Spring @MVC

기본기 다지기

✔ 목차

- 스프링 @MVC
  : 스프링 MVC
  : 스프링 @MVC
  : 스프링 @MVC에서 중요한 세가지
    * URL 연결하기
    * HttpServletRequest 다루기
    * 응답하기

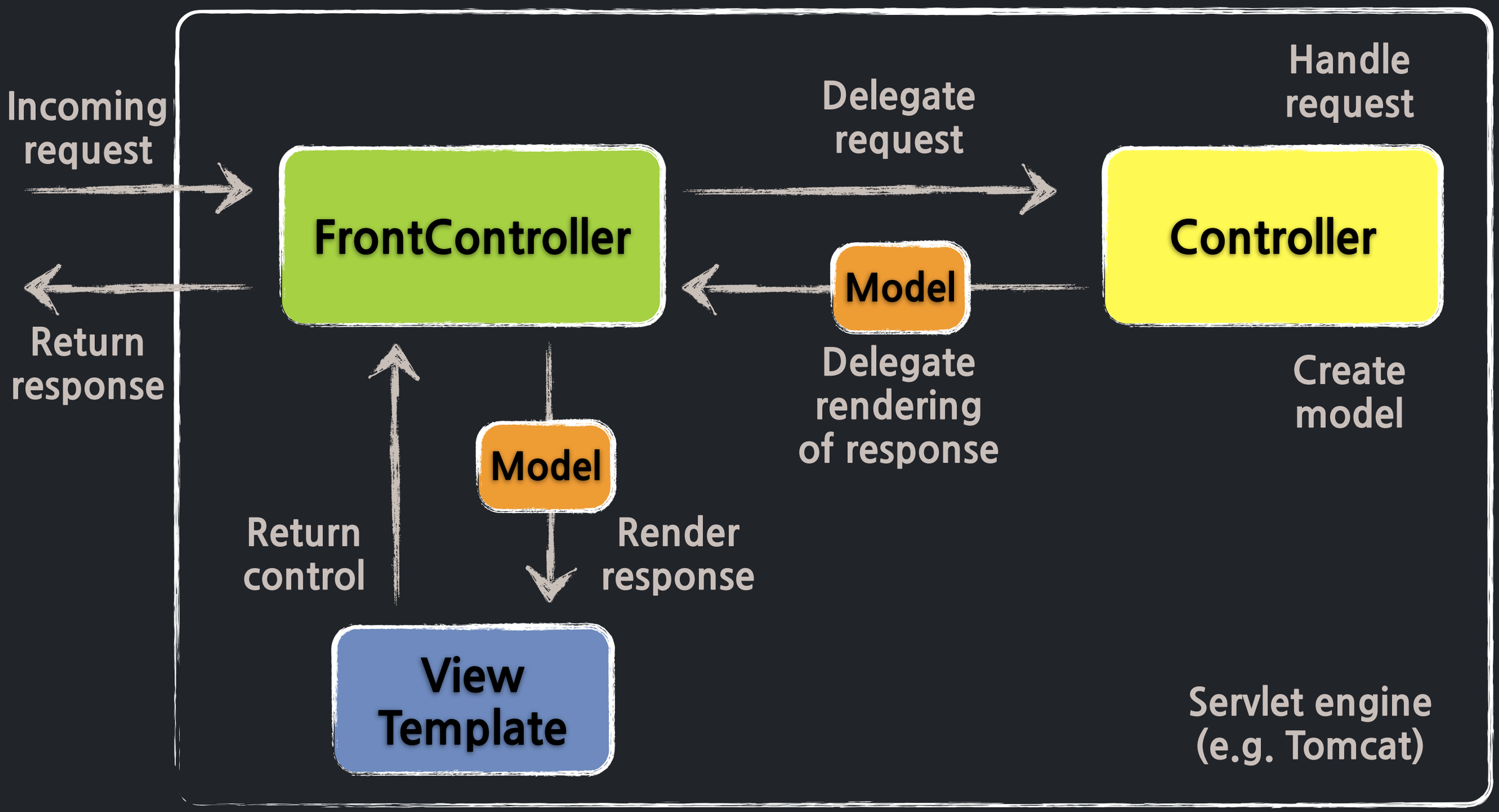
- 스프링 MVC 해부학
  : 스프링 MVC의 구조
  : @MVC의 기본 기능을 사용한 웹 페이지 데모 시연 및 코드 리뷰
  : Client(Request) -> Spring(?) -> Client(Response)까지 해부하기

스프링 @MVC

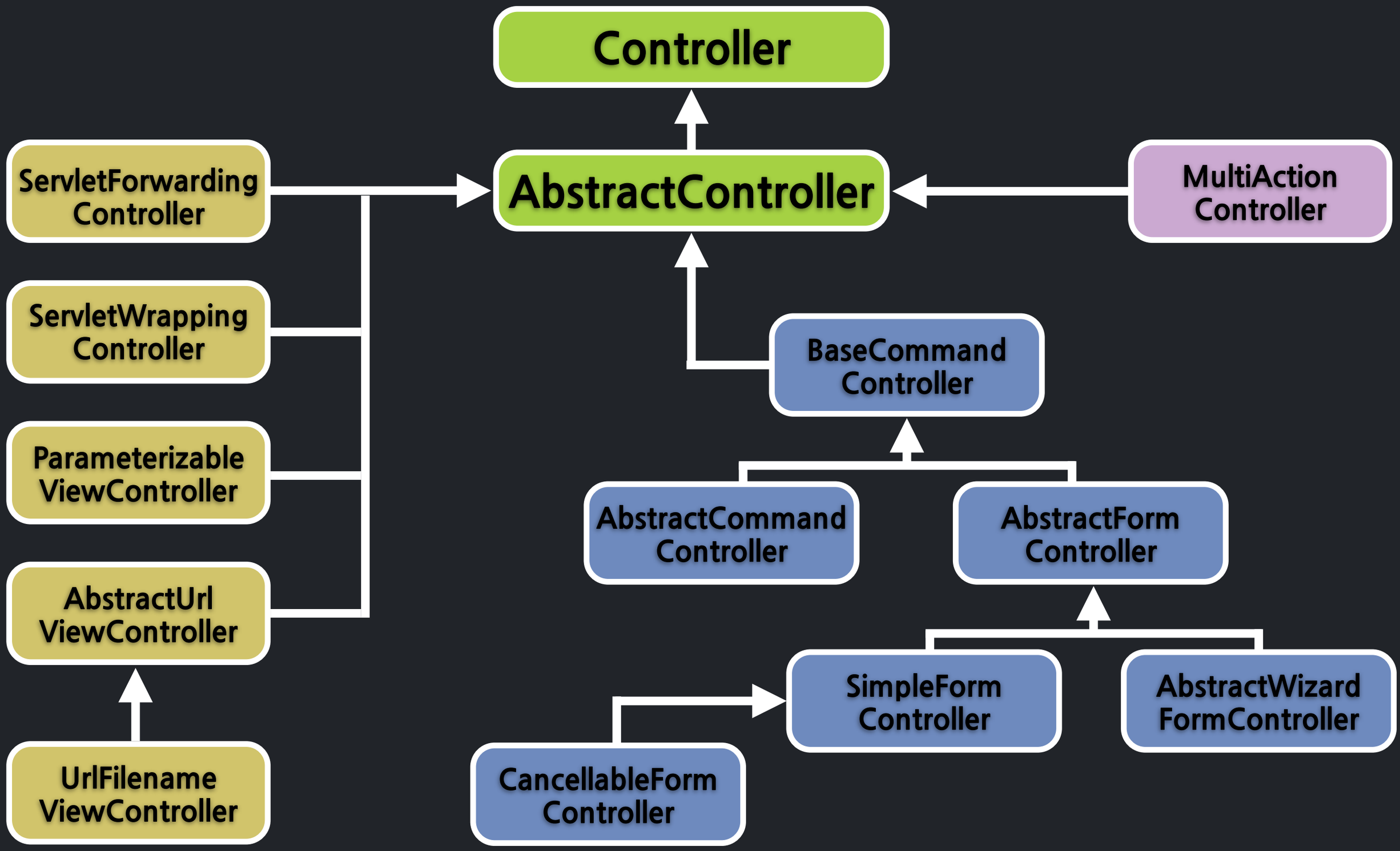
✔ 스프링 MVC

- SpringFramework에서 제공하는 웹 프레임워크
- 스프리 서블릿 또는 스프링 MVC라고 지칭
- 프론트 컨트롤러 패턴과 함께 MVC 아키텍처를 사용
- 풍부한 기능을 제공
- 견고한 웹 어플리케이션을 만드는데 도움
- Interface 기반으로 다양한 구현체 제공
- 다양한 확장 지점을 제공

✔ request processing workflow in Spring MVC



✔ Spring MVC Controller Hierarchy



✔ 스프링 @MVC

- Spring 2.5 부터 도입
- 자바 언어의 애노테이션(Annotation)을 사용
- 자바 언어의 리플렉션(Reflection) API를 사용
- CoC 스타일의 관례를 이용한 프로그래밍 모델
- Controller Interface 기반 클래스를 대체
- 극한의 유연성과 확장성을 제공
- Spring 3.x 지속적으로 강화되는 기능

✔ 애노테이션(Annotation)

- JAVA5(1.5)부터 추가된 새로운 스펙
- Class, Method, Field에 대해 보다 상세하게 설명하는 코드
- @{AnnotationName} 으로 표기

```
public class Annotation implements Ordered {
    @Override
    public int getOrder() {
        return 0;
    }
}
```

```
public class Date {
    @Deprecated
    public Date(int year, int month, int date) {
        // init code
    }
}
```

✔ 리플렉션 API(Reflection API)

런타임에서 Class나 Instance에서 메소드나 필드 등의 정보를 추출 할 수 있고, 객체의 생성이나 메소드의 호출 또는 동작 변경까지 가능한 강력한 기술

```
Class<?> clazz = Board.class;

out.println("class: " + clazz.getName());

for(Field field : clazz.getDeclaredFields())
    out.println("filed: " + field.getName());

for(Method method : clazz.getDeclaredMethods())
    out.println("method: " + method.getName());
```

↓

```
class: egovframe.model.Board    method: getName
filed: name                     method: getDescription
filed: description              method: getPosts
filed: posts                    ...
```

✔ 애노테이션 + 리플렉션 API

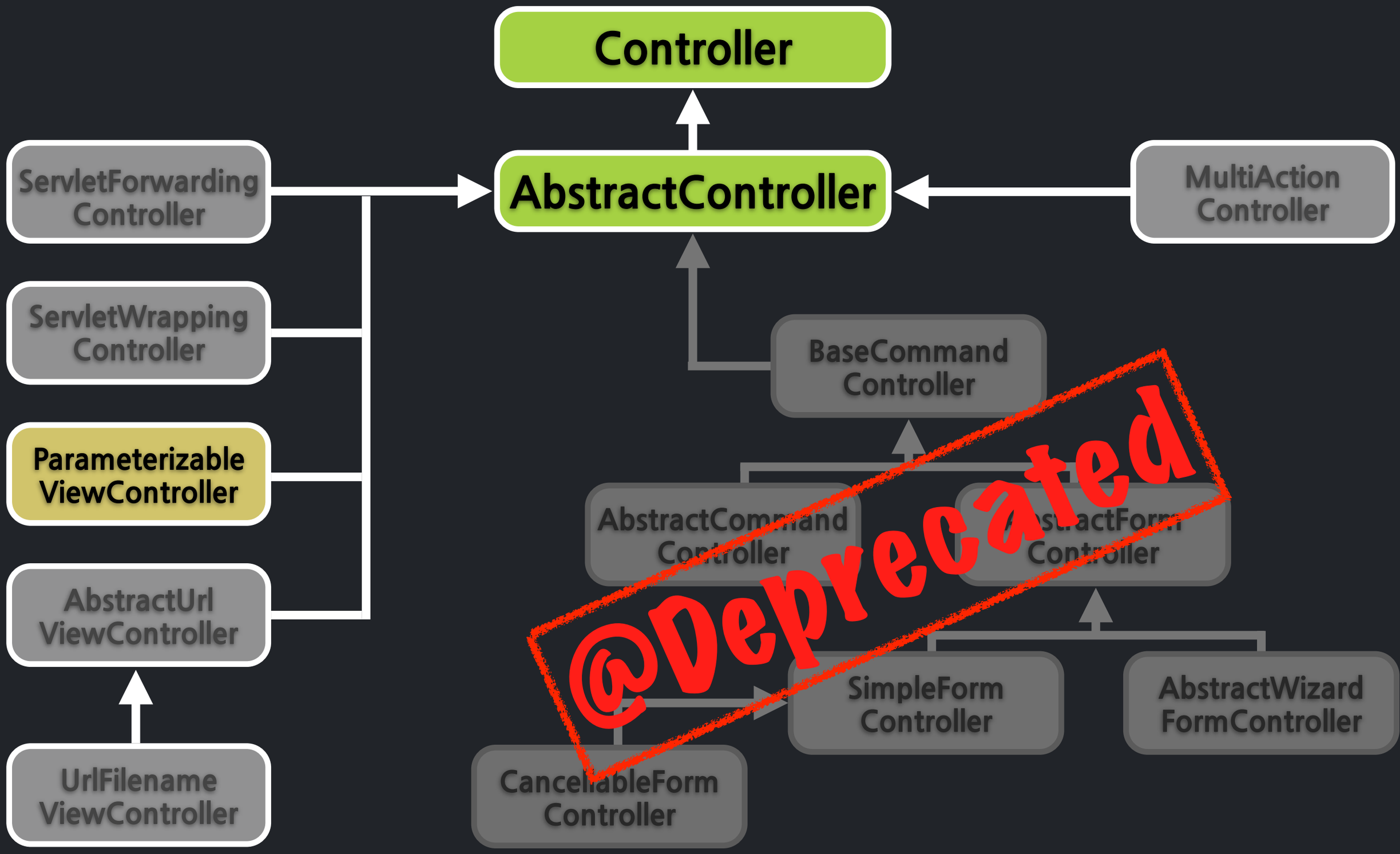
```
@Controller("egovCtrl")
class EgovController { + }
```

```
Class<?> clazz = EgovController.class;
String value = clazz.getAnnotation(Controller.class)
                    .value();
out.println("Controller Name: " + value);
```

