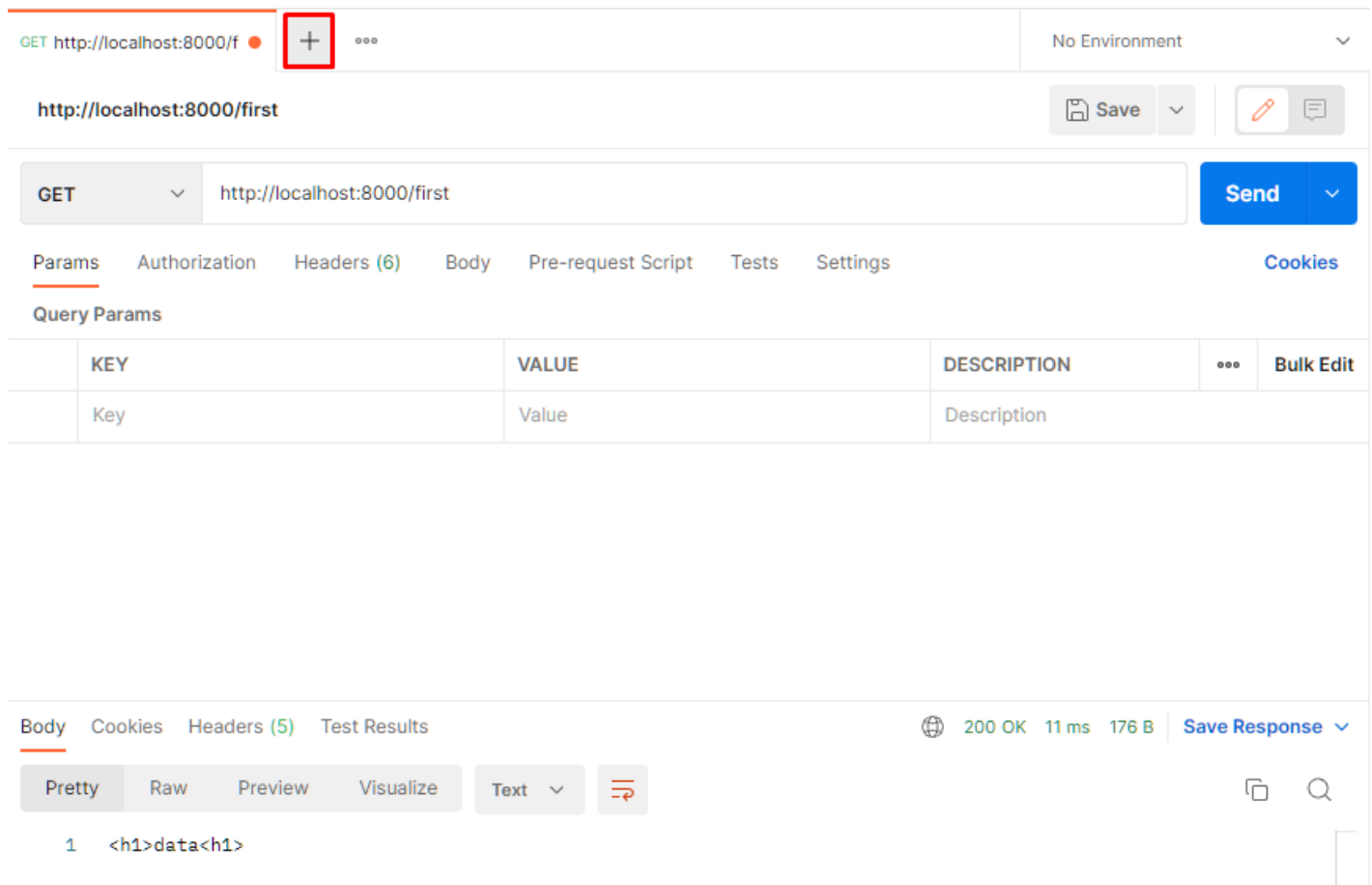
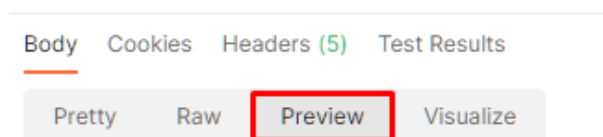


