



It's Your Life

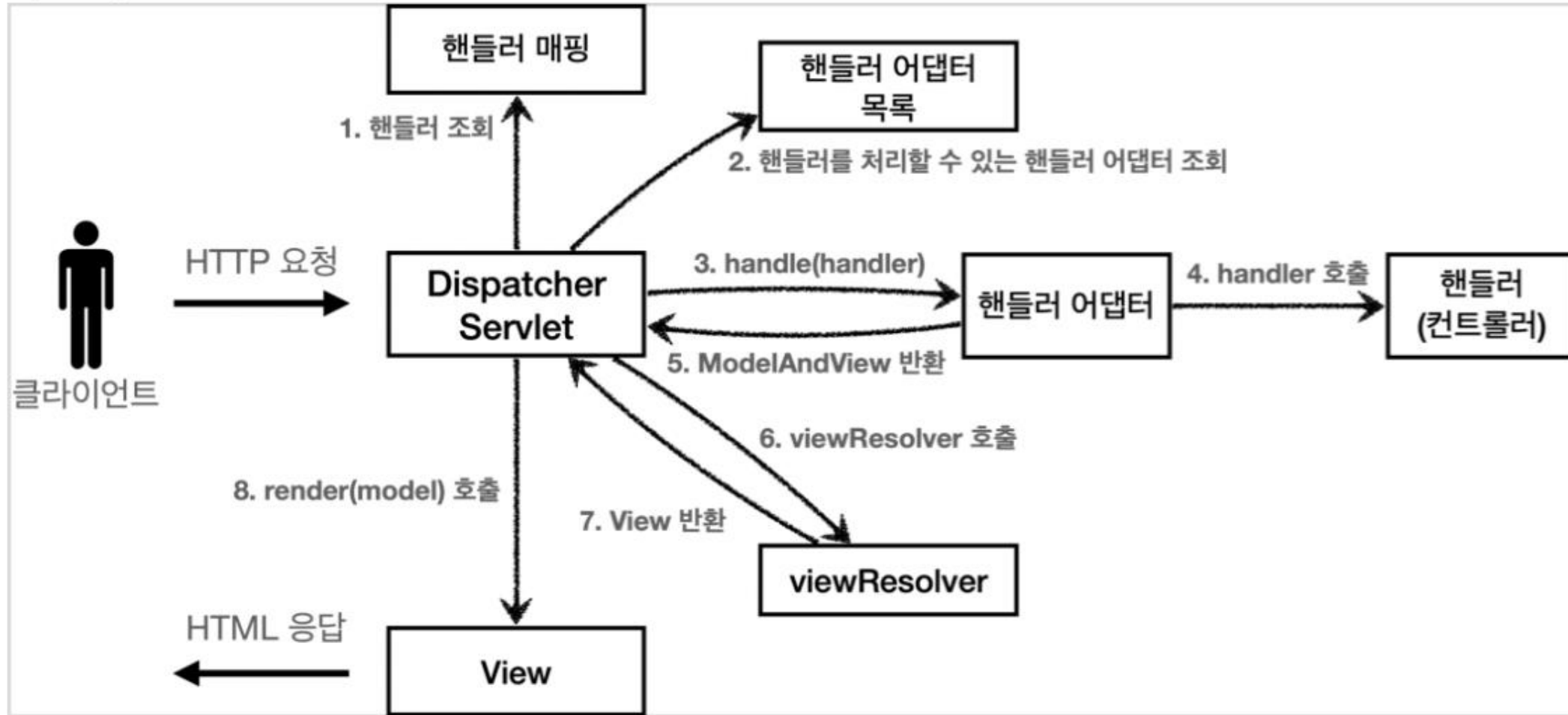
with



Spring MVC



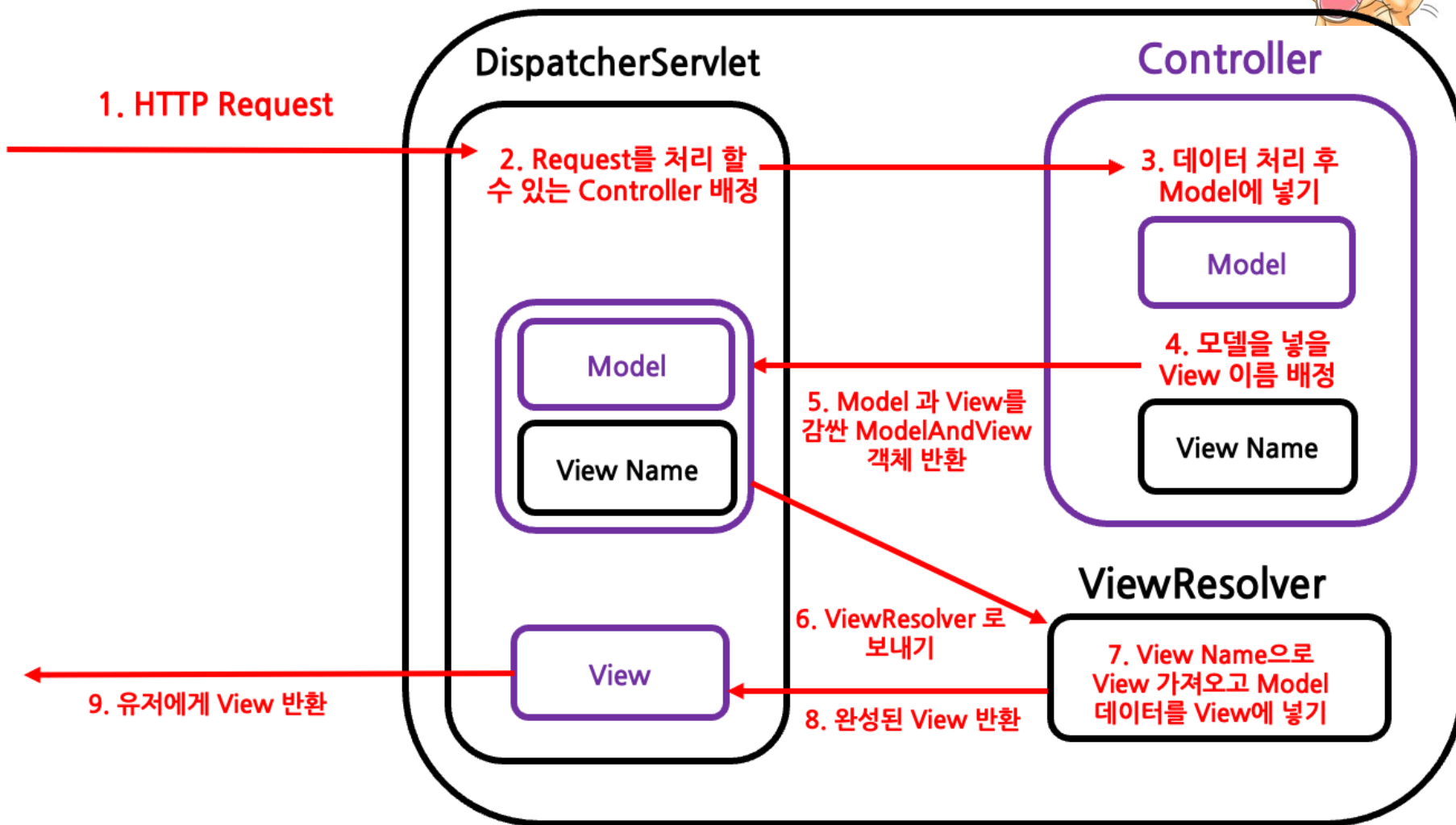
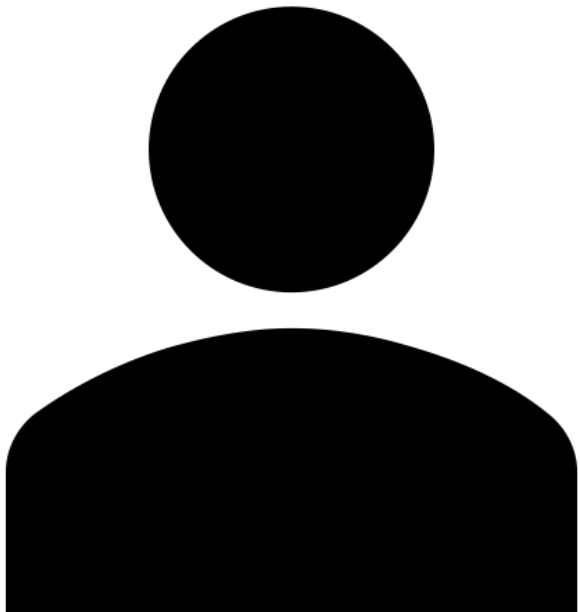
SpringMVC 구조