↓

```
Controller Name: egovCtrl
```

✔ Controller Interface 기반 클래스를 대체



✔ Spring MVC 3.x

- Explicit support for Servlet 3.0
- Java Config
- Consumes/Produces
- URI Variables
- Redirect & Flash Attributes
- Multipart Request Support
- UriComponentsBuilder
- New @MVC Infrastructure
- Asynchronous request processing
- Spring MVC Testing Support

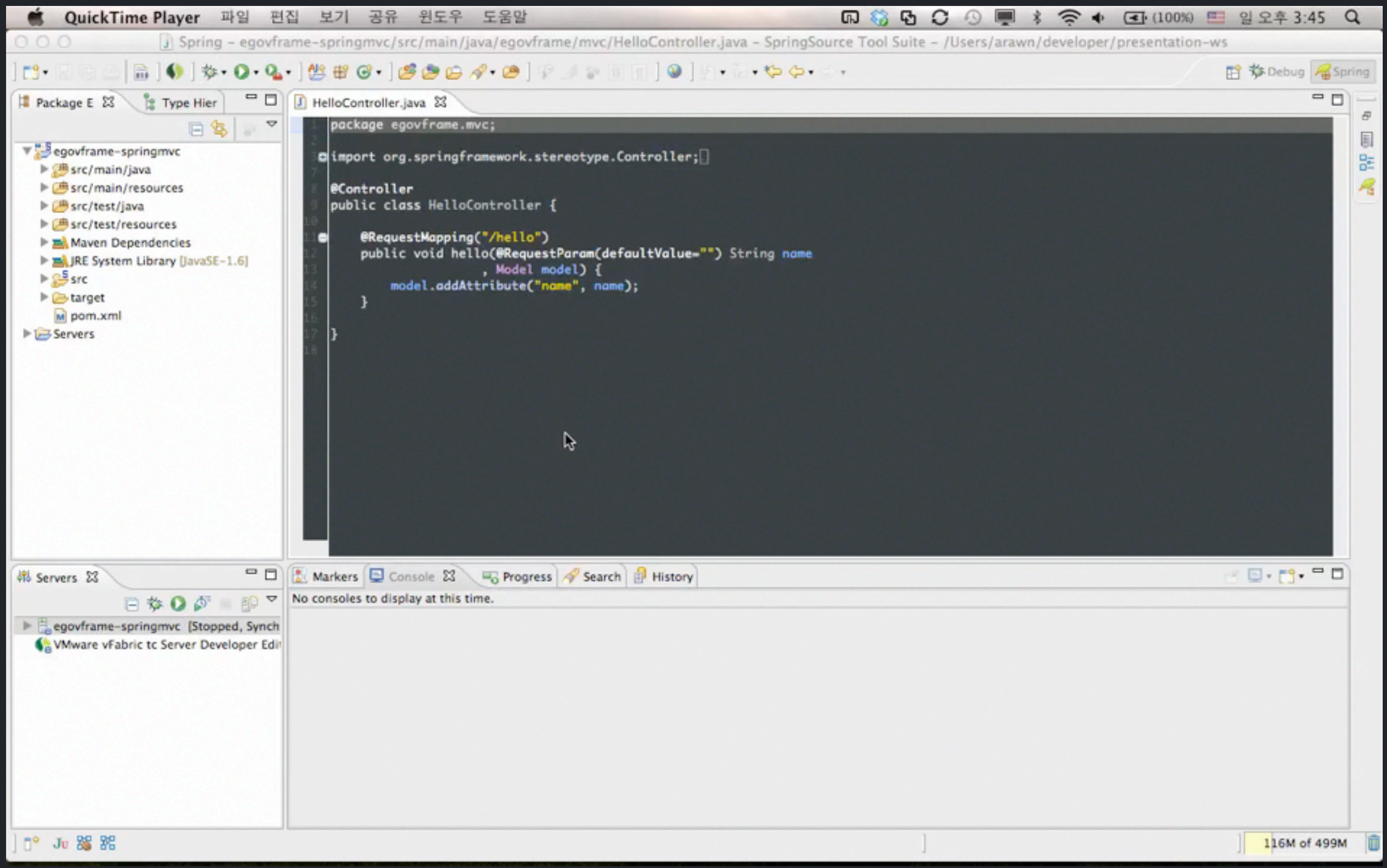
✔ Hello Spring @MVC - JSP

```
<html>
<head><title>Spring @MVC</title></head>
<body>
    <c:if test="${not empty name}">
        <h1>Hello ${name}!</h1>
    </c:if>
    <c:if test="${empty name}">
        <form>
            이름: <input type='text' name='name' />
            <button type='submit'>전송</button>
        </form>
    </c:if>
</body>
</html>
```

✔ Hello Spring @MVC - JAVA

```java
@Controller
public class HelloController {

    @RequestMapping("/hello")
    public void hello(@RequestParam(required=false) String name
                      , Model model) {
        model.addAttribute("name", name);
    }

}
```

