



It's Your Life

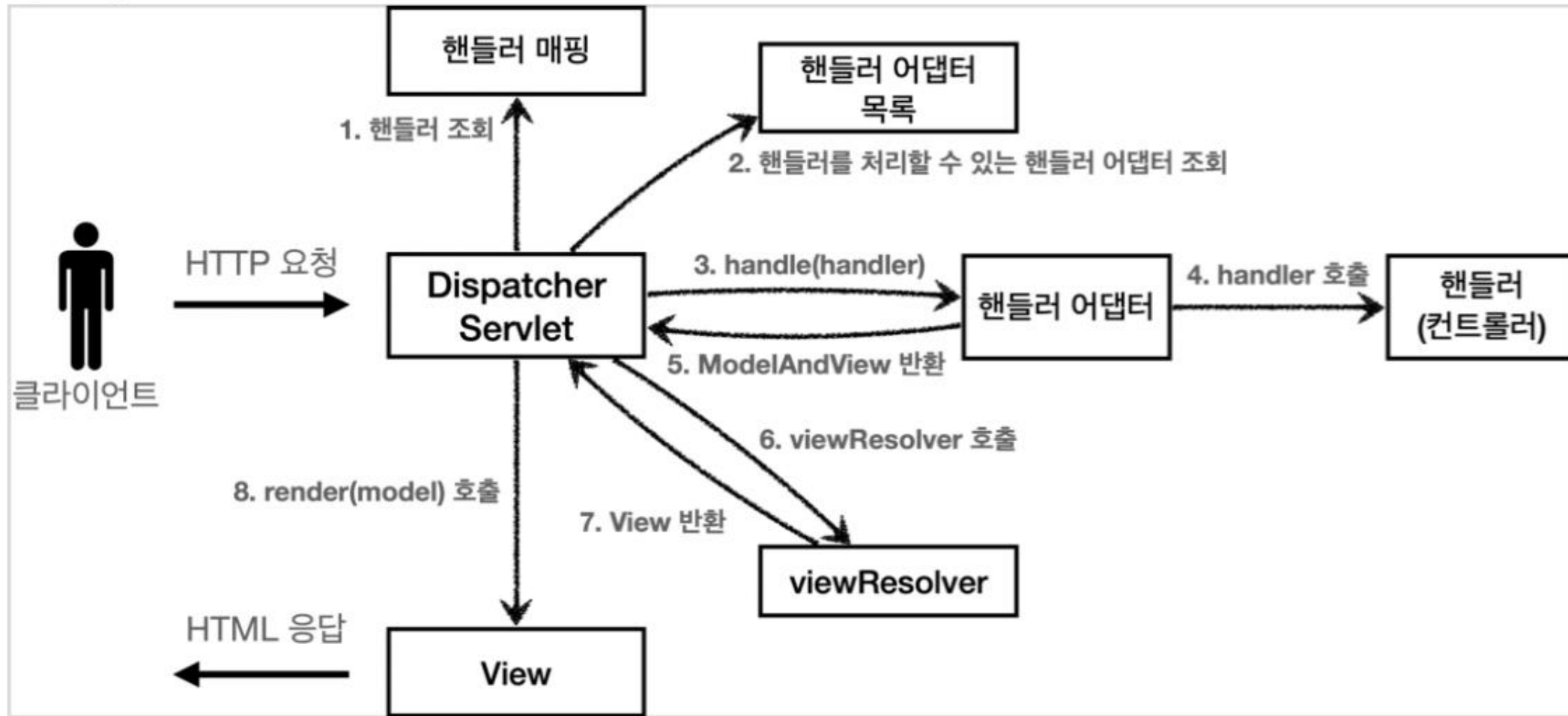
with



Spring MVC



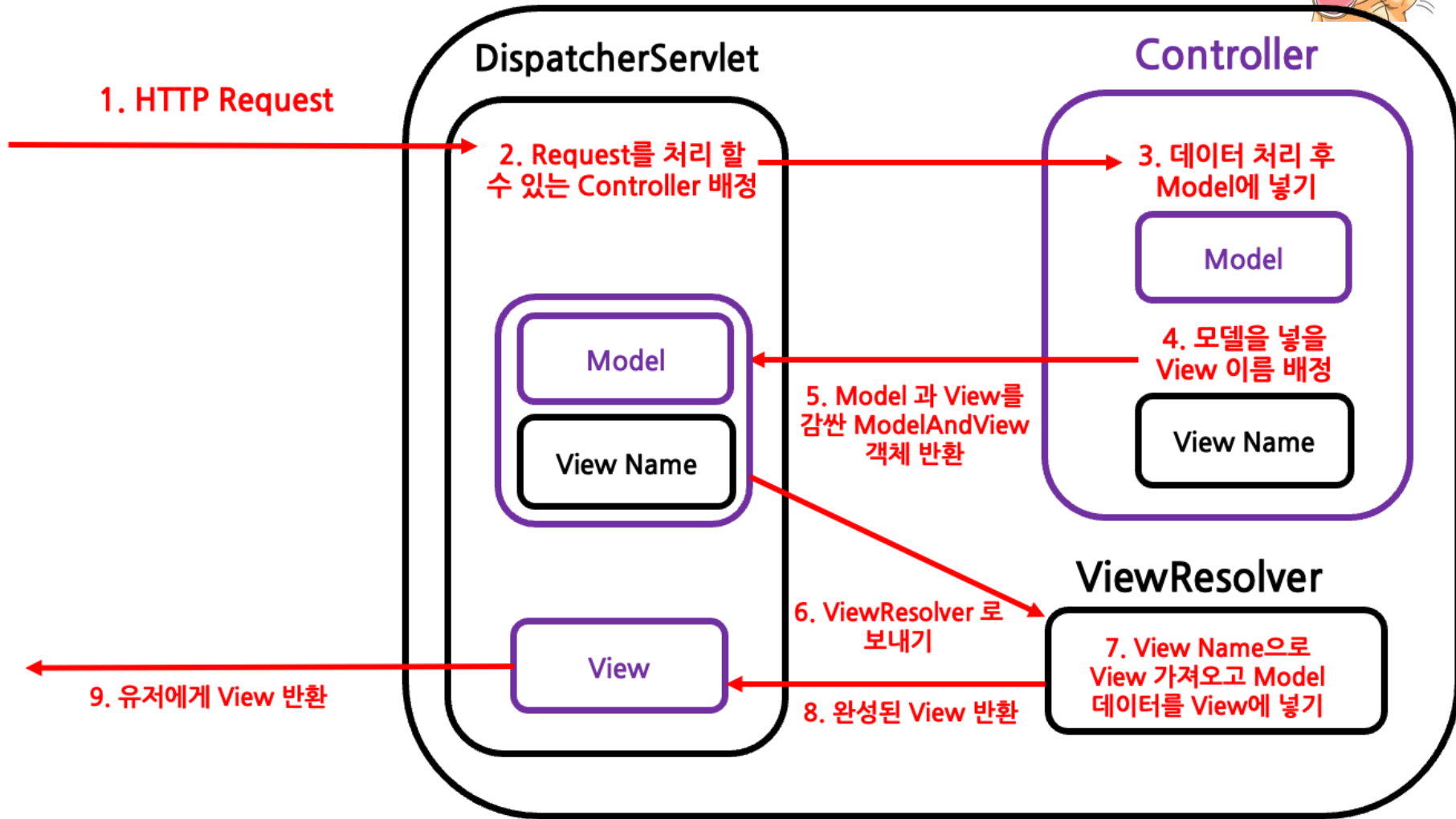
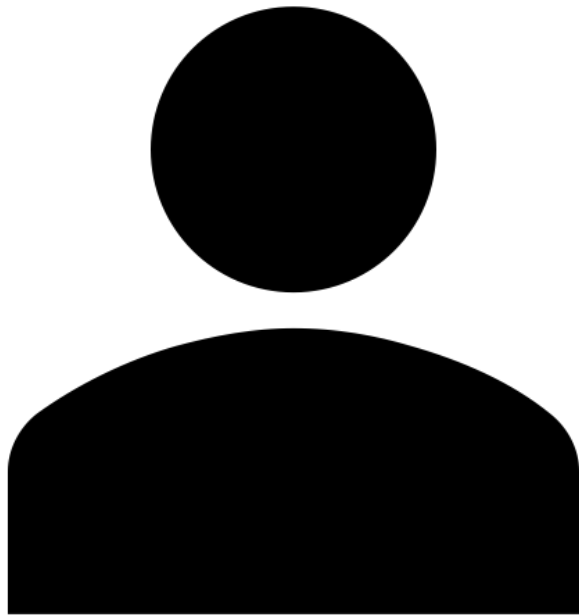
SpringMVC 구조



Server



유저





Controller 와

의존성 주입



열어라!





열어라!







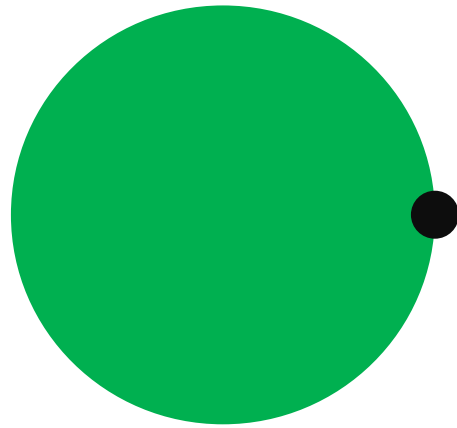
열어라!

