켜진 상태로
  - METHOD - GET
  - http://localhost:8080

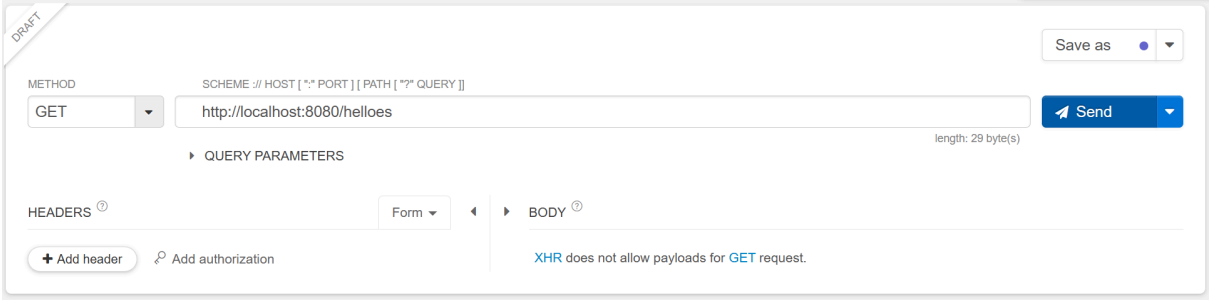
    를 입력하고 Send를 누르면 됩니다.



- Response 200 이 뜨고 아래와 같은 결과가 출력됩니다.



- METHOD - GET랑 http://localhost:8080/helloes 로 다시 Send를 보냅니다.

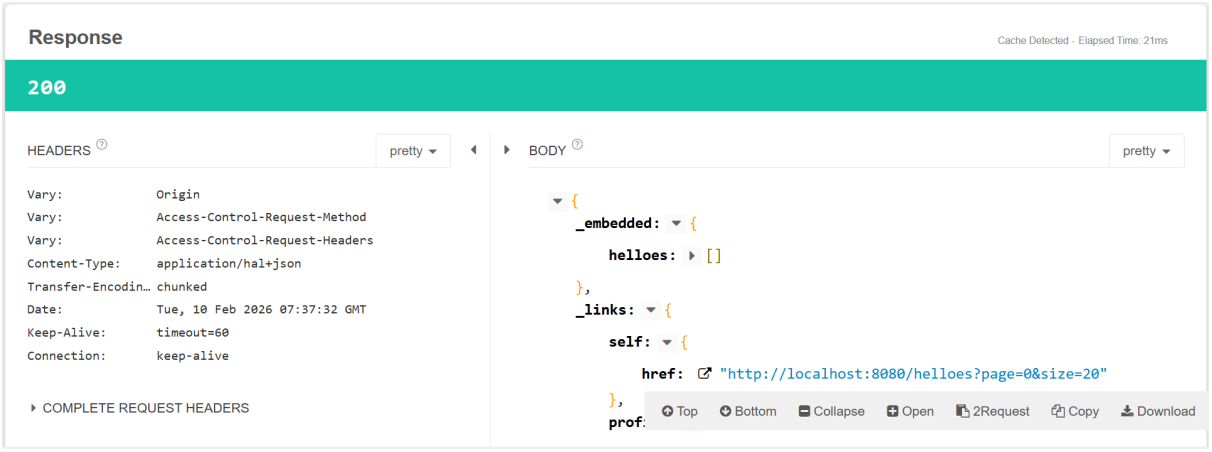


- 아래와 같이 _embedded: {

  hellos: []

  }

형태로 출력이 되면 성공입니다.



POST - Create (생성)

- @GeneratedValue 어노테이션을 통해 ID는 자동으로 생성됩니다.
  - METHOD - POST
  - http://localhost:8080/helloes
  - HEADERS - 체크
  - BODY에는 JSON 형태로

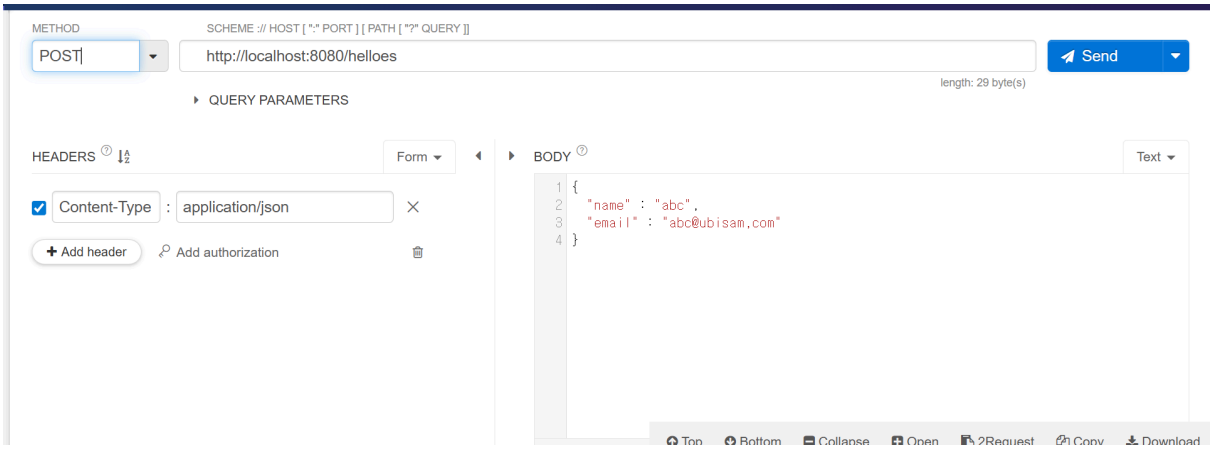
    {

    “name” : “아무거나”,

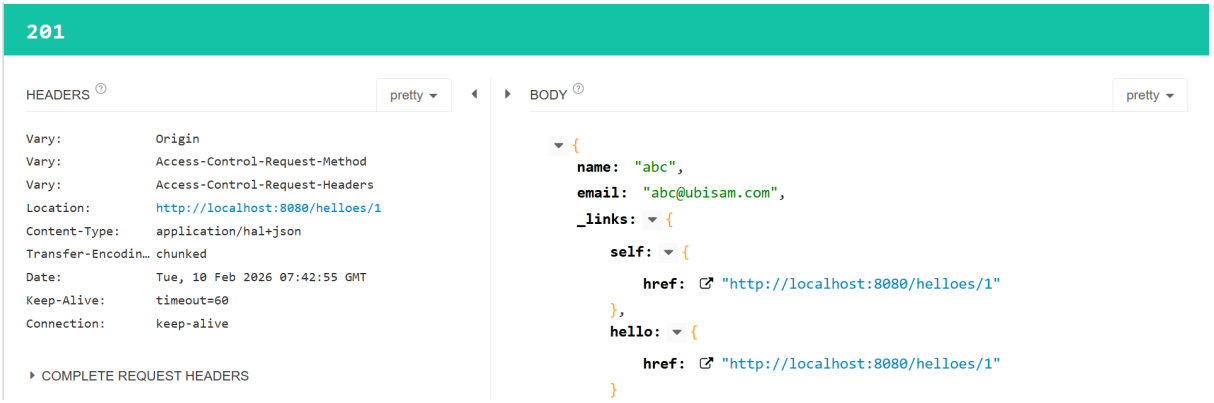
    “email” : “아무거나”