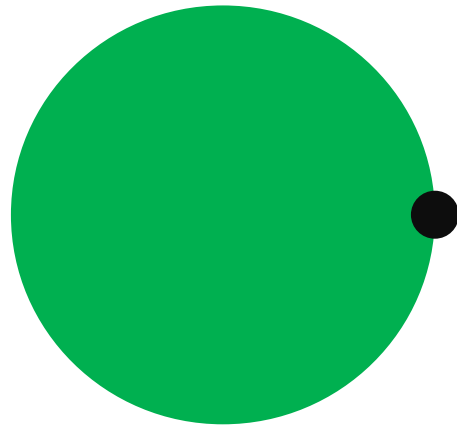


Server

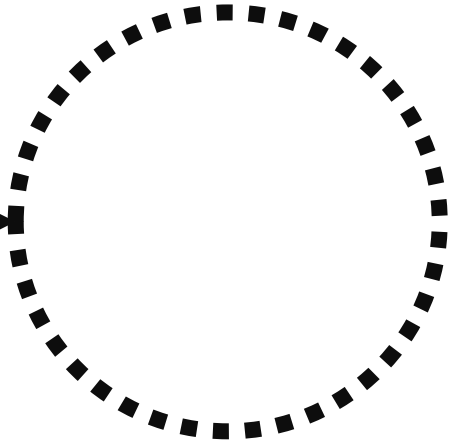


유저

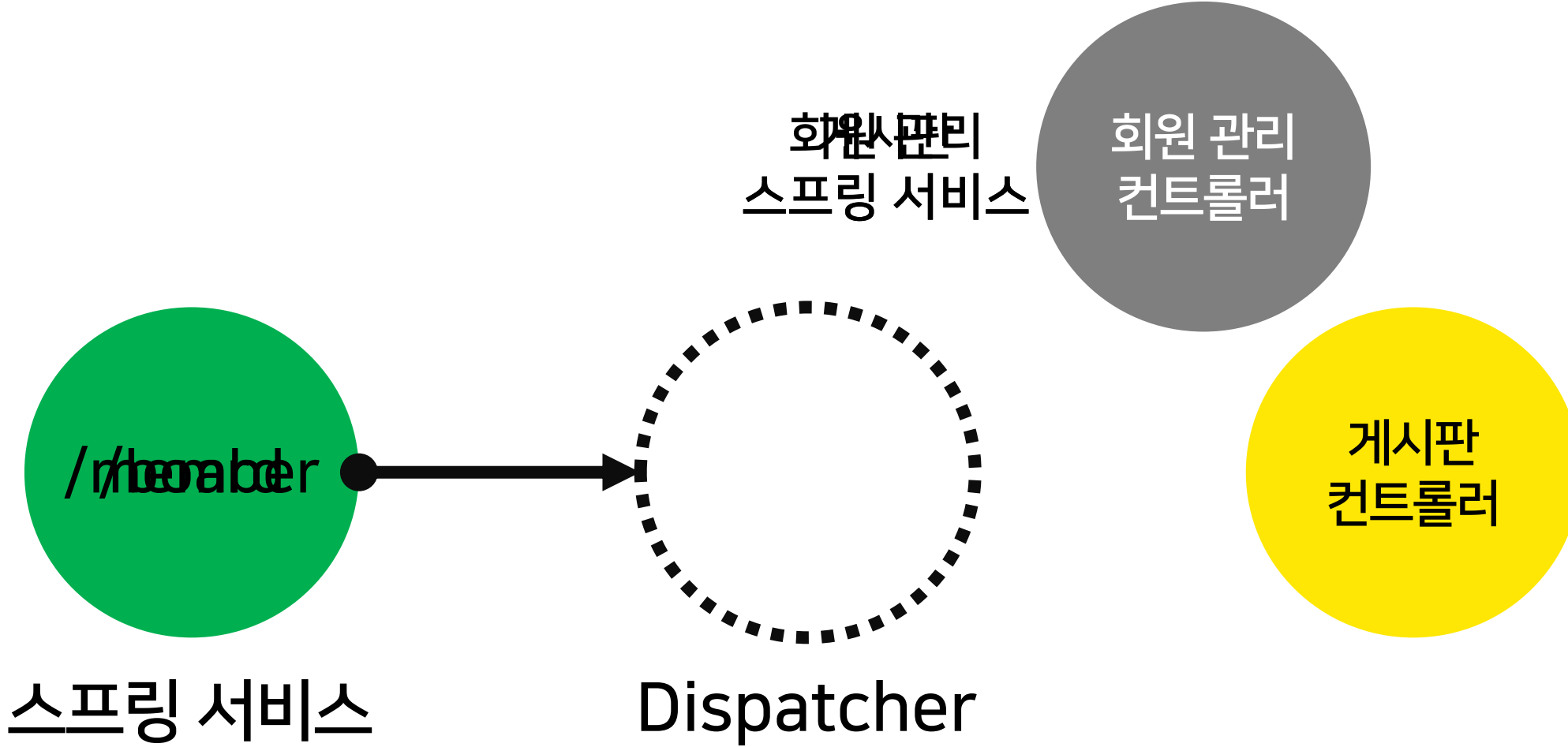




스프링 서비스



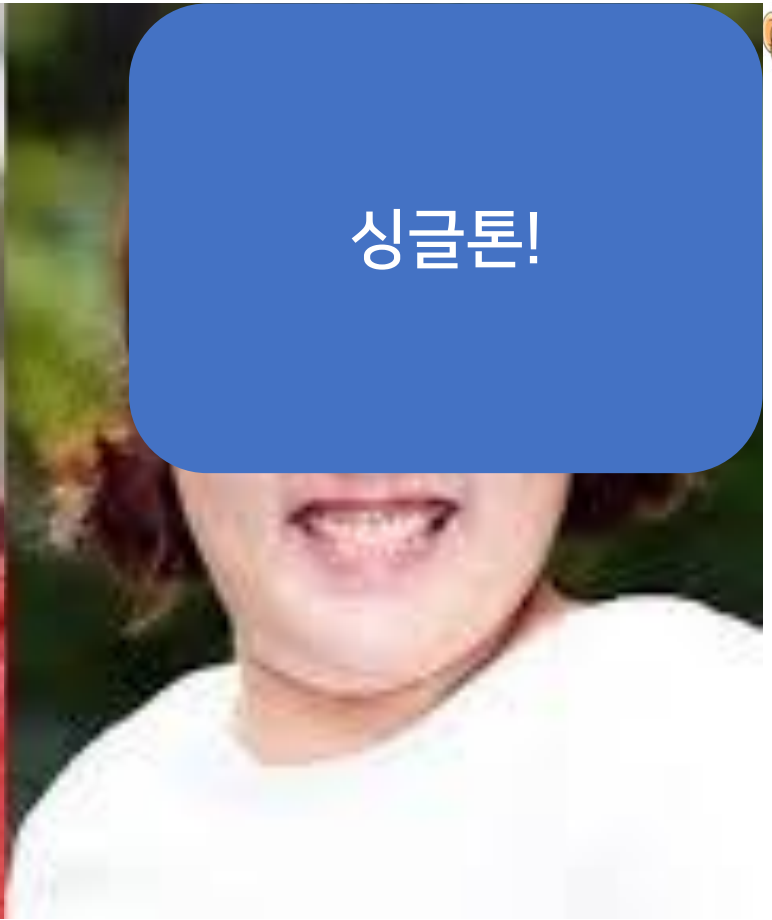
Dispatcher





스프링의 편의 기능

활용하기!



싱글톤!





```
@Controller  kdtTetz +1
```

```
@Slf4j
```

```
public class MemberShowControllerV1 {
```

```
    private MemberDtoListV1 memberList = MemberDtoListV1.get
```

```
@GetMapping("/member/show")  kdtTetz +1
```

```
public String process(HttpServletRequest request, HttpSe
```

```
    log.info("=====> 회원 조회 페이지 호출, /mem
```

```
    request.setAttribute("memberList", memberList.getList());
```

```
    return "member-show";
```

```
}
```

```
}
```

SCOPE

The word "SCOPE" is written in a large, bold, dark blue font. A magnifying glass with a dark blue handle and frame is positioned over the letter 'O'. Inside the lens of the magnifying glass, there is a small illustration of a factory with two smokestacks emitting blue smoke.



10:36



Spring Bean





스프링 빈 등록을 위한
@Component 어노테이션 붙이기!

```
@Component
```

```
public class MemberDtoListV2 {
```

```
    private List<MemberDto> memberDtoList;
```

```
    public MemberDtoListV2() {
```

```
        this.memberDtoList = new ArrayList<>();
```

```
        this.addList(id: "tetz", name: "이효석");
```

```
        this.addList(id: "siwan", name: "김시완");// List 초기화
```

```
    }
```

```
