회원 등록 API 만들기 과제

2022 tankware global 준비반 미니프로젝트

Github : https://github.com/W00hyun-Kim/SwaggerMiniProject

Name : 김 우 현

Mail : woohyun9509@gmail.com

Tel : 010 4128 9509

Blog : https://woohyun.tistory.com/

[목 차]

주 제

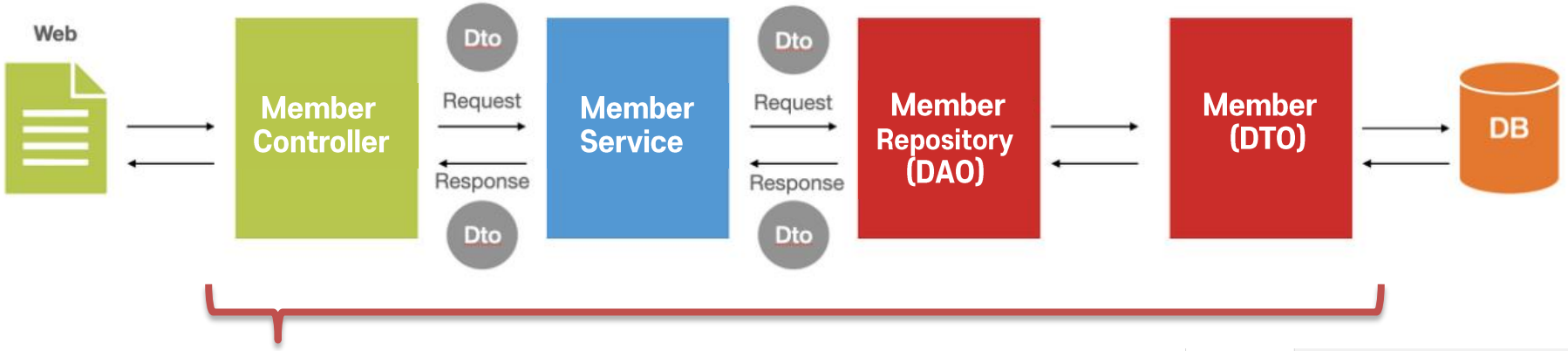
Spring Framework 기반으로 Rest API 개발

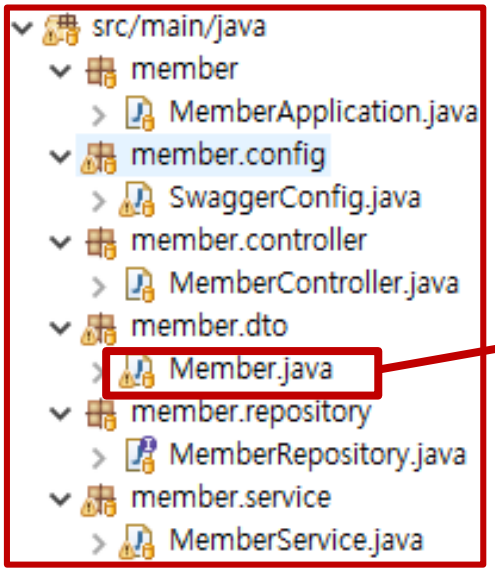
- 회원가입 및 회원정보 조회 API 개발 (CRUD)
- 회원정보 Table 구성(회원가입 정보 API 필드 구성)
- API 테스트 수행 (Jmeter 활용하여 1,000번 연속 호출)

개발 환경

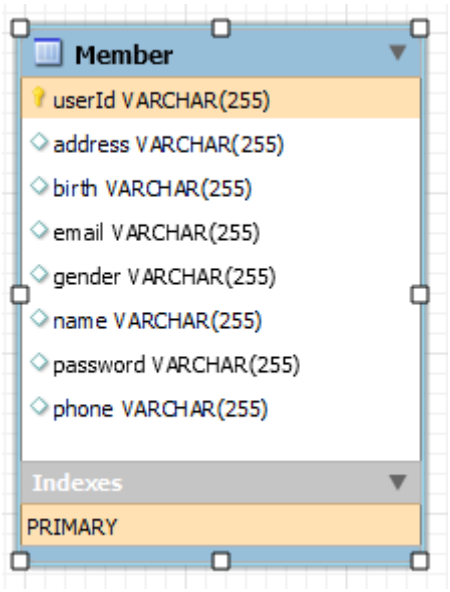
- DataBase : MySQL
- Tool : STS
- Build : Maven
- Language : JAVA
- Spring Boot Version : 2.7.3
- Java Version : 17.0.2

클래스 및 DB 설계





클래스 구분

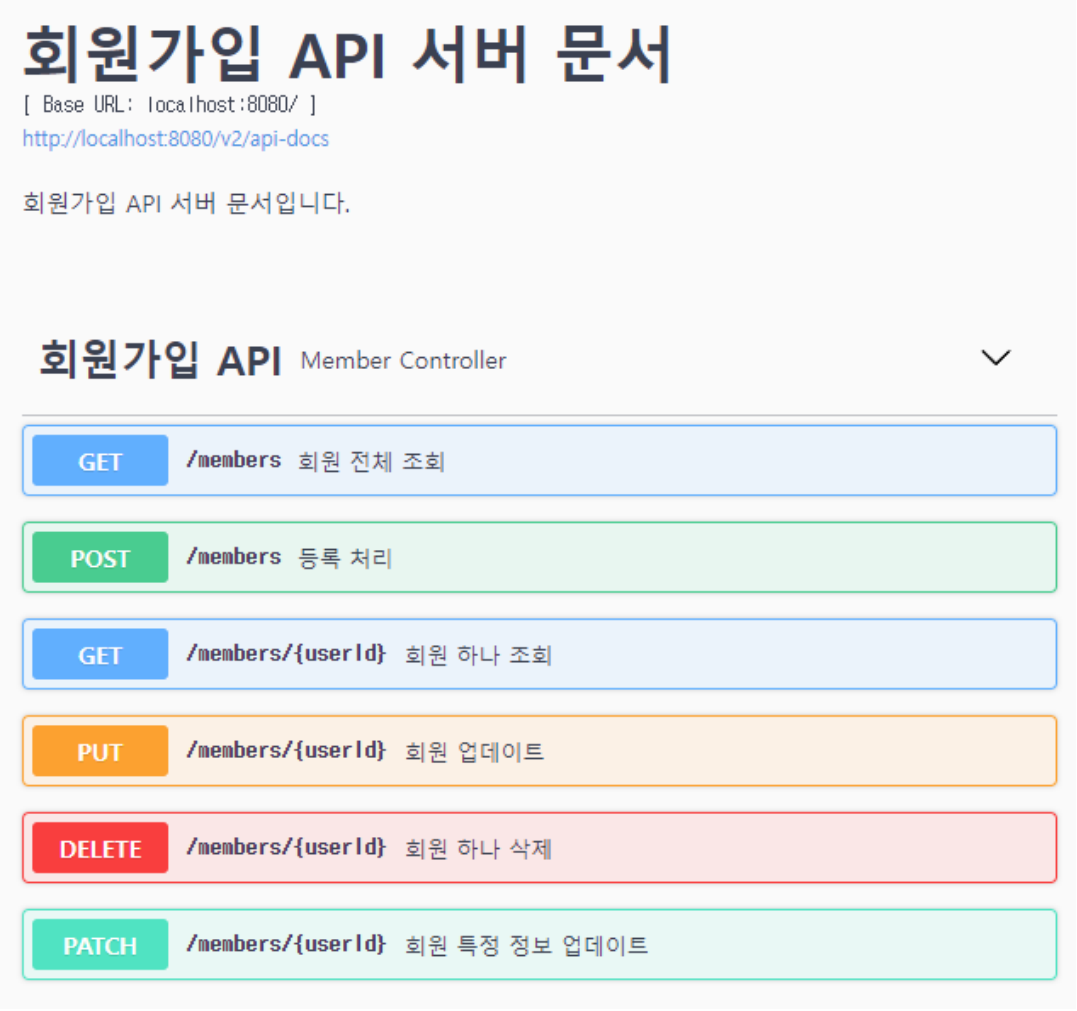


DB 테이블

Member.java

```java
package member.dto;

import javax.persistence.*;

@Entity
@Data
@Getter @Setter
public class Member {

    @Id
    @Column
    @ApiModelProperty(example = "kamyamy9509")
    private String userId;

    @Column
    @ApiModelProperty(example = "asdf123")
    private String password;

    @Column
    @ApiModelProperty(example = "김우현")
    private String name;

    @Column
    @ApiModelProperty(example = "19950905")
    private String birth;

    @Column
    @ApiModelProperty(example = "경기도 성남시")
    private String address;

    @Column
    @ApiModelProperty(example = "여자")
    private String gender;

    @Column
    @ApiModelProperty(example = "abc@naver.com")
    private String email;

    @Column
    @ApiModelProperty(example = "010-4128-9509")
    private String phone;

}
```

API URI 설계



리소스 식별, URI 계층 구조 활용

계층 구조상 상위를 컬렉션으로 보고 복수단어인 /members를 사용하였으며, 회원이라는 리소스만 식별이 가능하도록 설계하여 등록, 조회, 삭제 등의 행위는 분리되도록 설계하였습니다.

