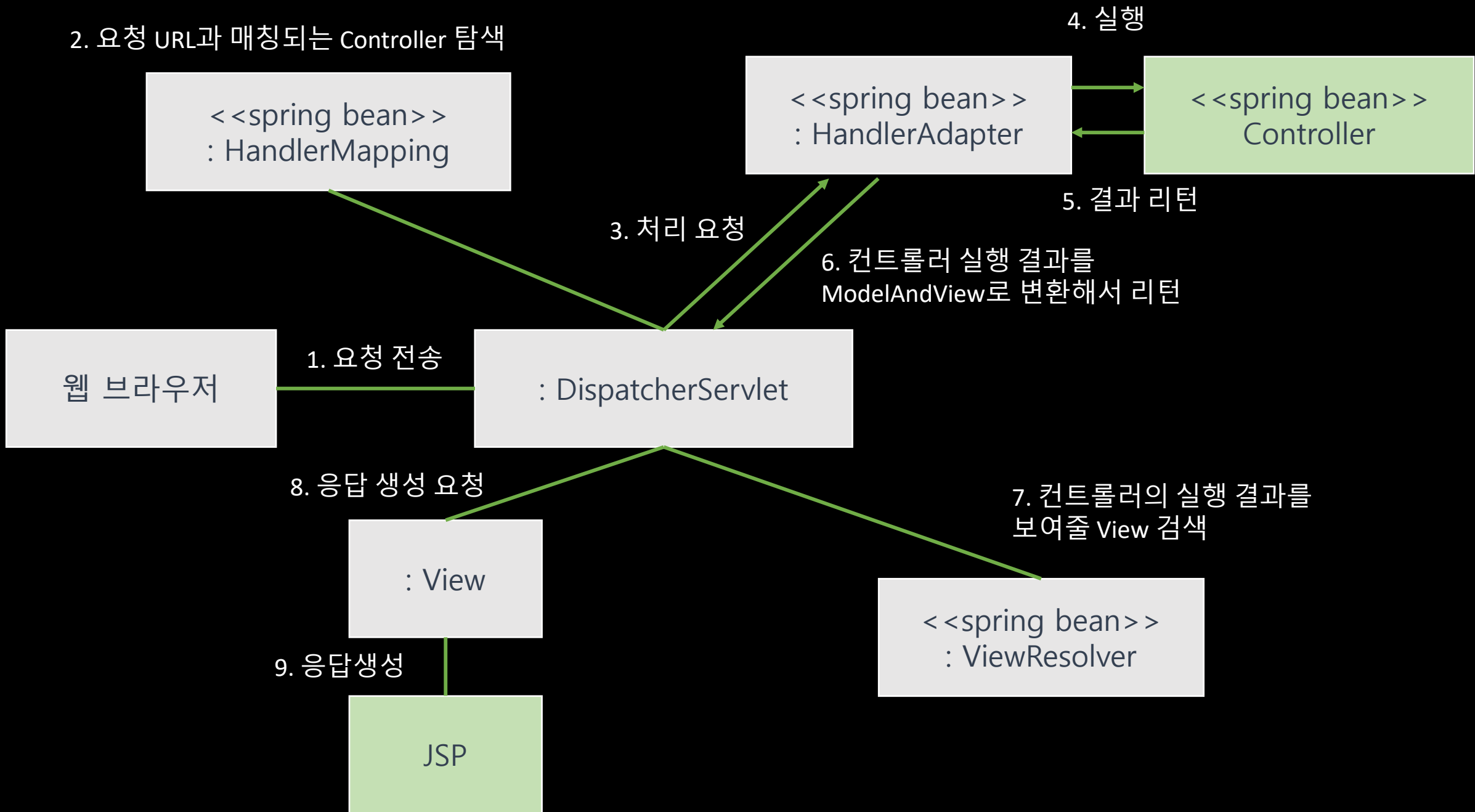


스프링 5

Chapter 10

정승혜

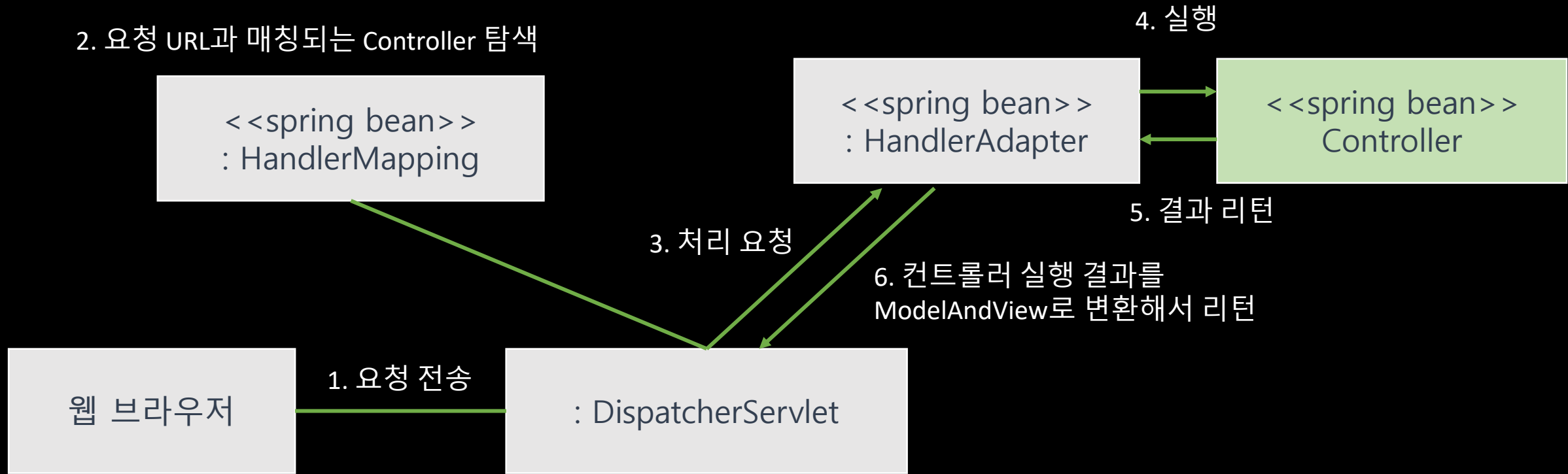


: DispatcherServlet

web.xml 파일에 등록된
스프링 컨테이너, 서블릿에 대한 설정

클라이언트의 요청을 받는 창구 역할

2. 요청 URL과 매칭되는 Controller 탐색



Controller와 Handler

HandlerMapping : 클라이언트의 요청에 해당하는 컨트롤러를 탐색

HandlerAdapter :

컨트롤러는

@Controller, Controller 인터페이스, HttpServletRequestHandler 인터페이스 등을
사용한 여러가지 형태로 존재

=> 모든 클래스들을 동일한 방식으로 처리해주기 위해 중간에 사용됨

=> DispatcherServlet에 ModelAndView 객체 반환

⇒ 직접 Configuration에 @Bean을 통해 등록 가능,
하지만 @EnableWebMvc 어노테이션 활용시 자동으로 스프링 빈 추가

WebMvcConfigurer 인터페이스 1

- 디폴트 핸들러와 HandlerMapping의 우선순위

- web.xml 파일에 DispatcherServlet에 대한 매핑경로를 '/'로 설정 시,
DispatcherServlet은 .jsp로 끝나는 요청을 제외한 모든 요청을 처리함

<web.xml>

```
<servlet-mapping>  
<servlet-name>dispatcher</servlet-name>  
<url-pattern>/</url-pattern>  
</servlet-mapping>
```

WebMvcConfigurer 인터페이스 1


-디폴트 핸들러와 HandlerMapping의 우선순위

- 하지만 @EnableWebMvc 어노테이션이 등록하는 HandlerMapping은
@Controller 어노테이션을 적용한 빈 객체가 처리하는 요청 경로에만 대응할 수 있다.

