spring_RESTful API

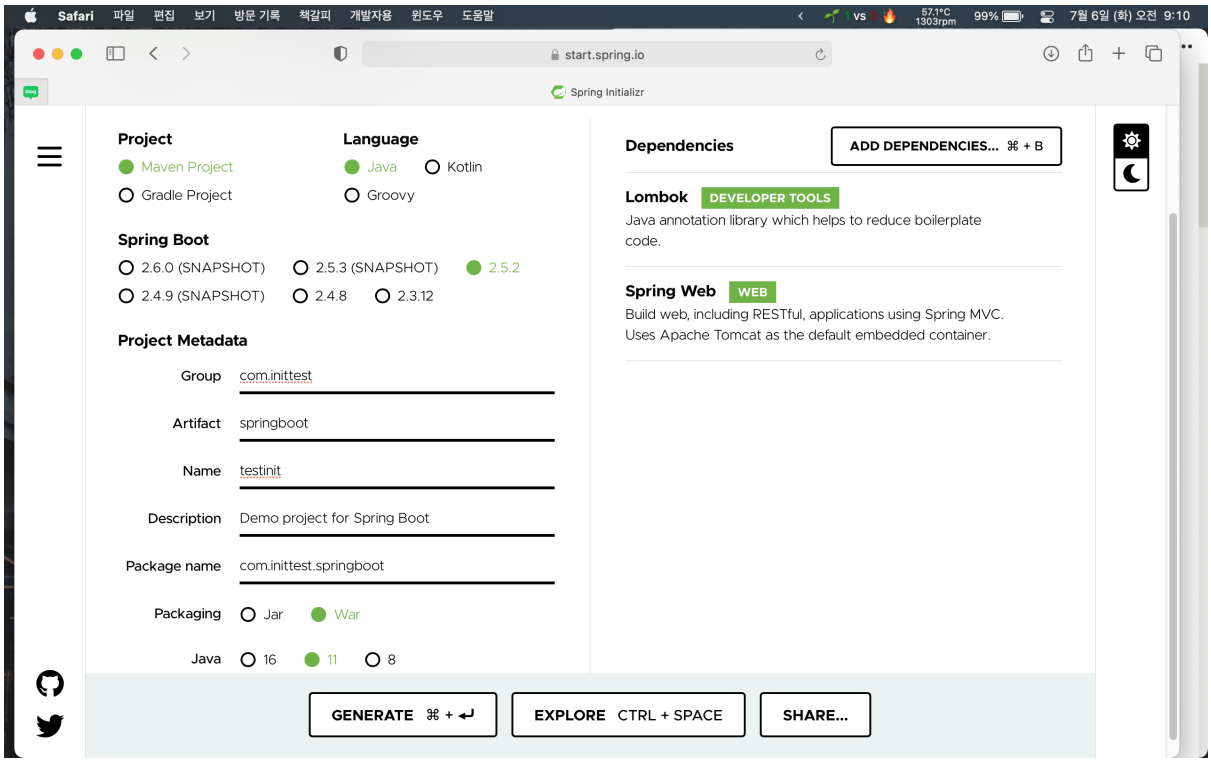
언어	
열	
유형	spring
출처	
태그	

설명 잘해놓은 블로그

https://mangkyu.tistory.com/46

https://mangkyu.tistory.com/46

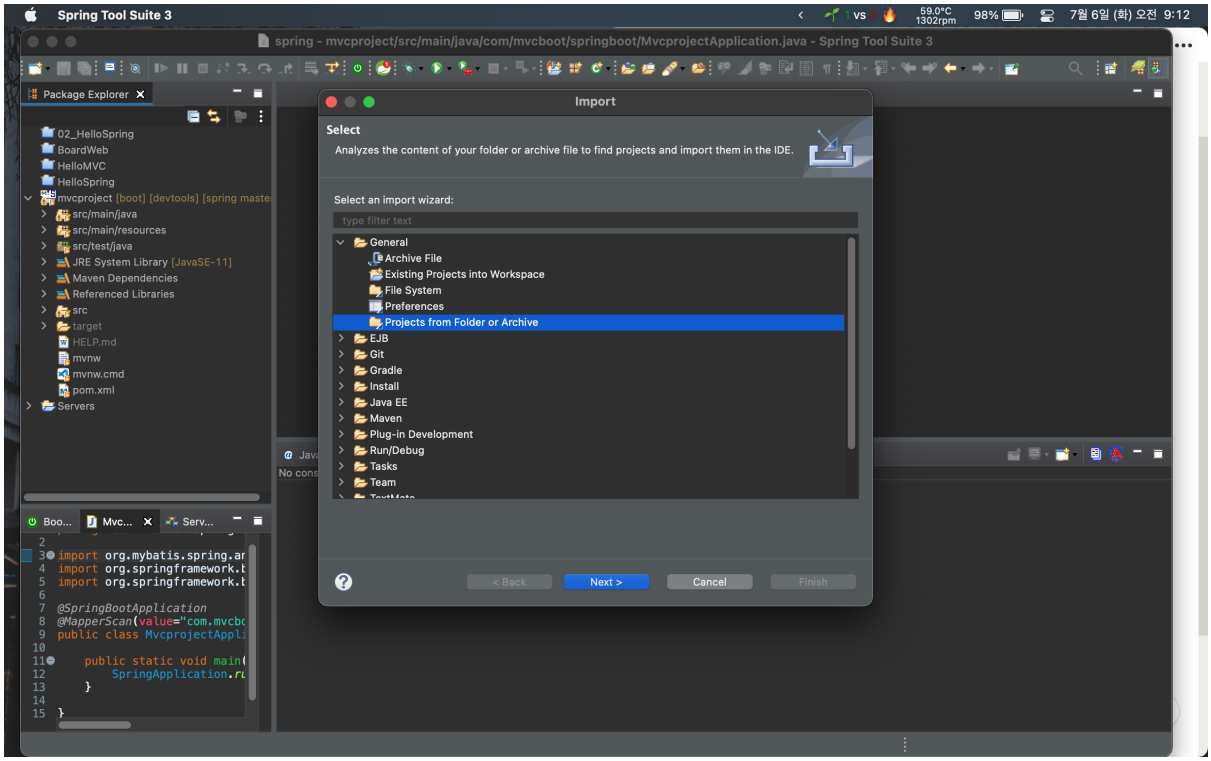
spring.io 접속