GET (/members) : 회원 전체 조회

POST(/members) : 회원 등록

GET(/members/{id}) : 특정 회원 조회

PUT(/members/{id}) : 회원 업데이트

PATCH(/members/{id}) : 회원 특정 정보 업데이트

DELETE(/members/{id}) : 특정 회원 삭제

SWAGGER STS 연동 방법

1) pom.xml (maven)에 dependency 추가

```
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger2</artifactId>
   <version>3.0.0</version>
</dependency>
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-swagger-ui</artifactId>
   <version>3.0.0</version>
</dependency>
<dependency>
   <groupId>io.springfox</groupId>
   <artifactId>springfox-boot-starter</artifactId>
   <version>3.0.0</version>
</dependency>
```

2) application.properties에 버전 일치를 위해 아래 코드 추가

```
20 spring.mvc.pathmatch.matching-strategy=ant-path-matcher
```

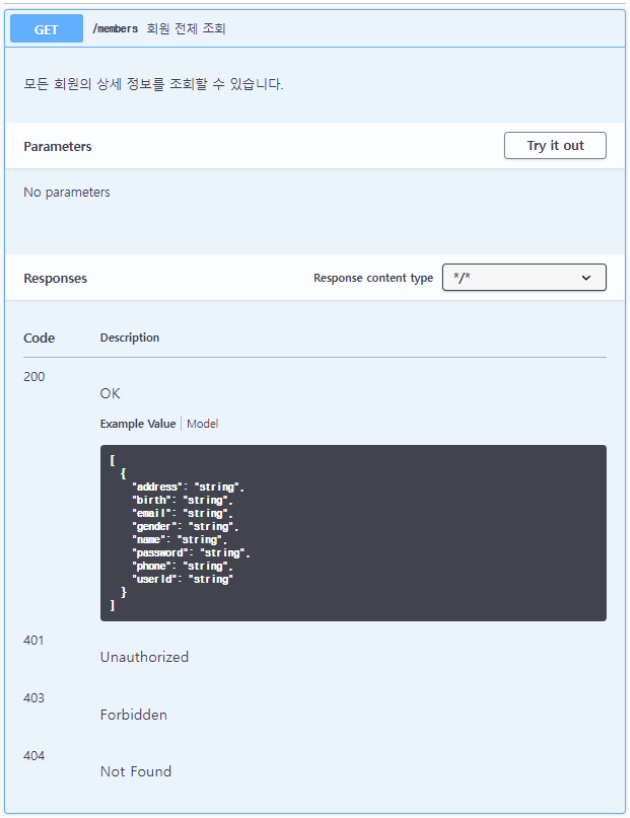
3) Swagger Configuration class 파일 추가하기

```
1  package member.config;
2
3  import org.springframework.context.annotation.Bean;
14
15 @Configuration
16 @EnableSwagger2
17 public class SwaggerConfig {
18
19     @Bean
20     public Docket docket() {
21         ApiInfoBuilder apiInfo = new ApiInfoBuilder();
22         apiInfo.title("회원가입 API 서버 문서");
23         apiInfo.description("회원가입 API 서버 문서입니다.");
24
25         Docket docket = new Docket(DocumentationType.SWAGGER_2);
26         docket.apiInfo(apiInfo.build());
27
28         ApiSelectorBuilder apis = docket.select().apis(RequestHandlerSelectors.basePackage("member.controller"));
29         apis.paths(PathSelectors.ant("/**"));
30
31         return apis.build();
32     }
33 }
```

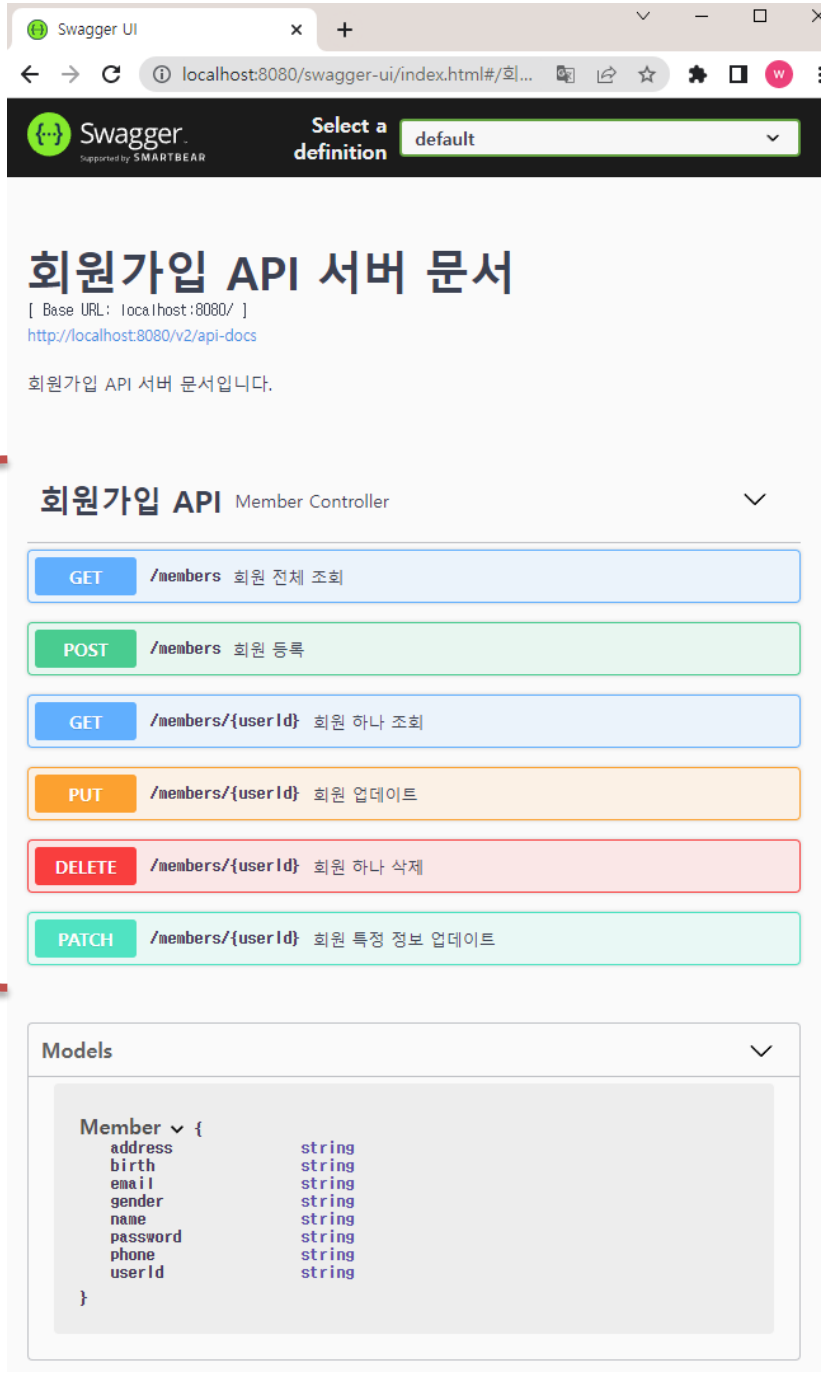
SWAGGER 연동 방법

4) http://localhost:8080/swagger-ui/index.html/

해당 링크로 Swagger API 문서를 확인할 수 있다.

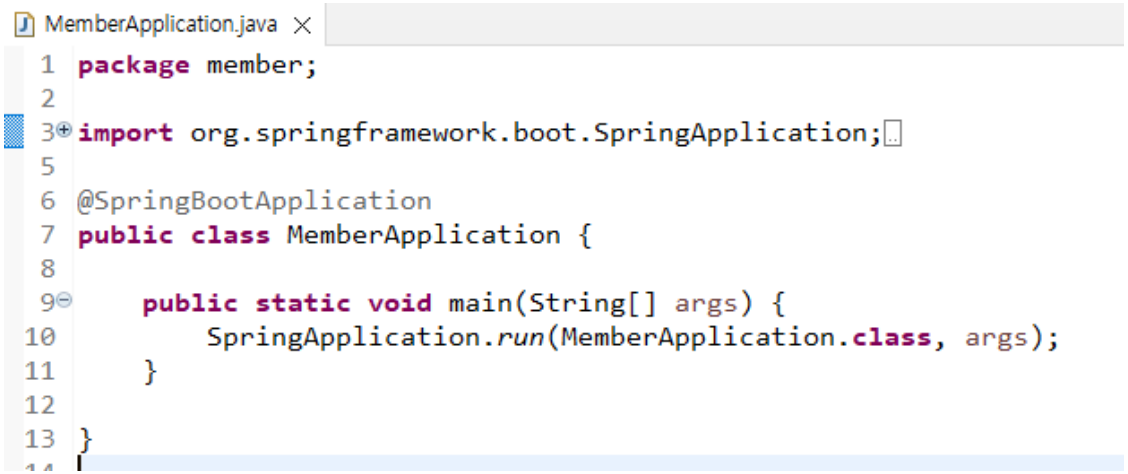


서버로 요청되는 URL 리스트를 HTML 화면으로 문서화 및 테스트할 수 있다.