✔ Hello Spring @MVC



✔ 스프링 @MVC에서 중요한 세가지

■ URL 연결하기

- @RequestMapping
- Handler

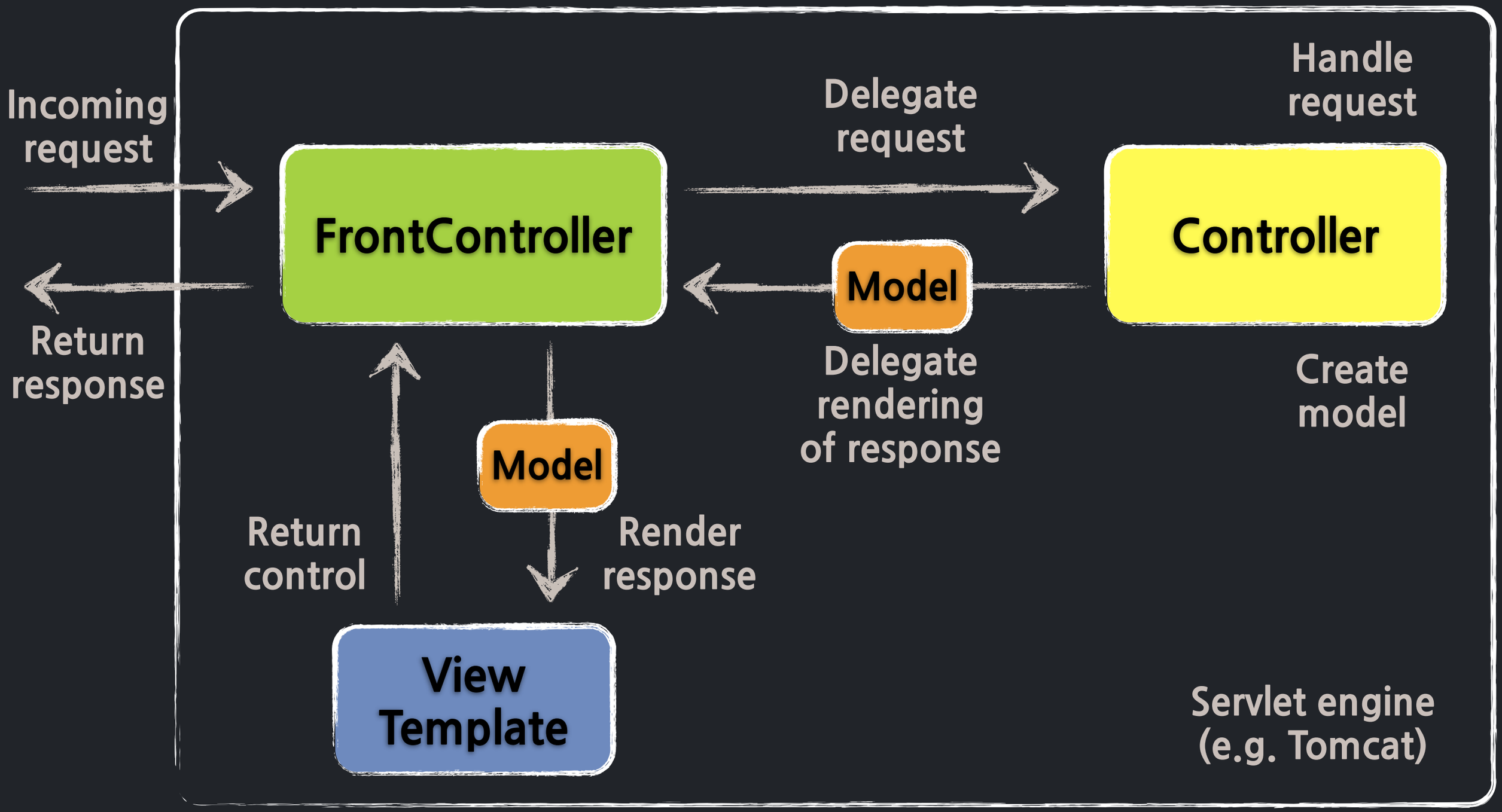
■ HttpServletRequest 다루기

- HandlerMethodArgumentResolver
- WebArgumentResolver
- DataBinding

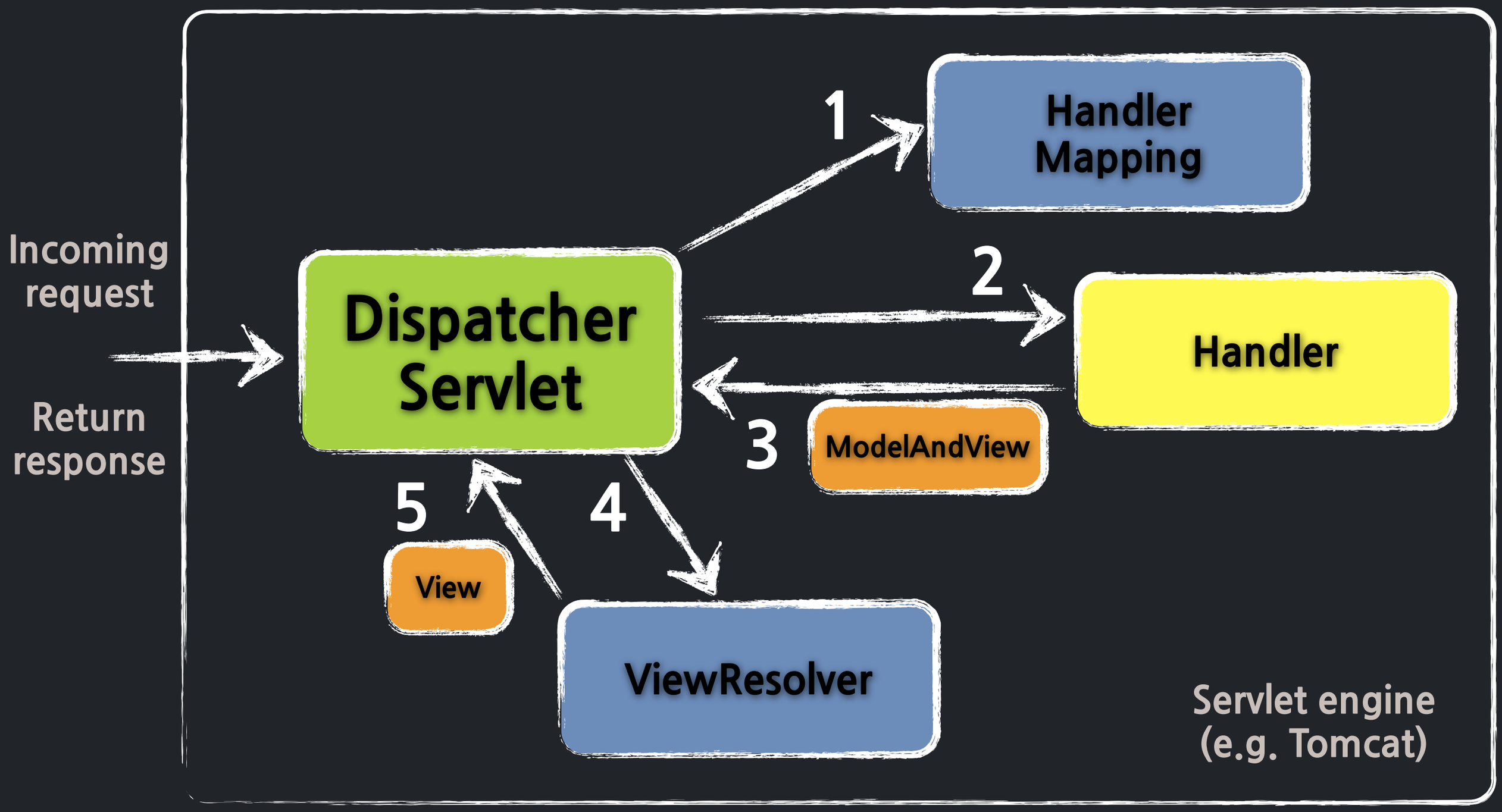
■ 응답하기

- HandlerMethodReturnValueHandler
- ModelAndViewResolver
- View & ViewResolver

✔ 스프링 @MVC에서 중요한 세가지

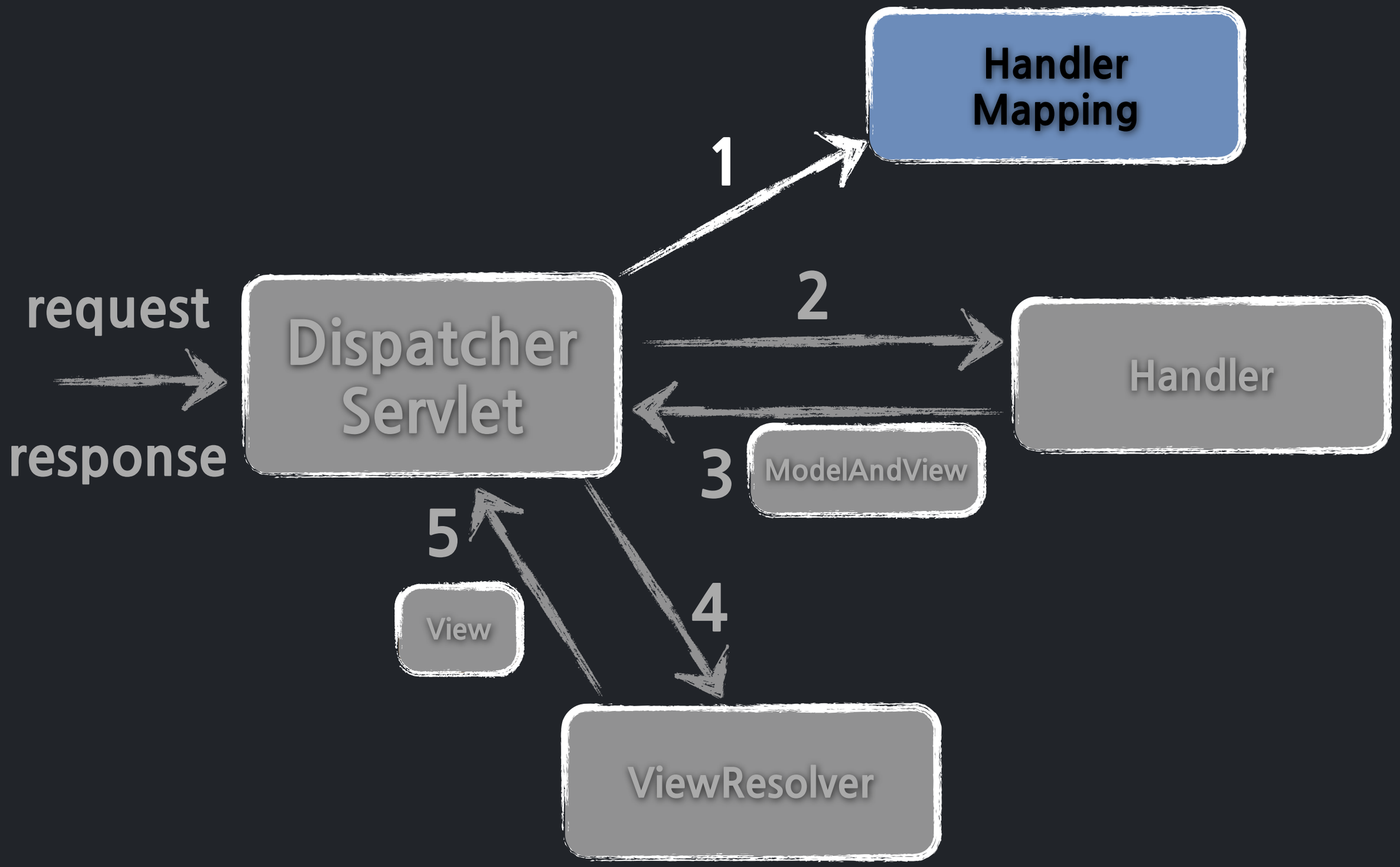


✔ 스프링 @MVC에서 중요한 세가지



URL 연결하기

✔ URL 연결하기


Handler Mapping
1
request
response
Dispatcher Servlet
2
Handler
3
ModelAndView
5
View
4
ViewResolver

✔ Handler

```
@Controller
public class MemberController {

    @RequestMapping("/member/list")
    public String list() { + }

    @RequestMapping(value="/member/view")
    public String view() { + }

}
```

✔ @RequestMapping - By Path (1)

```
@Controller
public class MemberController {

    @RequestMapping("/member/list")
    public String list() { + }

    @RequestMapping("/member/view")
    public String view() { + }

}
```

http://open.egovframe.co.kr/member/view

✔ @RequestMapping - By Path (2)

```
@Controller
@RequestMapping("/member/*")
public class MemberController {

    @RequestMapping
    public String list() { + }

    @RequestMapping
    public String view() { + }

}
```

http://open.egovframe.co.kr/member/view

✔ @RequestMapping - By HTTP method

```
@Controller
public class MemberController {

  @RequestMapping(method=RequestMethod.GET)
  public String list() { + }

  @RequestMapping(method=RequestMethod.POST)
  public String add() { + }

  @RequestMapping(method=RequestMethod.PUT)
  public String edit() { + }

  @RequestMapping(method=RequestMethod.DELETE)
  public String delete() { + }

}
```

✔ @RequestMapping의 전략들

■ By presence of query parameter

- @RequestMapping(value="path", params="name")
- http://open.egovframe.go.kr/path?name=springmvc

■ By presence of request header

- @RequestMapping(value="path", header="Accept=application/json")
- $.getJSON('http://open.egovframe.go.kr/path', function(data){ ... })

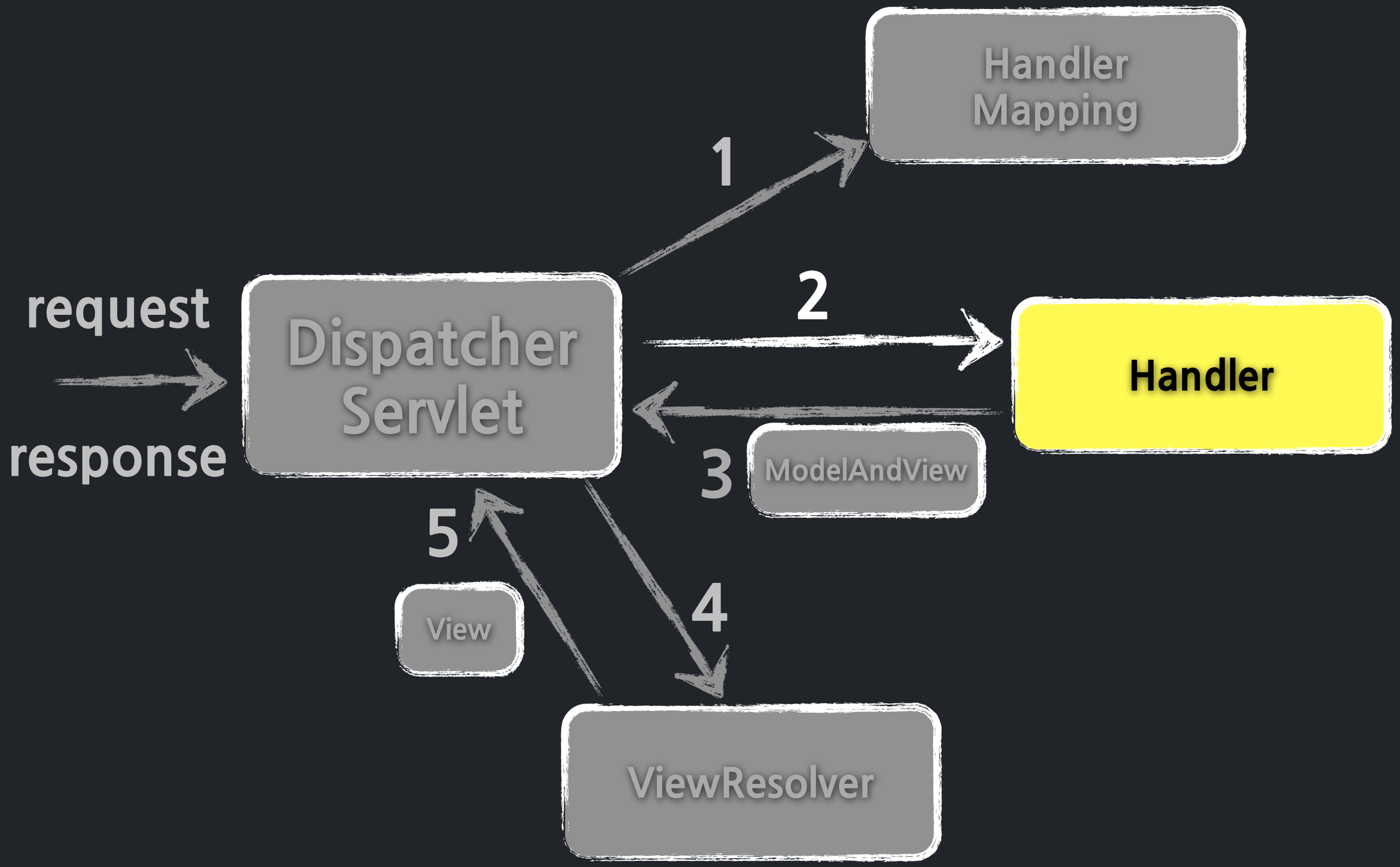
■ By Consumes/Produces

- @RequestMapping(value="path", consumes="application/json")
- @RequestMapping(value="path", produces="application/json")

HttpServletRequest

다루기

✔ HttpServletRequest 다루기



✔ MethodParameter

```
@Controller
@RequestMapping("/member/*")
public class MemberController {

    @RequestMapping
    public String list(HttpServletRequest request) {
        MemberCondition condition = new MemberCondition();
        condition.setName(request.getParameter("name"));
        condition.setNickName(request.getParameter("nickName"));

        return "member-list";
    }

}
```

✔ MethodParameter로 Request 데이터 받기

■ A query parameter value
- @RequestParam(“name”)

■ A group of query parameter values
- A custom JavaBean with a getName()/setName() pair for each parameter

■ A path element value
- @PathVariable(“var”)

■ A request header value
- @RequestHeader(“name”)

■ A cookie value
- @CookieValue(“name”)

■ The request body
- @RequestBody

■ The request body and any request header
- HttpEntity<T>

✔ @ReuqestParam

request.getParameter("name")

```
@RequestMapping
void action(@RequestParam("name") String name) {}
```

request.getParameter("age")

```
@RequestMapping
void action(@RequestParam Integer age) {}
```

✔ Custom JavaBean

```
MemberCondition condition = new MemberCondition();
condition.setName(request.getParameter("name"));
condition.setNickName(request.getParameter("nickName"));
```

```
@RequestMapping
public String list(MemberCondition condition) {
   return "member-list";
}
```

✔ @PathVariable

```
../member/arawn > name = "arawn";
```

```
@RequestMapping("/member/{memberName}")
public void action(
  @PathVariable("memberName") String name) {}
```

```
@RequestMapping("/member/{memberId}")
public void action(
  @PathVariable Integer memberId) { }
```

✔ @RequestHeader, @CookieValue

```
@RequestMapping
public void header(
 @RequestHeader("user-agent") String userAgent) {}
```

```
@RequestMapping
public void cookie(
 @CookieValue("memberId") String memberId) { }
```

✔ Injecting standard objects

- HttpServletRequest / HttpServletResponse
- HttpSession
- java.util.Locale
- java.io.InputStream / java.io.OutputStream
- java.io.Reader / java.io.Writer
- java.security.Principal
- org.springframework.ui.Model
- org.springframework.validation.Errors
- org.springframework.web.bind.support.SessionStatus
- org.springframework.http.HttpEntity
- org.springframework.web.context.request.WebRequest
- org.springframework.web.context.request.NativeWebRequest

✔ Injecting custom objects (1)

```java
@RequestMapping("/user-info")
public String userInfo(HttpSession httpSession) {
  Object user = httpSession.getAttribute(SESSION_USER_KEY);
  if(user == null)
    return "redirect:/notice/post";

  return "user-info";
}
```

```java
@RequestMapping("/user-info")
public String userInfo(User user) {
  if(user.isAnonymous())
    return "redirect:/login";

  return "user-info";
}
```

✔ Injecting custom objects (2)

MethodParameter Resolver Mechanism

WebArgumentResolver
(Spring 3.0 이하)

HandlerMethodArgumentResolver
(Spring 3.1 이상)

1) 지원되는 타입 발견

```
String userInfo(User user) { + }
```

2) Argument 생성 후 반환

```
(User) httpSession.getAttribute(SESSION_USER_KEY);
```

✔ Injecting custom objects (3)

```
public class UserMethodArgumentResolver implements HandlerMethodArgumentResolver {

  @Override
  public boolean supportsParameter(MethodParameter parameter) {
    return User.class.isAssignableFrom(parameter.getParameterType());
  }

  @Override
  public Object resolveArgument(MethodParameter parameter
                              , ModelAndViewContainer mavContainer
                              , NativeWebRequest webRequest
                              , WebDataBinderFactory binderFactory)
    throws Exception {

    HttpSession session =  webRequest.getNativeRequest(HttpServletRequest.class)
                                     .getSession();
    Object user = session.getAttribute(UserController.SESSION_USER_KEY);

    return user != null ? user : User.anonymous();
  }

}
```

✔ Injecting custom objects (4)

```
<mvc:annotation-driven conversion-service="conversionService">
    <mvc:argument-resolvers>
        <bean class="egovframe...UserMethodArgumentResolver"/>
    </mvc:argument-resolvers>
</mvc:annotation-driven>
```

✔ 데이터 타입 불일치 문제

```
@RequestMapping
void action(@RequestParam Integer age) {}
```

Integer != String

```
String age = request.getParameter("age");
```

```
public interface HttpServletRequest extends ServletRequest {

  String getParameter(String name);

  String[] getParameterValues(String name);

  ...
}
```

✔ Type Converson (1)

Type Conversion Mechanism

PropertyEditor
(Spring 2.5 이하)

ConversionService
(Spring 3.0 이상)