    }

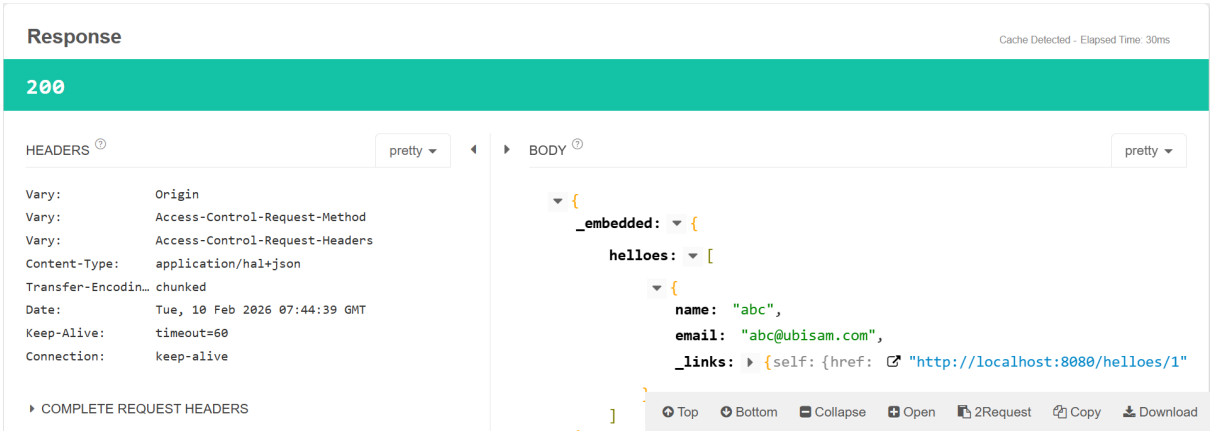
    를 입력한 후 Send를 누릅니다.



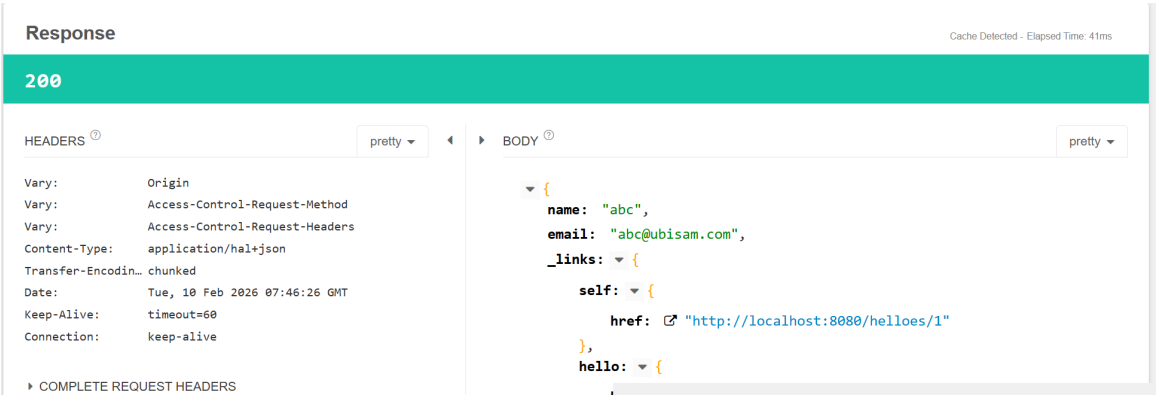
- 아래와 같이 201 코드 + BODY에 입력한 내용이 출력됩니다.



- 다시 METHOD를 GET으로 바꾼 후 http://localhost:8080/helloes 주소에 Send를 보냅니다.
  - helloes에 name, email에 저희가 입력한 내용이 추가됩니다.

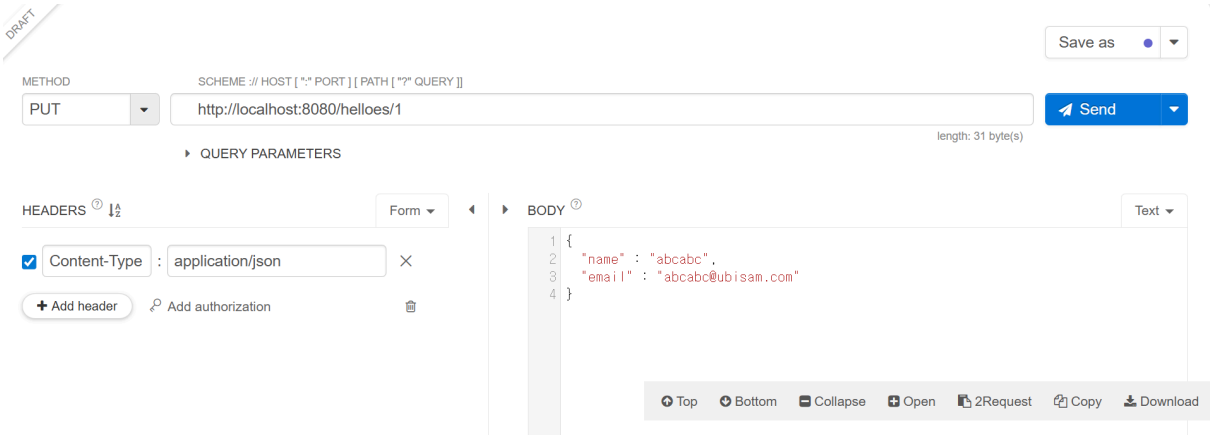


- METHOD는 그대로, http://localhost:8080/helloes/1 주소에 Send를 보냅니다.
  - 그럼 아까와 같은 결과를 보실 수 있습니다.

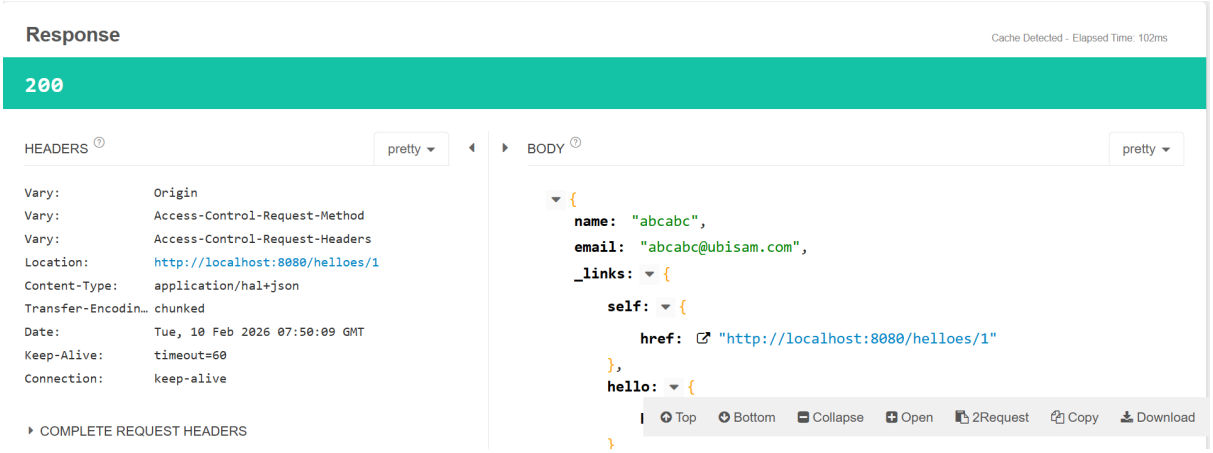


PUT - Update (수정)

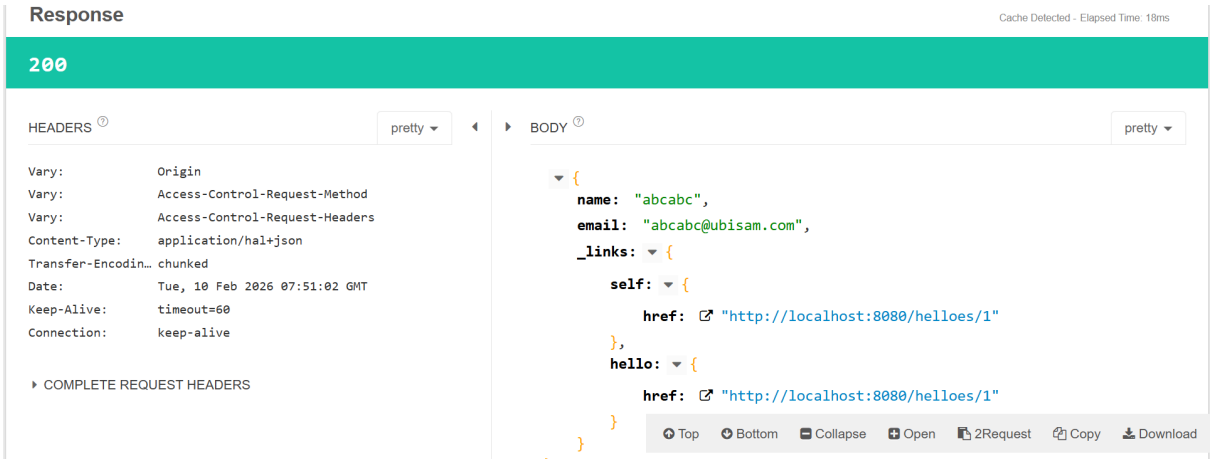
- METHOD - PUT
- http://localhost:8080/helloes/1
- 그 후 BODY에 원하는 내용으로 수정 후 Send를 누릅니다.