차 문을 여는
서비스

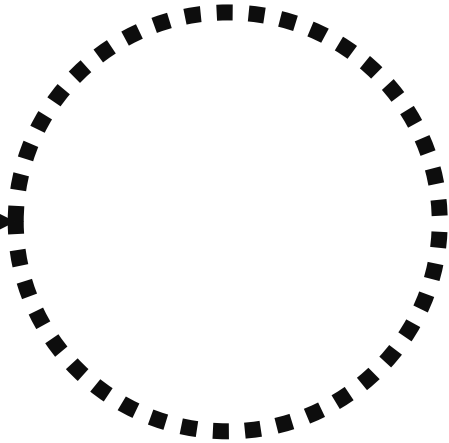




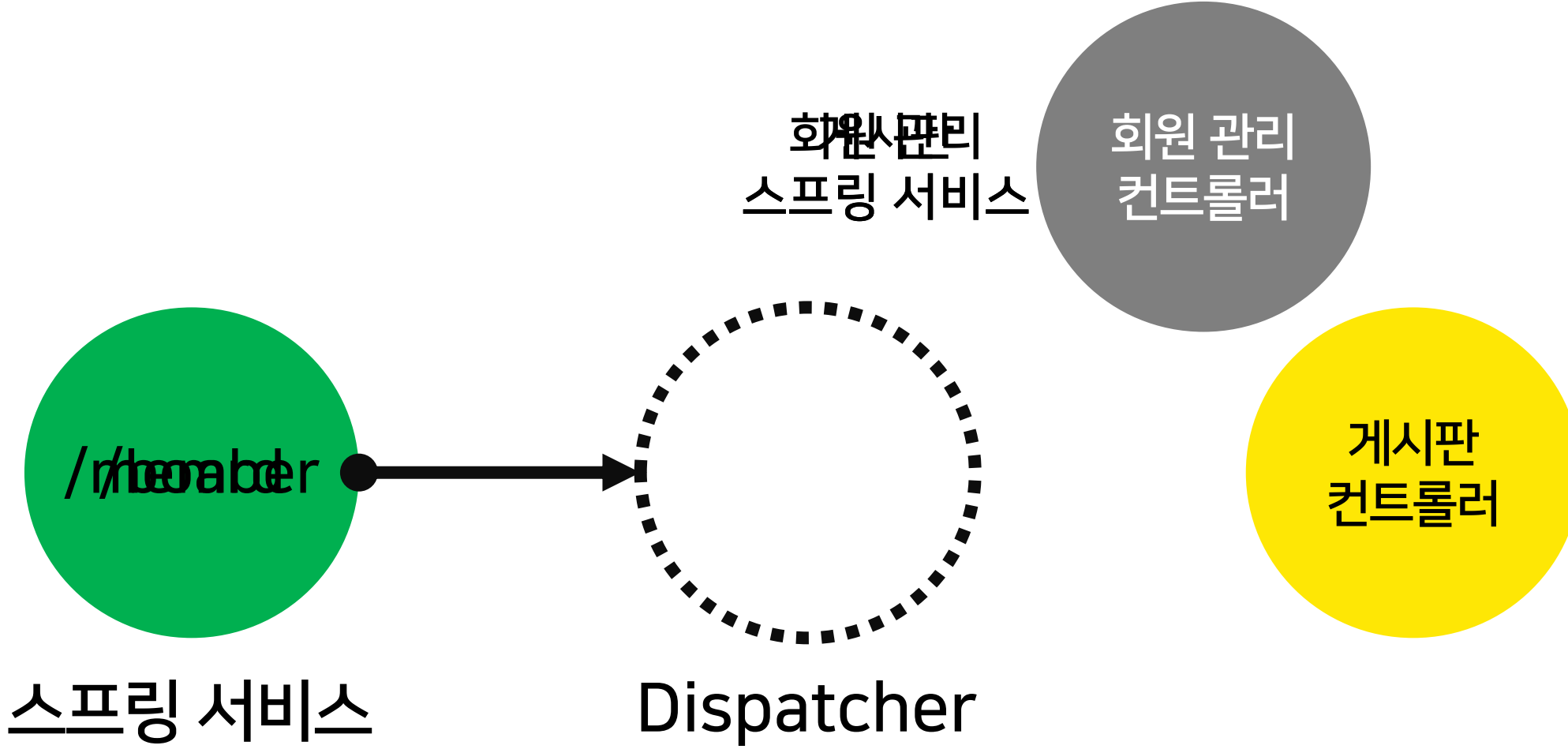
이걸 스프링에
적용하면!?



스프링 서비스



Dispatcher





와나...

난 이제 컨트롤러 생성 기계가 되겠구나...





스프링에
DTO 를 적용!

```
public class MemberDto { 6 usages  👤 kdtTetz
    private String id; 3 usages
    private String name; 3 usages
```

간단한 작업을 할 것이므로
id 와 name 만 사용!

```
public MemberDto(String id, String name) { 2 usages  👤
    this.id = id;
    this.name = name;
}
```

생성자, Getter / Setter 작업

```
public String getId() { return id; }
```

```
public void setId(String id) { this.id = id; }
```



그란데 말입니다...

싱글톤 패턴이 뭐죠!?

오랜만에 쿠폰 겁니다!! 😊

그란데 말입니다

MemberDtoList
인스턴스 1



MemberDtoList
인스턴스 2

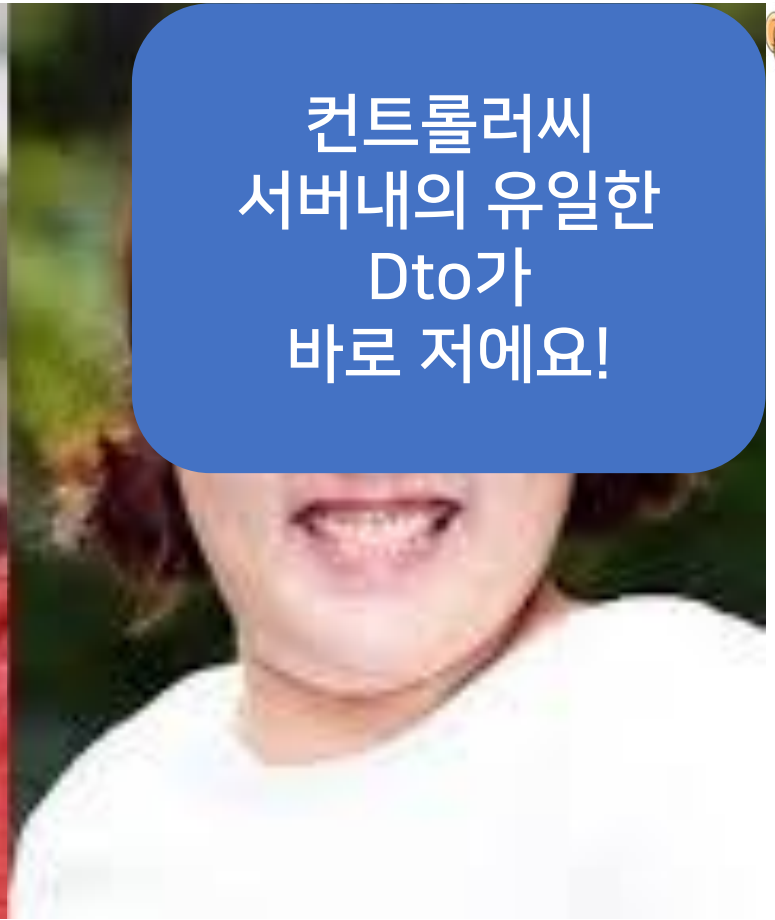


MemberDtoList
인스턴스 3





Controller



컨트롤러씨
서버내의 유일한
Dto가
바로 저예요!





회원 목록 보기

컨트롤러 작성



```
@Controller  🌐 kdtTetz *
@Slf4j

public class MemberShowControllerV1 {
    private MemberDtoListV1 memberList = MemberDtoListV1.getInstance(); 1 use

    @GetMapping(🌐 "/member/show")  🌐 kdtTetz *
    public String process(HttpServletRequest request, HttpServletResponse re
        log.info("=====> 회원 조회 페이지 호출, /member/show");

        request.setAttribute(s: "memberList", memberList.getList());
        return "member-show";
    }
}
```

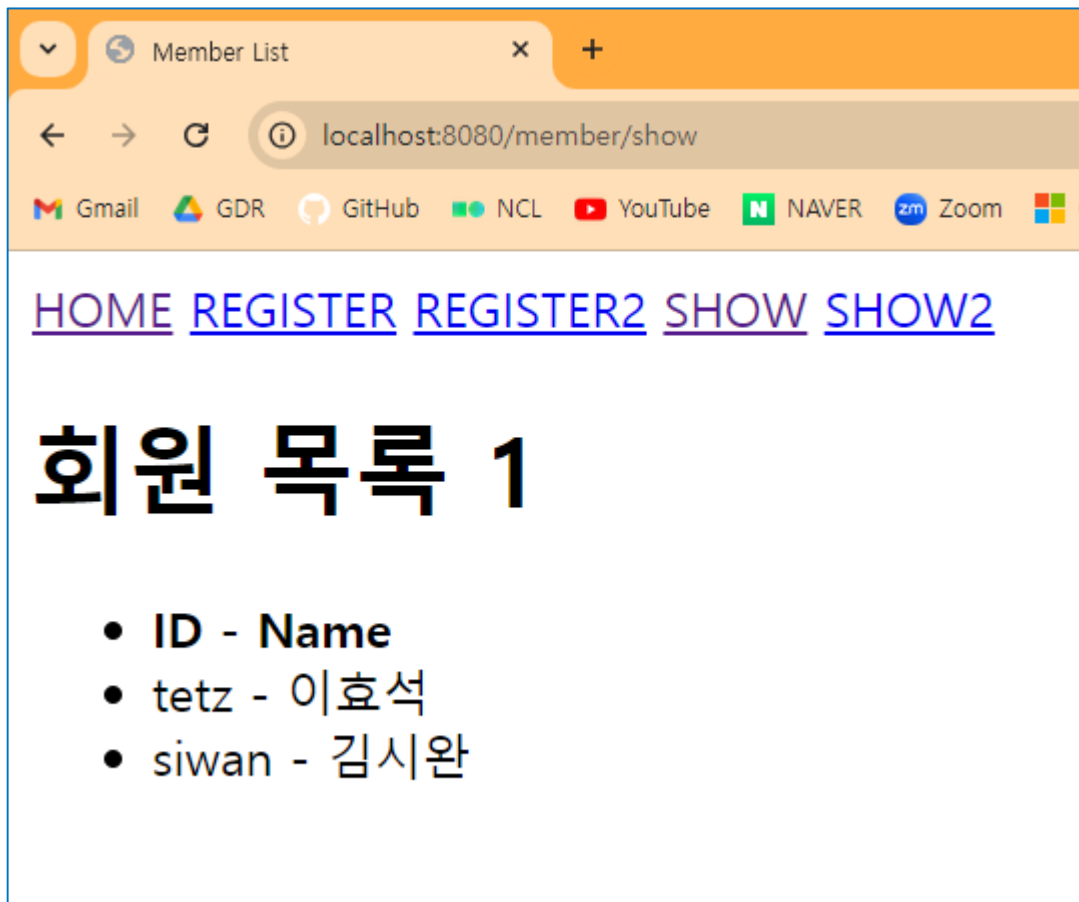


회원 목록 보기

jsp 페이지 작성



```
<%@ 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 List</title>
</head>
<body>
<%@ include file="header.jsp"%>
<h1>회원 목록 1</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>
```



회원 추가

jsp 페이지 작성



```
<%@ 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 1</h1>
<form method="get" action="/member/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>
</body>
</html>
```



```
✓ @Controller  🌐  👤 kdtTetz
@Slf4j
public class MemberFormControllerV1 {
    @GetMapping(🌐  👤 "/member/form")  👤 kdtTetz
    ✓ public String home(HttpServletRequest request, HttpServletResponse response) {
        log.info("=====> 회원 추가 페이지 호출, /member/register");

        return "member-form";
    }
}
```



회원 추가

컨트롤러 작성



Member Register

localhost:8080/member/form

Gmail GDR GitHub NCL YouTube NAVER Zoom O36

[HOME](#) [REGISTER](#) [REGISTER2](#) [SHOW](#) [SHOW2](#)

회원 추가 1

아이디 :

이름 :

회원 추가 버튼을 클릭하면
GET 방식으로 요청이 날아갑니다!



그란데 말입니다

Member List

x

+



localhost:8080/member/form/save?id=12&name=12



요러한 요청은 어떻게 받아줘야 할까요!?