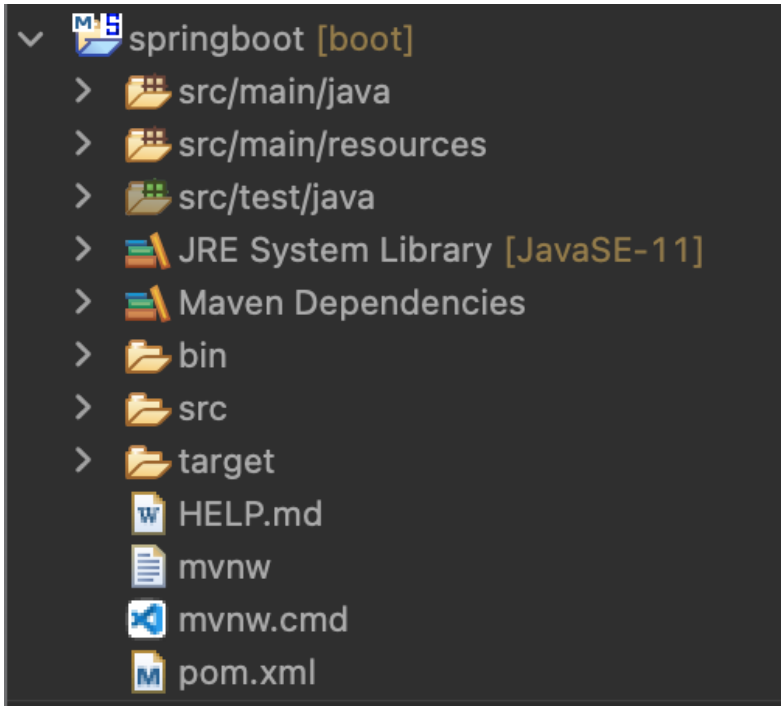
GENERATE → 압축파일로 다운로드된다

프로젝트 import

File > Import > General > Projects from Folder or Archive > Next > Archive버튼 클릭 > 압축폴더 클릭 > 프로젝트명.zip.expanded 는 제외하고 import (압축폴더 안의 프로젝트만 필요하니까) > Finish



