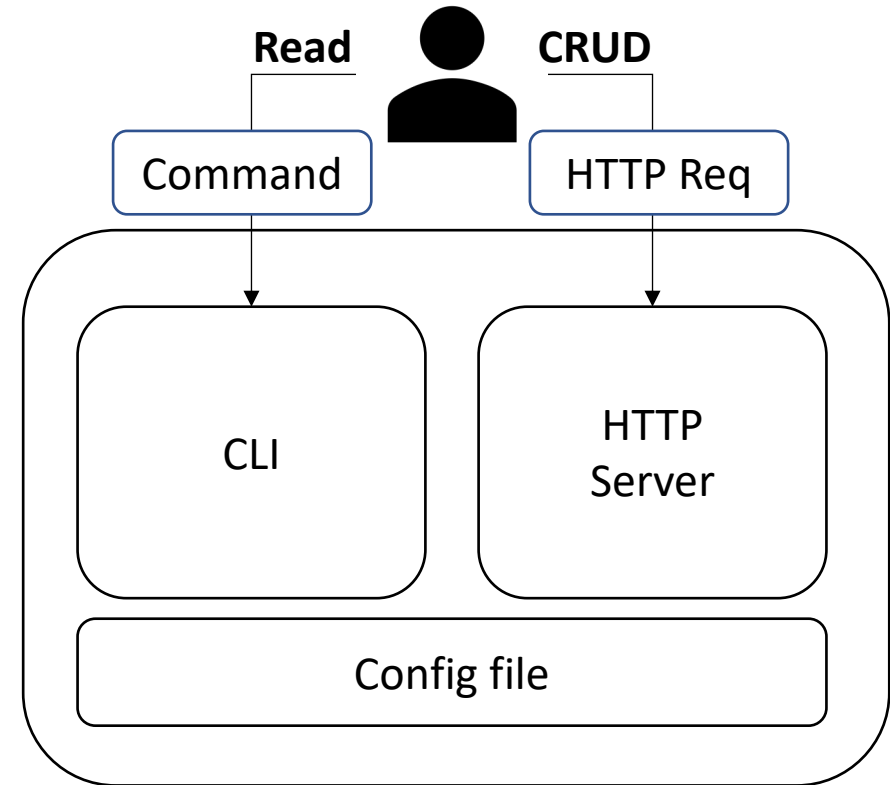


# 프로젝트 구조 : HTTP server

```
main.rs
use cli;
use http;

fn main() {
    cli::init_cli();
    http::init_http_server();
}
```



-> 실행 시 cli application 모듈과 http server 모듈이 실행됩니다.

cli application은 사용자가 터미널을 통해 생성된 설정파일을 읽을 수 있습니다

HTTP Server는 멀티 스레드로 TCP 리스너로 전달된 명령이 실행되며 설정파일 생성, 읽기 요청, 삭제, 업데이트가 가능합니다

# 실행 방법 : Cli application

```
configuration > {} config.json > ...
```

```
1 {  
2   "url" : "localhost",  
3   "port" : "8080",  
4   "serverName" : "boom-http",  
5   "admin" : "dongjunna"  
6 }
```

cargo run -- configuration/config.json

```
Running `target/debug/boomhttp configuration/config.json`
```

```
{  
  "url" : "localhost",  
  "port" : "8080",  
  "serverName" : "boom-http",  
  "admin" : "dongjunna"  
}
```

```
~/Desktop/boom-labs-http-server> █
```

11/21/2022 07:50:43 PM

-> 해당 명령어로 실행 시 기본 설정 파일인 configuration/config.json의 내용을 읽어 화면에 띄웁니다.

# Cli application 사용

```
~/Desktop/boom-labs-http-server> configuration/config.json
>> Configuration file path : "configuration/config.json"
file content:
{
  "url" : "localhost",
  "port" : "8080",
  "serverName" : "boom-http",
  "admin" : "dongjunna"
}
~/Desktop/boom-labs-http-server> █
```

11/21/2022 07:52:06 PM

11/21/2022 07:52:06 PM

-> cli application 실행 후 설정 파일의 경로를 입력하면 저장된 설정 파일의 내용을 화면에 뿌립니다.

```
~/Desktop/boom-labs-http-server> file
>> Configuration file path : "file"
Not found : No such file or directory (os error 2) : file
~/Desktop/boom-labs-http-server> █
```

11/21/2022 07:53:05 PM

11/21/2022 07:53:05 PM

-> 존재하지 않는 파일을 요청할 경우 Not found 에러가 발생합니다.

# HTTP Server

**POST** http://localhost:8080/filename&content

**GET** http://localhost:8080/filename

**PUT** http://localhost:8080/filename&modifiedContent

**DELETE** http://localhost:8080/filename

CRUD 처리를 위해 POST, Get, Put, Delete 명령어가 구현되어 있습니다.