⇒ @GetMapping("/hello")의 /hello 경로만 처리할 수 있고, 나머지는 404 error

```
@Controller
public class HelloController {

    @GetMapping("/hello")
    public String hello(Model model,
        @RequestParam(value = "name", required = false) String name) {
        model.addAttribute("greeting", "안녕하세요" + name);
        return "hello";
    }
}
```



WebMvcConfigurer 인터페이스 1

-디폴트 핸들러와 HandlerMapping의 우선순위

그렇다면 나머지 경로들은 어떻게 처리할까?

1. 각 경로를 처리하기 위한 컨트롤러 객체를 직접 구현
2. WebMvcConfigurer의 configureDefaultServletHandling() 메서드를 사용하면 편리

```
@Override
public void configureDefaultServletHandling(
    DefaultServletHandlerConfigurer configurator) {
    configurator.enable();
}
```

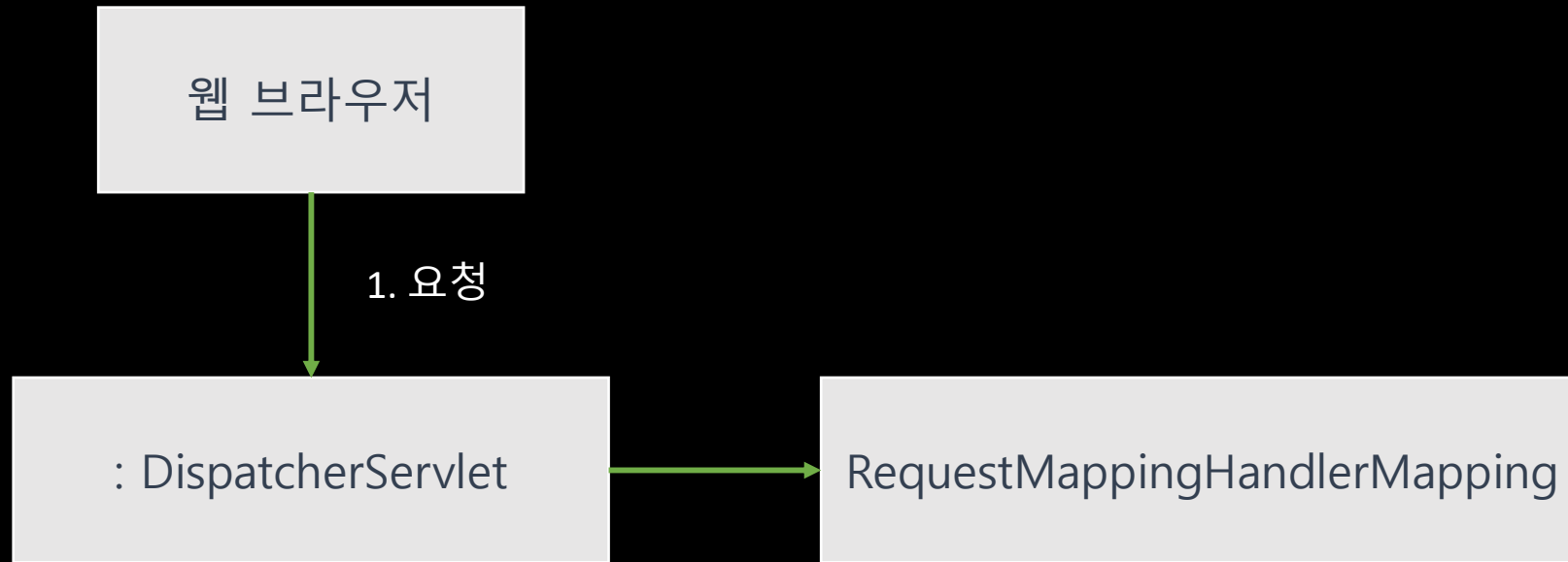
DefaultServletHandlerConfigurer.enable()는 다음 두 빈 객체를 추가함