1) 타입 불일치 발견

```
request.getParameter("age") != Integer
```

2) 타입 변환 후 주입

```
@RequestMapping
void action(@RequestParam Integer age) {}
```

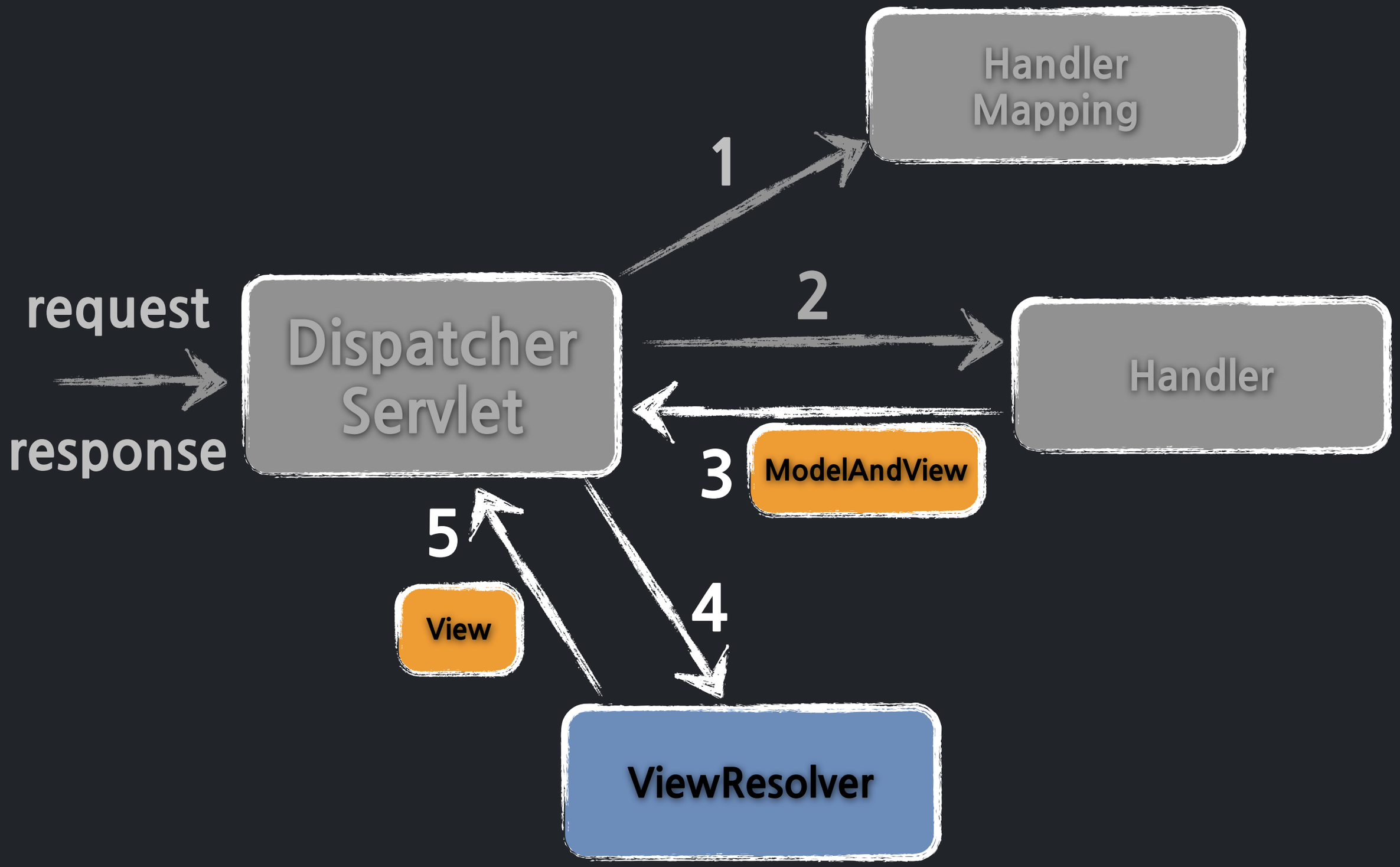
✔ Type Converson (2)

```
class StringToNumber<T extends Number> implements Converter<String, T> {

  private final Class<T> targetType;

  public StringToNumber(Class<T> targetType) {
    this.targetType = targetType;
  }
  public T convert(String source) {
    if (source.length() == 0) {
      return null;
    }
    return NumberUtils.parseNumber(source, this.targetType);
  }
}
```

```
<mvc:annotation-driven conversion-service="conversionService"/>
<bean id="conversionService"
   class="org.springframework...FormattingConversionServiceFactoryBean">
  <property name="converters">
    <list>
      <bean class="..."/>
    </list>
  </property>
</bean>
```

응답하기

✔ 응답하기


Handler Mapping
1
request
response
Dispatcher Servlet
2
Handler
3
ModelAndView
5
View
4
ViewResolver

✔ Handler Return Value = ModelAndView (1)

```
@Controller
@RequestMapping("/member/*")
public class MemberController {

    @RequestMapping
    public String list(MemberCondition mc, Model model) {
        model.addAttribute("members", dao.findAll(mc));

        return "member/list";
    }

}
```

```
return new ModelAndView("/member/list")
            .addObject("members", dao.findAll(mc));
```

✔ Handler Return Value = ModelAndView (2)

ModelAndView Resolver Mechanism

ServletHandlerMethodInvoker & ModelAndViewResolver
(Spring 3.0 이하)

HandlerMethodReturnValueResolver
(Spring 3.1 이상)

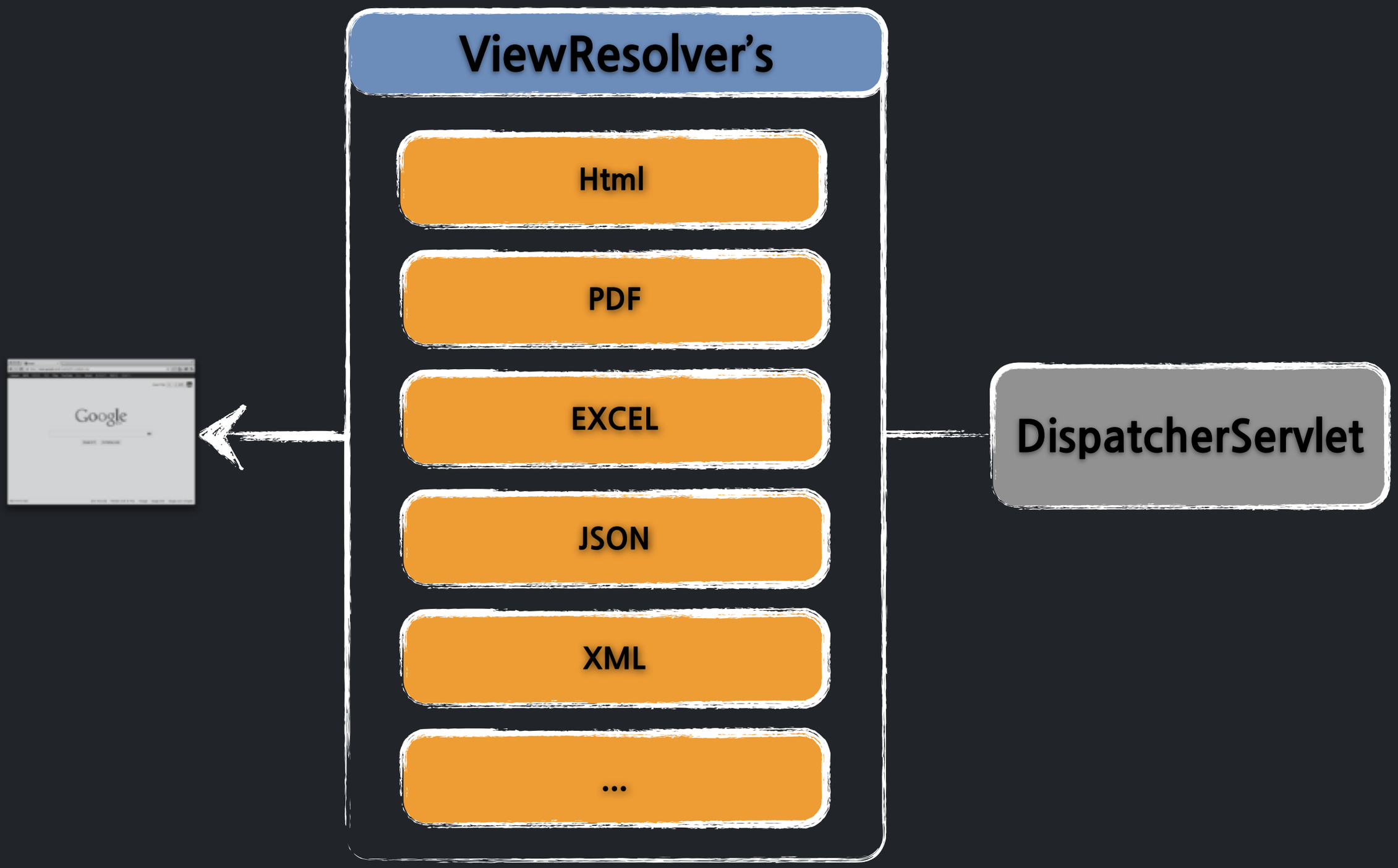
1) Handler 호출 결과 전달

```
return "member/list";
```

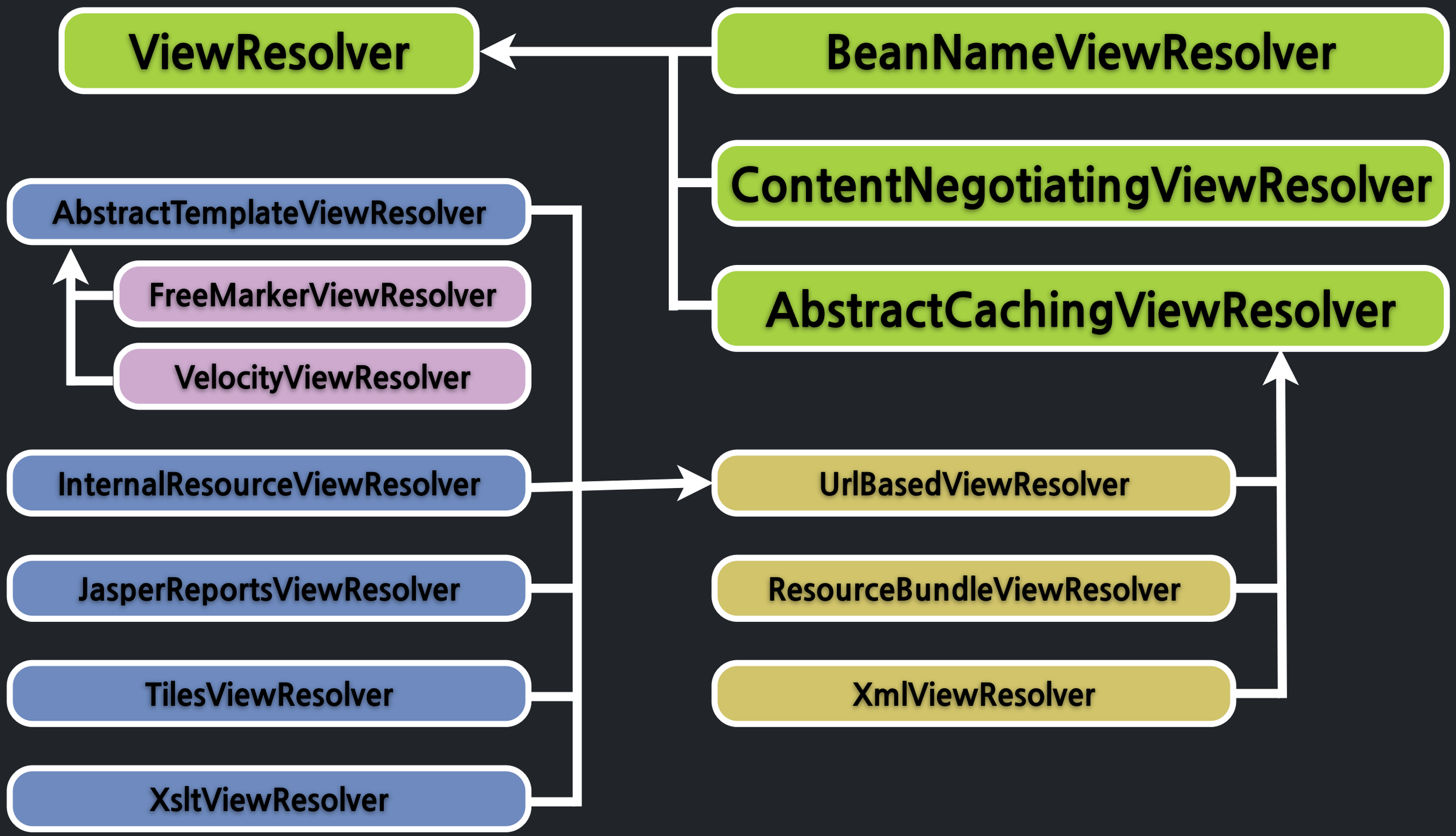
2) ModelAndView 객체 생성 후 반환

```
new ModelAndView("/member/list");
```