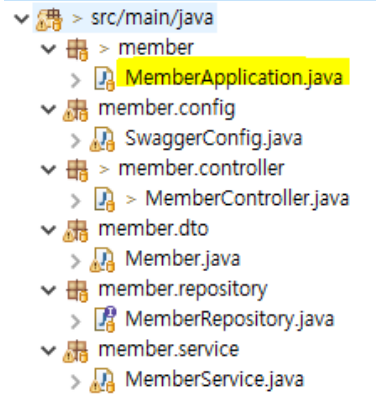


각 클래스별 코드 설명 - main

MemberApplication.java



스프링 부트 프로젝트가 실행되기 위한 프로덕션 소스 코드



각 클래스별 코드 설명 - Config

SwaggerConfig.java


```java
package member.config;

import org.springframework.context.annotation.Bean;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket docket() {
        ApiInfoBuilder apiInfo = new ApiInfoBuilder();
        apiInfo.title("회원가입 API 서버 문서");
        apiInfo.description("회원가입 API 서버 문서입니다.");

        Docket docket = new Docket(DocumentationType.SWAGGER_2);
        docket.apiInfo(apiInfo.build());

        ApiSelectorBuilder apis = docket.select().apis(RequestHandlerSelectors.basePackage("member.controller"));
        apis.paths(PathSelectors.ant("/**"));

        return apis.build();
    }
}
```

.apiInfo() : Swagger API 문서에 대한 설명을 표기하는 메서드

.apis() : Swagger API 문서로 만들기 원하는 BasePackage 경로 설정

.path() : URL 경로를 지정하여 해당 URL에 해당하는 요청만 Swaggger API로 만든다.

(컨트롤러 경로로 지정해줬고 컨트롤러에 각 URL 지정)

각 클래스별 코드 설명 - Domain

Member.java

```java
package member.dto;

import javax.persistence.*;

@Entity
@Data
@Getter @Setter
public class Member {

    @Id
    @Column
    @ApiParam(value = "아이디 입력(예시 : kamyamy)", required = true)
    private String userId;

    @Column
    @ApiParam(value = "비밀번호 입력(예시 : asdf123)", required = true)
    private String password;

    @Column
    @ApiParam(value = "이름 입력(예시 : 김우현)", required = true)
    private String name;

    @Column
    @ApiParam(value = "생년월일 입력(예시 : 19950905)", required = true)
    private String birth;

    @Column
    @ApiParam(value = "주소 입력(예시 : 경기도 성남시)", required = true)
    private String address;

    @Column
    @ApiParam(value = "성별 입력(예시 : 여자)", required = true)
    private String gender;

    @Column
    @ApiParam(value = "이메일 입력(예시 : amy@naver.com)", required = true)
    private String email;

    @Column
    @ApiParam(value = "핸드폰번호 입력(예시 : 010-4128-9509)", required = true)
    private String phone;

}
```

계층 간 데이터 교환을 하기 위해 사용하는
클래스로 직접 DB에 접근하기 위한 Member class

@Entity 어노테이션을 사용해 DB에 테이블을
만들고 Lombok 을 활용하여 getter setter를
자동으로 생성해주었다.

@ApiParam 어노테이션을 사용해 API 문서 상에서
파라미터를 입력할 때 해당 파라미터에 대한 설명을
추가하였고, required=true를 설정하여 반드시
입력을 하도록 설계하였다.

각 클래스별 코드 설명 - Repository

MemberRepository.java

```java
package member.repository;

import java.util.Optional;

@Repository
public interface MemberRepository extends JpaRepository<Member, Integer> {

    Optional<Member> findByUserId(String userId);

    @Transactional
    void deleteByUserId(String userId);

}
```

JPA에서 제공하는 메서드 외에 userId로 member 객체를 찾는 findByUserId를 만들어주었고, 특정 userId를 통해 삭제하는 메서드인 deleteByUserId를 만들었다.

@Transactional 어노테이션을 통해 DB에 접근할 때 직접적인 변화를 주는 메서드인 delete 메서드에 rollback을 하도록 해두었다.

각 클래스별 코드 설명(1) - 회원 등록

MemberController.java


```java
package member.controller;


import io.swagger.annotations.Api;

@RestController
@RequestMapping("/members")
@Api(tags =  "회원가입 API")
@RequiredArgsConstructor
public class MemberController {

    @Autowired
    private MemberService memberService;

    //회원 등록
    @PostMapping(value="")
    @ApiOperation(value = "회원 등록", notes = "신규 회원 등록이 가능합니다.")
    public Member save(Member member) throws Exception {

        member.setPassword(memberService.encrypt(member.getPassword()));
        member.setPhone(memberService.phoneNumForm(member.getPhone()));
        memberService.isValidEmail(member.getEmail());
        memberService.isValidGender(member.getGender());

        return memberService.save(member);
    }

```

@RestController : controller 등록

@RequestMapping : /members를 자동으로 uri 맵핑한다.

@RequiredArgsConstructor : 자동으로 생성자 만들어준다.

```java
    //입력 정보가 형식과 맞지 않을 때 예외 발생시키기
    @ExceptionHandler(NoSuchElementException.class)
    public Object nullEx(Exception e) {

        return "※예외 발생 : 형식에 맞춰 입력을 다시 해주세요.";
    }

```

Controller에서 회원 등록 부분에 해당하는 PostMapping 부분이다.
password 의 경우 서비스 단에서 만들어 놓은 encrypt 메서드를 통해 암호화하여 저장해주고, 핸드폰 번호의 경우 "-"를 없애서 DB에 저장하도록 phoneNumForm을 통해 저장하도록 해주었다.

email과 gender의 경우 유효성 체크를 통해 예외를 발생시켜 예외 발생 시 해당 문구가 return 되도록 하였다.

각 클래스별 코드 설명(1) - 회원 등록

Swagger 결과 값

POST /members 회원 등록

신규 회원 등록이 가능합니다.

Parameters Cancel

Name	Description
address * required string (query)	주소 입력(예시 : 경기도 성남시) 경기도 성남시
birth * required string (query)	생년월일 입력(예시 : 19950905) 19951206
email * required string (query)	이메일 입력(예시 : amy@naver.com) kerica@naver.com
gender * required string (query)	성별 입력(예시 : 여자) 여자
name * required string (query)	이름 입력(예시 : 김우현) 김우현
password * required string (query)	비밀번호 입력(예시 : asdf123) perfec
phone * required string (query)	핸드폰번호 입력(예시 : 010-4128-9509) 010-6659-8887
userId * required string (query)	아이디 입력(예시 : kamyamy) applebanana91

Execute

Responses Response content type */*

Curl

```
curl -X POST "http://localhost:8080/members?address=%EA%B2%BD%EA%B8%B0%EB%8F%84%20%EC%84%B1%EB%82%A8%EC%8B%9C&birth=19951206&email=kerica%40naver.com&gender=%EC%97%AC%EC%9E%90&name=%EA%B9%80%EC%9A%B0%ED%98%84&password=perfec&phone=010-6659-8887&userId=applebanana91" -H "accept: */*" -d ""
```

Request URL

```
http://localhost:8080/members?address=%EA%B2%BD%EA%B8%B0%EB%8F%84%20%EC%84%B1%EB%82%A8%EC%8B%9C&birth=19951206&email=kerica%40naver.com&gender=%EC%97%AC%EC%9E%90&name=%EA%B9%80%EC%9A%B0%ED%98%84&password=perfec&phone=010-6659-8887&userId=applebanana91
```

Server response

Code Details

200

Response body


```
{
  "userId": "applebanana91",
  "password": "980ac2476ed0a24a909e5e7f6d5549b8686145f81c01227de47da655f1218b31",
  "name": "김우현",
  "birth": "19951206",
  "address": "경기도 성남시",
  "gender": "여자",
  "email": "kerica@naver.com",
  "phone": "01066598887"
}
```


Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Sat03 Sep 2022 14:45:46 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

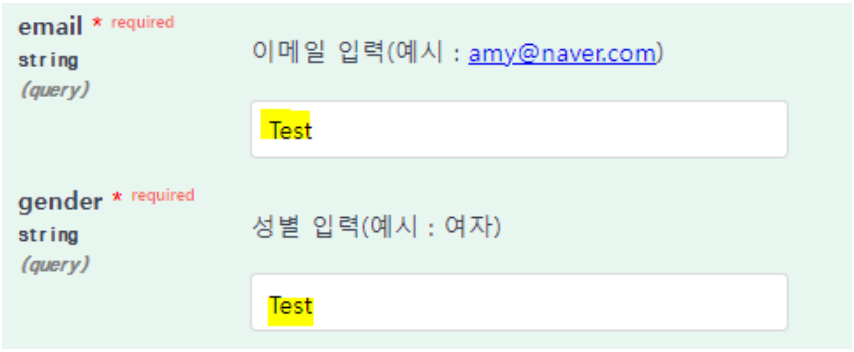
userId	address	birth	email	gender	name	password	phone
applebanana91	경기도 성남시	19951206	kerica@naver.com	여자	김우현	980ac2476ed0a24a909e5e7f6d5549b8686145f...	01066598887
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