postman 다운로드하기

http 메서드를 적용하려면 툴에 코드를 직접 적어주어야 하는 어려움이 있으므로, postman 프로그램을 이용한다. postman으로 4가지 요청을 쉽게 테스트 해볼 수 있다.



+를 누르면 접속을 할 수 있는 창이 뜬다.



data

Preview는 결과창을 보여준다.

1. Http Method(GET, POST, PUT, DELETE) 이해 (Mapping 기술)

1) http 메서드 실행 방법

가능한 메서드	요청 방법
GET	주소창(웹 브라우저), 하이퍼링크
POST, GET	form 태그
PUT, DELETE, GET, POST	JS

2) GetMapping

```
@RestController //테스트용으로 data만 응답받을 예정이라 RestController 사용, jsp 파일 만들 필요가 없음
public class FirstController {
    @GetMapping("/first")
    public String getData() {
        return "<h1>data</h1>";
    }
}
```

http://localhost:8000/first

Save

GET

http://localhost:8000/first

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (5) Test Results

200 OK 5 ms 177 B Save Response

Pretty Raw Preview Visualize

data

GET 설정을 하고 주소를 입력하면 return 값이 잘 출력된다.

3) PostMapping

```
@PostMapping("/first")
public String postData() {
    return "<h1>insert data</h1>";
    //원래 service.실행메서드를 하고 그 값을 return해주어야 하는 것이나
    //일단 연습용이기 때문에 임의의 값을 부여해놓았다. (태그 안의 내용은 차후 insert된 data를 return해줄)
}
```

POST

⌵

http://localhost:8000/first

Send

⌵

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

🌐

200 OK

5 ms

184 B

Save Response ⌵

Pretty

Raw

Preview

Visualize

insert data

💡 주소가 똑같아도 다른 메서드가 실행되면 다른 값이 return됨을 알 수 있다.

post는 post요청을 하는 코드가 있어야 한다.

Client 입장에서는 전송'버튼'을 누르는 것이 post요청을 하는 것이다. (버튼을 누르면 post요청 됨)

즉, post 요청을 service가 받으면 DB에 데이터를 insert하길 기대하며 버튼을 누르는 것이다.

-> 요청을 받고 DB에 insert를 해준 뒤 insert가 잘 되었다는 결과를 return 해준다.

4) PutMapping

```
@PutMapping("/first")
public String putData() {
    return "<h1>update data</h1>";
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

update data

지금 코드는 POST, PUT 메서드에 실제로 데이터를 담지 않고 테스트용 return값을 부여한 것이므로 return 값(<h1>update data</h1>)이 출력된다. (요청에 대해 응답이 잘 들어온 것이다.)

5) DeleteMapping

```
@DeleteMapping("/first")
public String deleteData() {
    return "<h1>delete data</h1>";
}
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize

delete data

마찬가지로 잘 응답되면 OK

2. Http Method(GET, POST, PUT, DELETE)에 데이터 받기

1) 메서드별 데이터를 받는 방법

GET : Form(GET), 브라우저(QueryString과 PathVariable(PK) 둘 다 가능)

POST, PUT : Form(POST) , JS(POST, PUT)

