Spring Boot Welcome Page (static)

File path: /resources/static/index.html

cf. view file path: /resource/template

Template Engine – Thymeleaf Engine

: Template + Input data -> output rendering

Client request ->

Spring Boot 1)Spring Container 내의 Controller 호출

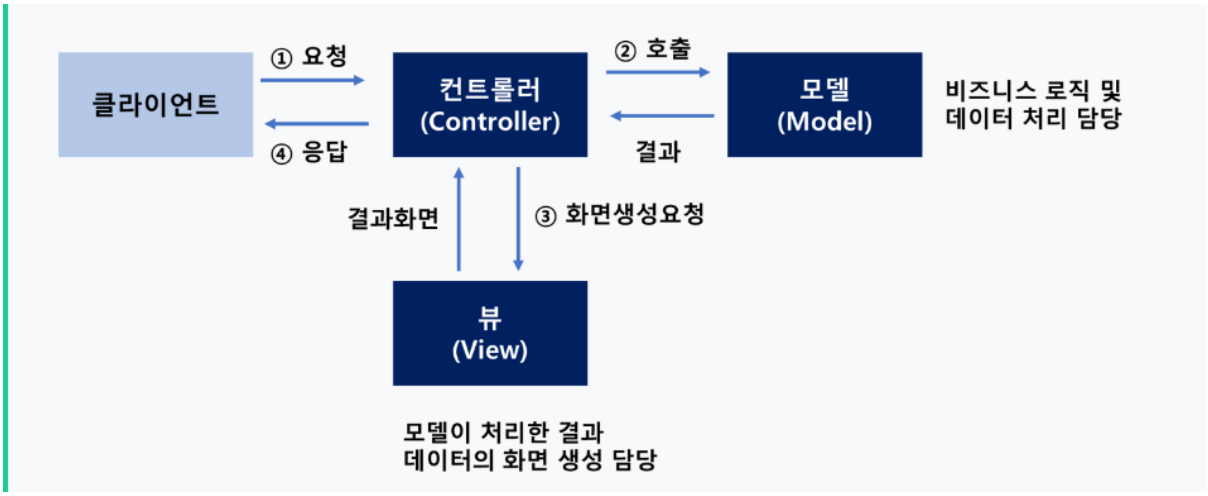
2)viewReslover를 통해 Controller가

resources:templates/ +{ViewName}+ .html Mapping

Cf. 참고 사항: Controller의 Get Post Mapping URL이 같을 때,

https://u-it.tistory.com/entry/%EC%BB%A8%ED%8A%B8%EB%A1%A4%EB%9F%AC%EC%9D%98-GETPOST-Mapping-%EB%A9%94%EC%84%9C%EB%93%9C%EC%97%90-%EB%93%B1%EB%A1%9D%EB%90%9C-uri%EA%B0%80-%EA%B0%99%EB%8B%A4%EB%A9%B4-Form-action-uri%EC%83%9D%EB%9E%B5-%EA%B0%80%EB%8A%A5-%EC%8B%9C%EC%9E%91%EA%B3%BC-%EB%81%9D%EC%9D%B4-%EA%B0%99%EC%9D%8C

MVC Pattern (Model-View-Controller): Business Logic (Domain)과 UI를 분리 (관심사의 분리)



1

https://velog.io/@se_bb/Spring-Framework-7%EA%B0%95-Spring-MVC-%EC%86%8C%EA%B0%9C

1. Model: data를 처리, 가공하는 component

   Business logic을 처리 후, Controller와 View에 전달

   @GetMapping @PostMapping Annotation

1

1) addAttribute method

parameter - attributename, attributevalue

return "(viewName)" //rendering by viewResolver

/resources/templates/$(ViewName)+`.html`

2. View: Client에 보여지는 부분(User Interface)
3. Controller: Model과 view사이의 **interface**

-Component Scan & Auto Dependency Injection

**객체를 일일이 생성하지말고 spring container 에 등록하여 쓴다. – Spring Bean

Cf. spring bean – spring IoC 가 관리하는 객체 (Singleton pattern)