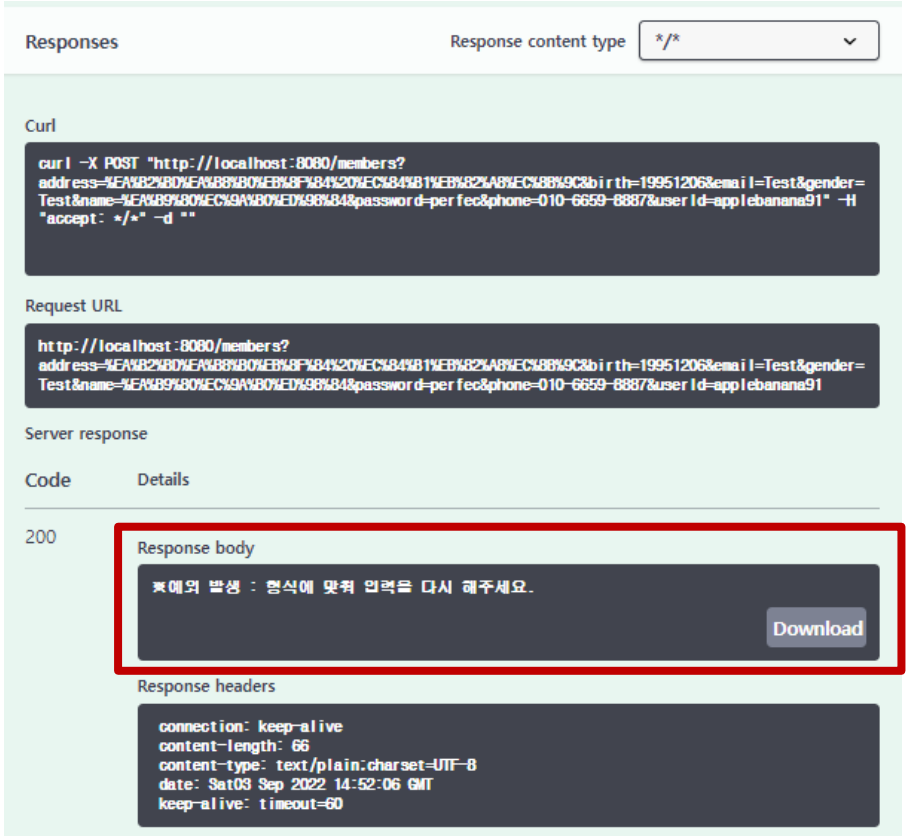
- src/main/java
  - member
    - MemberApplication.java
  - member.config
    - SwaggerConfig.java
  - member.controller
    - MemberController.java
  - member.dto
    - Member.java
  - member.repository
    - MemberRepository.java
  - member.service
    - MemberService.java

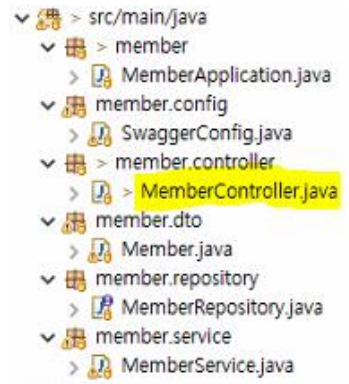
파라미터 값을 적어서 execute 하면 password가 암호화되고,
폰번호의 경우 "-"을 없애고 DB에 저장된다.

각 클래스별 코드 설명(1) - 회원 등록

Swagger 결과 값







이메일과 성별 파라미터 값에 형식이 다르게 입력을 할 경우 reponse body에 예외 발생이라는 문구가 기재되도록 설계하였다.

각 클래스별 코드 설명(2) - 회원 조회

MemberController.java

```
42      //전체 조회
43      @GetMapping(value = "") // user 테이블의 모든 정보를 읽어옴
44      @ApiOperation(value = "회원 전체 조회", notes = "모든 회원의 상세 정보를 조회할 수 있습니다.")
45      public List<Member> getList() {
46          return memberService.findAll();
47      }
48
49
50      //하나 조회
51      @GetMapping(value="/{userId}")
52      @ApiOperation(value = "회원 하나 조회", notes = "아이디를 가지고 특정 회원의 상세 정보를 조회할 수 있습니다.")
53      @ApiImplicitParams({
54          @ApiImplicitParam(name = "userId", value = "회원 아이디", example = "amy9595")
55      })
56      public Member oneView(@PathVariable String userId) {
57          Member member = memberService.findByUserId(userId).get();
58
59          return member;
60      }
```

회원 전체 조회의 경우 GetMapping으로 JPA의 findAll() 메서드를 사용해 List<Member>에 담아 반환하였다.

특정 회원 조회의 경우 GetMapping으로 JPA에 만들어 둔 findByUserId()메서드를 사용하여 member 객체를 반환하였다.

각 클래스별 코드 설명(2) - 회원 조회

Swagger 결과 값

GET /members 회원 전체 조회

모든 회원의 상세 정보를 조회할 수 있습니다.

Parameters Cancel

No parameters

Execute Clear

Responses Response content type */*

Curl

```
curl -X GET "http://localhost:8080/members" -H "accept: */*"
```

Request URL

```
http://localhost:8080/members
```

Server response

Code Details

200

Response body


```
[
  {
    "userId": "applebanana91",
    "password": "980ac2476ed0a24a909e5e7f6d5549b8686145f81c01227de47da655f1218b31",
    "name": "김우현",
    "birth": "19951206",
    "address": "경기도 성남시",
    "gender": "여자",
    "email": "kerica@naver.com",
    "phone": "01066598887"
  },
  {
    "userId": "dain9191",
    "password": "72fb1ed4c76952b3254cff2660ec3179f12edaac2b83928d76bd157f5a5e19ed",
    "name": "이다인",
    "birth": "19910403",
    "address": "대전 유성구",
    "gender": "여자",
    "email": "dain@gmail.com",
    "phone": "01066598887"
  },
  {
    "userId": "fishandchips",
    "password": "d895e3eab45791a39413e616e76d21f0e011249a6ea1ff7e90a932a962662ba",
    "name": "김희현",
    "birth": "19940203",
    "address": "서울시 강남구",
```


Download

Json 형식으로 전체 리스트 반환

GET /members/{userId} 회원 하나 조회

아이디를 가지고 특정 회원의 상세 정보를 조회할 수 있습니다.

Parameters Cancel

Name Description

userId * required
string
(path)

회원 아이디

applebanana91

Execute Clear

Responses Response content type */*

Curl

```
curl -X GET "http://localhost:8080/members/applebanana91" -H "accept: */*"
```

Request URL

```
http://localhost:8080/members/applebanana91
```

Server response

Code Details

200

Response body


```
{
  "userId": "applebanana91",
  "password": "980ac2476ed0a24a909e5e7f6d5549b8686145f81c01227de47da655f1218b31",
  "name": "김우현",
  "birth": "19951206",
  "address": "경기도 성남시",
  "gender": "여자",
  "email": "kerica@naver.com",
  "phone": "01066598887"
}
```


Download

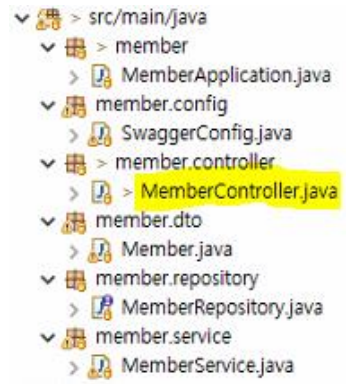
파라미터 값(회원 아이디) 입력하면 특정 회원 정보 반환

- src/main/java
  - member
    - MemberApplication.java
  - member.config
    - SwaggerConfig.java
  - member.controller
    - MemberController.java
  - member.dto
    - Member.java
  - member.repository
    - MemberRepository.java
  - member.service
    - MemberService.java

각 클래스별 코드 설명(3) - 회원 업데이트

MemberController.java

```
62      //전체 업데이트 (덮어쓰기)
63      @PutMapping(value="/{userId}")
64      @ApiOperation(value = "회원 업데이트", notes = "아이디를 가지고 회원의 상세정보를 업데이트할 수 있습니다.")
65
66      public Member update(@PathVariable("userId") String userId, @RequestBody Member member) {
67
68          return memberService.update(member);
69      }
70
71      //업데이트(특정부분)
72      @PatchMapping(value="/{userId}")
73      @ApiOperation(value = "회원 특정 정보 업데이트", notes = "아이디를 가지고 회원의 특정 정보를 업데이트할 수 있습니다.")
74
75      public Member update_part(@RequestBody Member member) {
76
77          return memberService.patch(member);
78
79      }
```



회원 전체 업데이트의 경우 특정 회원의 정보를 전체를 한꺼번에 입력하여 한꺼번에 업데이트를 하는 것이기 때문에 덮어쓰기의 기능이 있는 PUT을 사용하였다.