프로젝트명 중복되면 안된다



application.properties 대신에 application.yml 사용

```
server:
  port: 9090
```

RESTful 방식

com.inittest.springboot.controller 패키지에 RestTestController 클래스 생성

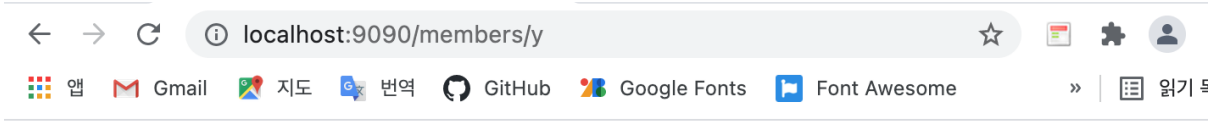
```
package com.inittest.springboot.controller;

import java.util.ArrayList;
import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import com.inittest.springboot.model.vo.Member;

@RestController
public class RestTestController {

  // RESTful방식(REST-api설계)은
  // 요청주소를 URI를 보고 판단할 수 있게 설계하고
  // 단위서비스로 구현하여 모듈단위로 이용할 수 있게 하는 것

  // 예를들어
  // localhost:9090/selectMemberAll 가 아니라
  // localhostL9090/members
  // 행위에 대한것은 method(get, post, put, delete)로 지정

  // method로 uri주소가 하는 행위를 지정한다.
  // uri설계시에는 그 행위에 대한 내용은 제외한다.
  // 명사로만 표현

  // method의 종류
  // GET : 조회(select)
  // POST : 입력(create)
  // PUT : 수정(update)
  // DELETE : 삭제(delete)

  // 예를들어 get .../members           -> 회원조회
  // GET :: localhost:9090/member/admin     -> 회원중 admin을 조회
  // GET :: localhost:9090/board/1      -> 1번 게시글 조회
  // POST :: localhost:9090/member        -> 회원등록 ( 회원정보가 requestBody에 있으니까 )
  // PUT :: localhost:9090/member/admin     -> admin회원을 수정
  // DELETE :: localhost:9090/member/admin  -> admin회원을 삭제

  // RESTful방식에서 서버는 데이터만 전송하는 역할을 하게 된다
  // JSON으로
  // 서버에서 session을 관리하지 않는다
  // 서버에서 클라이언트 상태를 관리하지 않는다

  private List<Member> list = List.of(Member.builder().userId("유병승").age(19).email("yoo@yoo.com").build(),
                    Member.builder().userId("김상현").age(29).email("sang@sang.com").build(),
                    Member.builder().userId("김예진").age(26).email("yj@yj.com").build(),
                    Member.builder().userId("양호준").age(27).email("ho@ho.com").build());

  @GetMapping("/members")
  public List<Member> selectAllMember() {
    return list;
  }

}
```