- 그럼 아래 사진과 같은 결과가 나옵니다.



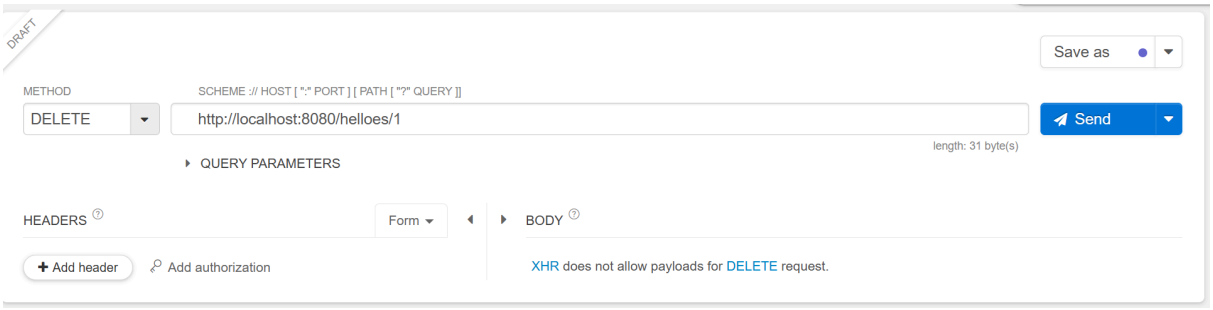
- METHOD - GET 으로 다시 Send할 경우 결과가 바뀐 것을 확인하실 수 있습니다.



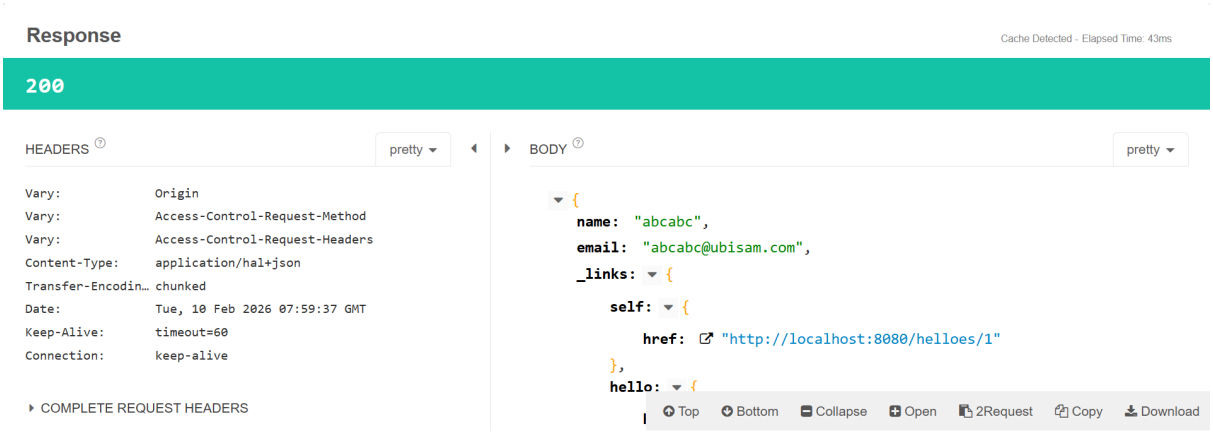
DELETE - Delete (삭제)

- METHOD - DELETE
- http://localhost:8080/helloes/1

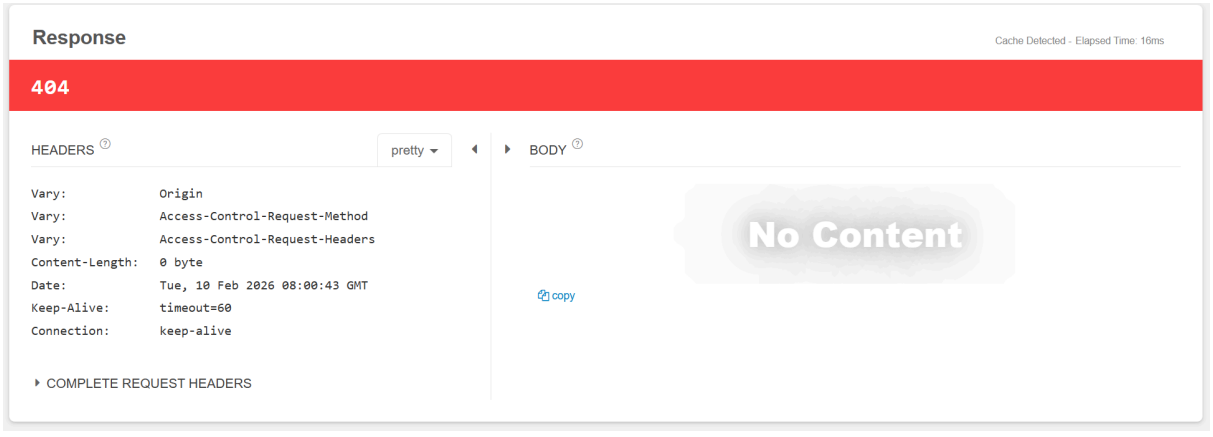
  로 Send를 보냅니다.



- 그럼 아래와 같이 정상적으로 삭제되었다는 결과가 나옵니다.

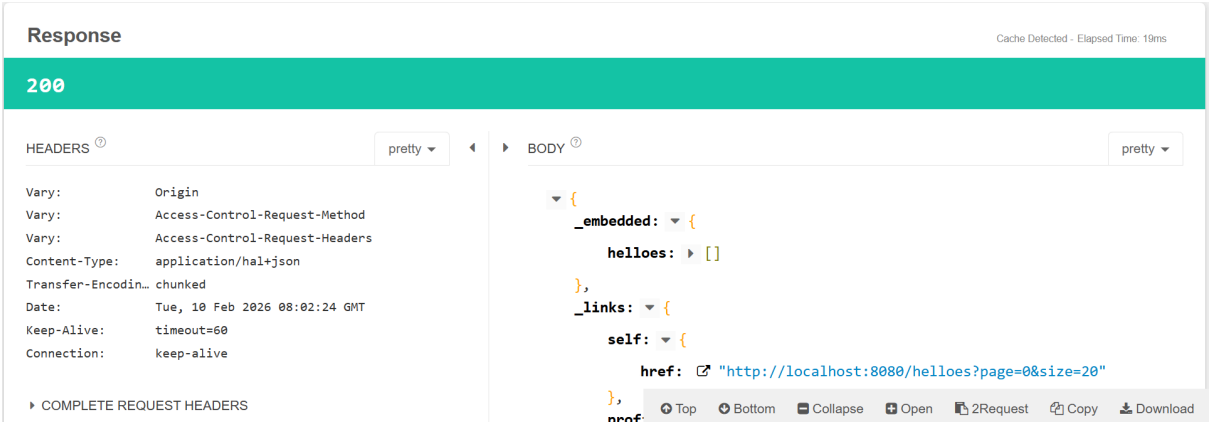


- 그 후 METHOD - GET / http://localhost:8080/helloes/1 을 요청합니다.
  - 아래 사진처럼 Content가 지워졌기에 없다고 뜹니다.