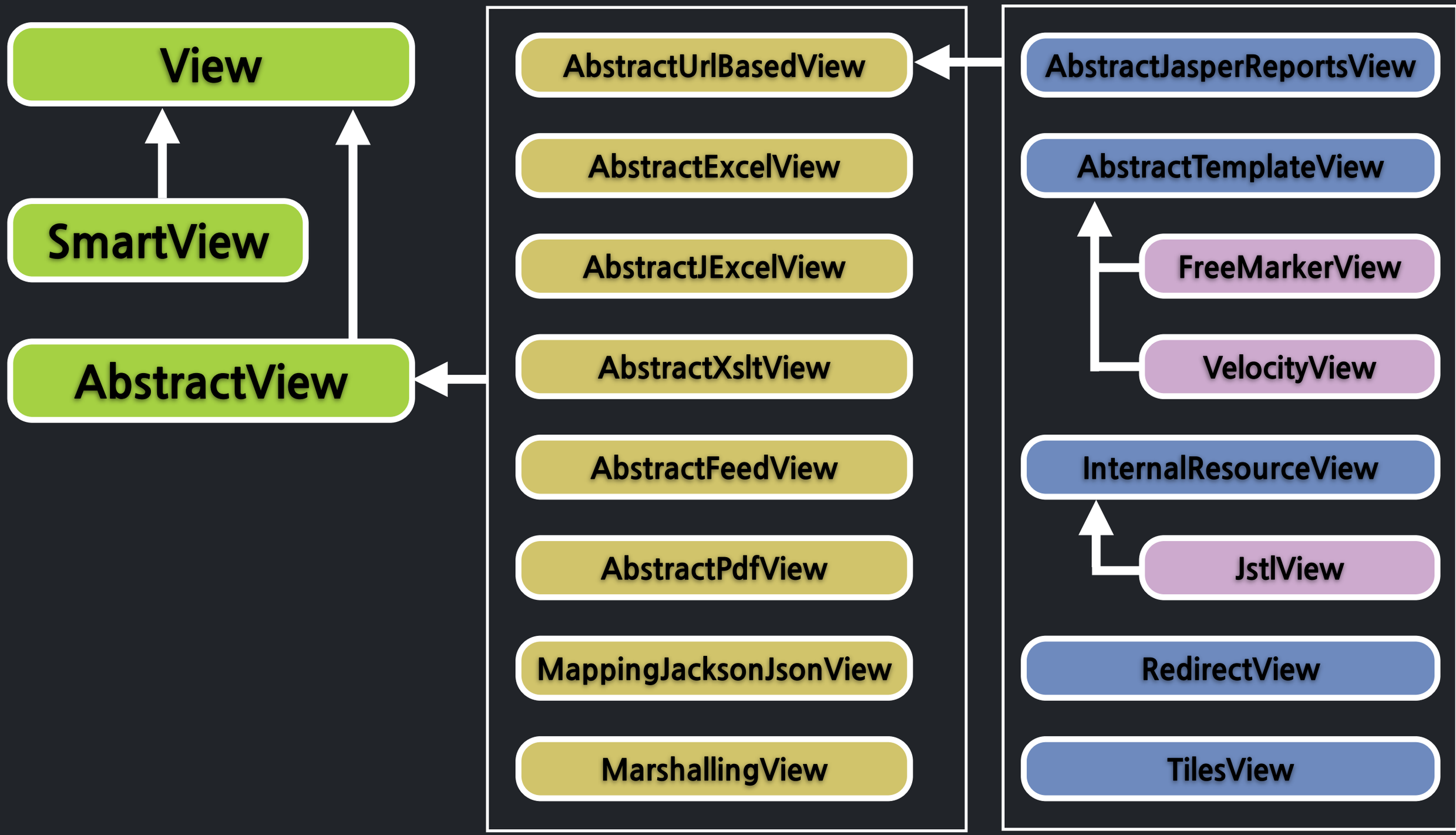
✔ 다양한 ViewResolver & View



✔ ViewResolver Hierarchy



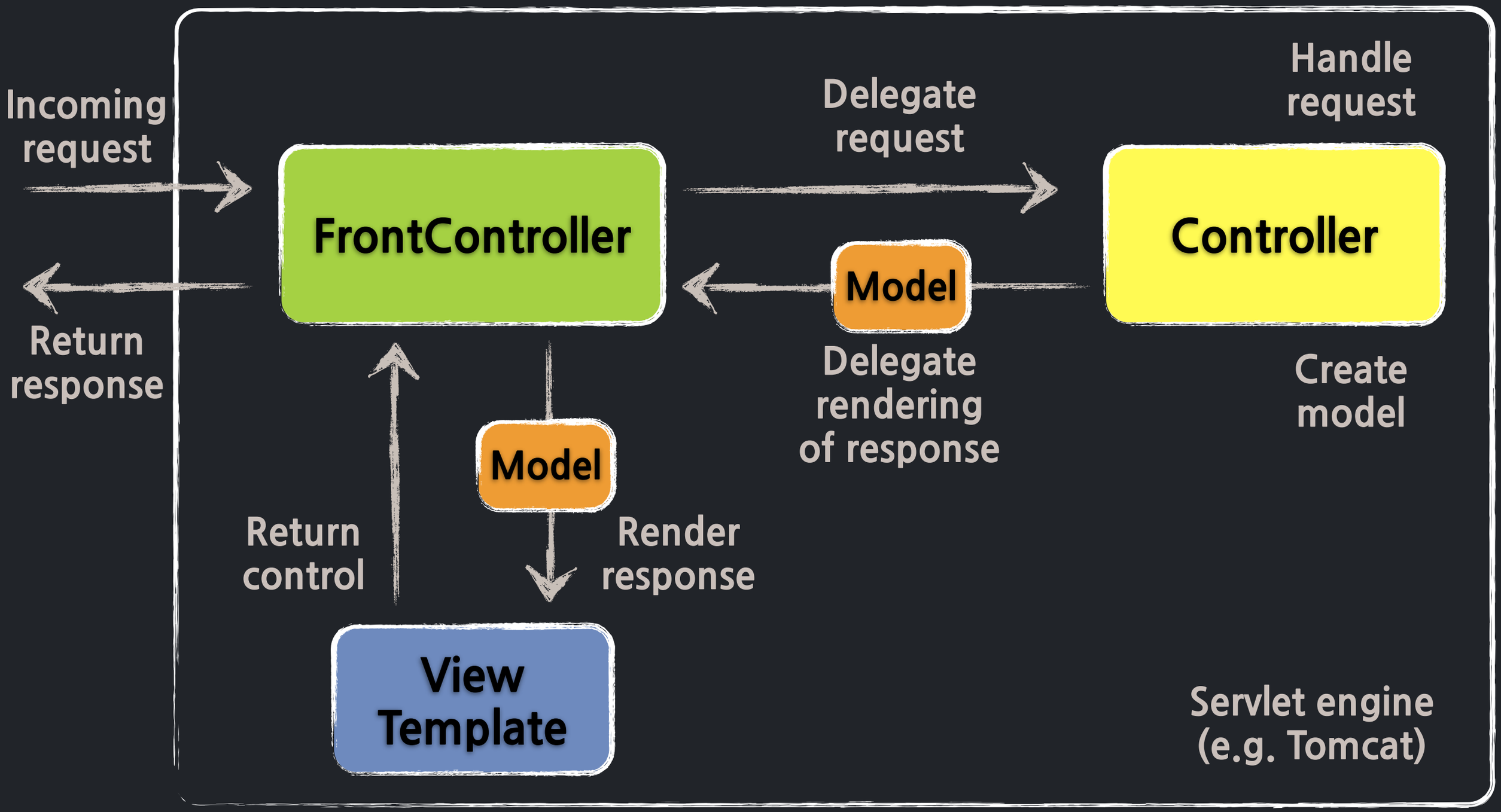
✔ View Hierarchy



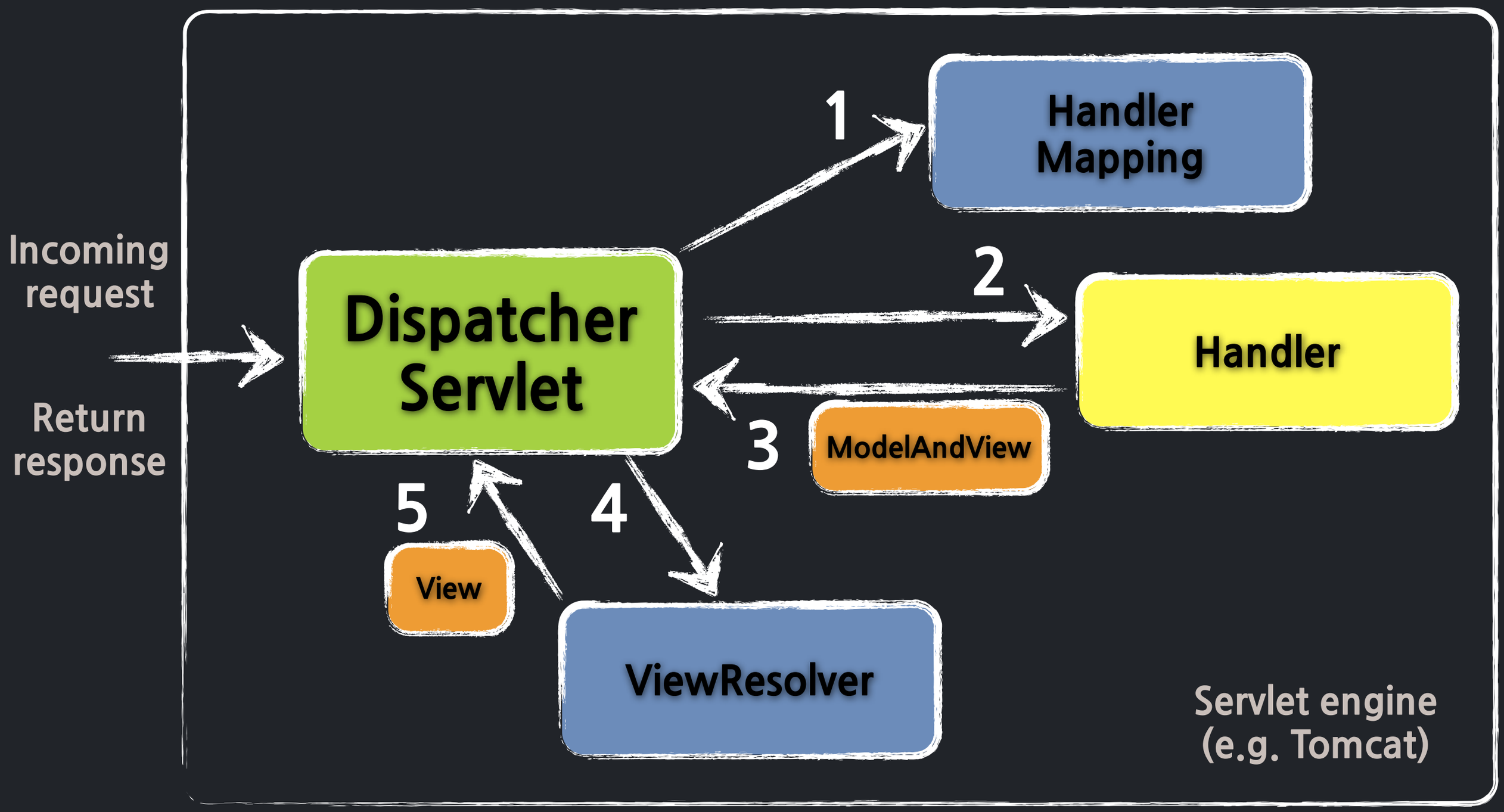
스프링 MVC

해부학

✔ request processing workflow in Spring MVC



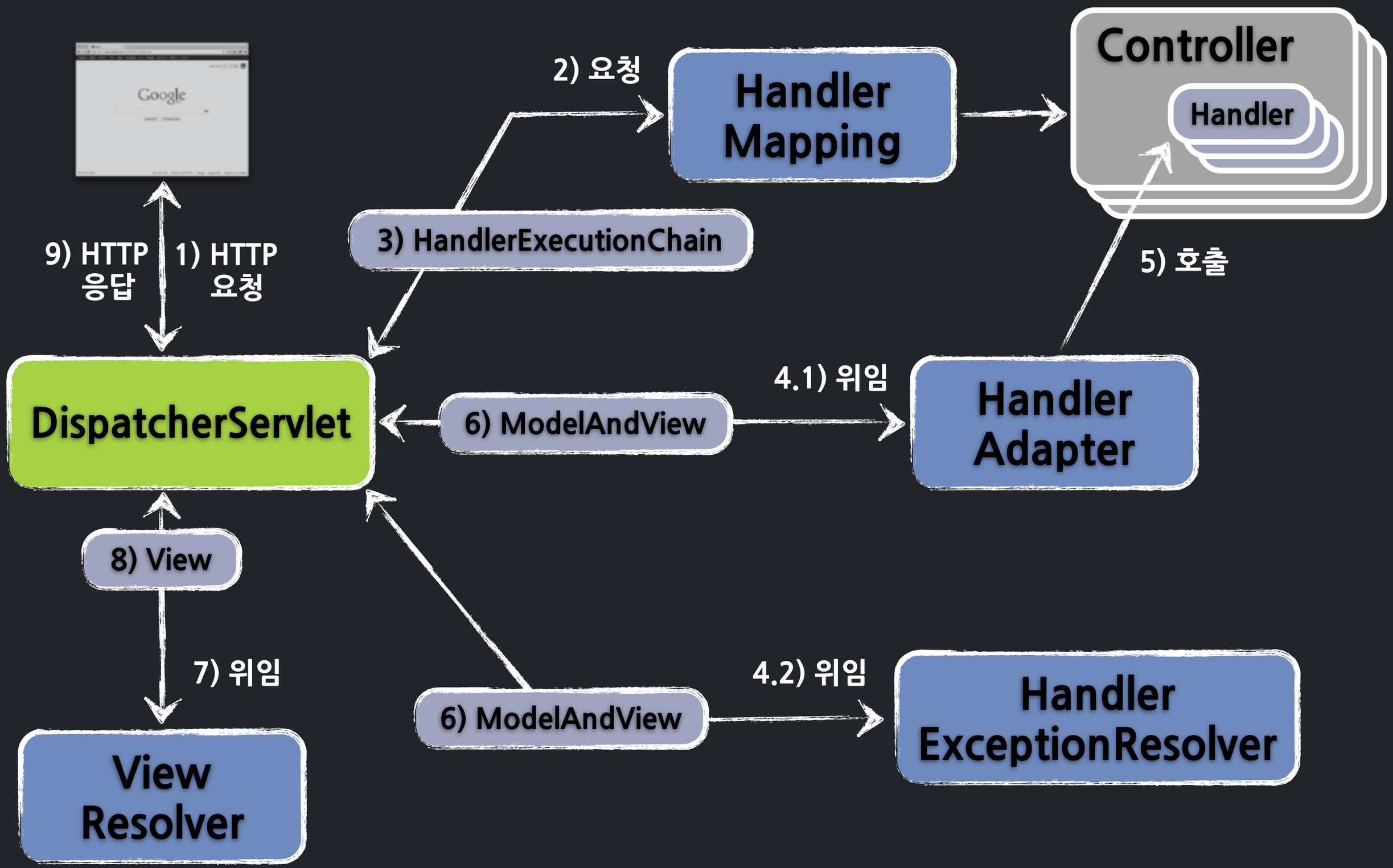
✔ request processing workflow in Spring MVC