회원 부분 업데이트의 경우 특정 회원의 정보의 일부분만을 업데이트하는 것이기 때문에 PATCH를 사용하여 설계해주었다.

각 클래스별 코드 설명(3) - 회원 정보 전체 업데이트

Swagger 결과 값

PUT /members/{userId} 회원 업데이트

아이디를 가지고 회원의 상세정보를 업데이트할 수 있습니다.

Parameters Cancel

Name	Description

member * required
object
(body)

member

Edit Value | Model

```
"address": "changed",
"birth": "changed",
"email": "changed",
"gender": "남자",
"name": "changed",
"password": "changed",
"phone": "010-1313-1313",
"userId": "applebanana91"
```

Cancel

Parameter content type
application/json

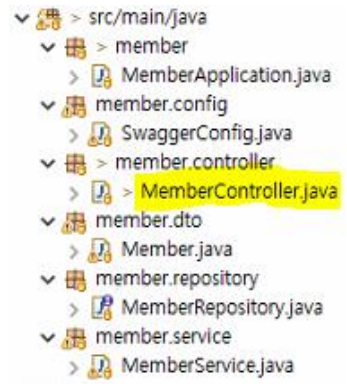
userId * required
string
(path)

userId

applebanana91

Execute Clear

특정 회원의 아이디를 pathVariable로 받아서 해당 회원 정보의 전체를 바꿔주었고, DB 상에서도 변경된 것을 확인할 수 있었다.



Server response

Code Details

200

Response body

```
{
  "userId": "applebanana91",
  "password": "changed",
  "name": "changed",
  "birth": "changed",
  "address": "changed",
  "gender": "남자",
  "email": "changed",
  "phone": "010-1313-1313"
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Sat03 Sep 2022 15:06:51 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

[변경 전 DB 테이블]

userId	address	birth	email	gender	name	password	phone
applebanana91	경기도 성남시	19950902	amy9595@naver.com	여자	김우현	perfectlyt	010-2222-2222
dain9191	대전 유성구	19910403	dain@gmail.com	여자	이다인	72fb1ed4c76952b32...	01066598887
fishandchips	서울시 강남구	19940203	kericak919@yahoo.com	남자	김희현	d895e3eab455791a...	01023221133

[변경 후 DB 테이블]

userId	address	birth	email	gender	name	password	phone
applebanana91	changed	changed	changed	남자	changed	changed	010-1313-1313
dain9191	대전 유성구	19910403	dain@gmail.com	여자	이다인	72fb1ed4c76952b32...	01066598887
fishandchips	서울시 강남구	19940203	kericak919@yahoo.com	남자	김희현	d895e3eab455791a...	01023221133

각 클래스별 코드 설명(3) - 회원 정보 부분 업데이트

Swagger 결과 값

PATCH /members/{userId} 회원 특정 정보 업데이트

아이디를 가지고 회원의 특정 정보를 업데이트할 수 있습니다.

Parameters Cancel

Name	Description

member * required
object
(body)
	member
	Edit Value | Model
	{
	"address": "미국",
	"userId": "applebanana91"
	}

Cancel

Parameter content type: application/json

Execute Clear

Server response

Code: 200

Details

Response body

```
{
  "userId": "applebanana91",
  "password": "changed",
  "name": "changed",
  "birth": "changed",
  "address": "미국",
  "gender": "남자",
  "email": "changed",
  "phone": "010-1313-1313"
}
```

Download

Response headers

```
connection: keep-alive
content-type: application/json
date: Sat03 Sep 2022 15:12:45 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

[변경 전 DB 테이블]

userId	address	birth	email	gender	name	password	phone
applebanana91	changed	changed	changed	남자	changed	changed	010-1313-1313
dain9191	대전 유성구	19910403	dain@gmail.com	여자	이다인	72fb1ed4c76952b32...	01066598887
fishandchips	서울시 강남구	19940203	kericak919@yahoo.com	남자	김희현	d895e3eab455791a...	01023221133

[변경 후 DB 테이블]

userId	address	birth	email	gender	name	password	phone
applebanana91	미국	changed	changed	남자	changed	changed	010-1313-1313
dain9191	대전 유성구	19910403	dain@gmail.com	여자	이다인	72fb1ed4c76952b32...	01066598887
fishandchips	서울시 강남구	19940203	kericak919@yahoo.com	남자	김희현	d895e3eab455791a...	01023221133

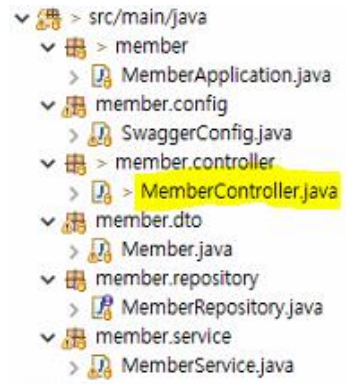
특정 회원의 정보 중 일부분만을 수정하기 위해서 address 부분을 미국으로 바꿨다.

나머지 정보들은 그대로이지만 특정 컬럼인 address 부분만 미국으로 바뀐 것을 확인할 수 있다.

각 클래스별 코드 설명(4) - 회원 삭제

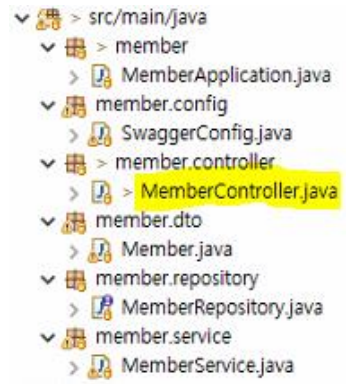
MemberController.java

```
82      //삭제 처리
83      @DeleteMapping(value="/{userId}")
84      @ApiOperation(value = "회원 하나 삭제", notes = "아이디를 가지고 회원의 정보를 삭제할 수 있습니다.")
85      @ApiImplicitParams({
86          @ApiImplicitParam(name = "userId", value = "회원 아이디", example = "amy9595")
87      })
88      public boolean delete(@PathVariable String userId) {
89          Member member = memberService.findByUserId(userId).get();
90          if(member == null) {
91              return false;
92          }
93          memberService.delete(userId);
94          return true;
95      }
```

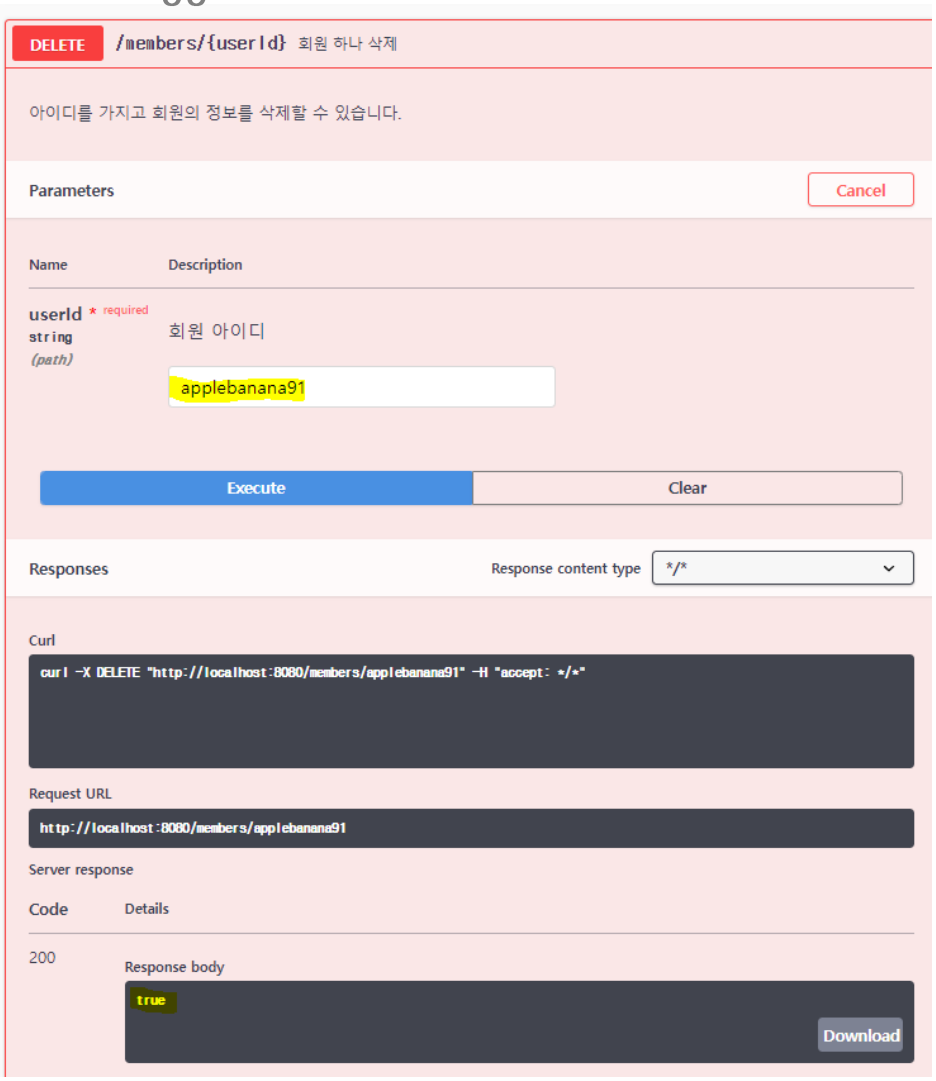


특정 회원 정보를 삭제하기 위해 pathvariable을 userId로 설정하여 DeleteMapping을 해주었다. 회원을 조회하여 삭제해야하는데, 조회하고자 하는 회원의 아이디가 없는 경우 삭제가 되지 않았기에 false를 리턴하도록 하였고, 삭제가 성공적으로 이뤄졌을 때 true를 반환하도록 설계하였다.