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



서블릿 기준이면 request 객체에서
getParameter 를 사용해서 받았습니다!



그란데 말입니다

페이지를 호출하는 요청과
저렇게 특정 작업을 하는
요청을 어떻게 구분할 수 있을까요?

만약 페이지가 많다면?
서비스의 수가 엄청나다면!?

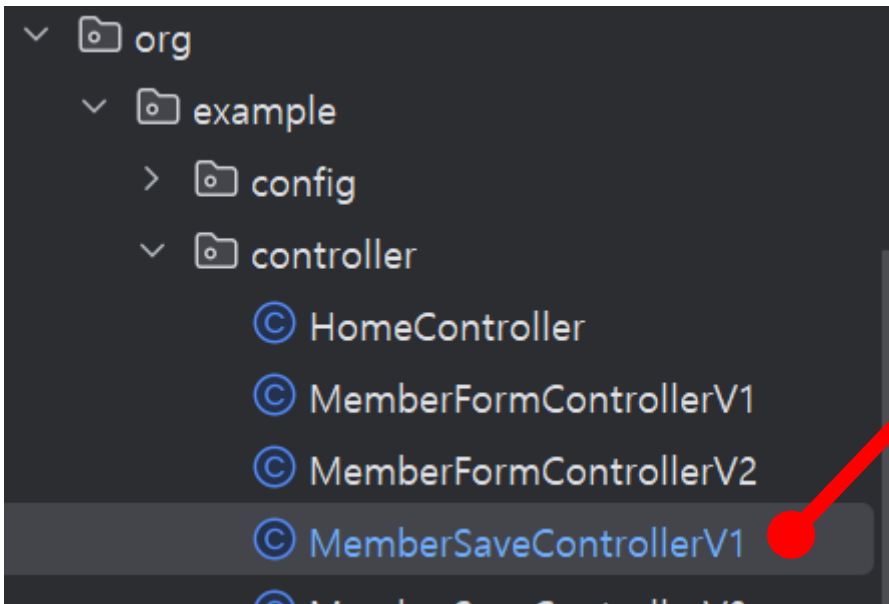
주소를 따로따로 분리하기가 매우 힘들어 집니다
그리고 메서드 별로 매핑하기도 힘들어 집니다!

@RequestMapping



Request 요청만 따로 받는 어노테이션이 등장!
(요게 스프링이 핫 해지는데 기여를 했습니다!)

이 몸, 등장



회원 저장 요청을 받아줄
MemberSaveControllerV1 을 생성

회원 추가를 해야하므로
회원 데이터를 관리하는 MemberDtoList 의
인스턴스 가져오기

```
@Controller
@Slf4j
public class MemberSaveControllerV1 {
    private MemberDtoListV1 memberList = MemberDtoListV1.getInstance(); 2 usages

    @RequestMapping(value = "/member/form/save", method = RequestMethod.GET)
    public String process(HttpServletRequest request, HttpServletResponse response) {
        log.info("=====> 회원 추가 Request 호출, /member/form/save");

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

        memberList.addList(id, name);

        request.setAttribute("memberList", memberList.getList());
        return "member-show";
    }
}
```



GET 방식 /member/form/save 로 전달되는
리퀘스트를 가져오도록
@RequestMapping 을 사용하여 매핑

@Controller  kdtTetz +1

@Slf4j

```
public class MemberSaveControllerV1 {
```

```
    private MemberDtoListV1 memberList = MemberDtoListV1.getInstance(); // usages
```

```
    @RequestMapping(value =  "/member/form/save", method = RequestMethod.GET) // kdtTe
```

```
    public String process(HttpServletRequest request, HttpServletResponse response) {
```

```
        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);
```

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

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

```
    }
```

```
}
```



@Controller kdtTetz +1

@Slf4j

```
public class MemberSaveControllerV1 {
```

```
    private MemberDtoListV1 memberList = MemberDtoListV1.getInstance(); 2 usages
```

```
    @RequestMapping(value =  "/member/form/save", method = RequestMethod.GET)  kdtTe
```

```
    public String process(HttpServletRequest request, HttpServletResponse response) {
```

```
        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);
```

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


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

```
    }
```

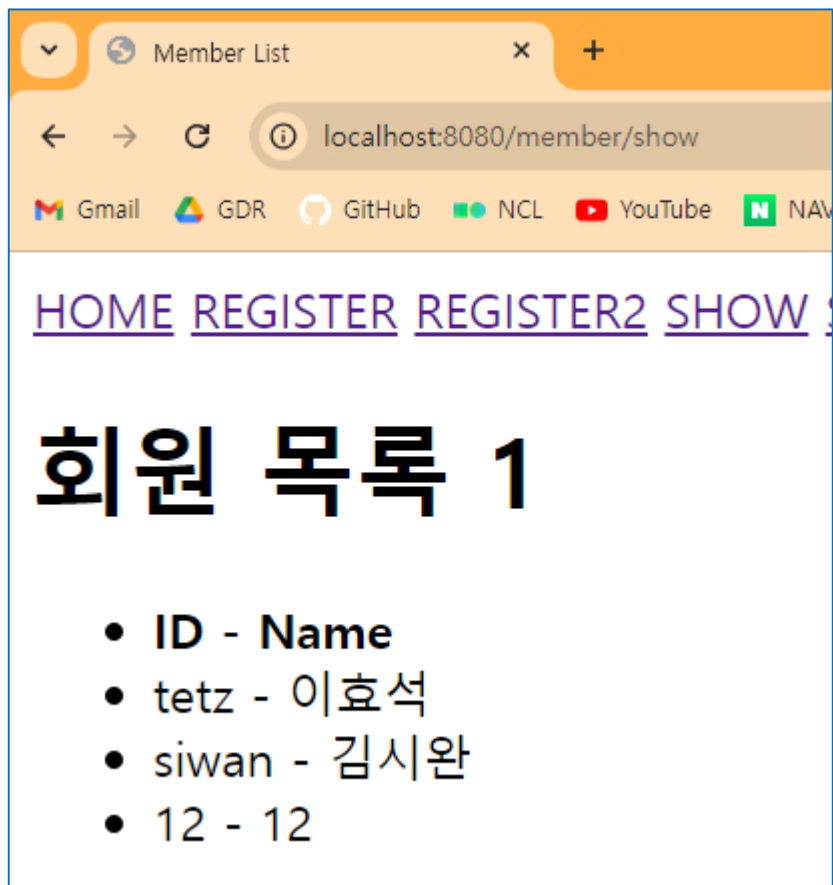
파라미터로 부터 데이터를 받아서
변수에 담기

```
}
```



```
public class MemberSaveControllerV1 {  
    private MemberDtoListV1 memberList = MemberDtoListV1.getInstance(); 2 usages  
  
    @RequestMapping(value =  "/member/form/save", method = RequestMethod.GET) kdtTe  
    public String process(HttpServletRequest request, HttpServletResponse response) {  
        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);  
  
        request.setAttribute(s: "memberList", memberList.getList());  
        return "member-show";  
    }  
}
```

멤버 리스트에 새로운 회원을 추가하고
Request 스코프에 담아서
member-show.jsp 페이지 호출





컨트롤러를 찍는 여러분의
미래 GIF

실습, Todo 기능을 구현해 봅시다!



- Todo 목록 보기와 추가 기능을 구현해 봅시다!
- 아래와 같이 Header 에 2가지 링크를 추가하고, 각각의 링크를 클릭하면 Todo 목록 보기와 Todo 추가 페이지가 보이면 됩니다
- Todo 추가 페이지에서 할 일을 입력하고 할 일 추가 버튼을 클릭하면 Todo 목록 페이지로 이동이 되며 추가 된 할 일 목록까지 출력이 됩니다!
- 위의 기능을 위한 TodoDtoListV1 / TodoShowControllerV1 / TodoFormControllerV1 / TodoSaveControllerV1 을 작성해 주세요!

실습, Todo 기능을 구현해 봅시다!



- TodoDto 코드입니다!

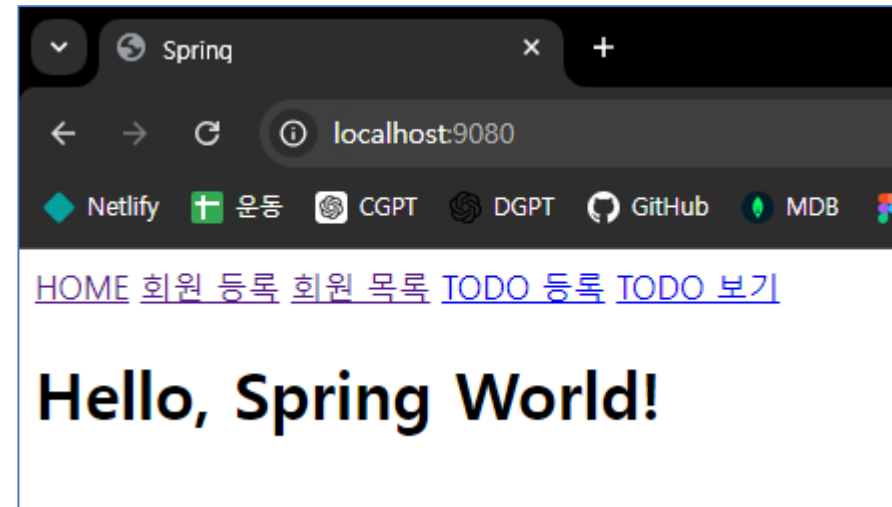
```
public class TodoDto { 3 usages  new *  
    private String todo; 3 usages  
  
    public String getTodo() { no usages  
        return todo;  
    }  
  
    public void setTodo(String todo) {  
        this.todo = todo;  
    }  
  
    public TodoDto(String todo) { 1 usage  
        this.todo = todo;  
    }  
}
```

실습, Todo 기능을 구현해 봅시다!



- 새로운 Header 코드 및 실제 화면 입니다!