    public void addList(String id, String name) { memberDtoList.add(new MemberD
```

```
    public List<MemberDto> getList() { return memberDtoList; }
```

```
}
```

스프링 빈에 등록되어 사용할
멤버 변수!



Spring Bean

설정! (중요!!!!)



@Component



너 왜 나 등록 안함!?

모르는데 어떻게 가요!



기본 설정을 담당하는
RootConfig 로 이동!

```
2  
3 > import ...  
7 @Configuration kdtTetz  
8 @ComponentScan(basePackages = "org.example")  
9 public class RootConfig {  
10  
11 }
```

Component(= Bean)를 어디서 찾을지
설정하는 부분입니다! (중요!!!)



그럼 어떻게

쓰나요!?



회원 정보 관리 데이터를 저장할
멤버 변수를 선언!

생성자에 @Autowired 어노테이션을
이용하여 Bean 에 등록 된
MemberDtoListV2 를 가져와서 바로 등록!

```
@Controller
```

```
@Slf4j
```

```
public class MemberShowControllerV2 {  
    private final MemberDtoListV2 memberDtoList;
```

```
@Autowired
```

```
public MemberShowControllerV2(MemberDtoListV2 memberDtoList) {  
    this.memberDtoList = memberDtoList;  
}
```


```
@GetMapping("/member/v2/show")
```

```
public String process(HttpServletRequest request, Model model) {  
    log.info("=====> 회원 조회 페이지 호출, " + request.getRequestURI());  
  
    model.addAttribute("memberList", memberDtoList.getList());  
    return "member-show2";  
}
```

```
}
```


@Autowired



@Controller  kdtTetz

@Slf4j

```
public class MemberShowControllerV2 {  
    private final MemberDtoListV2 memberDtoList; 2 usages
```

@Autowired  kdtTetz

```
public MemberShowControllerV2(MemberDtoListV2 memberDtoList) {  
    this.memberDtoList = memberDtoList;  
}
```