- 주소를 http://localhost:8080/helloes로 GET 요청을 보냅니다.

- 아래와 같이 helloes : [] 으로 정상적으로 지워진 것을 확인할 수 있습니다.



프로젝트 기준 helloes 가 자동생성된 이유!

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-rest</artifactId>
</dependency>
```

spring-book-starter-data-rest 의존성이 있고
HelloRepository가 JpaRepository를 상속하면
기본 규칙으로 /helloes 컬렉션 엔드포인트를 자동 생성합니다.

```
public interface HelloRepository extends JpaRepository<Hello, Long>{
```

여기서 helloes는 엔티티 이름을 '복수형'으로 만든 기본 경로입니다.

```
@Entity
@Data
@Table(name = "t_str")
public class Hello {
```

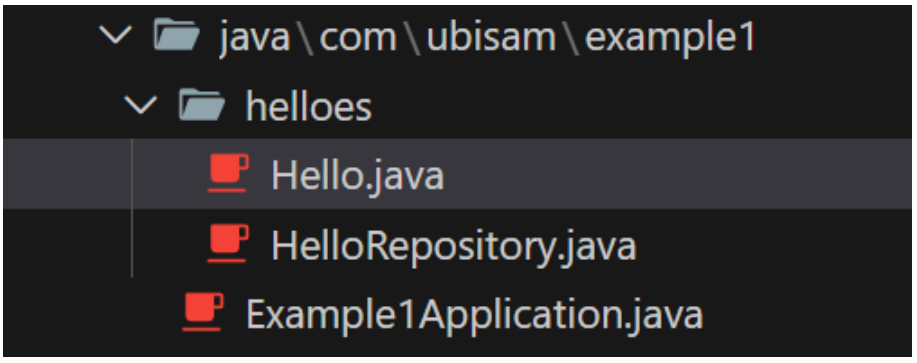
- COC - Convention Over Configuration
  - 소프트웨어 프레임워크에서 사용되는 디자인 패러다임
  - 프레임워크에서 관습적으로 사용되는 패턴을 사용자가 정의할 필요가 없도록 하는 것
  - 사용자는 의미없이 관습적인 내용을 정의하느라 고생할 필요가 없음

- 그러면서도 사용자가 직접 정의할 수 있어야하는 부분에는 제약이 없기에 유연성을 잃지 않음
  - https://scala0114.tistory.com/213

COC 패키지 구조 만들어보기

- 아래와 같이 com.ubisam.example1 안에

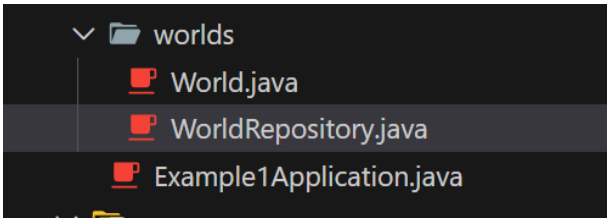
  helloes 폴더를 만든 후 Hello.java, HelloRepository.java 파일을 넣어줍니다.



- 중간에 무슨 박스가 뜨면 OK 누르시면 리팩토링 됩니다.

> 💡 리팩토링이란?
>
> - 기능을 그대로 유지하면서 코드의 구조를 개선하는 사전적 의미
> - 코드의 동작은 바꾸지 않고 구조(패키지/모듈/의존성)을 개선한 것 ← 위 리팩토링
>   - 즉, 구조적 리팩토링 이라고 합니다.

- 다음으로 COC를 예상해보기 위해 com.ubisam.example1 안에 worlds 폴더를 만듭니다.
- 그 후 Hello의 구조와 같이 World.java, WorldRepository.java를 만듭니다.




```java
package com.ubisam.example1.worlds;

import org.springframework.data.jpa.repository.JpaRepository;

public interface WorldRepository extends JpaRepository<World, Long>{

}
```

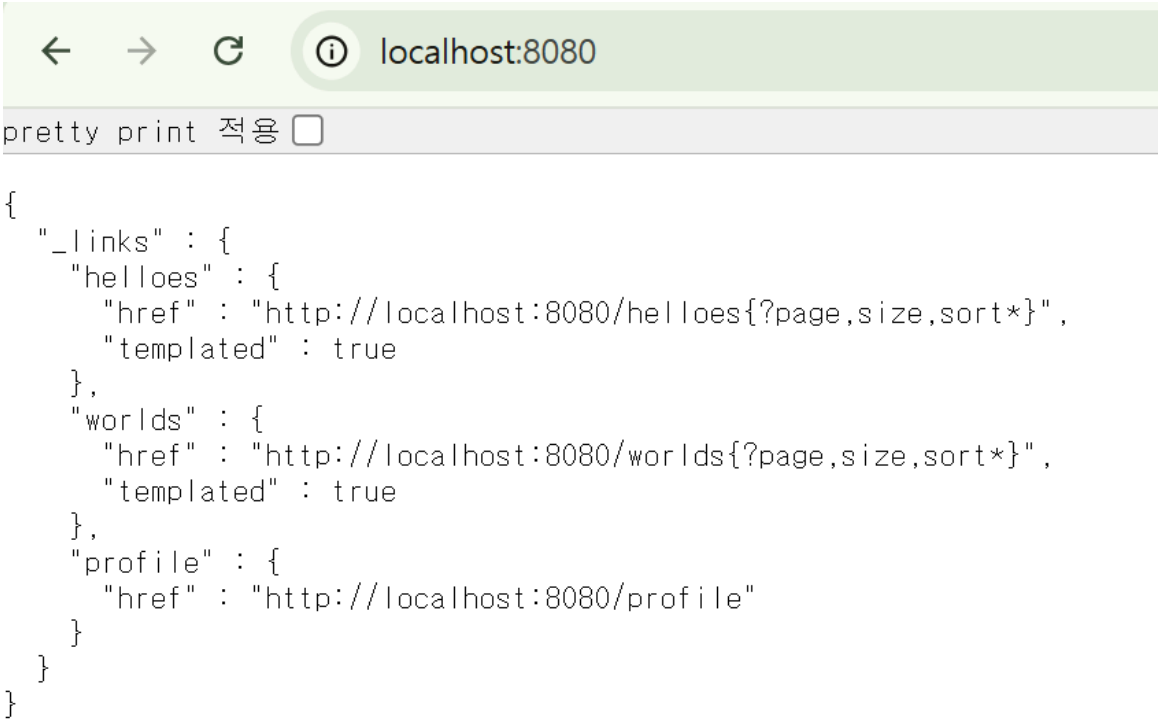
```java
@Entity
@Data
@Table(name = "t_str_world")
public class World {

    @Id
    @GeneratedValue
    private Long id;
    private String name;
    private String email;