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<header>
  <a href="/">HOME</a>
  <a href="/member/form">회원 등록</a>
  <a href="/member/show">회원 목록</a>
  <a href="/todo/form">TODO 등록</a>
  <a href="/todo/show">TODO 보기</a>
</header>
```

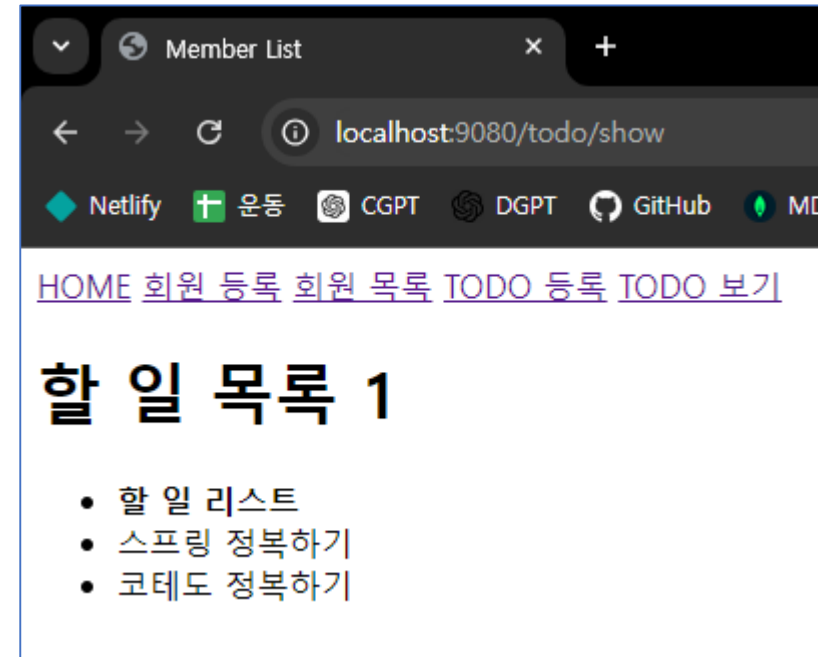


실습, Todo 기능을 구현해 봅시다!



- Todo 보기 페이지(todo-show.jsp) 코드 및 실제 화면 입니다

```
<%@ 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 List</title>
</head>
<body>
  <%@ include file="header.jsp"%>
  <h1>할 일 목록 1</h1>
  <ul>
    <li><b>할 일 리스트</b></li>
    <c:forEach var="todo" items="${todoList}">
      <li>${todo.todo}</li>
    </c:forEach>
  </ul>
</body>
</html>
```



실습, Todo 기능을 구현해 봅시다!



- Todo 등록 페이지(todo-form.jsp) 코드 및 실제 화면 입니다

```
<%@ 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>할 일 추가 1</h1>
    <form method="get" action="/todo/form/save">
        <label for="todo">할일 :</label>
        <input type="password" id="todo" name="todo" required>
        <br>
        <button type="submit">할일 추가</button>
    </form>
</body>
</html>
```

Member Register

localhost:9080/todo/form

Netlify 운동 CGPT DGPT GitHub ME

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

할 일 추가 1

할일 :



코드 빈칸

채우기 및 코드 제공

TodoDtoListV1 는 코드 전체를 드립니다! 😊



```
public class TodoDtoListV1 { 9 usages  new *
    private static TodoDtoListV1 instance; 3 usages
    private List<TodoDto> todoDtoList; 3 usages

    private TodoDtoListV1() { 1 usage  new *
        this.todoDtoList = new ArrayList<>(); // List 초기화
        // 테스트 데이터 추가
        this.addList(todo: "스프링 정복하기");
        this.addList(todo: "코테도 정복하기");
    }

    // 싱글톤 인스턴스 반환 메소드
    public static synchronized TodoDtoListV1 getInstance() {
        if (instance == null) {
            instance = new TodoDtoListV1();
        }
        return instance;
    }
}
```

```
public void addList(String todo) { 3 usages  new *
    todoDtoList.add(new TodoDto(todo));
}

public List<TodoDto> getList() { return todoDtoList; }
```


TodoShowControllerV1



```
@Controller  Tetz *  
@Slf4j  
public class TodoShowControllerV1 {  
    private TodoDtoListV1 todoList = TodoDtoListV1.getInstance(); 1 usage  
  
     Rename usages  
    @GetMapping( "/todo/show") Tetz *  
    public String todoShow(HttpServletRequest request, HttpServletResponse response) {  
  
  
    }  
}
```

요기를 완성하세요!!

TodoFormControllerV1



```
@Controller  
@Slf4j  
public class TodoFormControllerV1 {  
    @GetMapping("/todo/form")  
    public String todoForm(HttpServletRequest request, HttpServletResponse response) {  
  
        }  
}
```

여기를 완성하세요!!



@S7f4j

```
private TodoDtoListV1 todoDtoListV1 = TodoDtoListV1.getInstance(); 2 usages
```

요기를 완성하세요!!

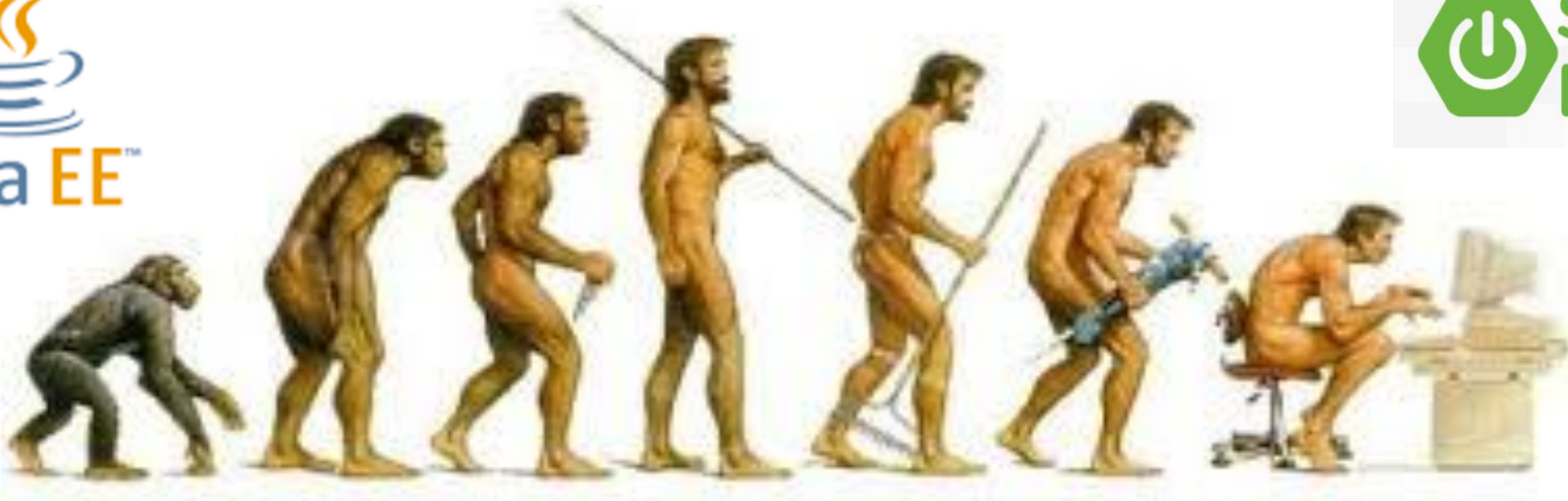
}

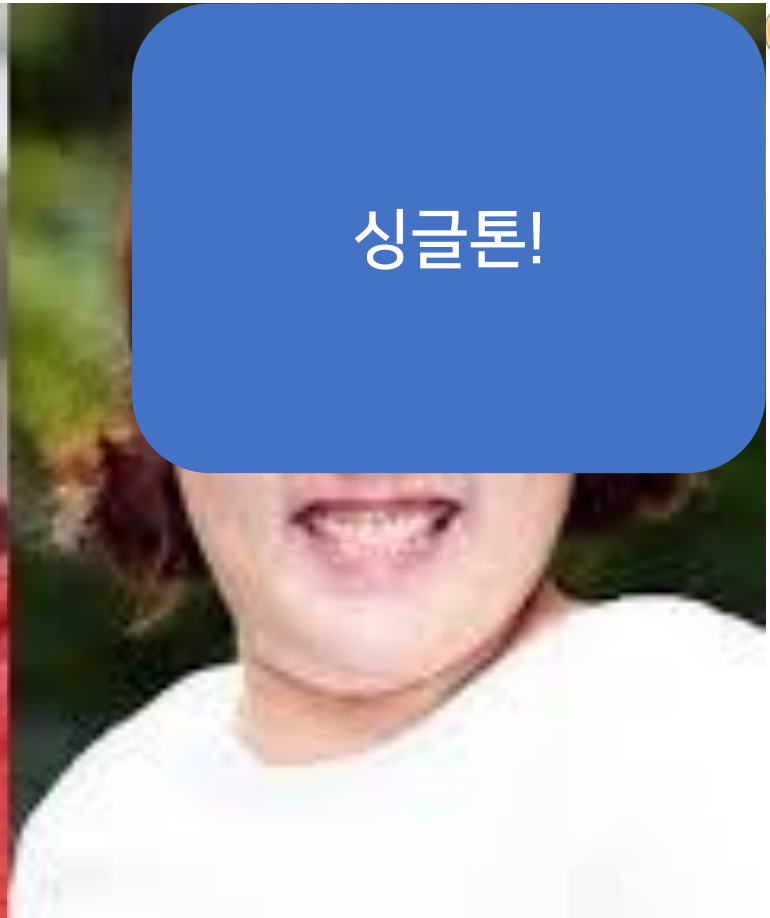
}



스프링의 편의 기능

활용하기! V2





싱글톤!



```
@Controller
```

```
@Slf4j
```

```
public class MemberShowControllerV1 {
```

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

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

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

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

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

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

```
}
```

```
}
```

SCOPE



10:36



컨트롤러를 찍는 여러분의
미래 GIF



여러분!





Spring Bean



```
public class MemberDtoListV1 { 16 usages  👤 kdtTetz
    private static MemberDtoListV1 instance; 3 usages
    private List<MemberDto> memberDtoList; 3 usages

    private MemberDtoListV1() { 1 usage  👤 kdtTetz
        this.memberDtoList = new ArrayList<>(); // List
        // 테스트 데이터 추가
        this.addList(id: "tetz", name: "이효석");
        this.addList(id: "siwan", name: "김시완");
    }

    // 싱글톤 인스턴스 반환 메소드
    public static synchronized MemberDtoListV1 getInstance() {
        if (instance == null) {
            instance = new MemberDtoListV1();
        }
        return instance;
    }
}
```

스프링 컨트롤러가 공통으로
데이터를 사용하기 위해서

싱글톤 패턴을 적용해줘야 합니다!



@Controller  kdtTetz +1

@Slf4j

```
public class MemberShowControllerV1 {  
    private MemberDtoListV1 memberList = MemberDtoListV1.getInstance();  
}
```



또한 사용을 위해서는
싱글톤 객체의 인스턴스를 가져와서
사용해줘야 했습니다!



스프링이 왜 나왔죠!?



그란데 말입니다



10:36



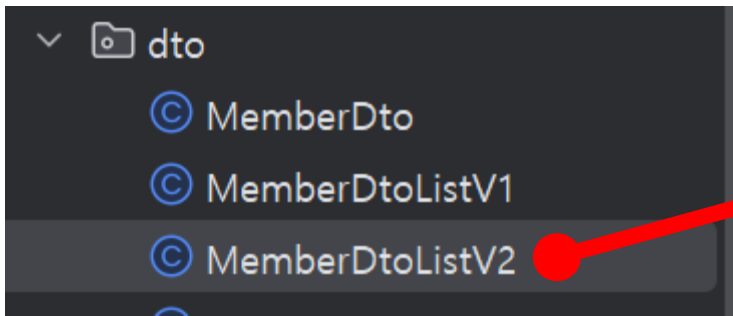




내가 @어노테이션으로
다 등록 해줄 테니까
편하게 등록하고 사용하면 됩니다!



10:36



스프링 빈을 사용하기 위한
MemberDtoListV2 추가





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!



@Component 10 usages kdtTetz

```
public class MemberDtoListV2 {
```

```
    private List<MemberDto> memberDtoList; 3 usages
```

```
    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; }
```

```
}
```

생성자를 사용하여
멤버 변수를 초기화!