@Autowired 가 붙으면
해당 생성자의 매개 변수의 타입을 봅니다

Component(=Bean) 목록

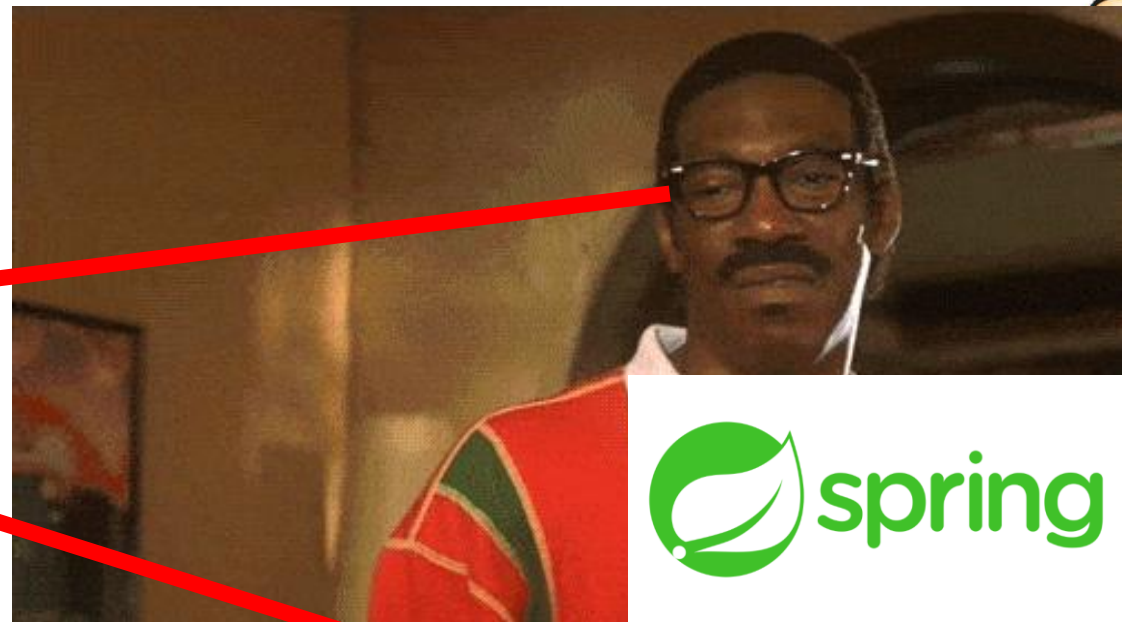
MemberDtoListV2 memberDtoList

...

...

...

...



```
@Autowired
public MemberShowControllerV2(MemberDtoListV2 memberDtoList) {
    this.memberDtoList = memberDtoList;
}
```

@Autowired 가 붙은 생성자의 매개변수
타입을 Bean 목록에서 찾아서 있으면
전달해 줍니다!!