//java bean 과 용어 구분

06:56
수정 삭제

@Autowired annotation spring bean 에 등록되어 있는 객체를 가져다가 넣어준다. -> DI(Dependency Injection): 의존성 주입

09:32
수정 삭제

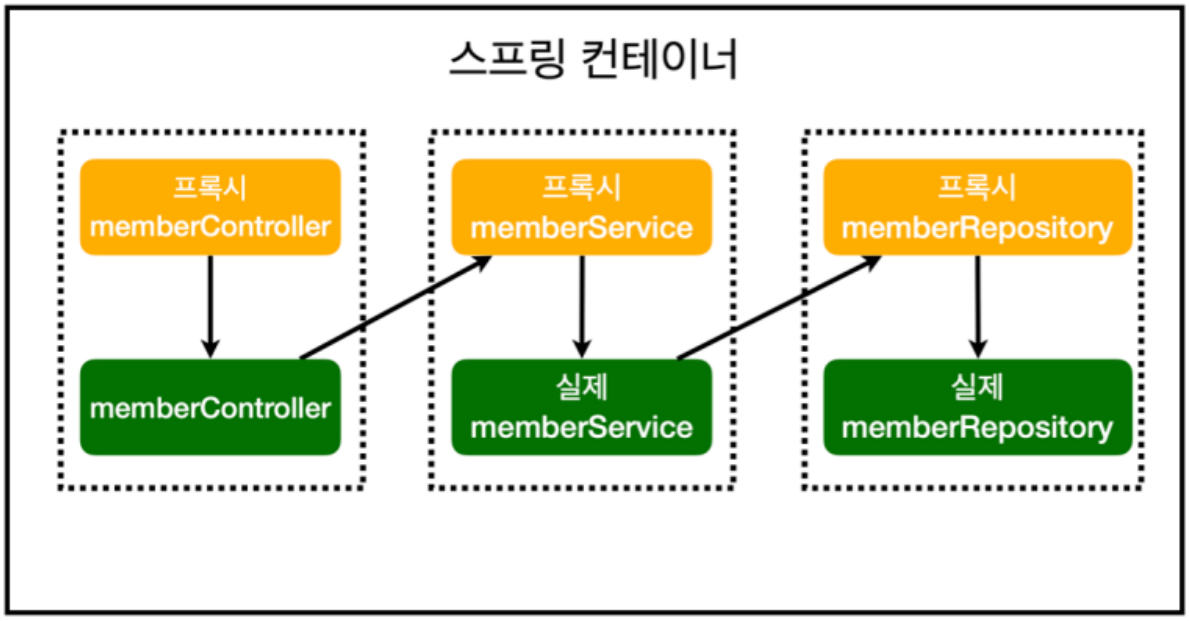
@Controller @Service @Repository-> Component scanning 방식

:component 와 관련된 annotation 이 있으면 spring 이 객체로 생성하여 spring container 에 등록

```java
package ac.kr.dankook.hellospring.controller;


import ac.kr.dankook.hellospring.domain.Member;
import ac.kr.dankook.hellospring.service.MemberService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class MemberController {

    private final MemberService memberService;


    //spring container 내 객체와 연결해주는 annotaion
    @Autowired
    public MemberController(MemberService memberService) {
        this.memberService = memberService;
    }

    @GetMapping("/members/new")
    public String createForm() {
        return "members/createMemberForm";
    }



    @PostMapping("/members/new")
    public String create(MemberForm form) {
        Member member = new Member();
        member.setName(form.getName()); //form.getName client 로 부터 input 된
name 의 value 를 return

        memberService.join(member);

        return "redirect:/";

    }
}
@GetMapping("/members")
public String list(Model model){
    List<Member> members = memberService.findMembers();
    model.addAttribute("members", members);
    return "members/memberList";
```

AOP: 핵심 비즈니스 로직(관심사)와 공통의 관심사 분리

AOP 적용 후 전체 그림



//Controller가 Proxy를 호출하는 것으로 변경됨.