```
@Component 10 usages  kdtTetz
public class MemberDtoListV2 {
    private List<MemberDto> memberDtoList; 3 usages

    public MemberDtoListV2() {  kdtTetz
        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; }
}
```

기존 메서드는 동일하지만
싱글톤 패턴 적용을 위한
getInstance() 메서드는 이제
없어도 됩니다!



Spring Bean

설정! (중요!!!!)



@Component



너 왜 나 등록 안함!?

모르는데 어떻게 가요!



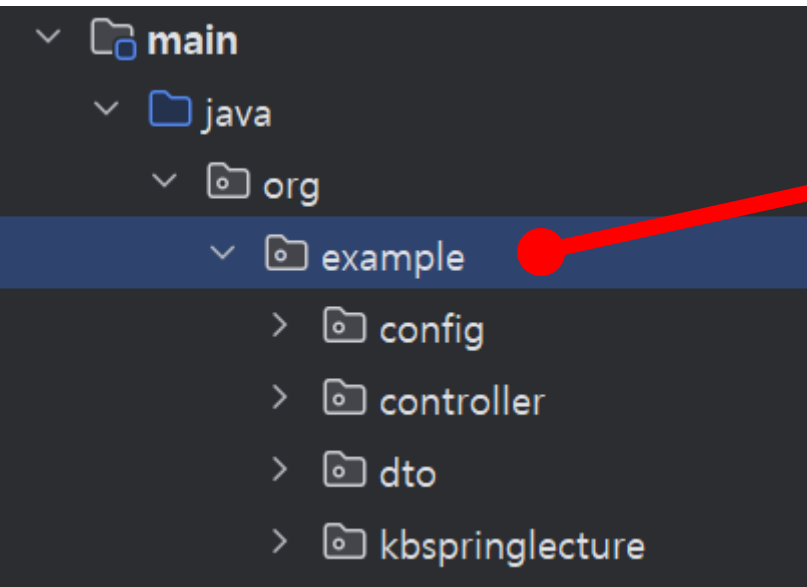
기본 설정을 담당하는
RootConfig 로 이동!

```
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!!!)



우리가 사용할 컴포넌트(=Bean)는
전부 example 패키지 하위에 존재할 것이므로
org.example 에서 찾으라고 시키면 됩니다!

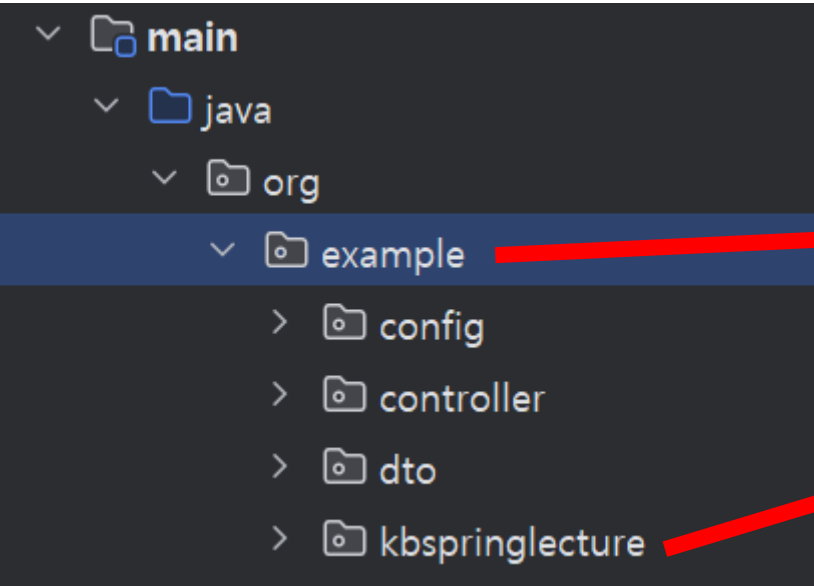


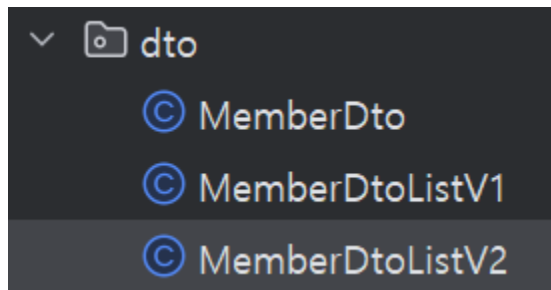
```
2  
3 > import ...  
7 @Configuration kdtTetz  
8 @ComponentScan(basePackages = "org.example")  
9 public class RootConfig {  
10  
11 }
```

컴포넌트 스캔의 범위를
org.example 로 설정!

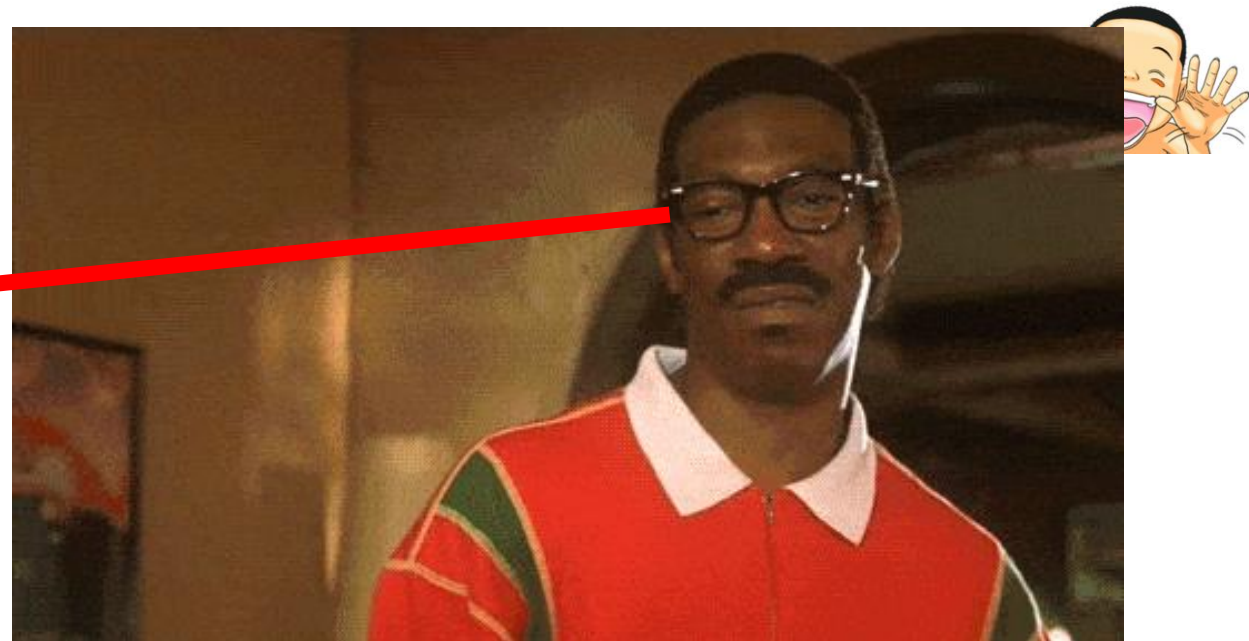


그럼 어떻게
작동할까요!?





```
@Component 10 usages kdtTetz
public class MemberDtoListV2 {
    private List<MemberDto> memberDtoList;
```





Component(=Bean) 목록



MemberDtoListV2 memberDtoList

...

...

...

...

"넌 내 마음속에 ㄱ ㄴ 으로 접을거야."



Component(=Bean) 목록

MemberDtoListV2 memberDtoList

...
...
...
...



Controller1

Controller2

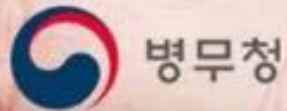
Controller3

Controller4



99년생님을
병무청 마음속에 저장

2018년도 병역판정검사 본인선택 안내





그럼 어떻게

쓰나요!?

controller

MemberShowControllerV1

MemberShowControllerV2

Bean 을 사용할 MemberShowControllerV2
컨트롤러 생성



기존과 동일하게 @어노테이션을 이용하여
컨트롤러를 등록 & 만들어 줍니다!



@Controller  kdtTetz

@Slf4j

public class MemberShowControllerV2 {


private final MemberDtoListV2 memberDtoList; 2 usages

@Autowired kdtTetz

public MemberShowControllerV2(MemberDtoListV2 memberDtoList) {

 this.memberDtoList = memberDtoList;

}

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

public String process(HttpServletRequest request, Model model) {

 log.info("=====> 회원 조회 페이지 호출, " + request.getRequestURI());

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

 return "member-show2";

}

}

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";  
}
```

```
}
```



@Controller kdtTetz

@Slf4j

public class MemberShowControllerV2 {

private final MemberDtoListV2 memberDtoList; 2 usages

@Autowired kdtTetz

public MemberShowControllerV2(MemberDtoList

this.memberDtoList = memberDtoList;

}

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

public String process(HttpServletRequest request, Model model) {

log.info("=====> 회원 조회 페이지 호출, " + request.getRequestURI());

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

return "member-show2";

}


}

주소로 V2 를 구분해야 하므로
주소 요청을 /member/v2/show 로 변경

나머지 코드는 동일!


@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!!

전달받은 Bean 의 인스턴스를
잘 사용하기만 하면 끝!



@Controller  kdtTetz

@Slf4j

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

@Autowired  kdtTetz

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

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

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