# HTTP Server : POST, GET

POST

http://localhost:8080/test1&hi

Params

Send

Save

AuthorizationHeadersBodyPre-request ScriptTests

TypeNo Auth

BodyCookiesHeaders (1)Test Results

Status: 200 OKTime: 13 ms

-> POST 요청시 URL을 통해 파일이름&파일내용 형식으로 전달되어 **파일이름 : test1, 내용 : hi**의 파일이 생성됩니다.

GET

http://localhost:8080/test1

Params

Send

Save

AuthorizationHeadersBodyPre-request ScriptTests

TypeNo Auth

BodyCookiesHeaders (1)Test Results

Status: 200 OKTime: 9 ms

PrettyRawPreviewText

1hi

-> GET 요청시 URL을 통해 파일이름을 전달할 경우 해당 파일의 내용을 반환합니다.

# HTTP Server : PUT

PUT ▾

http://localhost:8080/test1&bye

Params

Send ▾

Save ▾

AuthorizationHeadersBody ●Pre-request ScriptTests

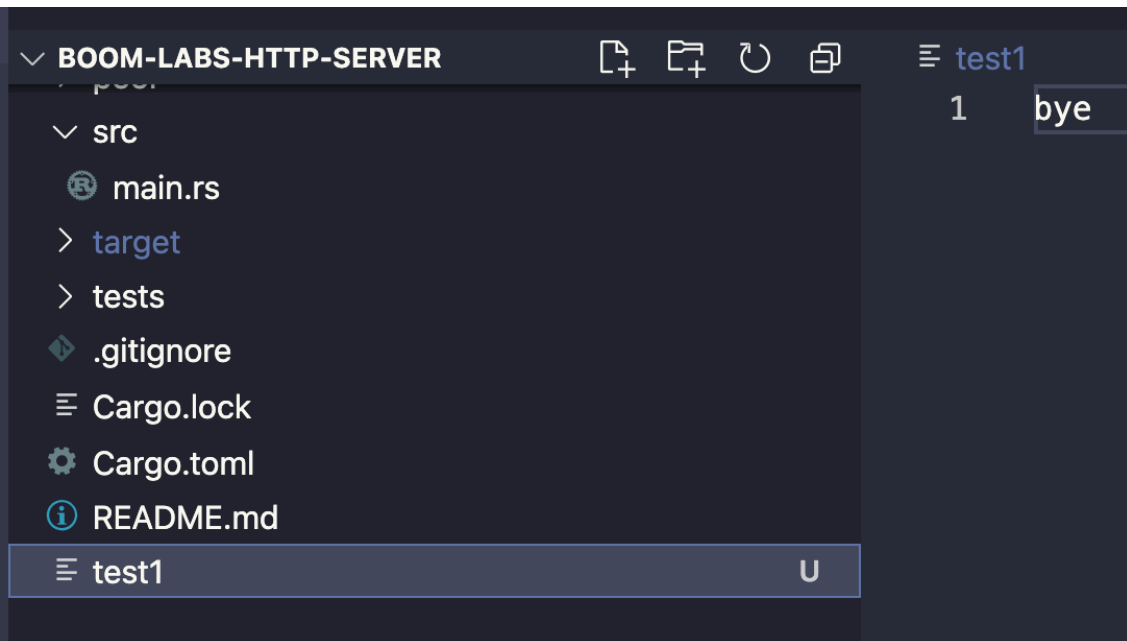
TypeNo Auth ▾

BodyCookiesHeaders (1)Test Results

Status: 200 OKTime: 8 ms

Code

-> PUT 요청시 URL을 통해 파일이름&수정할 파일내용 형식으로 전달됩니다.



-> 요청 실행 결과 파일 내용이 수정되었습니다.

# HTTP Server : DELETE

The screenshot displays the VS Code interface with an HTTP client extension. The top bar shows a DELETE request to `http://localhost:8080/test1` with a status of 200 OK and a time of 8 ms. The 'Body' tab is selected, showing the response body `bye`. The Explorer sidebar on the left shows the project structure for `BOOM-LABS-HTTP-SERVER`, with `test1` highlighted. The main editor shows the response body `bye`.

DELETE `http://localhost:8080/test1` Params Send Save

Authorization Headers Body Pre-request Script Tests Code

Type No Auth

Body Cookies Headers (1) Test Results Status: 200 OK Time: 8 ms

EXPLORER

- BOOM-LABS-HTTP-SERVER
  - pool
  - src
    - main.rs
  - target
  - tests
  - .gitignore
  - Cargo.lock
  - Cargo.toml
  - README.md

test1

1 bye

-> 파일이름을 URL에 포함하여 전달할 경우 파일이 삭제됩니다.