지금 시작합니다

데이터 전달을 위한



Model 객체

PAGE

jsp 페이지에서만 생존

REQUEST

요청에 대한 응답이 끝날 때까지 생존

SESSION

세션이 유지되는 동안 생존

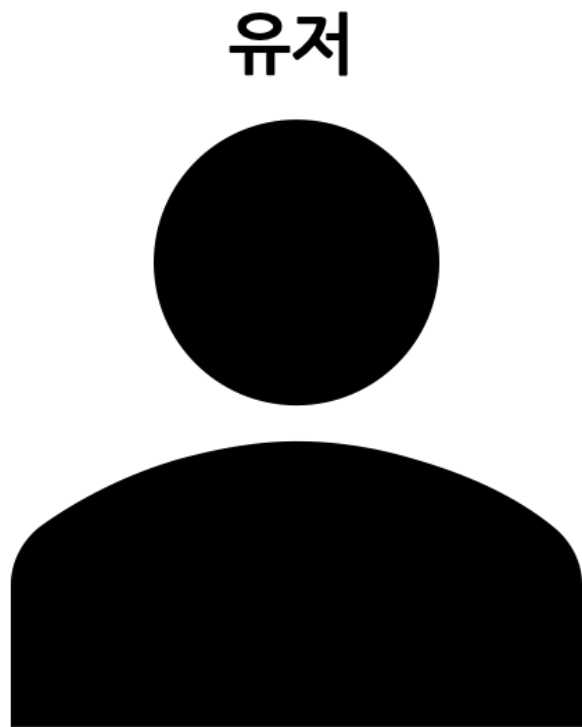
APPLI
CATION

서버가 꺼질 때까지 생존

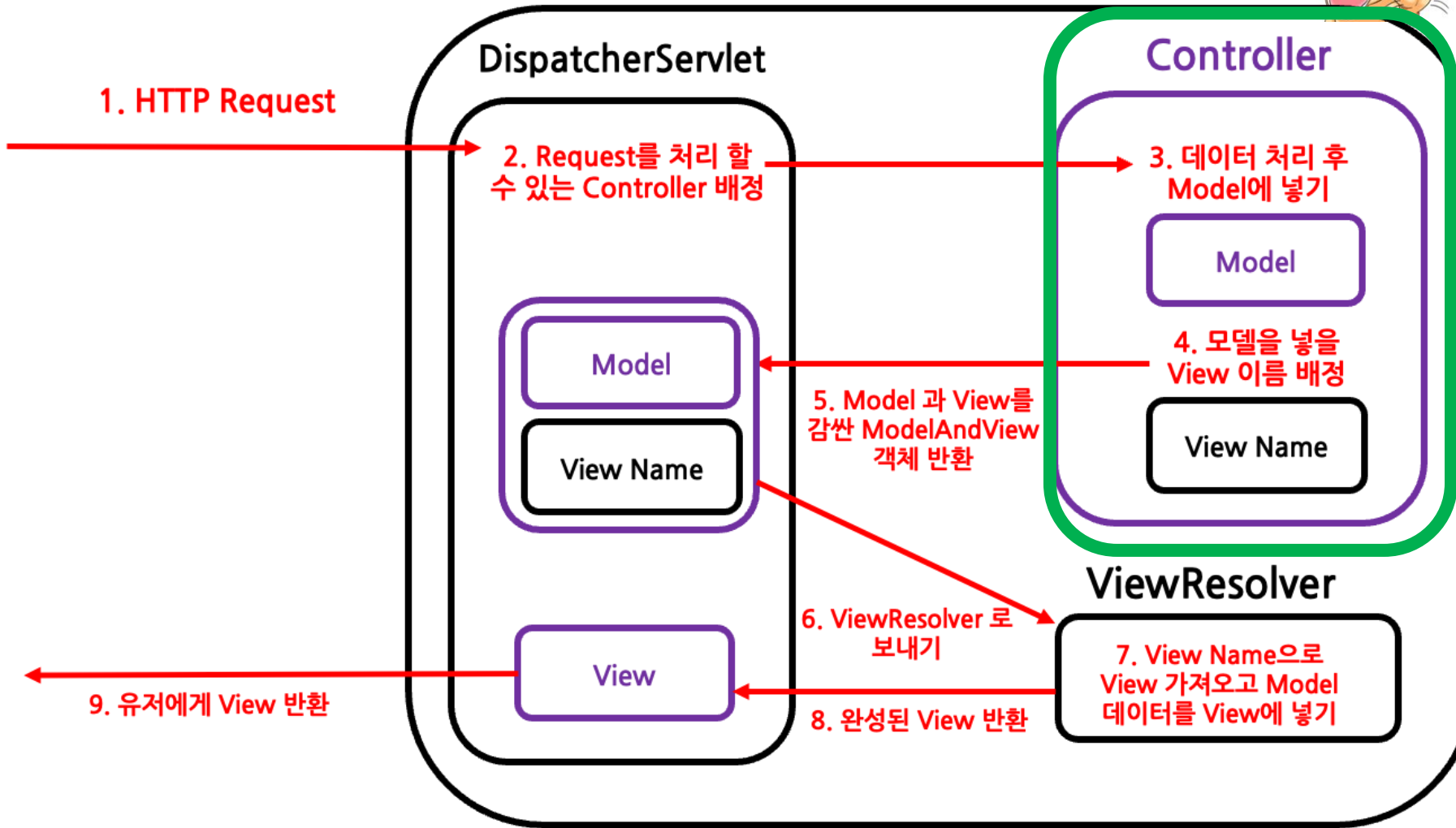


Spring 은 컨트롤러와 View 의
데이터 전달을 위한 도구로
Model 을 사용합니다!





Server





대뜸

모델 적용하기!

model 은 매개변수에
Model 타입으로 전달하면
Spring 이 자동으로 주입 해줍니다!

```
@GetMapping("/member/v2/show")
public String process(HttpServletRequest request, Model model) {
    log.info("=====> 회원 조회 페이지 호출, " + request.getRequestURI());

    model.addAttribute(attributeName: "memberList", memberDtoList.getList());
    return "member-show2";
}
```

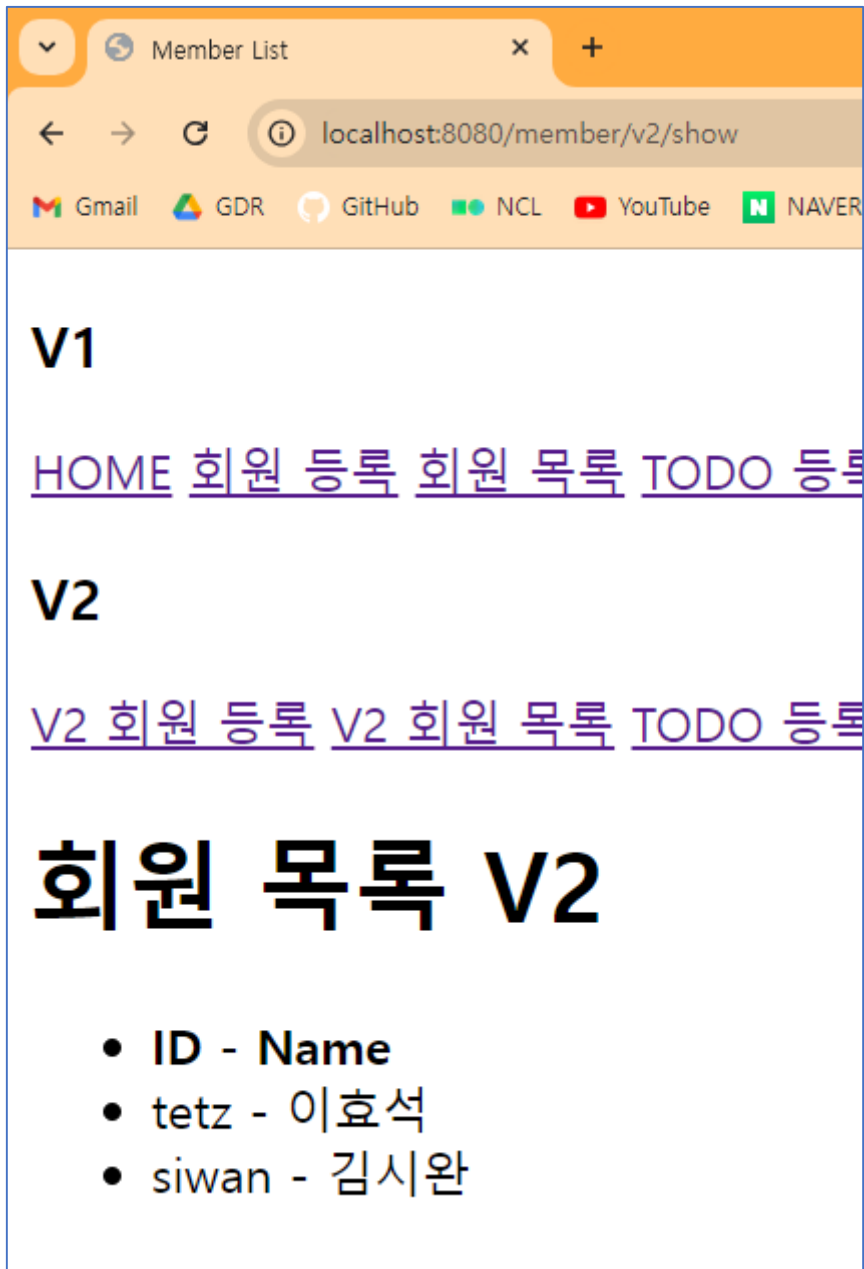
(주의) model 은 setAttribute 가 아닌
addAttribute 를 사용합니다!!

메서드에 매개 변수는
해당 작업에서 필요한 객체만 가져와서 씁니다
request 객체로 주소 값을 받기 위해서 포함

```
@GetMapping("/member/v2/show")
public String process(HttpServletRequest request, Model model) {
    log.info("=====> 회원 조회 페이지 호출, " + request.getRequestURI());

    model.addAttribute("memberList", memberDtoList.getList());
    return "member-show2";
}
```

주소 값을 직접 기입하는 것이 아니라
요청 주소에서 바로 가져 오도록 수정!



문제 없이 데이터가 전달 되는 것을
확인 할 수 있습니다!