```
}
```



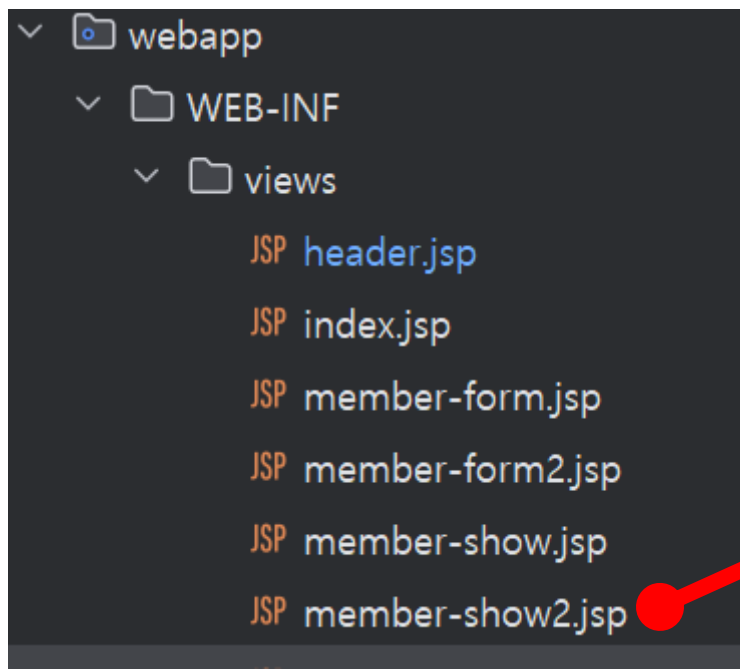
확인을 위한

View 파트 변경

```
<%@ page contentType="text/html; charset=UTF-8"
<header>
    <h3>V1</h3>
    <a href="/">HOME</a>
    <a href="/member/form">회원 등록</a>
    <a href="/member/show">회원 목록</a>
    <a href="/todo/form">TODO 등록</a>
    <a href="/todo/show">TODO 보기</a>
    <br>
    <h3>V2</h3>
    <a href="/member/v2/form">V2 회원 등록</a>
    <a href="/member/v2/show">V2 회원 목록</a>
</header>
```

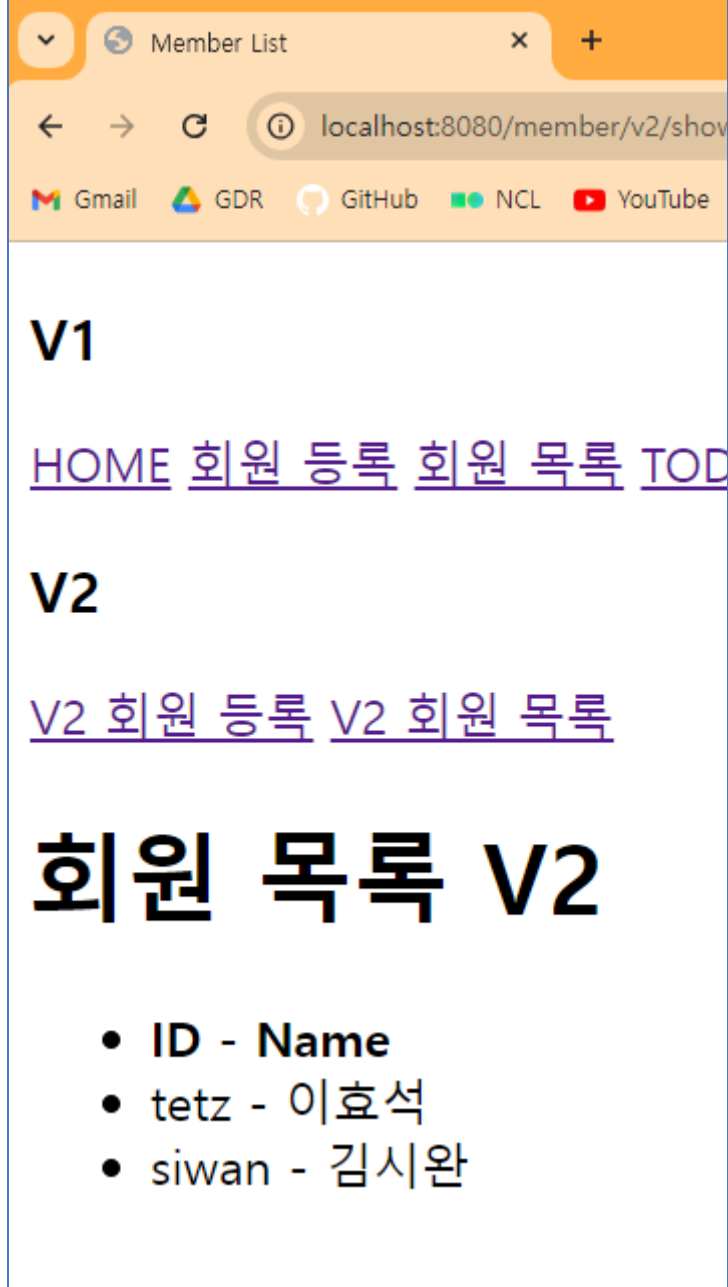
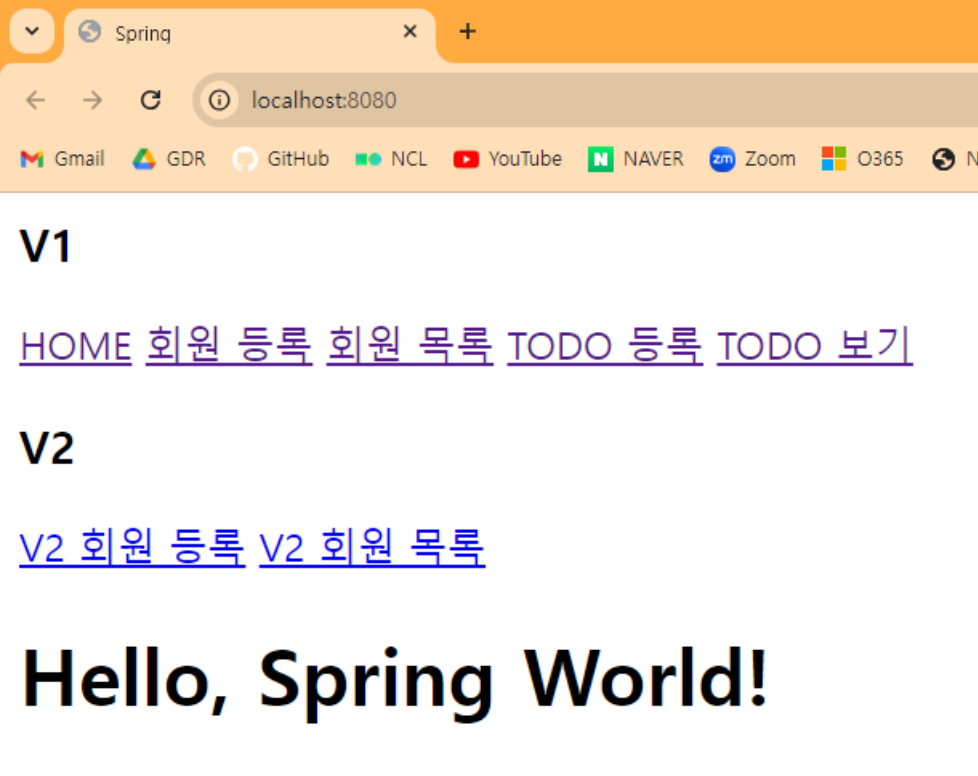


버전 구분을 위해 header 를
변경 했습니다~!



member-show.jsp 파일을 카피해서
member-show2.jsp 만들기

구분을 위해서 <h1> 은
회원 목록 V2 로 변경하기





회원 추가 페이지

컨트롤러 추가

controller

HomeController

MemberControllerV3

MemberFormControllerV1

MemberFormControllerV2

회원 등록 페이지 추가를 위해서
MemberFormControllerV2 추가



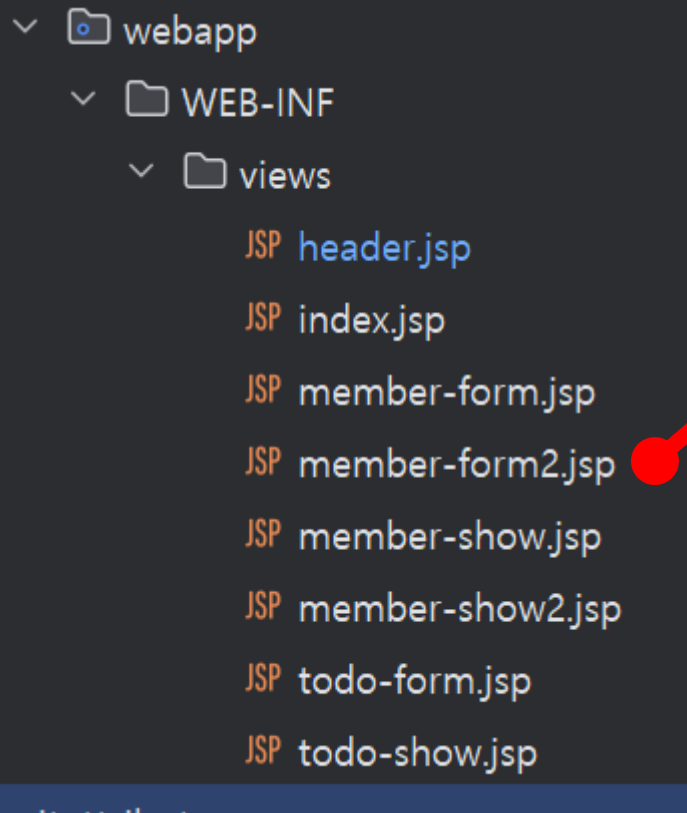


주소 요청을 /member/v2/form 으로
변경해 줍니다!

```
@Controller
@Slf4j
public class MemberFormControllerV2 {
    @GetMapping("/member/v2/form")
    public String home(HttpServletRequest request, HttpServletResponse response) {
        log.info("=====> 회원 추가 페이지 호출, /member/register");

        return "member-form2";
    }
}
```

추가 요청 주소가 달라질 예정이므로
jsp 페이지는 member-form2 로 호출



member-form2.jsp 파일 추가



```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<title>Member Register</title>
</head>
<body>
<%@ include file="header.jsp"%>
<h1>회원 추가 V2</h1>
<form method="get" action="/member/v2/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>
</body>
</html>
```

구분을 위해서 회원 추가 V2 로 변경

등록 요청을
/member/v2/form/save 로 변경



회원 추가 기능

컨트롤러 추가



controller

HomeController

MemberControllerV3

MemberFormControllerV1

MemberFormControllerV2

MemberSaveControllerV1

MemberSaveControllerV2


회원 등록 기능 추가를 위해서
MemberSaveControllerV2 추가



@Controller  kdtTetz *

@Slf4j

```
public class MemberSaveControllerV2 {  
    private final MemberDtoListV2 memberList; }
```

@Autowired  kdtTetz

```
public MemberSaveControllerV2(MemberDtoListV2 memberList) {  
    this.memberList = memberList;  
}
```

@Autowired 어노테이션을 사용하여
Bean 을 주입

Form 요청이 오는 주소인
/member/v2/form/save 를 매핑



```
@RequestMapping("/member/v2/form/save")
public String process(HttpServletRequest request, HttpServletResponse response) {
    log.info("=====> 회원 추가 Request 호출, /member/form/save");

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

    memberList.addList(id, name);

    request.setAttribute("memberList", memberList.getList());
    return "member-show2";
}
```

Request 의 파라미터로 부터
id 와 name 을 받기



```
@RequestMapping("/member/v2/form/save")
public String process(HttpServletRequest request, HttpServletResponse response) {
    log.info("=====> 회원 추가 Request 호출, /member/form/save");