}
```

- 다시 Spring 프로젝트를 엽니다.

그 후 localhost:8080 에 접속 시 worlds가 추가된 것을 확인 할 수 있습니다.

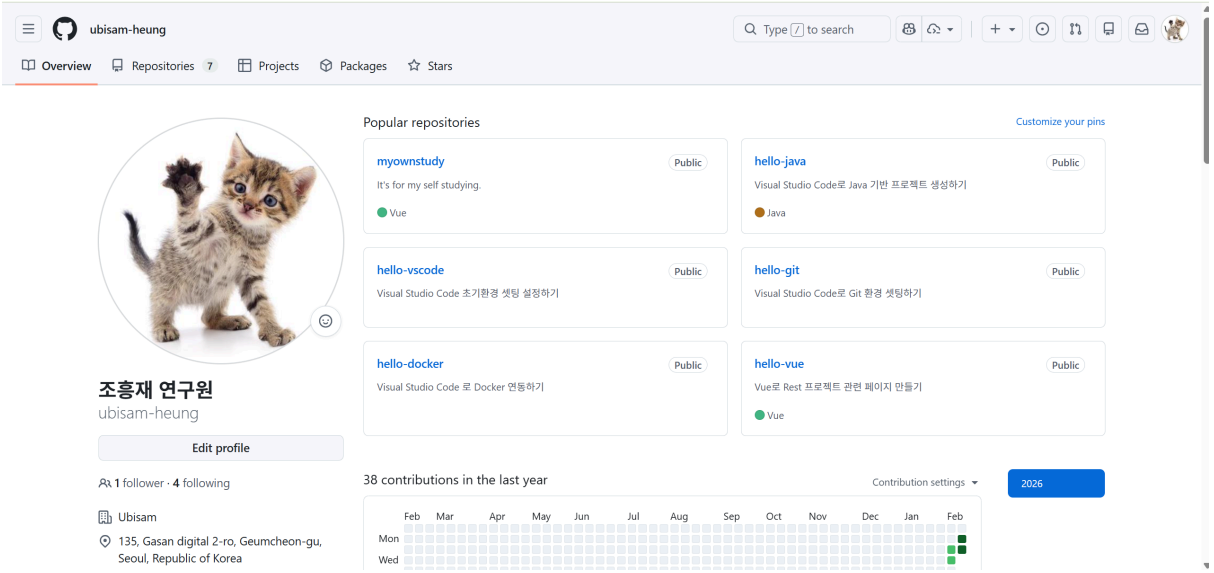


- 위와 같이 Self Description적이고 State Transfer한 것을 묶어서
  REST(Representational State Transfer)라고 합니다.

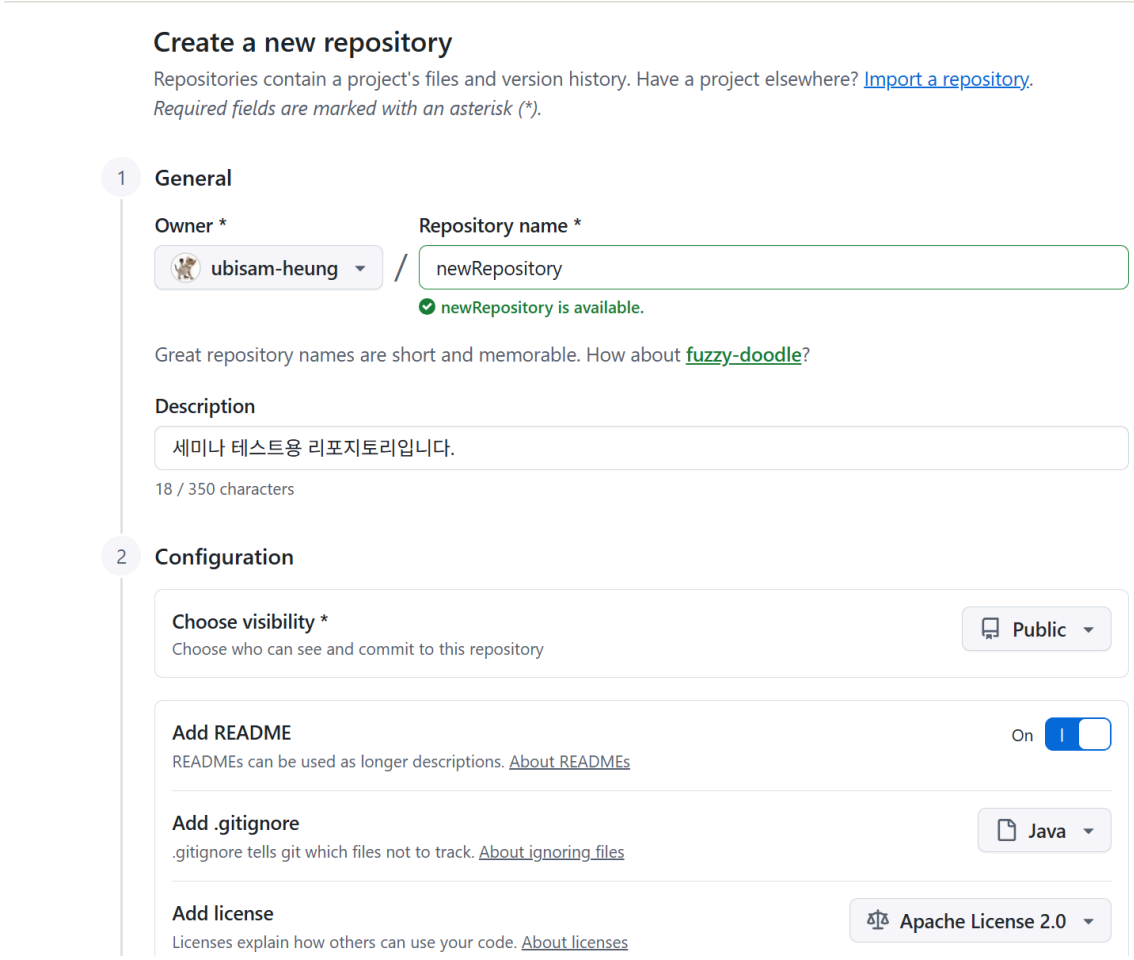
- 아래 영상을 통해 추가적인 정보를 얻을 수 있습니다
  https://deview.kr/2017/schedule/212

GitHub에 대하여

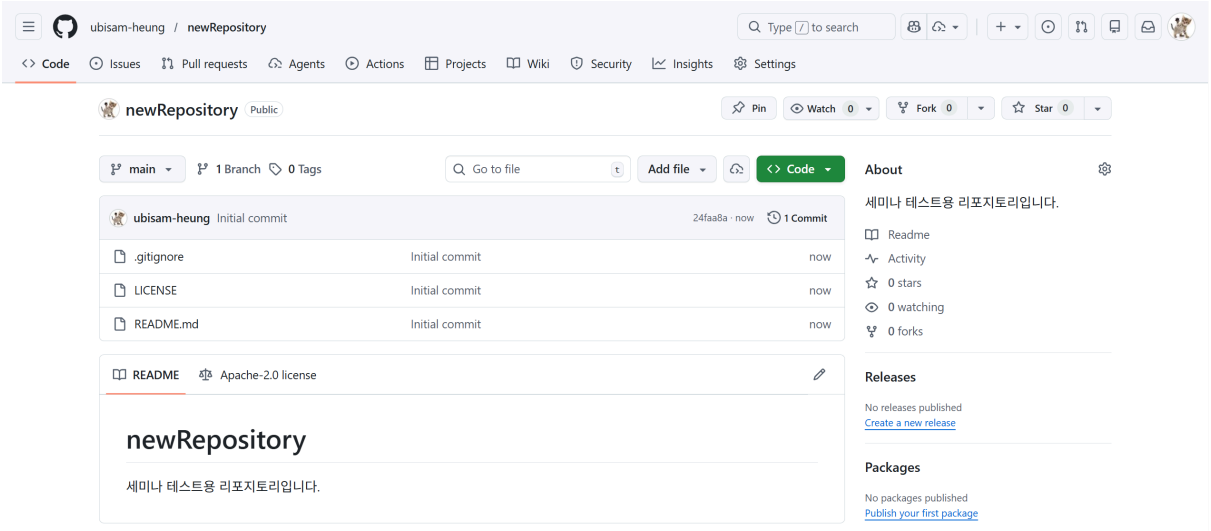
- Wiki나 README.md를 작성할 것.
  - 아래는 제 유비샘 Github 계정 예시입니다.



- Repositories 에 들어가서 New 로 새 Repositories를 생성합니다.
  - 아래와 같이 테스트용 리포지토리를 생성합니다.
    - Public