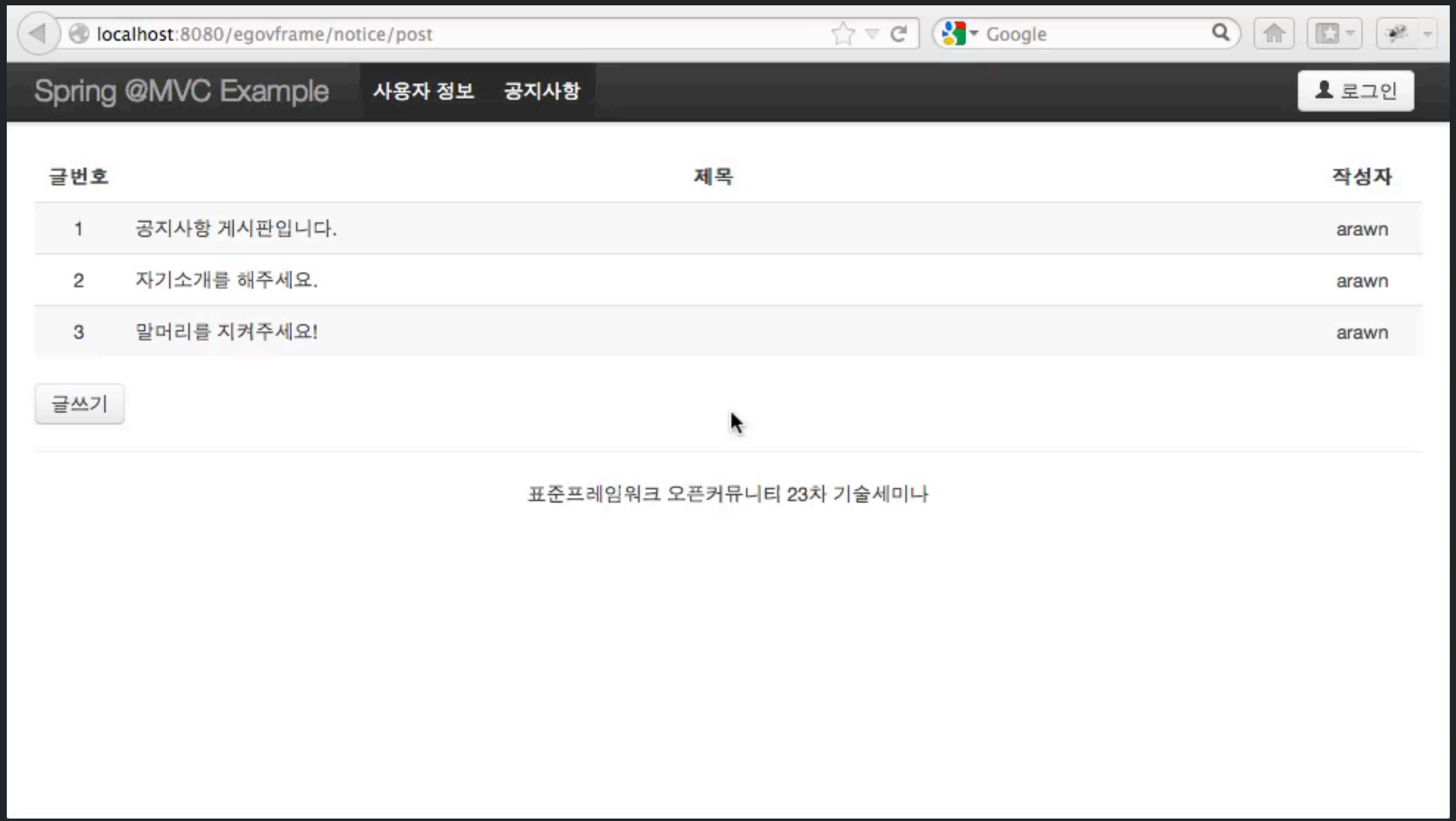
✔ DispatcherServlet이 가지고 있는 것

- WebApplicationContext
- MultipartResolver
- LocaleResolver
- ThemeResolver
- RequestToViewNameTranslator
- FlashMapManager
- HandlerMapping
- HandlerAdapter
- ViewResolver
- HandlerExceptionResolver

✔ request processing workflow in Spring MVC



✔ Spring @MVC를 사용한 웹 애플리케이션



✔ WebApplicatoinContext (servlet-context.xml)

```
<context:component-scan base-package="egovframe"/>

<mvc:annotation-driven conversion-service="conversionService"/>

<mvc:resources mapping="/resources/**" location="/resources/"/>

<bean class="org.springframework...InternalResourceViewResolver"
      p:prefix="/WEB-INF/views/"
      p:suffix=".jsp"/>

<bean id="conversionService"
      class="org.springframework...FormattingConversionServiceFactoryBean">
    <property name="converters">
        <list>
            <bean class="egovframe.mvc.BoardConverter"/>
        </list>
    </property>
</bean>
```

✔ PostController.java

```
@Controller
@RequestMapping("/{board}/post")
public class PostController {

    @RequestMapping(method=RequestMethod.GET)
    public String list(@PathVariable Board board
                       , Model model) {
        model.addAttribute("posts", board.getPosts());
        return "post/list";
    }

    @RequestMapping(value="{postNo}", method=RequestMethod.GET)
    public String view(@PathVariable Board board
                       , @PathVariable Long postNo
                       , Model model) {
        model.addAttribute(board.findPost(postNo));
        return "post/view";
    }

    ...
```

✔ list.jsp

```
<%@ taglib prefix="mvc"    tagdir="/WEB-INF/tags" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %>
<%@ taglib prefix="c"      uri="http://java.sun.com/jsp/jstl/core" %>
<mvc:layout>
    <table class="table table-striped">
    <thead>
    <tr>
        <th>글번호</th>
        <th>제목</th>
        <th>작성자</th>
    </tr>
    </thead>
    <tbody>
    <c:forEach items="${posts}" var="post">
    <tr>
        <td class="txt_c">${post.no}</td>
        <td>${post.title}</td>
        <td class="txt_c">${post.writer}</td>
    </tr>
    </c:forEach>
    </tbody>
    </table>
    <a href='<spring:url value="/${board.name}/post/new"/>' class='btn'>글쓰기</a>
</mvc:layout>
```

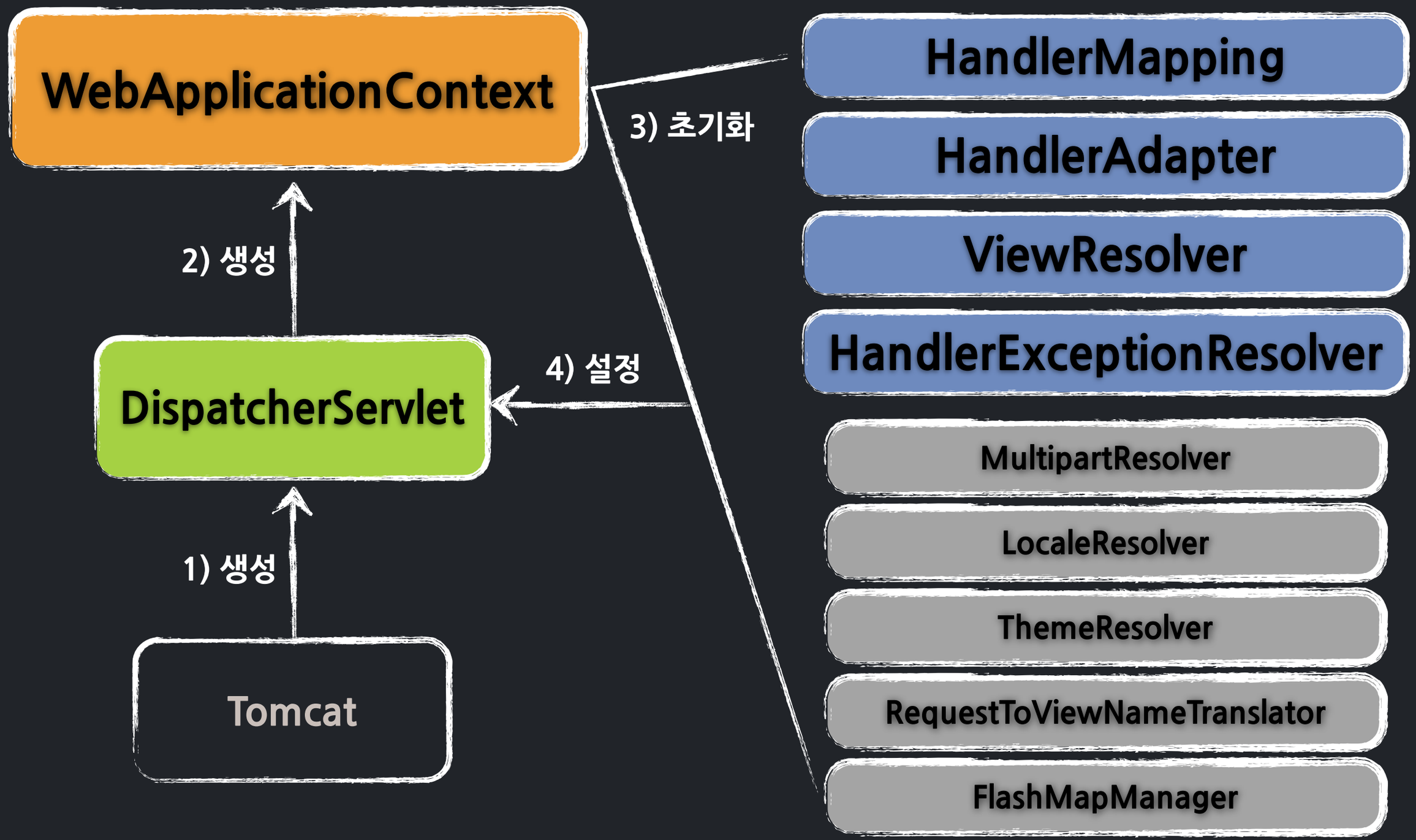
Request
-> Spring @MVC
-> Response

✔ Tomcat Start

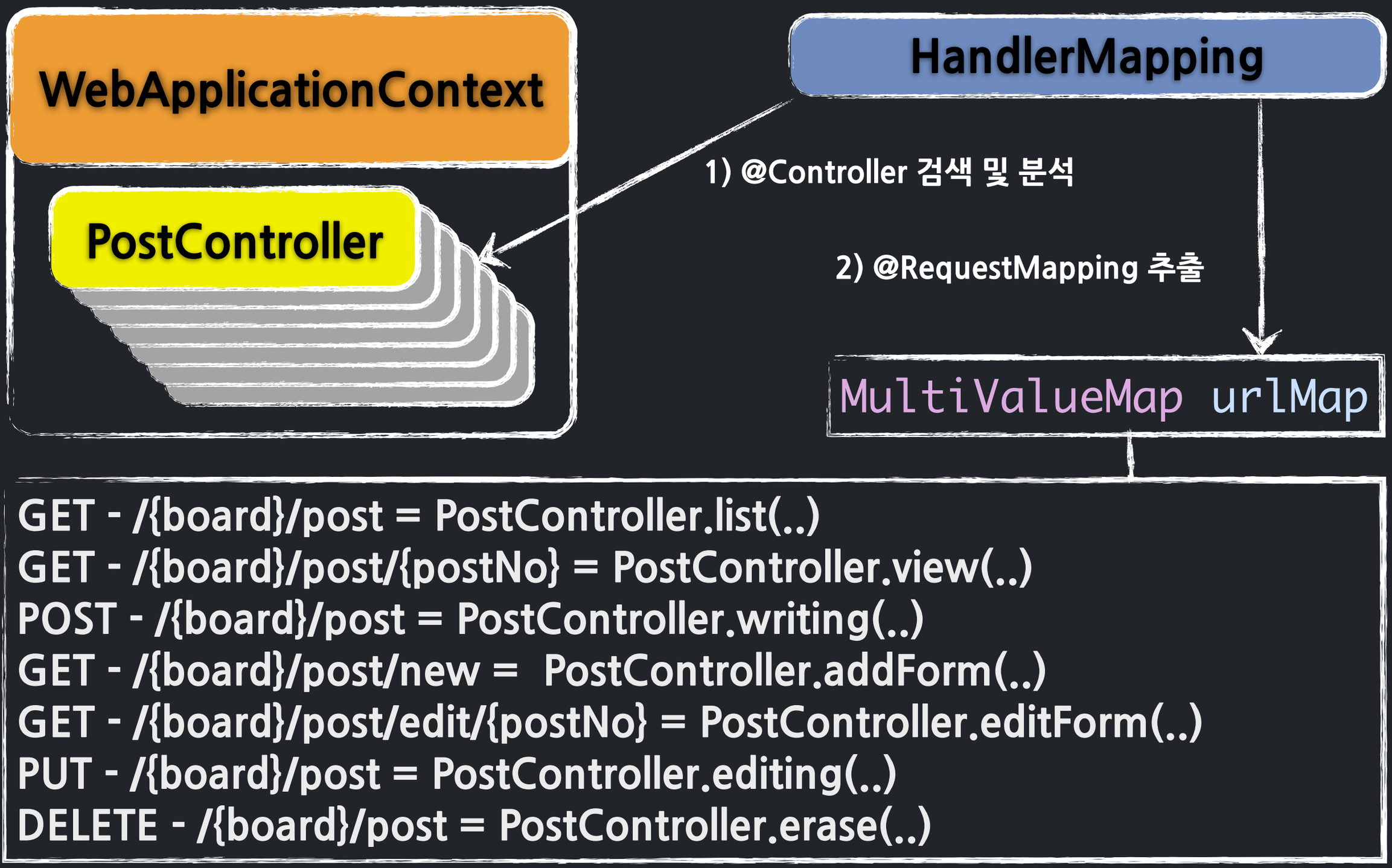
1. dreaminfra@ubuntu: ~/bitnami/apache-tomcat/bin (ssh)

```
dreaminfra@ubuntu:~/bitnami/apache-tomcat/bin$ ls
bootstrap.jar                configtest.sh     setclasspath.sh  tomcat-native.tar.gz
catalina-tasks.xml           cpappend.bat      setenv.sh        tool-wrapper.bat
catalina.bat                 daemon.sh         shutdown.bat     tool-wrapper.sh
catalina.sh                  digest.bat        shutdown.sh      version.bat
commons-daemon-native.tar.gz digest.sh         startup.bat      version.sh
commons-daemon.jar           logs              startup.sh
configtest.bat               setclasspath.bat  tomcat-juli.jar
dreaminfra@ubuntu:~/bitnami/apache-tomcat/bin$ ./catalina.sh start
Using CATALINA_BASE:   /home/dreaminfra/bitnami/apache-tomcat
Using CATALINA_HOME:   /home/dreaminfra/bitnami/apache-tomcat
Using CATALINA_TMPDIR: /home/dreaminfra/bitnami/apache-tomcat/temp
Using JRE_HOME:        /usr/lib/jvm/java-6-sun
Using CLASSPATH:       /home/dreaminfra/bitnami/apache-tomcat/bin/bootstrap.jar:/home/drea
minfra/bitnami/apache-tomcat/bin/tomcat-juli.jar
dreaminfra@ubuntu:~/bitnami/apache-tomcat/bin$
```