각 클래스별 코드 설명(4) - 회원 삭제



Swagger 결과 값



[변경 전 DB 테이블]

userId	address	birth	email	gender	name	password	phone
applebanana91	경기도 성남시	19950902	amy9595@naver.com	여자	김우현	perfectlyt	010-2222-2222
dain9191	대전 유성구	19910403	dain@gmail.com	여자	이다인	72fb1ed4c76952b32...	01066598887
fishandchips	서울시 강남구	19940203	kericak919@yahoo.com	남자	김희현	d895e3eab455791a...	01023221133

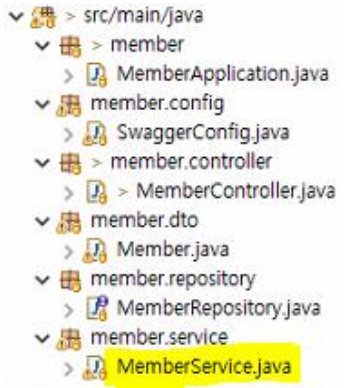
[변경 후 DB 테이블]

userId	address	birth	email	gender	name	password	phone
dain9191	대전 유성구	19910403	dain@gmail.com	여자	이다인	72fb1ed4c76952b32...	01066598887
fishandchips	서울시 강남구	19940203	kericak919@yahoo.com	남자	김희현	d895e3eab455791a...	01023221133

applebanana91이라는 user의 아이디를 파라미터 값으로 넘겨주면 responsebody에선 true값을 리턴하고 DB테이블 상에서는 해당 userId의 정보가 삭제된 것을 확인할 수 있다.

각 클래스별 코드 설명(1)

MemberService.java



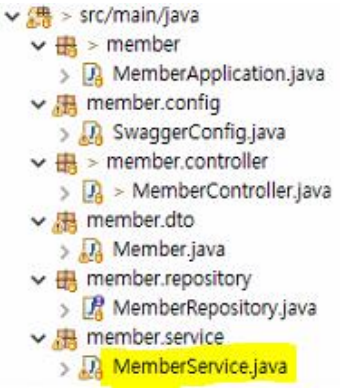

```java
package member.service;


import org.springframework.beans.factory.annotation.Autowired;

@Service
public class MemberService {

    @Autowired
    private MemberRepository memberRepository;

    //가입하기
    public Member save(Member member) {
        //기존에 가입 아이디가 존재한다면
        if(findByUserId(member.getUserId()).isPresent()) {
            throw new IllegalStateException("아이디가 존재합니다.");
        }

        return memberRepository.save(member);
    }

    //조회하기(회원 전체 정보 받기)
    public List<Member> findAll () {
        return memberRepository.findAll();
    }

    //조회하기(특정 ID의 회원 정보만 받기)
    public Optional<Member> findByUserId (String userId) {
        return memberRepository.findByUserId(userId);
    }

    public void delete(String userId) {
        memberRepository.deleteByUserId(userId);
    }
```

비즈니스 로직을 처리하는 부분인 service 단에서는
CRUD의 부분을 처리하기 위한 메서드들을
생성해주었다.

repository를 autowired해서 각 부분에 있어서
필요한 메서드들을 바로 사용할 수 있도록 하였다.

각 클래스별 코드 설명(2)

MemberService.java




```
50      //회원정보 업데이트(모든 정보 한번에 수정) key - ID
51      public Member update(Member member) {
52
53          Member selected = findByUserId(member.getUserId()).get();
54          //회원 아이디가 존재하지 않는 경우
55          if(selected == null) {
56              throw new NoSuchElementException();
57          }
58          selected.setUserId(member.getUserId());
59          selected.setPassword(member.getPassword());
60          selected.setName(member.getName());
61          selected.setBirth(member.getBirth());
62          selected.setAddress(member.getAddress());
63          selected.setGender(member.getGender());
64          selected.setEmail(member.getEmail());
65          selected.setPhone(member.getPhone());
66
67          return memberRepository.save(selected);
68
69      }
```

회원 정보 전체 업데이트 부분의 서비스단에서는 멤버 객체를 파라미터 값으로 받아서 그 멤버의 userId를 get으로 부른 후 findByUserId를 통해 변수에 넣어준다.

그 변수 값이 null인 경우는 해당 회원이 없는 것이니까 예외를 발생시켜주고,

나머지의 경우는 파라미터로 받은 애들의 정보들을 전부 setting 해준다.

세팅된 멤버 객체 변수를 리턴해주는 로직으로 작성하였다.

각 클래스별 코드 설명(3)

MemberService.java

```
71    //회원정보 업데이트(특정 필드만 수정) , 수정 필드명과 값 지정
72    public Member patch(Member member) {
73        Member selected = findByUserId(member.getUserId()).get();
74        //회원 아이디가 존재하지 않는 경우
75        if(selected == null) {
76            throw new NoSuchElementException();
77        }
78
79        //내가 바꾸고자 하는 게 패쓰워드면 받아온 member 값이 null값이 아니지
80        if(member.getPassword() != null) {
81            selected.setPassword(member.getPassword());
82        }
83        if(member.getName() != null) {
84            selected.setName(member.getName());
85        }
86        if(member.getBirth() != null) {
87            selected.setBirth(member.getBirth());
88        }
89        if(member.getAddress() != null) {
90            selected.setAddress(member.getAddress());
91        }
92        if(member.getGender() != null) {
93            selected.setGender(member.getGender());
94        }
95        if(member.getEmail() != null) {
96            selected.setEmail(member.getEmail());
97        }
98        if(member.getPhone() != null) {
99            selected.setPhone(member.getPhone());
100       }
101
102       return memberRepository.save(selected);
103   }
```

회원 정보 부분 업데이트 부분 역시도 파라미터 값을 멤버 객체를 받아서 그 파라미터의 id가 현재 없으면 예외를 만들어내도록 설계하였다.

파라미터 값으로 넘어오는 것은 특정 필드의 값일테니까 그 특정 필드의 값이 null이 아니면 setting을 진행해주는 방식으로 로직을 짰다.

if문으로 파라미터로 전달받은 멤버 객체의 필드 값이 null이 아니면 해당 필드의 값을 받은 파라미터의 필드 값으로 대체해주는 방식이다.

각 클래스별 코드 설명(4)

MemberService.java


```java
    //비밀번호 암호화
    public String encrypt(String pw) throws Exception {
        String str = "";
        try {
            MessageDigest sh = MessageDigest.getInstance("SHA-256");
            sh.update(pw.getBytes());
            byte byteData[] = sh.digest();
            StringBuffer sb = new StringBuffer();
            for (int i = 0; i < byteData.length; i++) {
                sb.append(Integer.toString((byteData[i]&0xff) + 0x100, 16).substring(1));
            }
            str = sb.toString();
        } catch (NoSuchElementException e) {
            e.printStackTrace();
            str = null;
        }
        return str;
    }
```

[값이 변경되어 들어간 DB 테이블]

password
d895e3eab455791a39413e616e76d21f0e011249a6ea1ff...
72fb1ed4c76952b3254cff2660ec3179f12edaac2b33928d...
d895e3eab455791a39413e616e76d21f0e011249a6ea1ff...

DB에 저장하기 전에 password를 암호화하는 메서드로 encrypt를 생성하였다.

파라미터로 받는 pw strin값을 SHA-256 암호화 방식으로 build 해주어서 값을 리턴해주었고,

```java
    //핸드폰번호 '-'없애기
    public String phoneNumForm(String phoneNum) {
        String changedForm = phoneNum.replace("-", "");

        return changedForm;
    }
```

[값이 변경되어 들어간 DB 테이블]

phone
01023221133
01066598887
01023221133

해당 리턴 값을 controller단에서 DB에 save될 때 변경되어 저장되도록 설계하였다.

phoneNumForm 역시도 param으로 받은 값은 "-"가 포함된 string 값이므로 DB에 저장될 땐 순수하게 번호만 저장된 string 값으로 들어가도록 설계하였다.