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

    memberList.addList(id, name);

    request.setAttribute("memberList", memberList.getList());
    return "member-show2";
}
```

Bean 으로 부터 주입받은
회원 목록 데이터에 새로운 회원을 추가

request 에 데이터를 담아서
view 페이지에 전달!



Member Register

localhost:8080/member/v2/form

HOME 회원 등록 회원 목록 TODO 등

V1

V2

[V2 회원 등록](#) [V2 회원 목록](#)

회원 추가 V2

아이디 :

이름 :

회원 추가

Member List

localhost:8080/member/v2/s

HOME 회원 등록 회원 목록 TO

V1

V2

[V2 회원 등록](#) [V2 회원 목록](#)

회원 목록 V2

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



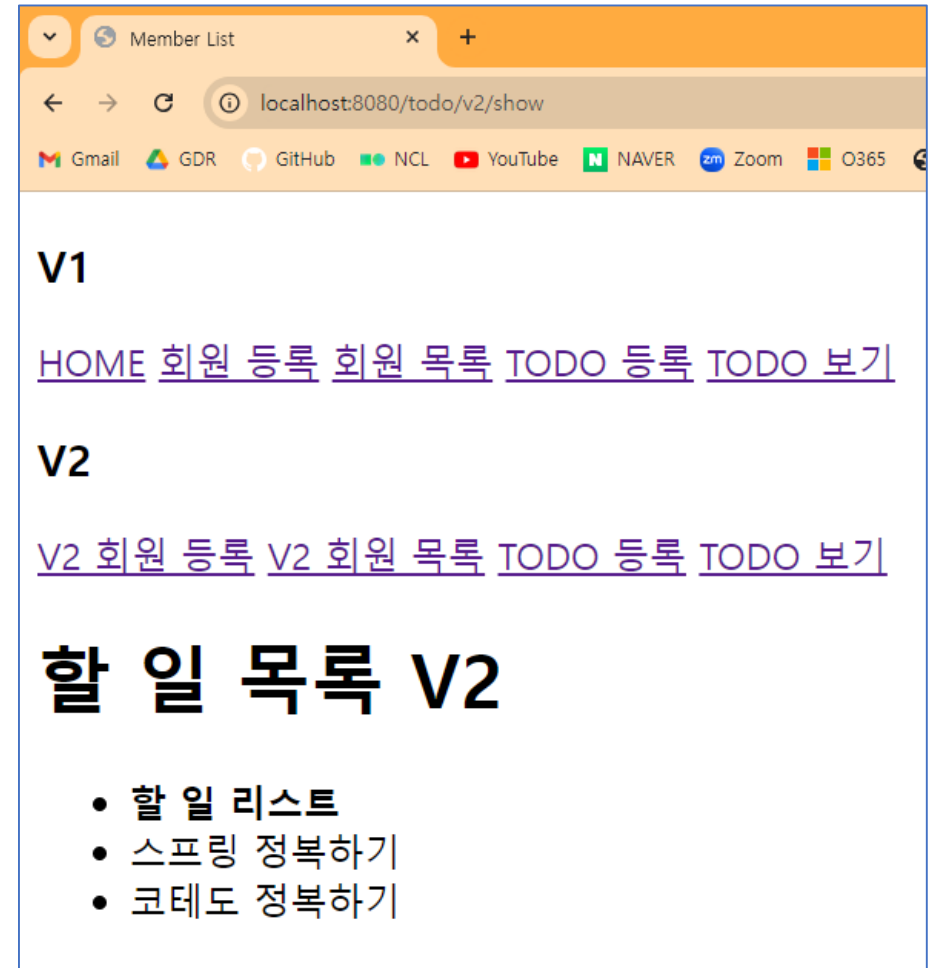
실습, Todo 에 Bean 을 적용 V2로 업그레이드

- 스프링 Bean 을 적용하여 Todo 기능을 V2 로 업그레이드 해봅시다
- 먼저 TodoDtoListV2 를 만들어서 스프링 빈으로 등록 시켜 줍니다 (Feat. @Component)
- Todo 목록 보기, Todo 등록 페이지 보기, Todo 등록 처리를 위해서
TodoShowControllerV2 / TodoFormControllerV2 /
TodoSaveControllerV2 컨트롤러를 만들어 주세요
- 모든 컨트롤러는 스프링 Bean 을 사용하여 구현해야 합니다

실습, Todo 에 Bean 을 적용 V2로 업그레이드

- TodoShowControllerV2 에 대응하는 todo-show2.jsp 의 코드

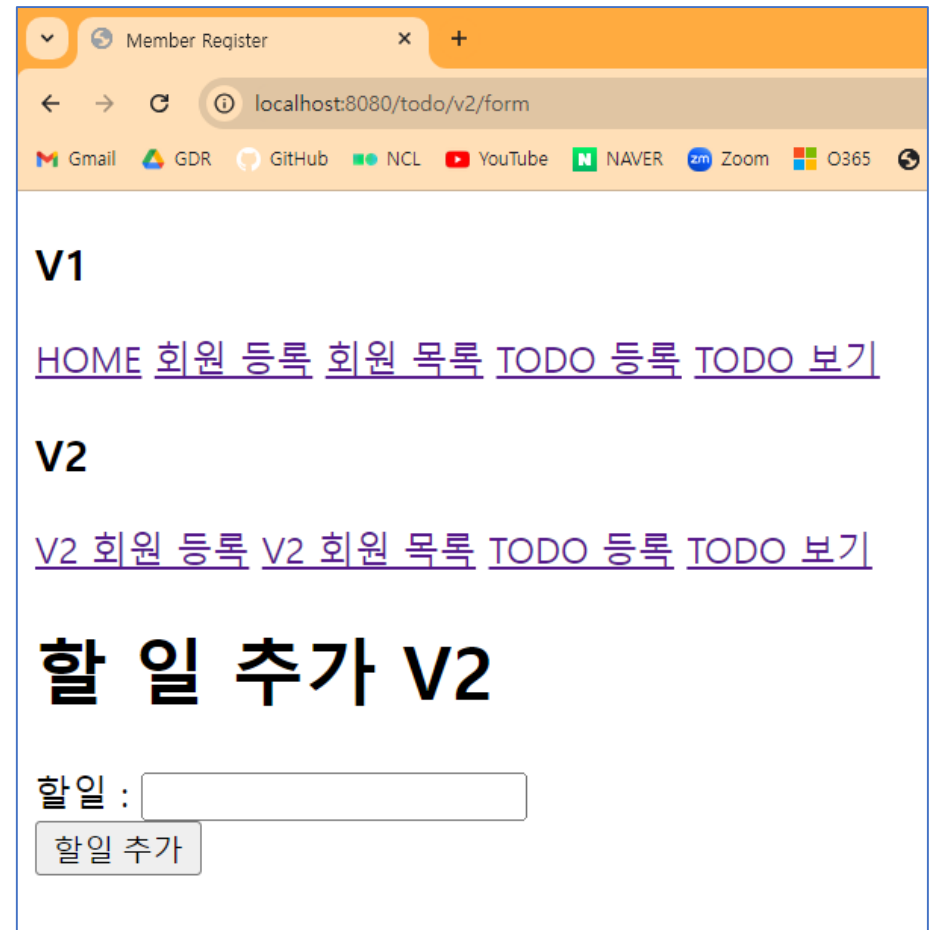
```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
<title>Member List</title>
</head>
<body>
<!-- @ include file="header.jsp" -->
<h1>할 일 목록 V2</h1>
<ul>
<li><b>할 일 리스트</b></li>
<c:forEach var="todo" items="${todoList}">
<li>${todo.todo}</li>
</c:forEach>
</ul>
</body>
</html>
```



실습, Todo 에 Bean 을 적용 V2로 업그레이드

- TodoFormControllerV2 에 대응하는 todo-form2.jsp 의 코드

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
    <title>Member Register</title>
</head>
<body>
<%@ include file="header.jsp"%>
<h1>할 일 추가 V2</h1>
<form method="get" action="/todo/v2/form/save">
    <label for="todo">할일 :</label>
    <input type="password" id="todo" name="todo" required>
    <br>
    <button type="submit">할일 추가</button>
</form>
</body>
</html>
```



Member Register

localhost:8080/todo/v2/form

Gmail GDR GitHub NCL YouTube NAVER Zoom O365

V1

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

V2

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

할 일 추가 V2

할일 :



코드 빈칸

채우기 및 코드 제공

TodoDtoListV2



```
> import ...
```

요기를 완성하세요!!

```
public class TodoDtoListV2 {  
    private List<TodoDto> todoDtoList; 3 usages
```

요기를 완성하세요!!

```
}
```

```
}
```

TodoShowControllerV2



```
@Controller  new *
@Slf4j
public class TodoShowControllerV2 {
    private TodoDtoListV2 todoDtoList; 2 usages
```

요기를 완성하세요!!

```
@GetMapping("/todo/v2/show")  new *
public String todoShow(HttpServletRequest request, HttpServletResponse response) {
    log.info("=====> TODO 리스트 보기 페이지 호출, /todo/show");

    request.setAttribute("todoList", todoDtoList.getList());
    return "todo-show2";
}
```

TodoFormControllerV2



```
@Controller  new *  
@Slf4j  
public class TodoFormControllerV2 {
```

여기를 완성하세요!!



```
}
```

TodoSaveControllerV2



```
@Controller  new *
@Slf4j
public class TodoSaveControllerV2 {
    private TodoDtoListV2 todoDtoList; 3 usages
```

요기를 완성하세요!!

```
public TodoSaveControllerV2(TodoDtoListV2 todoDtoList) {
    this.todoDtoList = todoDtoList;
}
```

```
@RequestMapping(value = "/todo/v2/form/save", method = RequestMethod.GET) new
public String todoSave(HttpServletRequest request, HttpServletResponse response)
    log.info("=====> 회원 추가 Request 호출, /member/form/save");
```

요기를 완성하세요!!

```
request.setAttribute(s: "todoList", todoDtoList.getList());
return "todo-show2";
```

```
}
```

```
}
```