모델의 장점



- 일반 객체로 구성된 Scope(Request, Session, Application) 이 아니라 Map 을 사용하여 데이터 추가 및 삭제가 용이합니다
- 더 빠른 속도를 제공하여 효율이 높습니다!
- 편하게 가져다 쓰면 되기 때문에 일단 쓰기 편합니다!!



Model Data

실습, Todo 에 Model 적용!



- Todo V2 버전에 model 을 적용하여 model 로 데이터를 전달 할 수 있도록 수정해 주세요!
- 여러분들이 수정해야할 컨트롤러는 TodoShowControllerV2, TodoSaveControllerV2 2개 입니다!



@RequestMapping

Client

Request



GET 방식
POST 방식
PUT 방식
DELETE 방식

Tomcat
Server

Response



Web Browser





그렇다면 Request 와
GET / POST / PUT / DELETE 방식 중에서
어떤 것이 더 상위 개념일까요!?



그란데 말입니다

Request



GET
Method

Post
Method

PUT
Method

DELETE
Method



그란데 말입니다

아래의 코드는 어떤 메서드를 받을 수 있을까요?

```
@RequestMapping(🌐"/member/v2/form/save")  👤 kdtTetz  
public String process(HttpServletRequest request, Http  
    log.info("=====> 회원 추가 Request 호출,
```



그란데 말입니다

그렇다면

아래의 코드는 어떤 메서드를 받을 수 있을까요?

```
@GetMapping(🌐✓"/member/v2/form")  👤 kdtTetz  
public String home(HttpServletRequest request,  
    log.info("=====> 회원 추가 페이지 3
```



그란데 말입니다

그렇다면2

아래의 코드는 어떤 메서드를 받을 수 있을까요?

```
@PostMapping(🌐"/member/v2/form")  👤 kdtTetz  
public String home(HttpServletRequest request,  
    log.info("=====> 회원 추가 페이지
```

```
@PutMapping(🌐"/member/v2/form")  👤 kdtTetz  
public String home(HttpServletRequest request,  
    log.info("=====> 회원 추가 페이지
```

```
@DeleteMapping(🌐"/member/v2/form")  👤 kdtTetz  
public String home(HttpServletRequest request,  
    log.info("=====> 회원 추가 페이지
```



그란데 말입니다

그렇다면3

위와 아래 중 어떤 것이 더 좋은 컨트롤러 일까요?

```
@RequestMapping(🌐"/member/v2/form/save")  👤 kdtTetz  
public String process(HttpServletRequest request, Http  
    log.info("=====> 회원 추가 Request 호출,
```

```
@GetMapping(🌐"/member/v2/form")  👤 kdtTetz  
public String home(HttpServletRequest request,  
    log.info("=====> 회원 추가 페이지 3
```


리퀘스트 매핑의 경우!



- 더 구체적인 메서드와 경로를 매핑 시키는 것이 좋습니다!!
- 즉, RequestMapping 을 쓰는 것 보다는 Get/Post/Put/Delete Mapping 으로 구체화를 시키는 편이 좋습니다!



두 번째 토론

MBC



명쾌한 정리!

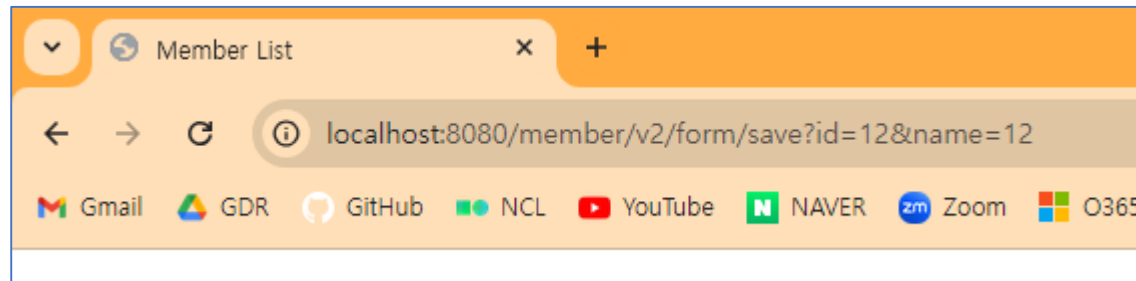


@RequestParam



그란데 말입니다

여러분은 프론트에서 전달하는 데이터를
어떤 방식으로 받고 있었나요!?



```
@GetMapping("/member/v2/form/save") kdtTetz *
public String process(HttpServletRequest request, Model model) {
    log.info("=====> 회원 추가 Request 호출, /member/form/save");

    String id = request.getParameter("id");
    String name = request.getParameter("name");

    memberList.addList(id, name);

    model.addAttribute("memberList", memberList.getList());
    return "member-show2";
}
```



아래의 코드 중에서 옛날 개발자들이 귀찮음을 느꼈을 부분을 고르시오



```
@GetMapping(🌐✓"/member/v2/form/save")  👤 kdtTetz *
public String process(HttpServletRequest request, Model model) {
    log.info("=====> 회원 추가 Request 호출, /member/form/save");

    String id = request.getParameter(s: "id");
    String name = request.getParameter(s: "name");

    memberList.addList(id, name);

    model.addAttribute(attributeName: "memberList", memberList.getList());
    return "member-show2";
}
```



```
@GetMapping(🌐✓"/member/v2/form/save")  👤 kdtTetz*  
public String process(HttpServletRequest request, Model model) {  
    log.info("=====> 회원 추가 Request 호출, /member/form/save");  
  
    String id = request.getParameter(s: "id");  
    String name = request.getParameter(s: "name");  
  
    memberList.addList(id, name);  
  
    model.addAttribute(attributeName: "memberList", memberList.getList());  
    return "member-show2";  
}
```

요거 빼고는 사실
다 바꿨습니다 😊

아 파라미터 요청에서
데이터 빼내기 귀찮다.....