    - Add README On
    - Add .gitignore - Java
    - Add license - Apache License 2.0



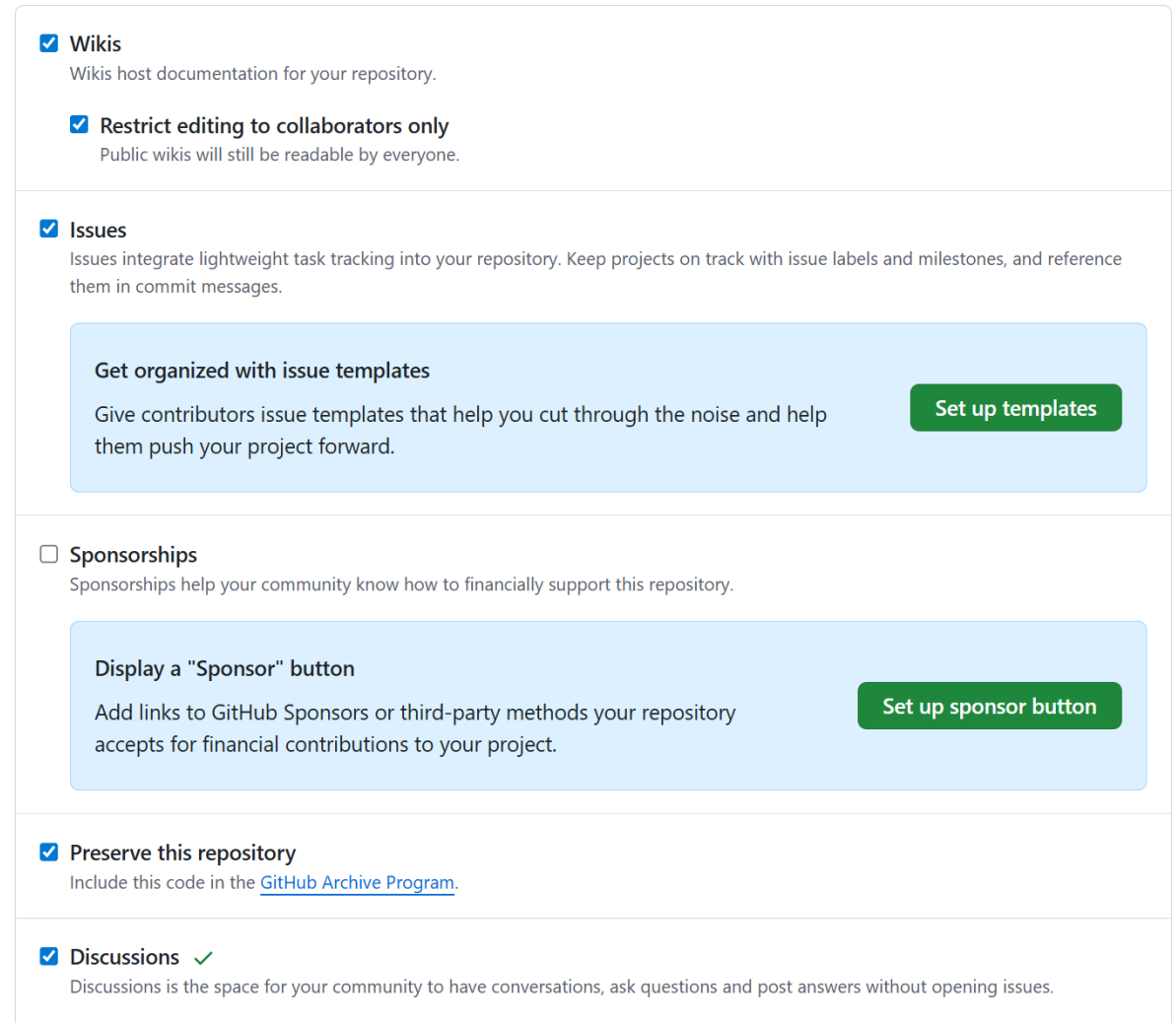
- 리포지토리가 생성되었으면 해당 페이지로 이동합니다.



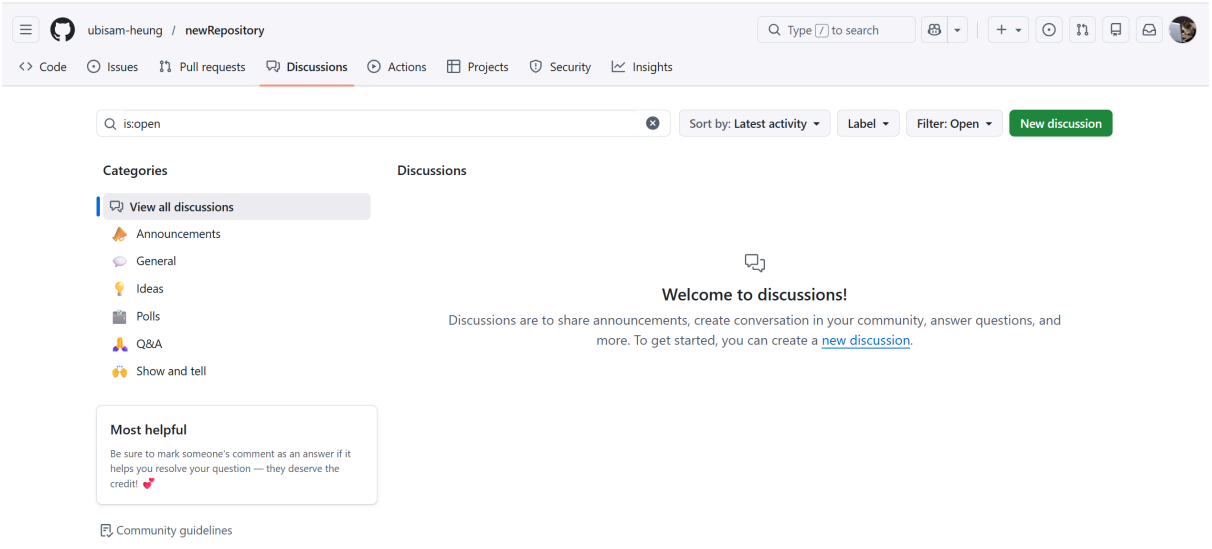
- 위 메뉴중 Settings → General 에서 드래그해서 내리면 아래 사진과 같이
  - Wikis와 Dicussion이 보입니다.

    체크해줍니다

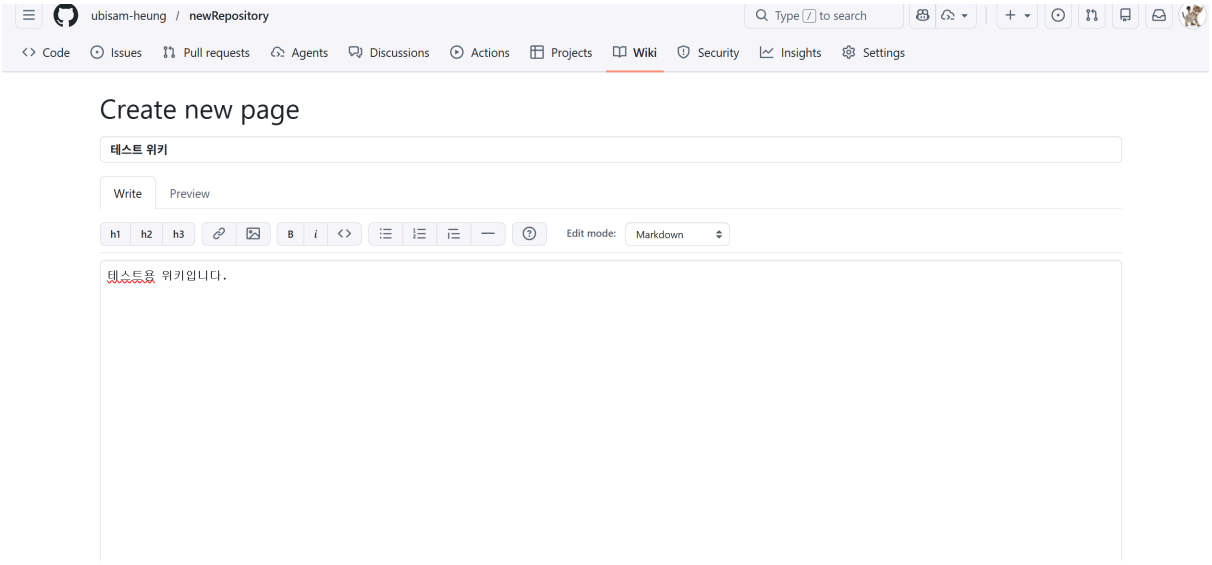
자동저장입니다.



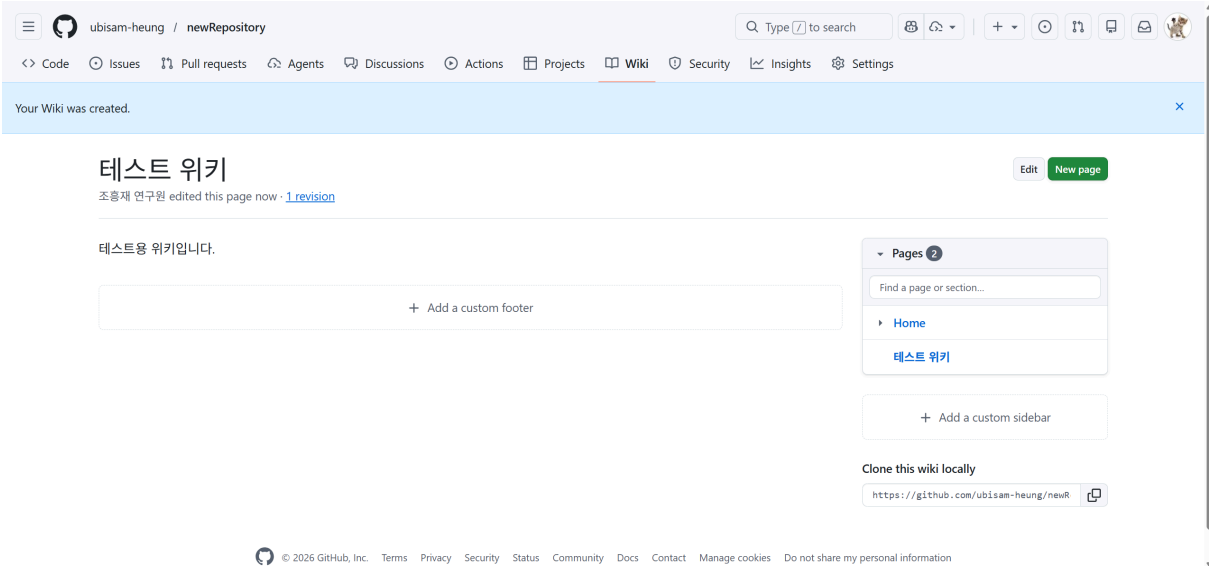
- 다른 사람 Repository의 Discussions이 켜져있으면 아래와 같이 보입니다.



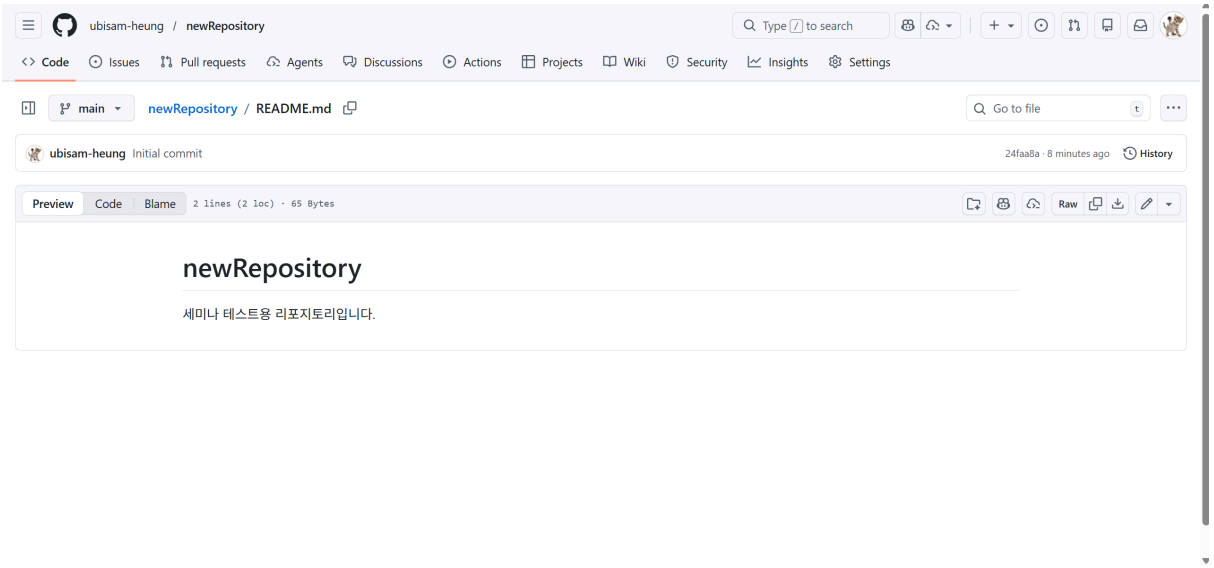
- 위키 생성 예시입니다.
  - 아래는 예시입니다. (빠른 진행을 위해 대략적으로 입력하였습니다)