데이터 전달을 위한



Model 객체

PAGE

jsp 페이지에서만 생존

REQUEST

요청에 대한 응답이 끝날 때까지 생존

SESSION

세션이 유지되는 동안 생존

APPLI
CATION

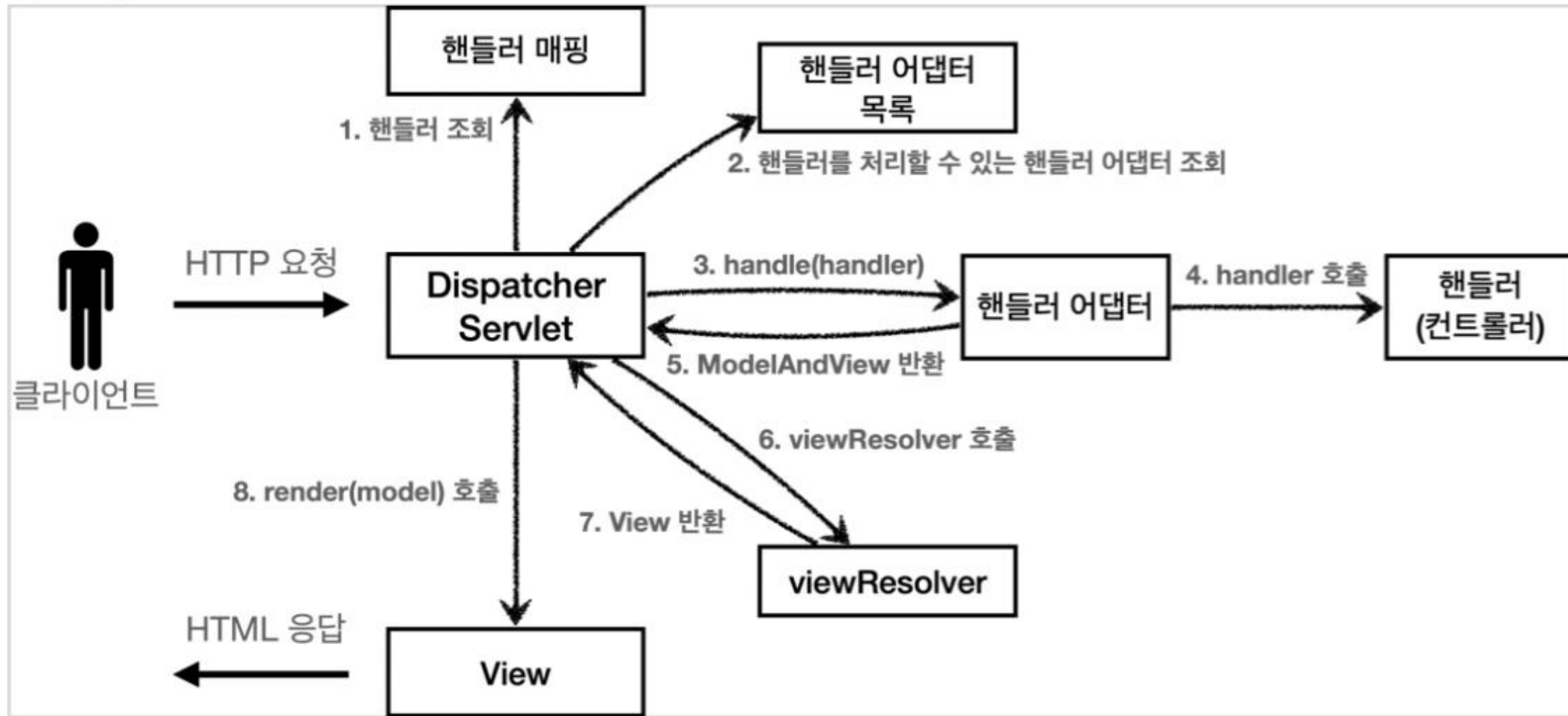
서버가 꺼질 때까지 생존



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



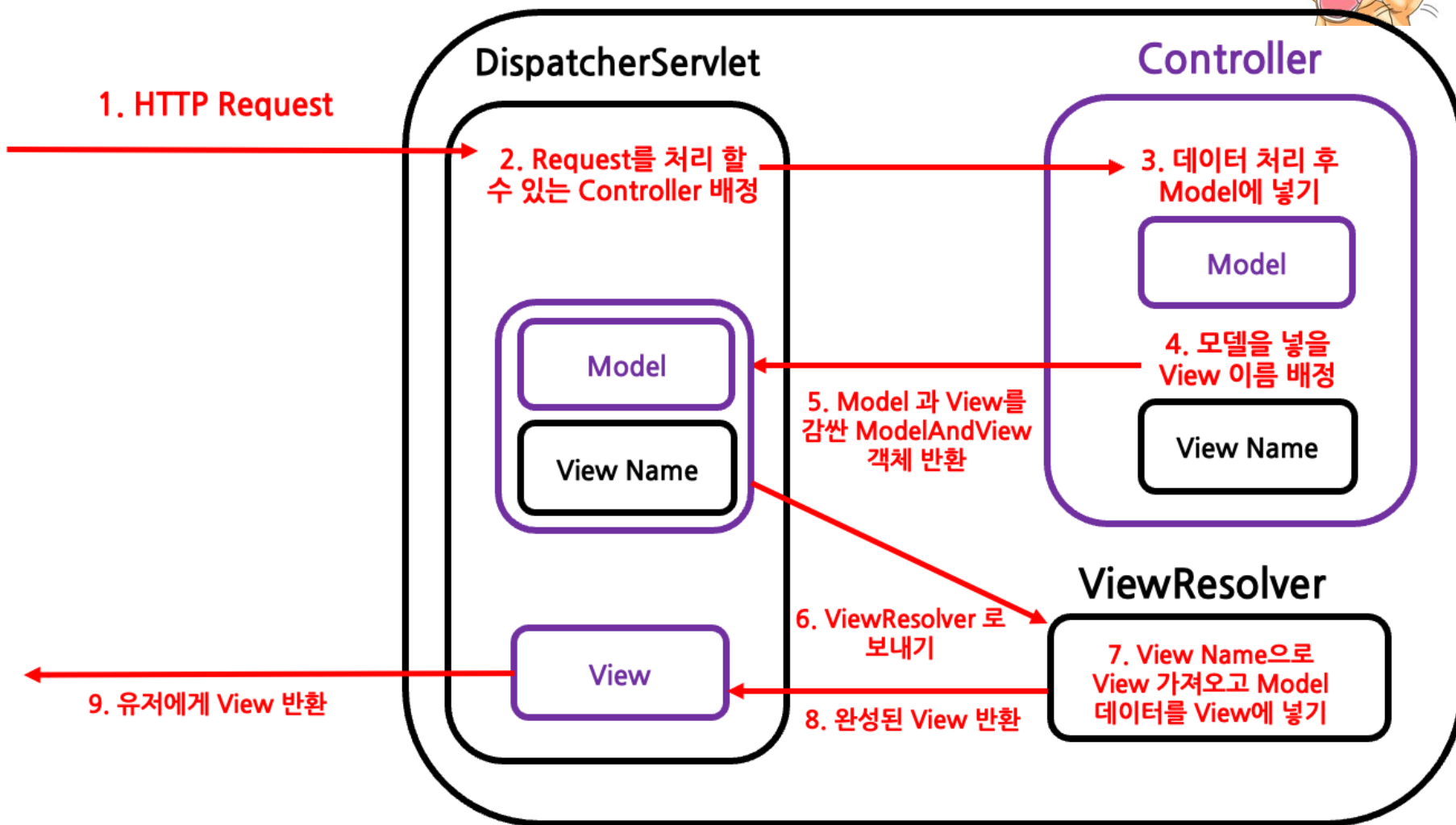
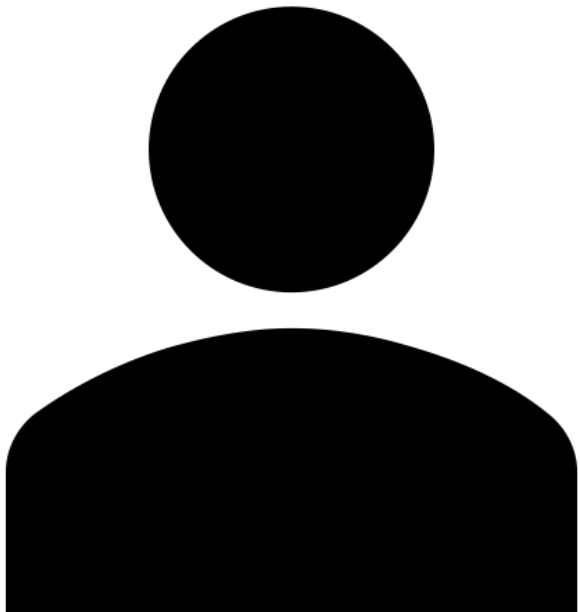
SpringMVC 구조



Server



유저





대뜸

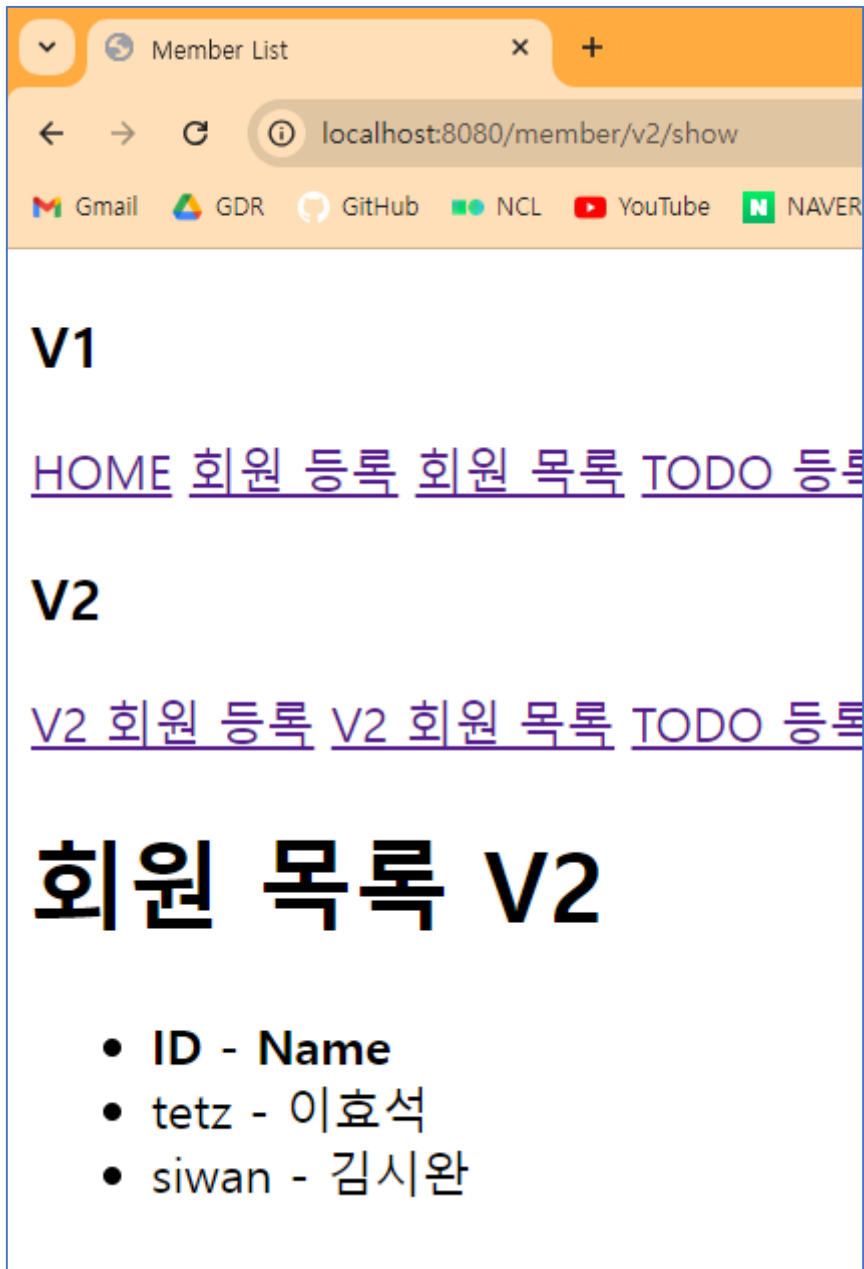
모델 적용하기!

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!!



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!



Model Data

실습, Todo 에 Model 적용!



- Todo V2 버전에 model 을 적용하여 model 로 데이터를 전달 할 수 있도록 수정해 주세요!