@RequestParam 등장



@GetMapping(🌐"/member/v2/form/save") kdtTetz *

```
public String process(  
    @RequestParam("id") String id,  
    @RequestParam("name") String name,  
    Model model
```

```
) {
```

```
    log.info("=====> 회원 추가 Request 호출, /member/form/save");
```

```
    memberList.addList(id, name);
```

```
    model.addAttribute(attributeName: "memberList", memberList.getList());
```

```
    return "member-show2";
```

```
}
```

이제는 Request 를 받아서
거기서 뽑는게 아니라
매개 변수에서 바로 뽑아서 씁니다!



```
@RequestParam("id") String id,
```

@RequestParam 지시자 사용

쿼리스트링으로 전달 된
파라미터의 이름 전달

전달 받은 데이터의 타입과
이름을 지정!



```
@GetMapping("/member/v2/form/save")  kdtTetz *
public String process(
    @RequestParam("id") String id,
    @RequestParam("name") String name,
    Model model
) {
    log.info("=====> 회원 추가 Request 호출, /member/form/save");

    memberList.addList(id, name);

    model.addAttribute(attributeName: "memberList", memberList.getList());
    return "member-show2";
}
```

매개 변수에서 이미 뽑은
id, name 을 직접 사용



CHD
F1 코리아 결승 D-1

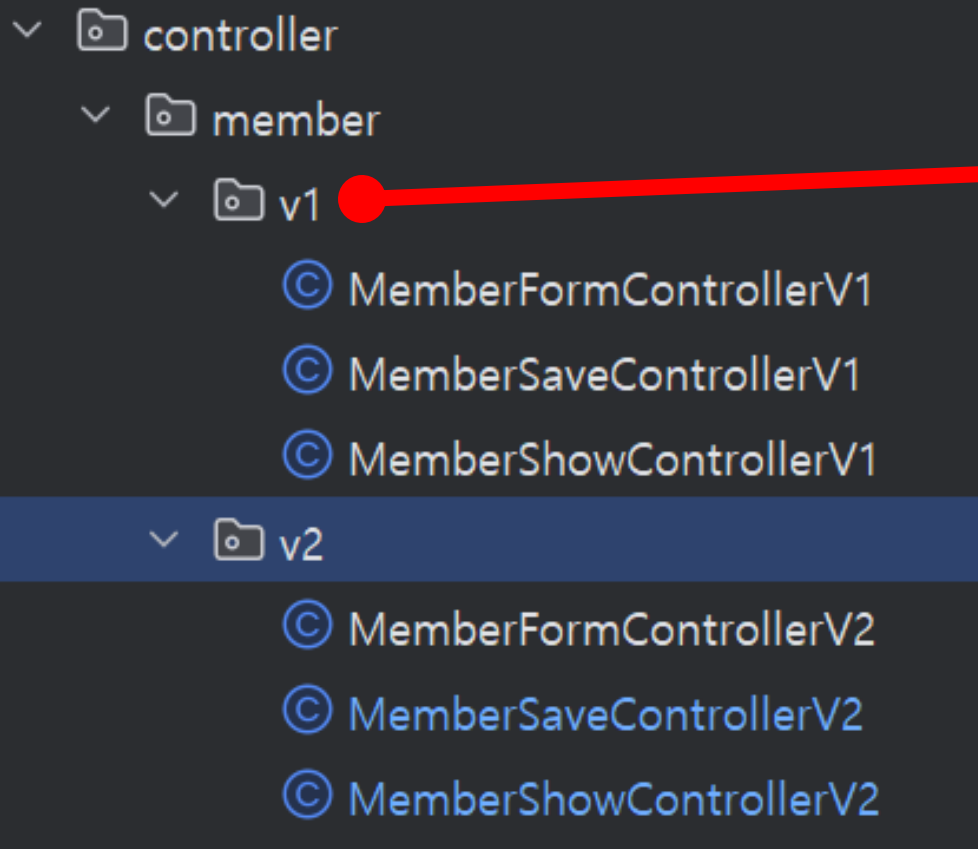


인생은 실전이야 동훈아...







실무 버전의 컨트롤러로 변경





일단 정신 건강을 위해서
복잡한 컨트롤러를 패키지로 나누어
정리 합시다!



- ▼  controller
 - ▼  member
 - ▼  v1
 - © MemberFormControllerV1
 - © MemberSaveControllerV1
 - © MemberShowControllerV1
 - ▼  v2
 - © MemberFormControllerV2
 - © MemberSaveControllerV2
 - © MemberShowControllerV2





컨트롤러

통합하기!






▼ v2


- © MemberFormControllerV2
- © MemberSaveControllerV2
- © MemberShowControllerV2
- © MemberControllerV3

3개로 나누어진 회원 관리 컨트롤러를
하나의 MemberControllerV3 로
통합해 봅시다!

@Controller  kdtTetz

@Slf4j

```
public class MemberControllerV3 {  
    private final MemberDtoListV2 memberDtoList; 4
```

@Autowired  kdtTetz

```
public MemberControllerV3(MemberDtoListV2 memberDtoList) {  
    this.memberDtoList = memberDtoList;  
}
```

기본적으로 필요한 것들을
V2 의 코드로 부터 가져 옵니다!





```
@Controller  kdtTetz *  
@Slf4j  
public class MemberControllerV3 {  
    private final MemberDtoListV2 memberDtoList; 4 usages
```

```
@Autowired  kdtTetz  
public MemberControllerV3(MemberDtoListV2 memberDtoList) {  
    this.memberDtoList = memberDtoList;  
}
```

R Rename usages

```
@GetMapping(" /member/v3/show") kdtTetz *  
public String process(Model model) {  
    log.info("=====> 회원 조회 페이지 호출, /member/list");  
  
    model.addAttribute(attributeName: "memberList", memberDtoList.getList());  
    return "member-show3";  
}
```

MemberShowControllerV2 의
메서드를 가지고 온 다음
주소 값과 JSP 파일명을 변경해 줍니다!



그란데 말입니다

그란데 말입니다

V3 버전의 회원 관리 요청은
모두 어떤 주소로 시작을 하나요!?

R Rename usages

```
@GetMapping(🌐 "/member/v3/show")
```

```
@GetMapping(🌐 "/member/v3/form")
```

```
@PostMapping(🌐 "/member/v3/form/save")
```





해로운
중독이다

개발자



RequestMapping

을 이용하여 통합 처리

```
@Controller kdtTetz
```

```
@Slf4j
```

```
@RequestMapping("/member/v3")
```

```
public class MemberControllerV3 {
```

```
    private final MemberDtoListV2 memberDtoList; 4 usages
```

```
@Autowired kdtTetz
```

```
public MemberControllerV3(MemberDtoListV2 memberDtoList) {
```

```
    this.memberDtoList = memberDtoList;
```

```
}
```

```
@GetMapping("/show") kdtTetz
```

```
public String memberList(Model model) {
```

```
    log.info("=====> 회원 조회 페이지 호출, /member/list");
```

MemberControllerV3 클래스 자체에
중복되는 주소(=컨텍스트)인 /member/v3 를
매핑시킵니다!