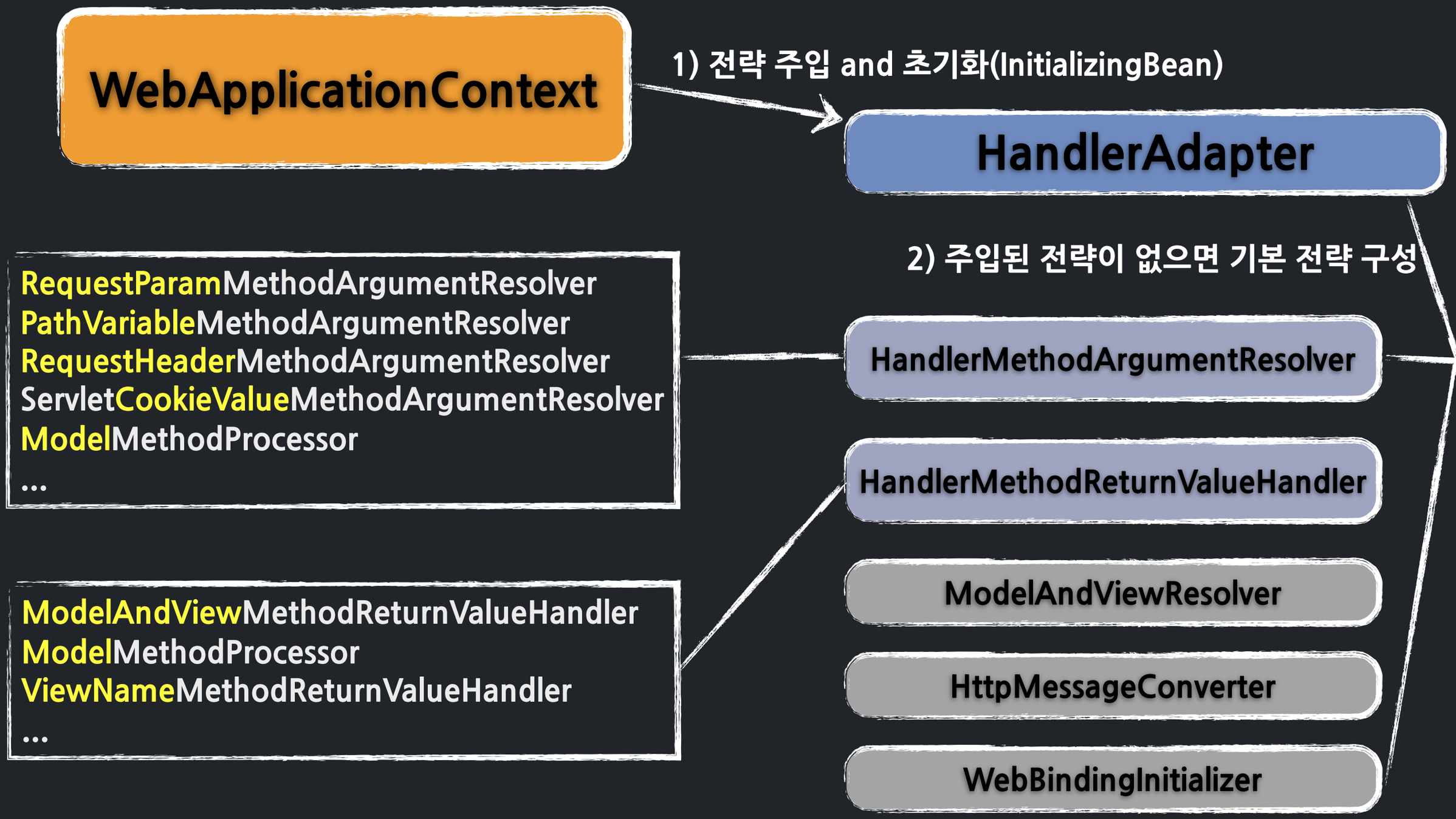
✔ Tomcat.WebApplication이 구동되며 하는 일



✔ HandlerMapping이 초기화되며 하는 일

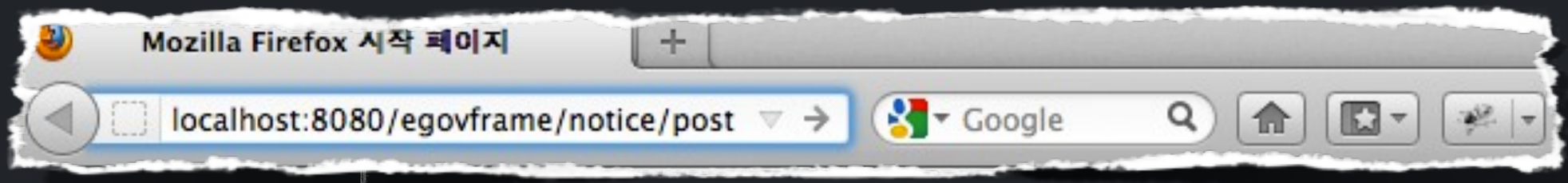


✔ HandlerAdapter가 초기화되며 하는 일



✔ /notice/post 접속

✔ Request: /notice/post



1) HTTP - GET - /notice/post 요청

Tomcat

DispatcherServlet

2) Handler 요청

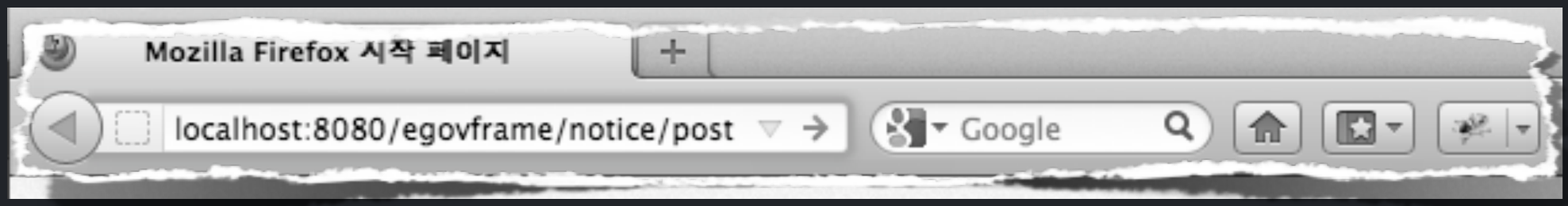
3) PostController .list(..) 반환

Handler Mapping

MultiValueMap urlMap

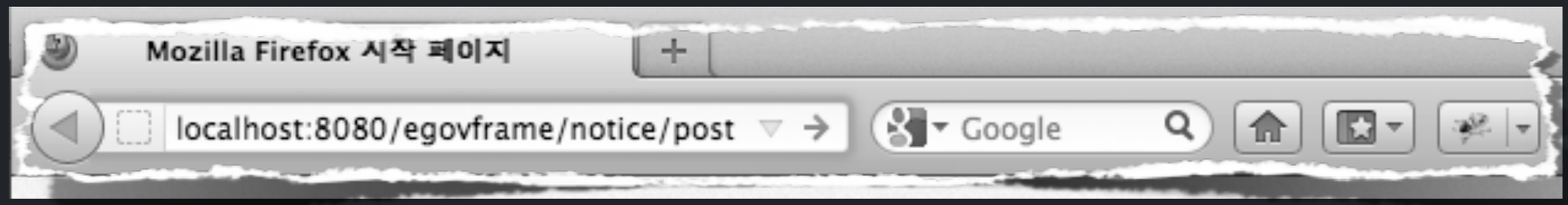
GET - /{board}/post = PostController.list(..)
GET - /{board}/post/{postNo} = PostController.view(..)
POST - /{board}/post = PostController.writing(..)
GET - /{board}/post/new = PostController.addForm(..)
GET - /{board}/post/edit/{postNo} = PostController.editForm(..)
PUT - /{board}/post = PostController.editing(..)
DELETE - /{board}/post = PostController.erase(..)

✔ Handler(PostController.list) 실행




```
@RequestMapping(value="/{board}/post"
                , method=RequestMethod.GET)
public String list(@PathVariable Board board
                   , Model model) {

    model.addAttribute("posts", board.getPosts());

    return "post/list";
}
```

✔ Handler(PostController.list) 실행



Tomcat

DispatcherServlet

1) PostController.list(..) 위임

5) ModelAndView 반환

Handler Adapter

HandlerMethodArgumentResolver

2) Board, Model 생성

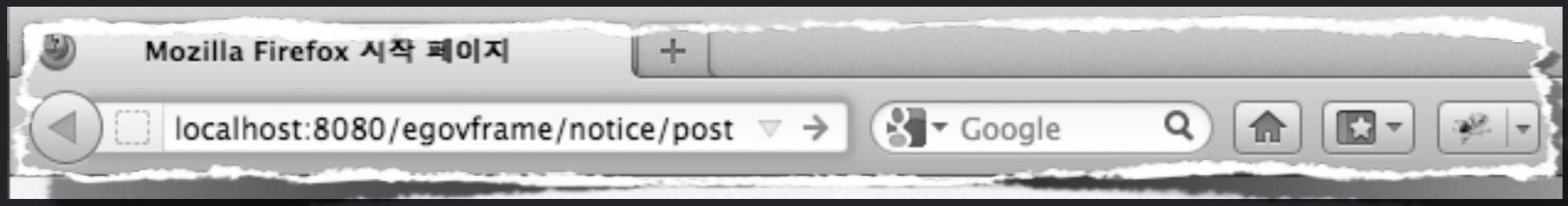
String list(@PathVariable Board , Model)

3) 호출

4) '/post/list' 처리

HandlerMethodReturnValueHandler

✔ HandlerMethodArgumentResolver 동작 (1)



@PathVariable Board board

Handler Adapter

1) 선택 및 처리 요청

PathVariableMethodArgumentResolver

2) URL에서 값 추출

/{board}/post

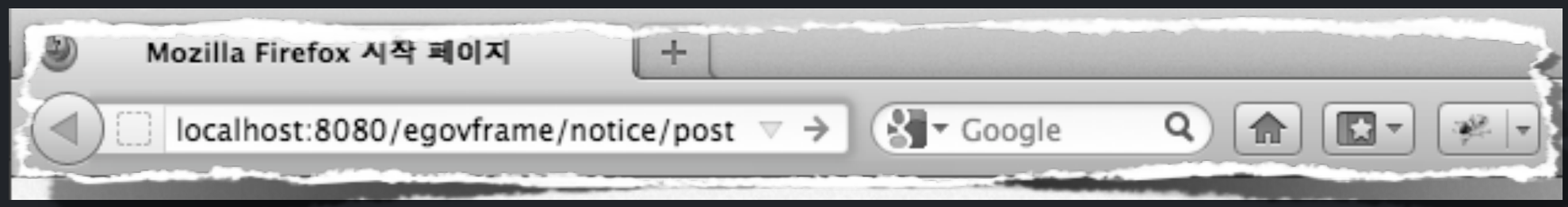
/notice/post

3) 타입 변환

String notice > Board notice

Type Conversion(BoardConverter)

✔ HandlerMethodArgumentResolver 동작 (2)



Model model

Handler Adapter

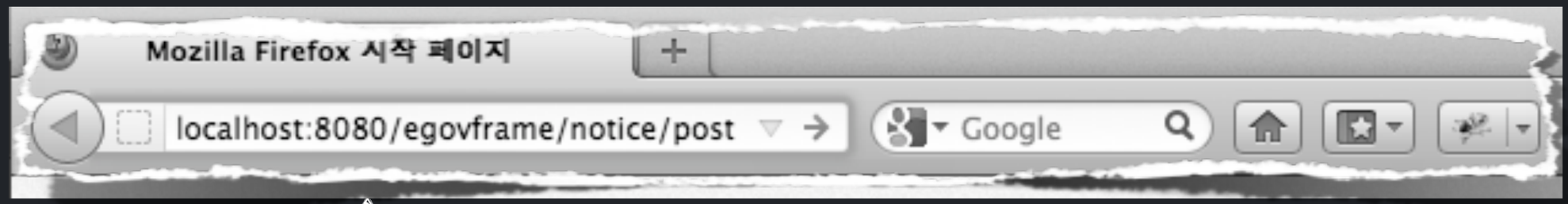
1) 선택 및 처리 요청

ModelMethodProcessor

2) 생성

```
new ModelMap();
```

✔ View("/post/list") 선택 및 응답



4) HTTP 응답

Tomcat

DispatcherServlet

1) View 요청
"/post/list"

3) View 반환

View Resolver

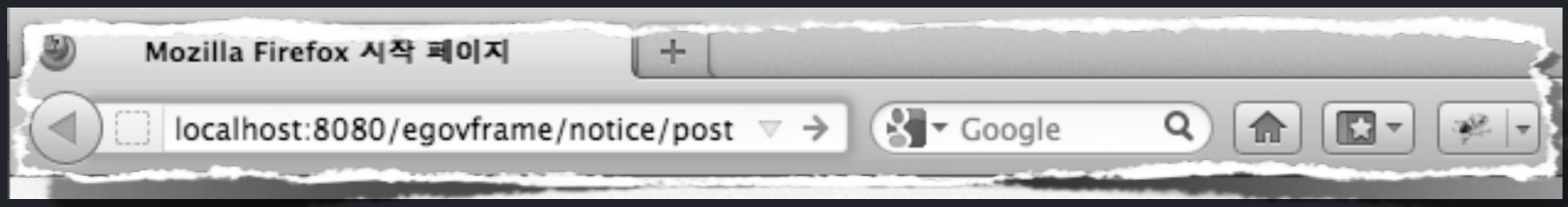
2) View 생성

/WEB-INF/views/post/list.jsp

/WEB-INF/views/post/view.jsp

/WEB-INF/views/post/form.jsp

✔ DispatcherServlet.resolveViewName(...)