DELETE : QueryString, PathVariable(PK) 둘 다 가능, JS

🌿 → 🌳 GET,DELETE(구체적 질의) 와 POST, PUT(바디 데이터)의 차이

- GET, DELETE : GET과 DELETE는 데이터를 담아 보내는 요청이 아니다. 특정한 데이터를 찾는 구체적 질의를 하는 것으로, **where절에 걸 내용을** 주소에 적어 보내는 요청이다.
- POST, PUT : 데이터를 **http body에 담아** 요청한다. body의 데이터가 무엇인지 확인하기 위해서는 여러 정보가 필요한데 이 정보는 header에 담겨 있다.
- GET과 DELETE는 위치를 찾는 쪽지를 보내는 것에 가깝고, POST와 PUT은 '데이터가 들어있는 짐꾸러미'를 직접 들고 가는 것이라고 생각해보자. => 따라서 **GET과 DELETE는 http body가 없다.**

🌿→🌳 최근 프로토콜 : PK값을 찾을 때에는 *PathVariable*을 사용한다.

게시글을 불러올 때에는 제목명을 검색해서 찾기보다(QueryString), 목차의 PK값(id)로 찾는 편이 간편하고 깔끔하다.

반대로 어떠한 키워드가 들어 있는 게시물을 찾을 때에는 QueryString을 사용한다.

- 목차에 있는 게시글을 불러올 때 : GET localhost:8000/board/1
- 스프링과 관련된 게시글을 찾을 때 : GET localhost:8000/board?content=스프링
- 특정한 게시글을 지울 때 : DELETE localhost:8000/board/1

2) GetMapping

1. PathVariable 방법

```
@RestController
public class SecondController {

    @GetMapping("/second/{id}") //PK(id)가 (변수)인 페이지를 찾고 싶다
    public String getData(@PathVariable Integer id) {
        return "id : "+id;
    }
}
```

GET

⌵

http://localhost:8000/second/1

Params

Authorization

Headers (6)

Body

Pre-request Scri

Query Params

	KEY	VALUE
	Key	Value

Body

Cookies

Headers (5)

Test Results


Pretty

Raw

Preview

Visualize

Text ⌵



1 id : 1

GET

⌵

http://localhost:8000/second/2

Params

Authorization

Headers (6)

Body

Pre-request Script

Query Params

	KEY	VALUE
	Key	Value

Body

Cookies

Headers (5)

Test Results

Pretty

Raw

Preview

Visualize

Text ⌵

⌵↻

1 id : 2

2. QueryString 방법

```
@GetMapping("/second")
public String getData2(String title, String content) {
    return "title:"+title+", content :"+content;
}
```


GET

http://localhost:8000/second?title=test&content=value

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	title	test			
<input checked="" type="checkbox"/>	content	value			
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK 7 ms 190 B Save Response

Pretty

Raw

Preview

Visualize

title:test, content :value

주소에 직접 적어도 되지만 postman에서는 params에 key와 value값을 넣어 자동완성할 수 있다.

3) PostMapping

```
@PostMapping("/second")
public String postData(String title, String content) {
    return "title:"+title+", content :"+content;
}
```

PostMapping은 body에 데이터를 들고가는 메서드다. 따라서 Body에 KEY값과 VALUE값을 넣어주어야 한다.
이 때, 바디 데이터 유형(Content Type)을 툴(나같은 경우 STS)이 읽을 수 있는 유형으로 선택하여야 한다.

POST http://localhost:8000/second Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none form-data **x-www-form-urlencoded** raw binary GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	title	제목	...		
<input checked="" type="checkbox"/>	content	내용	...		
	Key	Value	Description		