실제 메서드는 중복 되는 부분을 제외한
실제 요청 주소만 남깁니다!



```
@Controller  👤 kdtTetz
@Slf4j
@RequestMapping(🌐"/member/v3")
public class MemberControllerV3 {
    private final MemberDtoListV2 memberDtoList; 4 usages

    @Autowired  👤 kdtTetz
    public MemberControllerV3(MemberDtoListV2 memberDtoList) {
        this.memberDtoList = memberDtoList;
    }

    @GetMapping(🌐"/show")  👤 kdtTetz
    public String memberList(Model model) {
        log.info("=====> 회원 조회 페이지 호출, /member/list");
    }
}
```

메서드 명이 process 로 중복되면
에러가 발생하므로 각각의 목적에 맞는
이름으로 변경하기!



```
@GetMapping(🌐"/form")  👤 kdtTetz
public String memberForm(HttpServletRequest request, HttpServletResponse response) {
    log.info("=====> 회원 추가 페이지 호출, /member/register");

    return "member-form3";
}
```

회원 추가 페이지 컨트롤러도 복붙 후

1. 컨텍스트를 뺀 요청 주소만 매핑
2. 전달하는 View 파일명 변경
3. 적절한 메서드 이름으로 변경

```
@PostMapping("/form/save")
public String memberSave(@RequestParam("id") String id, @RequestParam("name") String name, Model model) {
    log.info("=====> 회원 추가 Request 호출, /member/form3/save2");

    memberDtoList.addList(id, name);
    model.addAttribute("memberList", memberDtoList.getList());
    return "member-show3";
}
```

회원 추가 기능 컨트롤러도 복붙 후

1. 컨텍스트를 뺀 요청 주소만 매핑
2. 파라미터는 @RequestParam 을 이용해서 받기
3. 전달하는 View 파일명 변경
4. 적절한 메서드 이름으로 변경



Request

/member/v3/show

GET 방식

Tomcat Server

/member/v3

MemberControllerV3

GET /show

memberShow() {}

GET /form

memberForm() {}

POST /form/save

memberSave() {}



Request

/member/v3/form

GET 방식

/member/v3

Tomcat Server

MemberControllerV3

GET /form

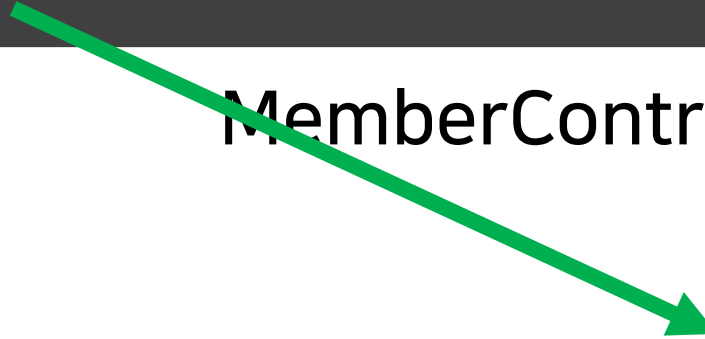
memberForm() {}

GET /show

memberShow() {}

POST /form/save

memberSave() {}





Request

/member/v3/form/save

POST 방식

Tomcat Server

/member/v3

MemberControllerV3

GET /show

memberShow() {}

GET /form

memberForm() {}

POST /form/save

memberSave() {}



통합 컨트롤러

기능 확인!



```
<header>
```

```
<a href="/todo/form">TODO 등록</a>
```

```
<a href="/todo/show">TODO 보기</a>
```

```
<br>
```

```
<h3>V2</h3>
```

```
<a href="/member/v2/form">회원 등록 V2</a>
```

```
<a href="/member/v2/show">회원 목록 V2</a>
```

```
<a href="/todo/v2/form">TODO 등록 V2</a>
```

```
<a href="/todo/v2/show">TODO 보기 V2</a>
```

```
<br>
```

```
<h3>V3</h3>
```

```
<a href="/member/v3/form">회원 등록 V3</a>
```

```
<a href="/member/v3/show">회원 목록 V3</a>
```

```
<a href="/todo/v3/form">TODO 등록 V3</a>
```

```
<a href="/todo/v3/show">TODO 보기 V3</a>
```

```
</header>
```

header 에 V3 기능 확인을 위해
각각의 링크 추가!



member-show3.jsp 파일 추가

구분을 위해 제목에 V3 추가!

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" pr
<%@ page contentType="text/html; charset=UTF-8" langua
<!DOCTYPE html>
<html>
<head>
    <title>Member List</title>
</head>
<body>
<%@ include file="header.jsp"%>
<h1>회원 목록 V3</h1>
<ul>
    <li><b>ID - Name</b></li>
    <c:forEach var="member" items="${memberList}">
        <li>${member.id} - ${member.name}</li>
    </c:forEach>
</ul>
</body>
</html>
```



member-form3.jsp 파일 추가

구분을 위해 제목에 V3 추가!

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
    <title>Member Register</title>
</head>
<body>
<%@ include file="header.jsp"%>
<h1>회원 추가 V3</h1>
<form method="post" action="/member/v3/form/save">
    <label for="id">아이디 :</label>
    <input type="text" id="id" name="id" required>
    <br>
    <label for="name">이름 :</label>
    <input type="password" id="name" name="name" required>
    <br>
    <button type="submit">회원 추가</button>
</form>
```

Member Register

localhost:8080/member/v3/form

Gmail GDR GitHub NCL YouTube NAVER

V1

[HOME](#) [회원 등록](#) [회원 목록](#) [TODO 등록](#)

V2

[회원 등록 V2](#) [회원 목록 V2](#) [TODO 등록](#)

V3

[회원 등록 V3](#) [회원 목록 V3](#) [TODO 등록](#)

회원 추가 V3

아이디 :

이름 :

회원 추가

Member List

localhost:8080/member/v3/show

Gmail GDR GitHub NCL YouTube

V1

[HOME](#) [회원 등록](#) [회원 목록](#) [TODO](#)

V2

[회원 등록 V2](#) [회원 목록 V2](#) [TODO](#)

V3

[회원 등록 V3](#) [회원 목록 V3](#) [TODO](#)

회원 목록 V3

- ID - Name
- tetz - 이효석
- siwan - 김시완
- 12 - 12



실습, Todo 컨트롤러 통합하기



- Todo 컨트롤러 TodoShowControllerV2 / TodoFormControllerV2 / TodoSaveControllerV3 3개를 하나의 TodoControllerV3 로 통합해 주세요
- @RequestMapping 을 사용하여 중복 주소는 클래스에 매핑해 주세요
- 각각의 메서드의 이름도 기능에 맞게 잘 지어 주세요
- 파라미터는 @RequestParam 어노테이션으로 받아서 처리해 주세요