각 클래스별 코드 설명(5)

MemberService.java


```java
    //이메일 체크
    public void isValidEmail(String email) {
        boolean err = false;
        String regex = "^[_a-z0-9-]+(.[_a-z0-9-]+)*@(?:\\w+\\.)+\\w+$";
        Pattern p = Pattern.compile(regex);
        Matcher m = p.matcher(email);
        if (m.matches()) {
            err = true;
        }
        if(err == false) {
            throw new NoSuchElementException();
        }
    }

    //성별체크
    public void isValidGender(String gender) {
        if(!(gender.equals("여자") || gender.equals("남자"))) {
            throw new NoSuchElementException();
        }
    }
```

```java
    //입력 정보가 형식과 맞지 않을 때 예외 발생시키기
    @ExceptionHandler(NoSuchElementException.class)
    public Object nullEx(Exception e) {

        return "※예외 발생 : 형식에 맞춰 입력을 다시 해주세요.";
    }
```

Response body

※예외 발생 : 형식에 맞춰 입력을 다시 해주세요.

Download

유효성 검사 메서드들로 이메일과 성별체크를 만들었다.

이메일의 경우 영문과 숫자가 포함되고 @가 들어간 형식으로 설정하여 해당 값이 아닌 값이 들어왔으르 경우 예외를 발생시키도록 하였다.

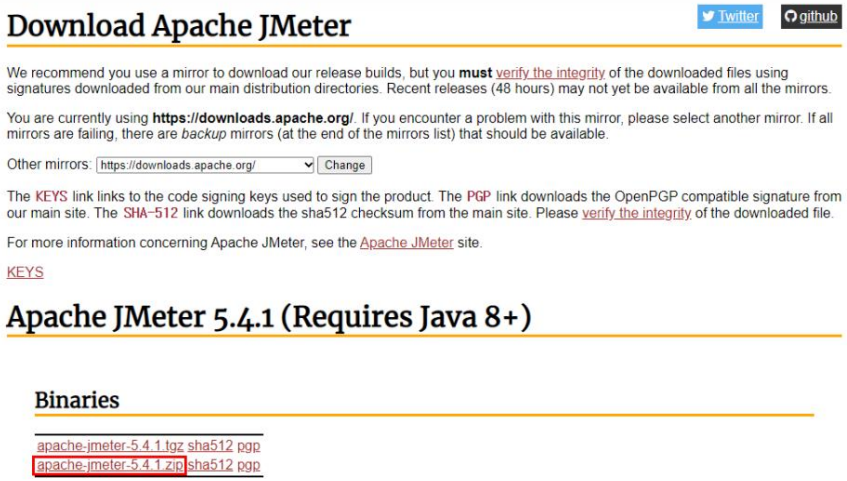
성별체크 역시 "남자" 혹은 "여자"로만 입력받을 수 있도록 설계하였고, 다른 값이 들어오면 예외가 발생하도록 하였다.

예외는 exceptionHandler 어노테이션으로 해당 String을 반환하도록 만들어주었다.

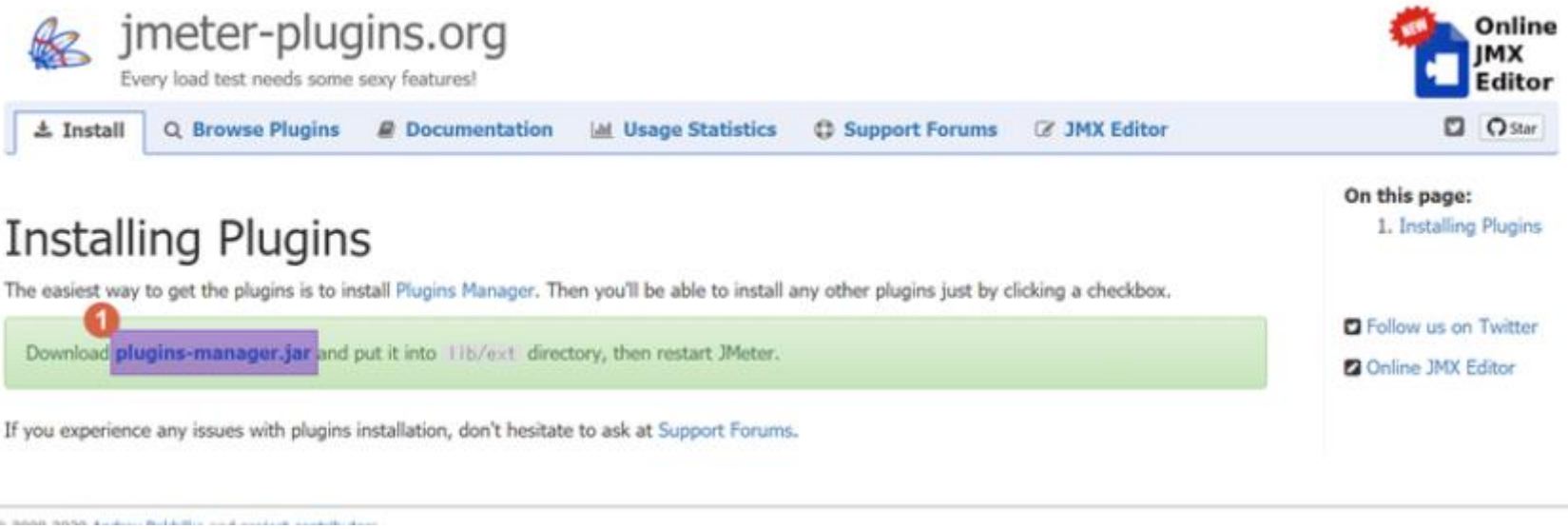
API 테스트 실행 Jmeter 성능테스트 도구

https://jmeter.apache.org/

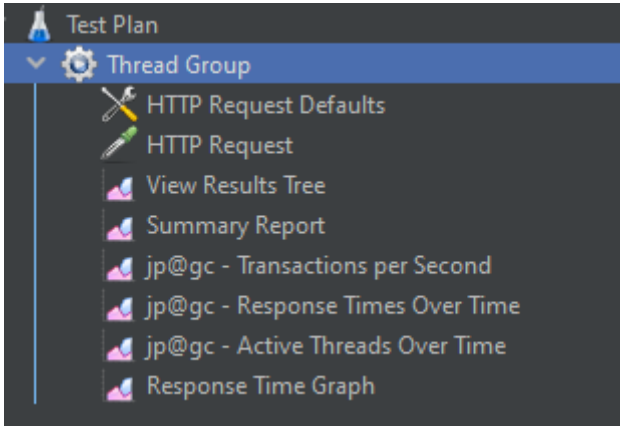
서버가 제공하는 서비스에 대한 성능을 측정하고 사용자에게 보여주는 테스트 도구이며, 서버에 요청을 많이 줘서 서버가 얼마나 버틸 수 있는 지 확인할 수 있다.

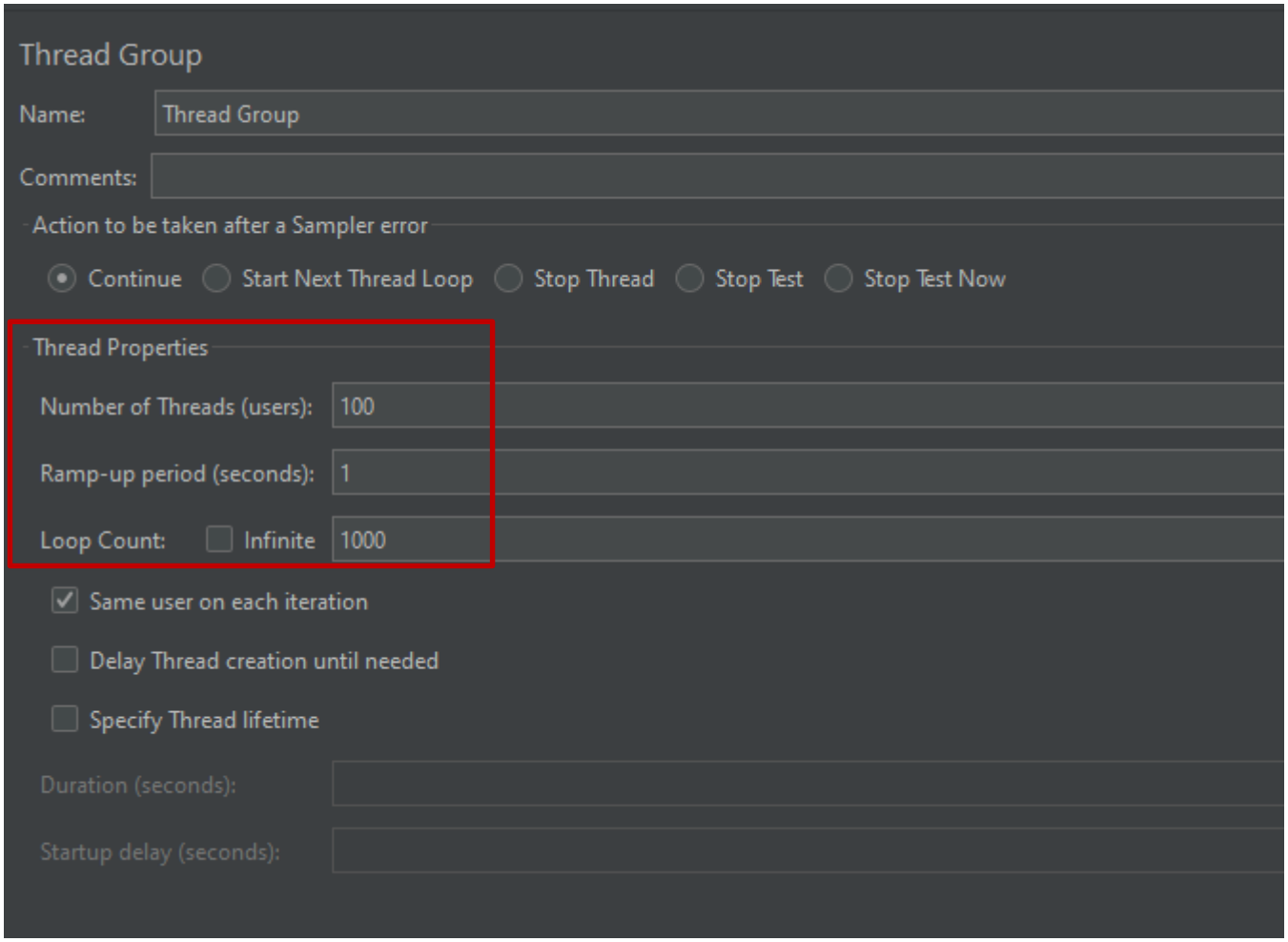


Apache Jmeter설치를 진행하고,
결과의 그래프 시각화를 위해 플러그인을
설치해준다.