바디 데이터 유형

Body Cookies Headers (5) Test Results 200 OK 6 ms 195 B Save Response

Pretty Raw Preview Visualize Text

```

1 title:제목
2 , content :내용
3

```

 **Content Type** : 웹의 데이터 전송 유형. 바디 데이터의 유형을 의미한다.

- Spring의 기본 파싱 타입은 'x-www-form-urlencoded'로 hello=안녕 의 형태이다.
먼저 Body에 담겨있는 데이터를 BR가 읽는다. 그러나 BR는 그저 문자열을 읽을 뿐이다. (BR이 읽은 문자열 : title=제목&content=내용)
이것을 parsing해야 하는데, parsing을 하기 위해 header에서 Content Type이 무엇인지 읽는다.
→ 파싱할 수 있는 근거 = 헤더의 content-type : x-www-form-urlencoded
- 바디 데이터가 x-www-form-urlencoded 타입이고, Spring 또한 기본적으로 x-www-form-urlencoded 타입으로 파싱하므로 두 Content Type이 일치하여 변수에 값을 넣어 줄 수 있다. (바디 데이터를 다른 타입으로 주면 Spring이 파싱하지 못한다.)

Params Authorization **Headers (8)** Body Pre-request Script Tests Settings Cookies

Headers Hide auto-generated headers

	KEY	VALUE	DESCRIPTION	...	Bulk Edit	Presets
<input checked="" type="checkbox"/>	Postman-Token	<calculated when request is sent>				
<input checked="" type="checkbox"/>	Content-Type	application/x-www-form-urlencoded				
<input checked="" type="checkbox"/>	Content-Length	<calculated when request is sent>				
<input checked="" type="checkbox"/>	Host	<calculated when request is sent>				
<input checked="" type="checkbox"/>	User-Agent	PostmanRuntime/7.29.2				
<input checked="" type="checkbox"/>	Accept	*/*				
<input checked="" type="checkbox"/>	Accept-Encoding	gzip, deflate, br				
<input checked="" type="checkbox"/>	Connection	keep-alive				
	Key	Value	Description			

- Web의 디폴트 문자 전송 유형도 x-www-form-urlencoded 타입이다.

! GetMapping과 PostMapping의 비교

1. GetMapping은 http body값을 받는 메서드가 아니다. 따라서 GET요청에 Body값을 넣으면 null값이 나온다.

The screenshot shows a REST client interface. The top bar indicates a GET request to `http://localhost:8000/second`. The 'Body' tab is selected, showing form-data with two fields: 'title' and 'content'. The response section shows a 200 OK status with a response time of 3 ms and a body size of 189 B. The response body is displayed in JSON format: `{\"title\":null, \"content\":null}`.

KEY	VALUE	DESCRIPTION
title	제목	...
content	내용	...

1 title:null, content :null

2. PostMapping에 body가 아닌 params에 값을 넣어도 값이 정상적으로 출력되는 이유는 params는 GET, POST, PUT, DELETE 모든 값이 나오기 때문이다. PostMapping을 한다면 속지말고 body에 값을 넣어주자!

POST

▼

http://localhost:8000/second?title=제목&content=내용

Send

▼

Params

Auth

Headers (7)

Body

Pre-req.

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	title	제목			
<input checked="" type="checkbox"/>	content	내용			
	Key	Value	Description		

Body

Cookies

Headers (5)

Test Results

200 OK

4 ms

194 B

Save Response

▼

Pretty

Raw

Preview

Visualize

title:제목, content :내용

4) PutMapping

```
@PutMapping("/second")
public String putData(String title, String content) {
    return "title:"+title+", content :"+content;
}
```

PUT ▼ http://localhost:8000/second Send ▼

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	title	제목변경 ↗	...		
<input checked="" type="checkbox"/>	content	내용변경 ↗	...		
	Key	Value	Description		

Body Cookies Headers (5) Test Results 200 OK 8 ms 207 B Save Response ▼

Pretty Raw Preview Visualize Text ▼ ≡

```

1  title:제목변경
2  , content :내용변경
3

```

PostMapping과 같은 원리로 동작한다.

5) DeleteMapping

```

@DeleteMapping("/second/{id}")
//쿼리스트링 해도 됨
public String deleteData(@PathVariable Integer id) {
    return id+"delete ok";
}

```

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input type="checkbox"/>	title	제목			
<input checked="" type="checkbox"/>	content	내용			
	Key	Value	Description		

```
1 1delete ok
```