  - 원하는 내용을 위키에 입력합니다.

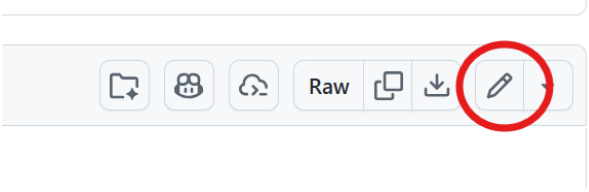


- 그 후 Save 하시면 아래와 같이 위키가 생성됩니다.



- 그 다음 자신의 위키 주소를 복사합니다.
  - https://github.com/ubisam-heung/newRepository/wiki ← 예시로 제 위키주소입니다.
- 아까 만들어 두었던 Repository의 README.md 를 눌러 이동합니다.





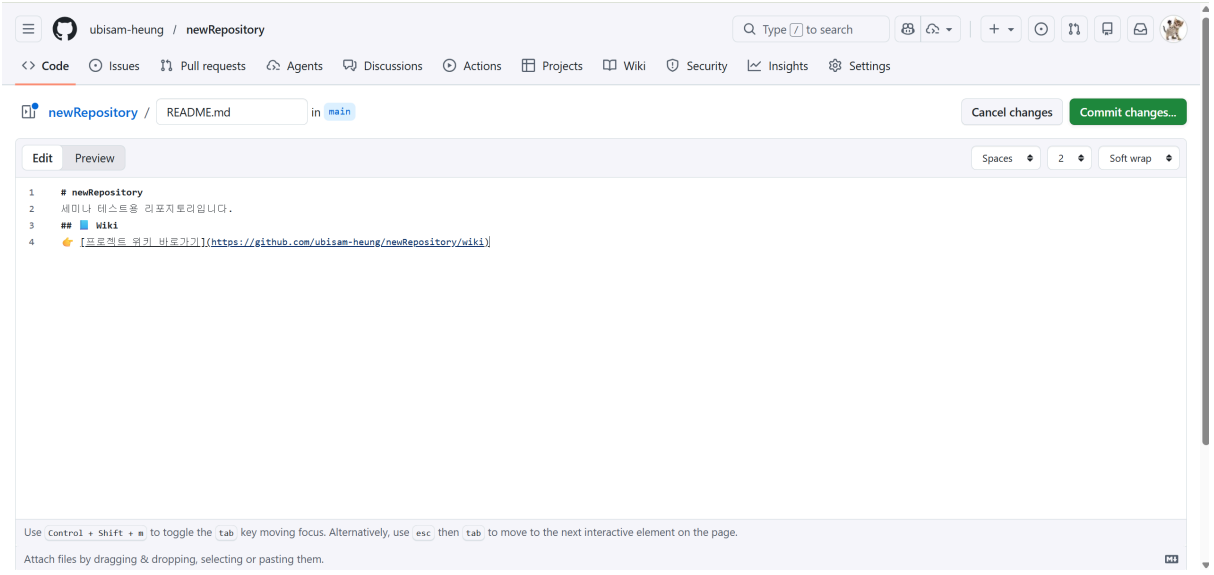
- 해당 버튼으로 수정모드에 들어갑니다.

- 아래 사진과 같이 복사해두었던 주소를 꾸며서 적어줍니다.

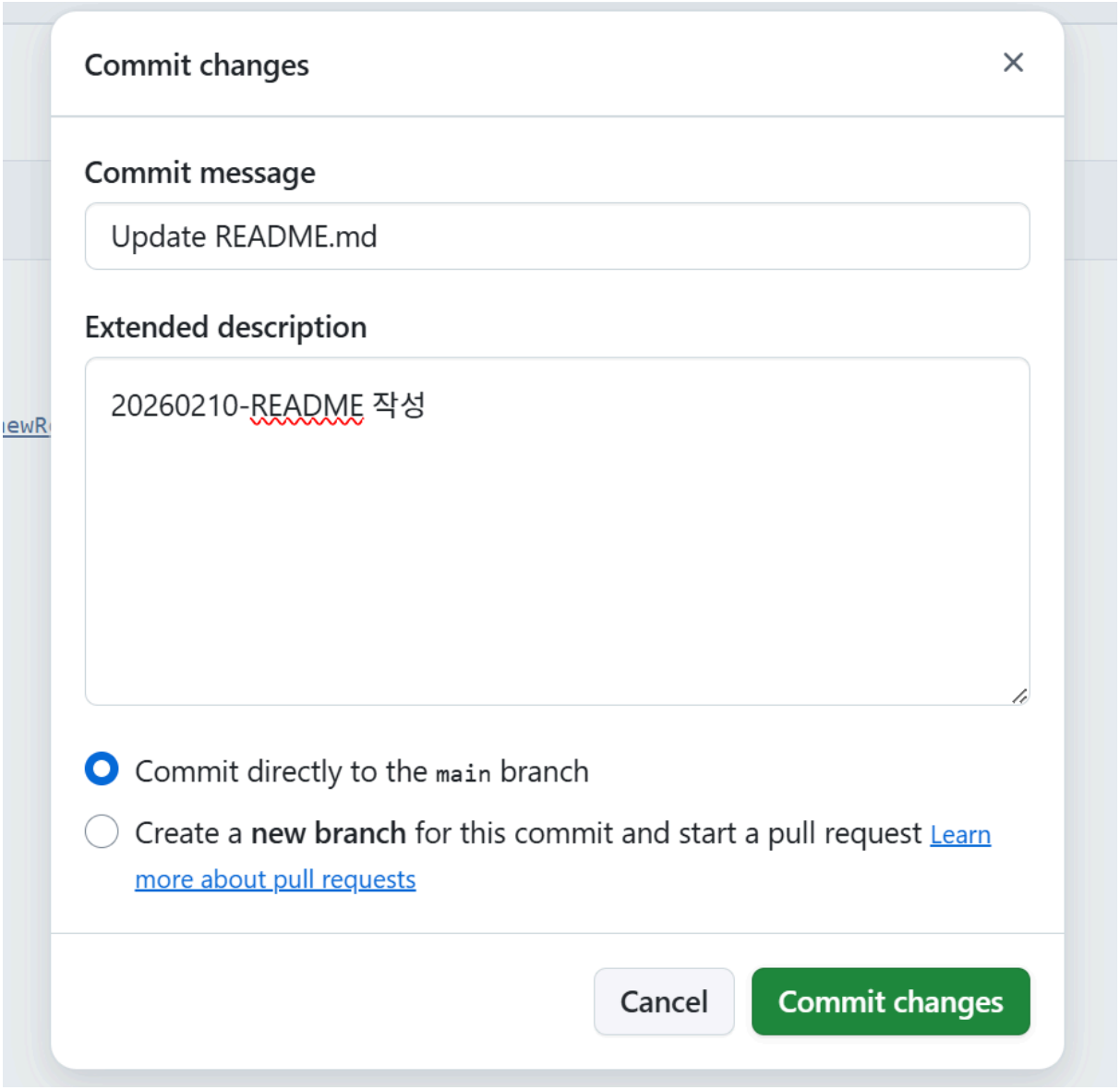
  Markdown 문법은

  [보여줄 텍스트](주소)

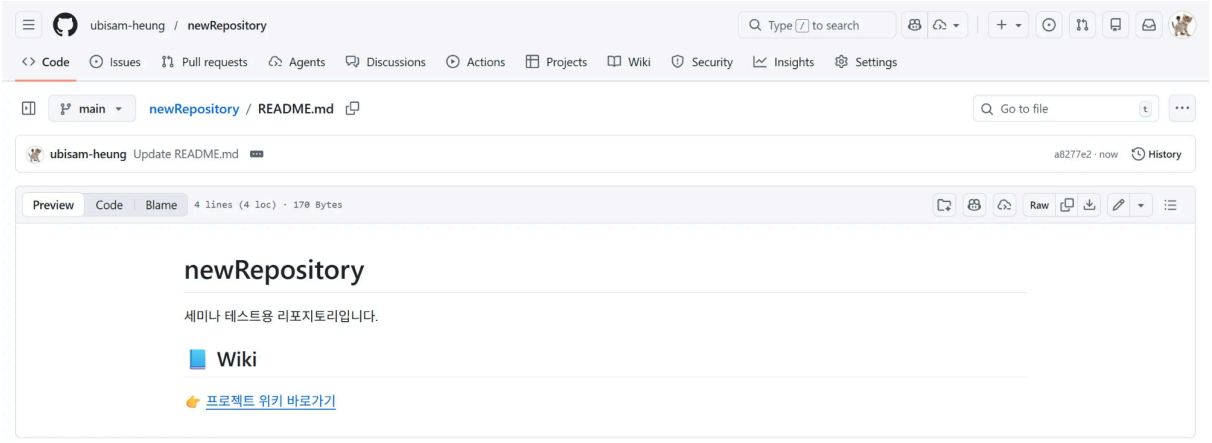
  형식입니다.
- 그 후 Commit changes... 버튼을 누릅니다.



- 커밋메시지 입력 후 Commit directly to the main branch 체크 후 Commit changes를 누릅니다.



- 아래와 같이 README.md 가 잘 작성된 것을 확인하실 수 있습니다.



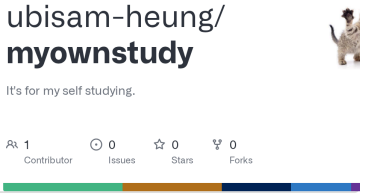
- README 예시는 아래 박진원 연구원님 Github을 추천드립니다.

https://github.com/ParkJinwon1025/sample-rest-basic

이렇게 2월 10일 세미나에 대한 내용 공유였습니다.

제 소스 파일은 github에 올려놨습니다.

myownstudy/Semina/example2 at main · ubisam-heung/myownstudy
It's for my self studying. Contribute to ubisam-heung/myownstudy development by creating an account on GitHub.
https://github.com/ubisam-heung/myownstudy/tree/main/Semina/example2



긴 글 이였는데도 읽어주셔서 감사합니다 😊