- DefaultServletHttpRequestHandler
- SimpleUrlHandlerMapping

WebMvcConfigurer 인터페이스 1

-디폴트 핸들러와 HandlerMapping의 우선순위

- 일치하는 컨트롤러가 있을때

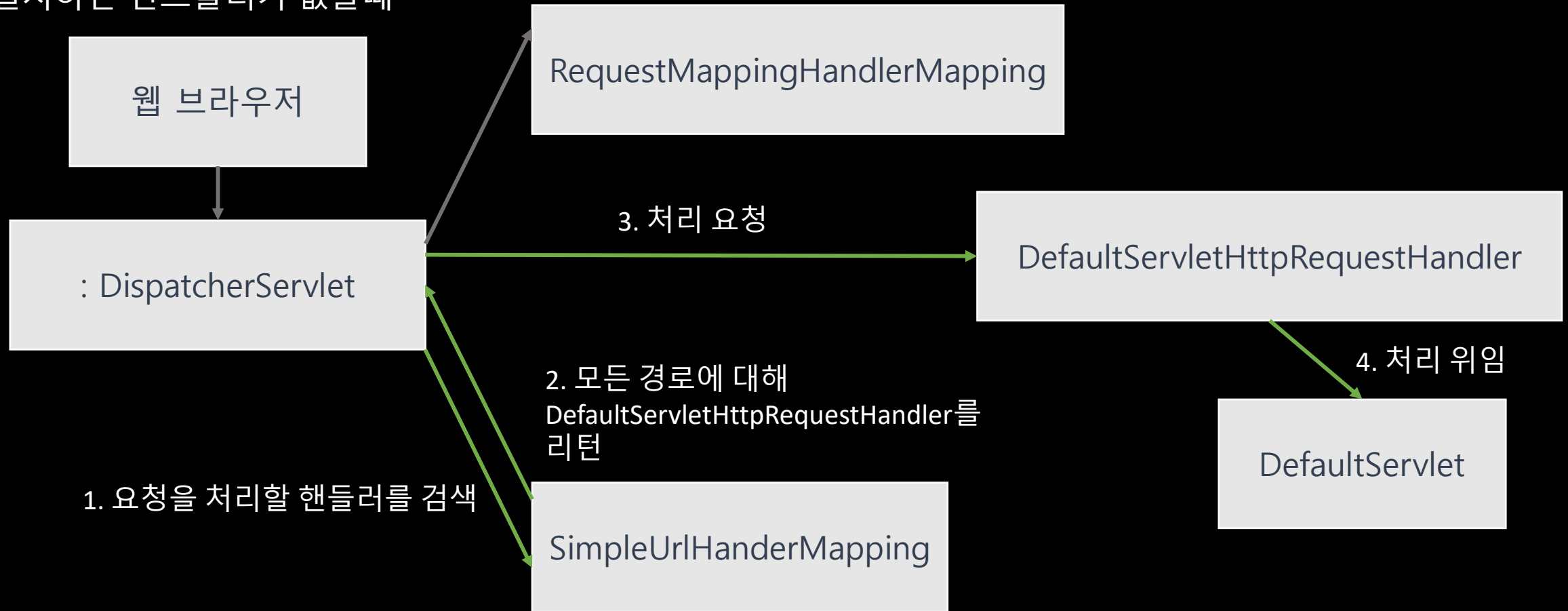


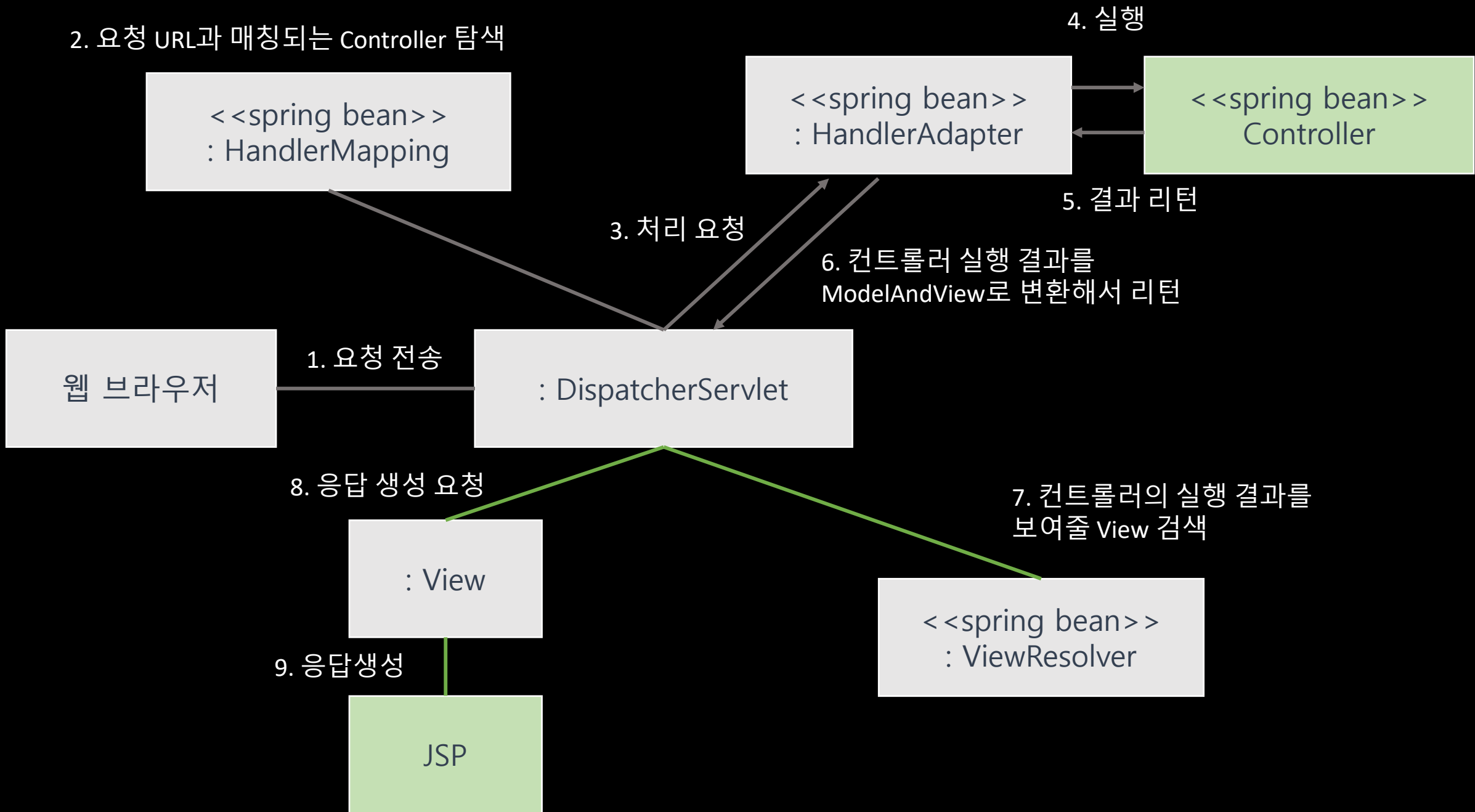
2. 일치하는 컨트롤러가 있는지 탐색,
있다면 처리 (@Controller에 대한 경로)

WebMvcConfigurer 인터페이스 1

-디폴트 핸들러와 HandlerMapping의 우선순위

- 일치하는 컨트롤러가 없을때





WebMvcConfigurer 인터페이스 2

- JSP를 위한 ViewResolver

- public void configureViewResolvers(ViewResolverRegistry registry) { }

=> Controller에서 DispatcherServlet을 통해 넘어온 Map 객체를 jsp에 보냄

```
@Override
public void configureViewResolvers(ViewResolverRegistry registry) {
    registry.jsp("/WEB-INF/view/", ".jsp");
}
```