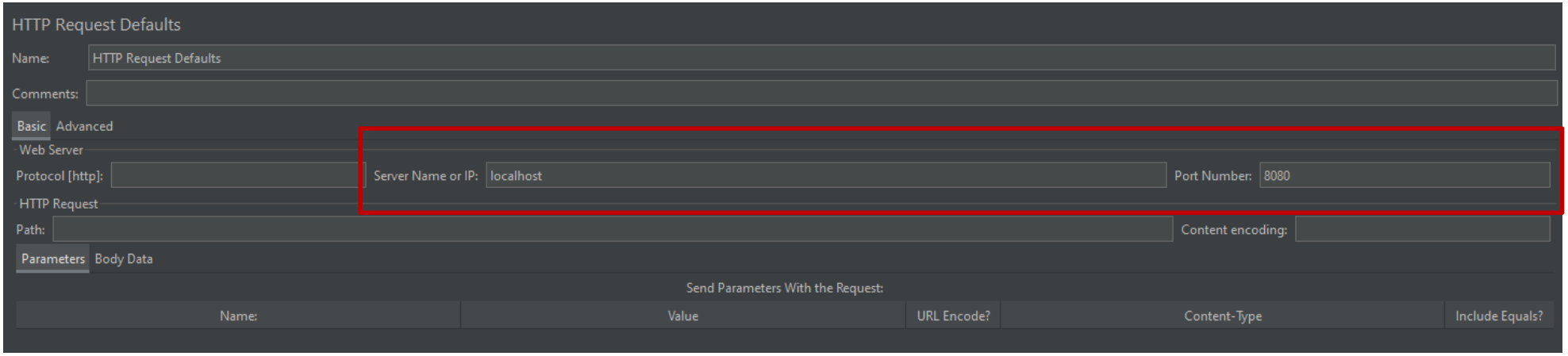
API 테스트 실행 Jmeter 성능테스트 도구





thread는 생성될 쓰레드의 수로 한명의 테스트 유저로 간주된다. loop count는 테스트를 반복할 횟수를 지정해주는 것이다. 100명의 유저가 1000번의 횟수로 API를 호출했을 때의 테스트를 진행해보고자 한다.

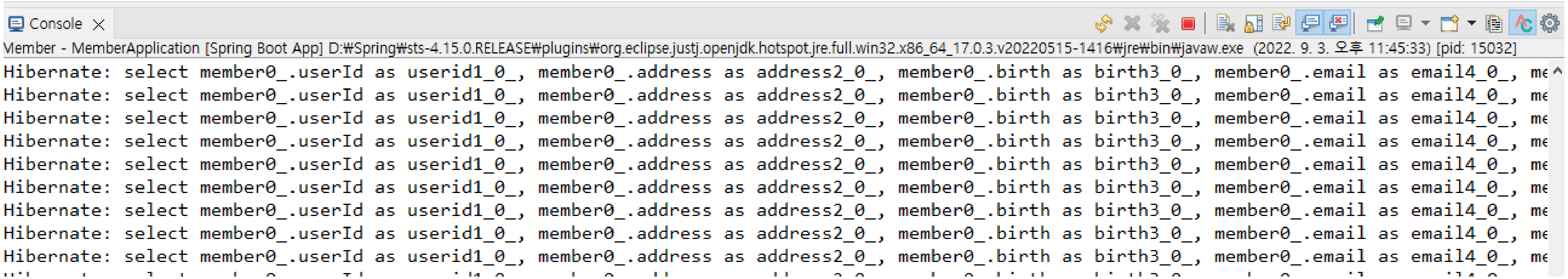
API 테스트 실행 Jmeter 성능테스트 도구



HTTP Request의 Web Server 설정은 localhost, 8080 포트로 설정해준다.

테스트 해볼 HTTP Request 메서드는 GET 조회 API이기 때문에 해당 path인 /members를 기재해준다.

API 테스트 실행 Jmeter 성능테스트 도구



테스트가 진행됨과 동시에 console 창에는 지속적으로 select문이 실행되고 아래와 같이 결과가 지속적 업데이트 된다.

API 테스트 결과분석

Summary Report

Name: Summary Report

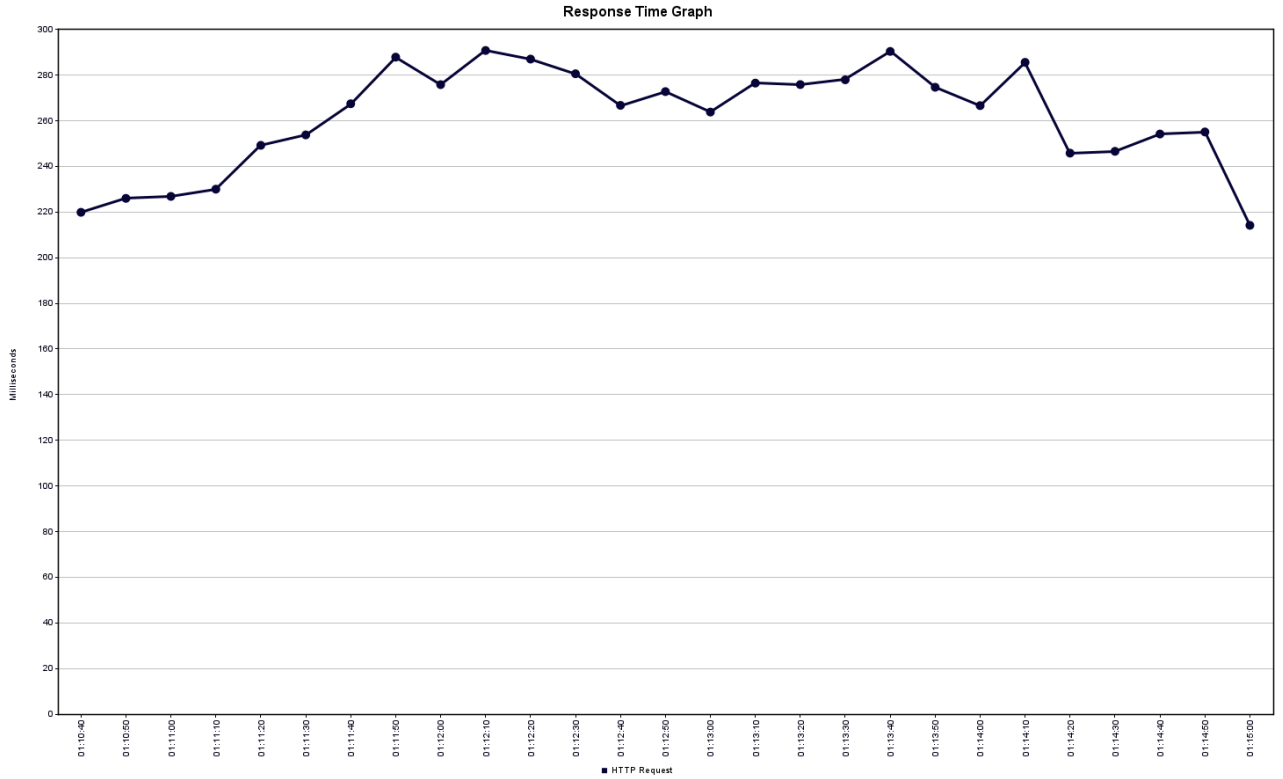
Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	202000	246	2	1875	109.49	0.00%	1.8/sec	1.21	0.22	667.9
TOTAL	202000	246	2	1875	109.49	0.00%	1.8/sec	1.21	0.22	667.9

100 쓰레드로 1000 루프를 돌린 결과 error율은 0%로 결과가 나왔기에 API는 잘 작동하는 것을 확인할 수 있었다.
응답시간의 경우 Average가 246이 나왔는데, 이는 평균 0.246초 안에 모든 응답을 처리했다는 말이다.



감사합니다 !