RestTestController

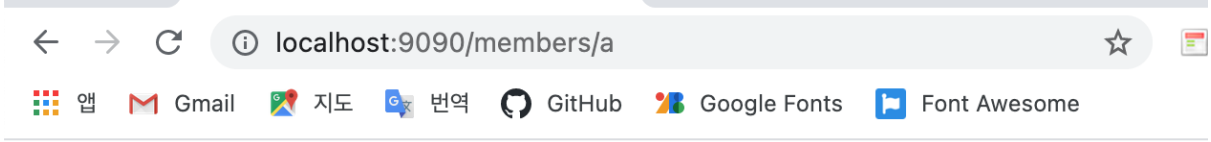
```
package com.inittest.springboot.controller;

import java.util.ArrayList;
import java.util.List;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import com.inittest.springboot.model.vo.Member;

@RestController
public class RestTestController {

  // RESTful방식(REST-api설계)은
  // 요청주소를 URI를 보고 판단할 수 있게 설계하고
  // 단위서비스로 구현하여 모듈단위로 이용할 수 있게 하는 것

  // 예를들어
  // localhost:9090/selectMemberAll 가 아니라
  // localhostL9090/members
  // 행위에 대한것은 method(get, post, put, delete)로 지정

  // method로 uri주소가 하는 행위를 지정한다.
  // uri설계시에는 그 행위에 대한 내용은 제외한다.
  // 명사로만 표현

  // method의 종류
  // GET : 조회(select)
  // POST : 입력(create)
  // PUT : 수정(update)
  // DELETE : 삭제(delete)

  // 예를들어 get .../members           -> 회원조회
  // GET :: localhost:9090/member/admin     -> 회원중 admin을 조회
  // GET :: localhost:9090/board/1      -> 1번 게시글 조회
  // POST :: localhost:9090/member        -> 회원등록 ( 회원정보가 requestBody에 있으니까 )
  // PUT :: localhost:9090/member/admin     -> admin회원을 수정
  // DELETE :: localhost:9090/member/admin  -> admin회원을 삭제

  // RESTful방식에서 서버는 데이터만 전송하는 역할을 하게 된다
  // JSON으로
  // 서버에서 session을 관리하지 않는다
  // 서버에서 클라이언트 상태를 관리하지 않는다

  private List<Member> list = List.of(Member.builder().userId("유병승").age(19).email("yoo@yoo.com").build(),
                  Member.builder().userId("김상현").age(29).email("sang@sang.com").build(),
                  Member.builder().userId("김예진").age(26).email("yj@yj.com").build(),
                  Member.builder().userId("양호준").age(27).email("ho@ho.com").build());

  @GetMapping("/members")
  public List<Member> selectAllMember() {
    return list;
  }


  @GetMapping("/members/{email}")
  public List<Member> searchEmail(@PathVariable(value="email") String email) {

    List<Member> searchlist = new ArrayList<>();
    for(Member m : list) {
```

```
      if(m.getEmail().contains(email)) {
        searchlist.add(m);
      }
    }
    return searchlist;
  }
}
```


```
[{"userId":"김예진","password":null,"age":26,"email":"yj@yj.com"}]
```


```
[{"userId":"유병승","password":null,"age":19,"email":"yoo@yoo.com"},{"userId":"김예
진","password":null,"age":26,"email":"yj@yj.com"}]
```

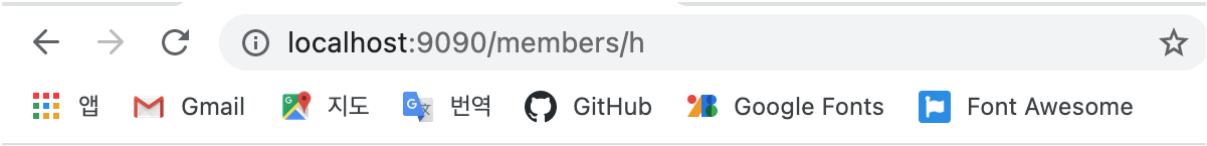




```
[{"userId":"김상현","password":null,"age":29,"email":"sang@sang.com"}]
```


webapp > jsp 페이지 생성

```
package com.inittest.springboot.controller;

import java.util.ArrayList;
import java.util.List;
import java.util.stream.Collectors;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import com.inittest.springboot.model.vo.Member;

@RestController
public class RestTestController {

  private List<Member> list = List.of(Member.builder().userId("유병승").age(19).email("yoo@yoo.com").build(),
                  Member.builder().userId("김상현").age(29).email("sang@sang.com").build(),
                  Member.builder().userId("김예진").age(26).email("yj@yj.com").build(),
                  Member.builder().userId("양호준").age(27).email("ho@ho.com").build());

  @GetMapping("/members")
```

```
  public List<Member> selectAllMember() {
    return list;
  }


  @GetMapping("/members/{email}")
  public List<Member> searchEmail(@PathVariable(value="email") String email) {

//    List<Member> searchlist = new ArrayList<>();
//    for(Member m : list) {
//      if(m.getEmail().contains(email)) {
//        searchlist.add(m);
//      }
//    }

    // stream 객체 사용
    // 람다식, 함수용 인터페이스 사용
    List<Member> searchlist = list.stream().filter(m -> m.getEmail().contains(email)).collect(Collectors.toList());


    return searchlist;
  }
}
```



[{"userId":"양호준","password":null,"age":27,"email":"ho@ho.com"}]