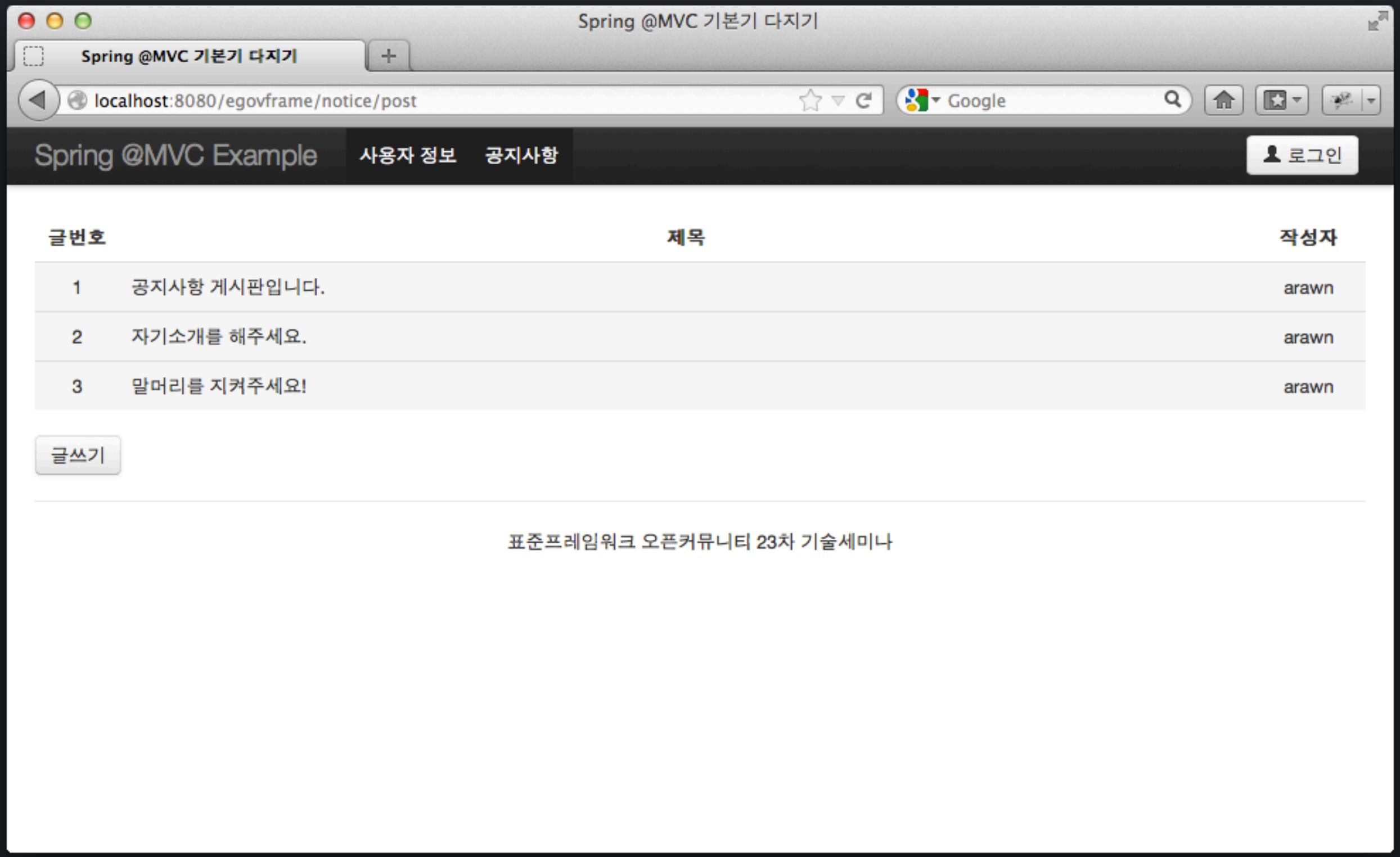
"/post/list"

```
View resolveViewName(String viewName, Locale locale) {

  for (ViewResolver viewResolver : this.viewResolvers) {
    View view = viewResolver.resolveViewName(viewName, locale);
    if(view != null) {
      return view;
    }
  }

  return null;
}
```

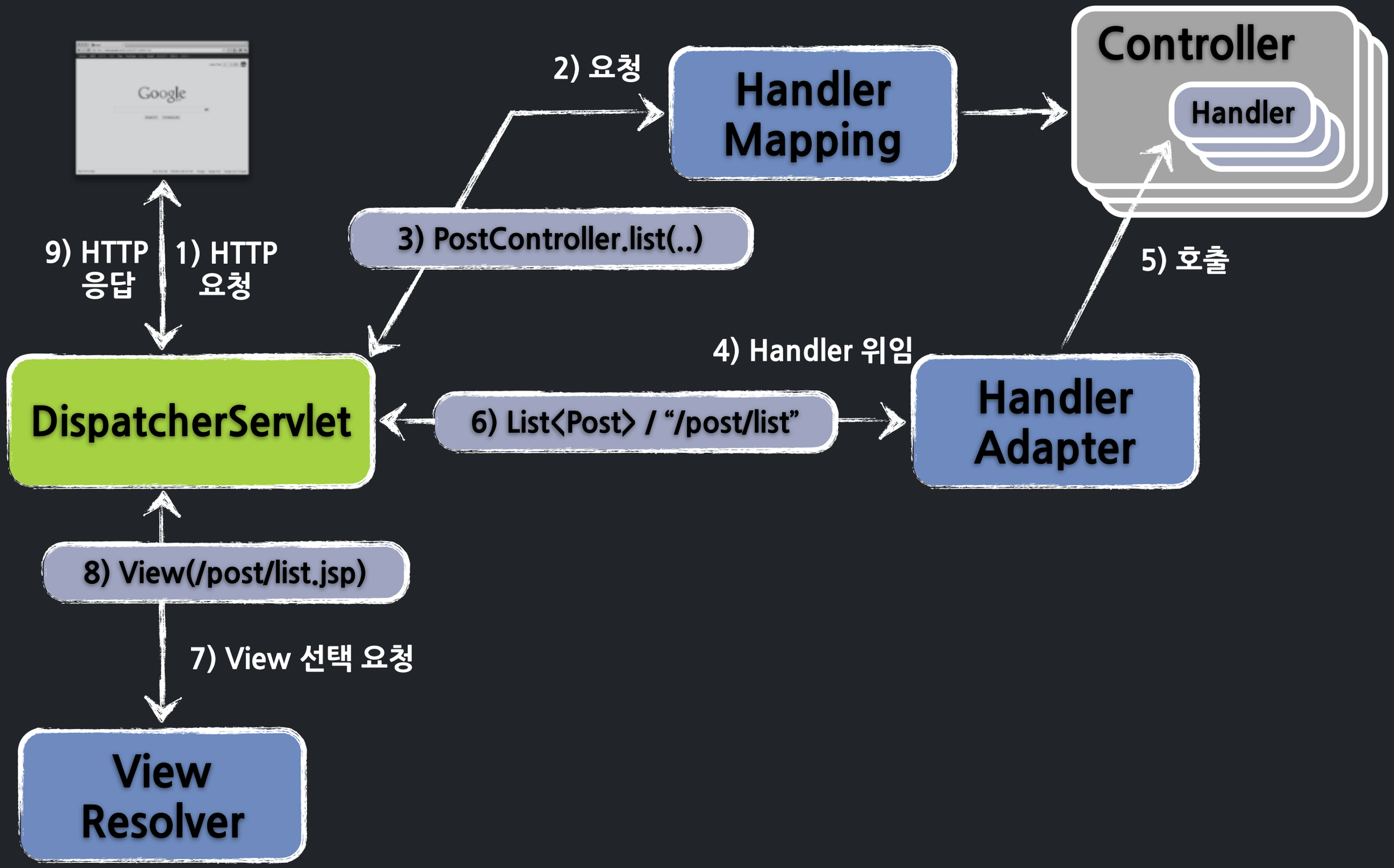
JstlView("WEB-INF/views/post/list.jsp")

스프링 MVC 해부학 : Request -> Spring @MVC -> Response

✔ 결과

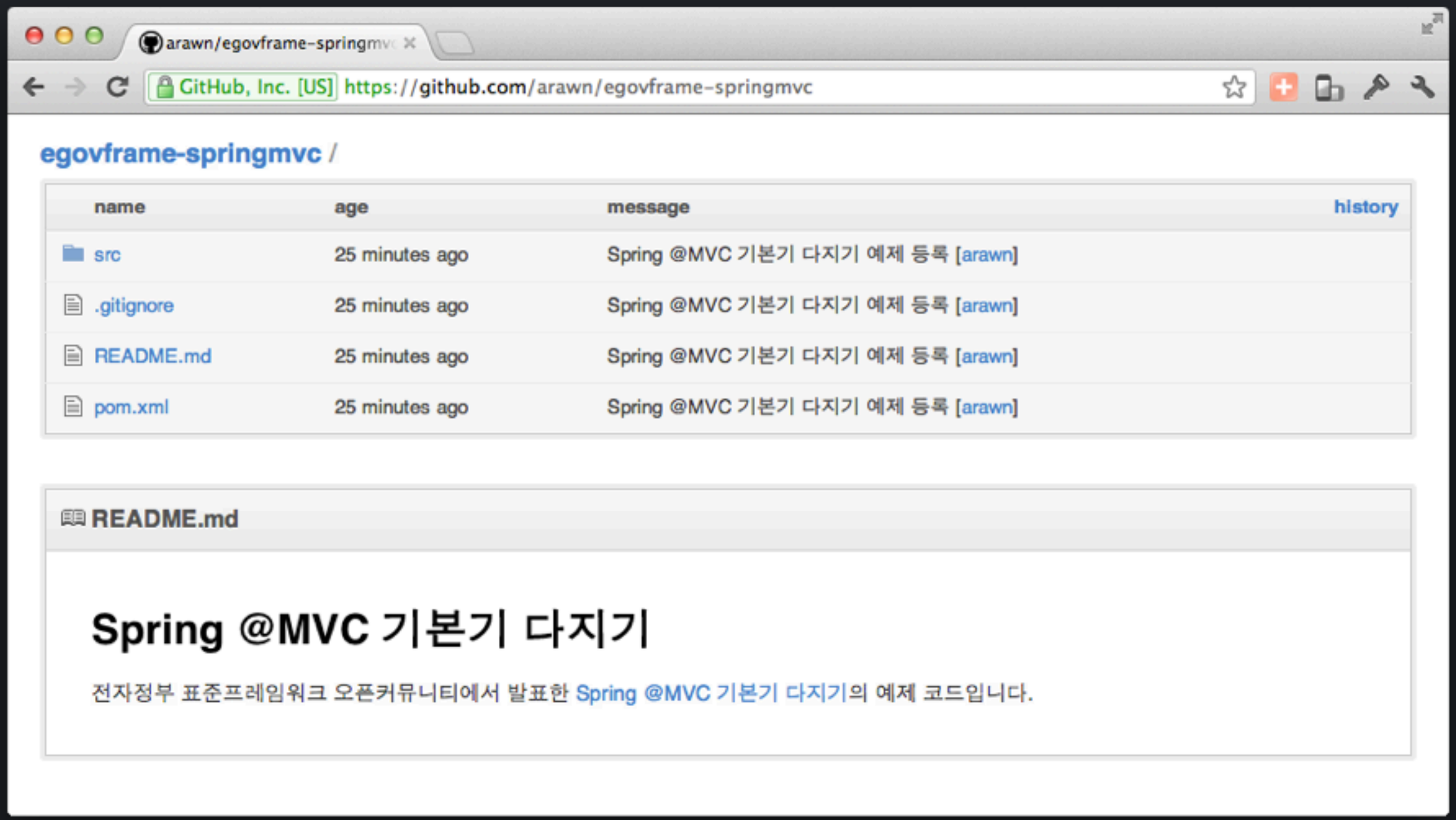


✔ Request -> Spring @MVC -> Response



✔ 예제코드

https://github.com/arawn/egovframe-springmvc



✔ 발표자

박용권

(주)드림인프라 / 애플리케이션 아키텍쳐 / 개발

- 봄싹(SpringSprout)
- 한국 스프링 사용자 그룹(KSUG)
- 라 스칼라 코딩단(La Scala Coding Dan)
- eGov-Data 프로젝트팀