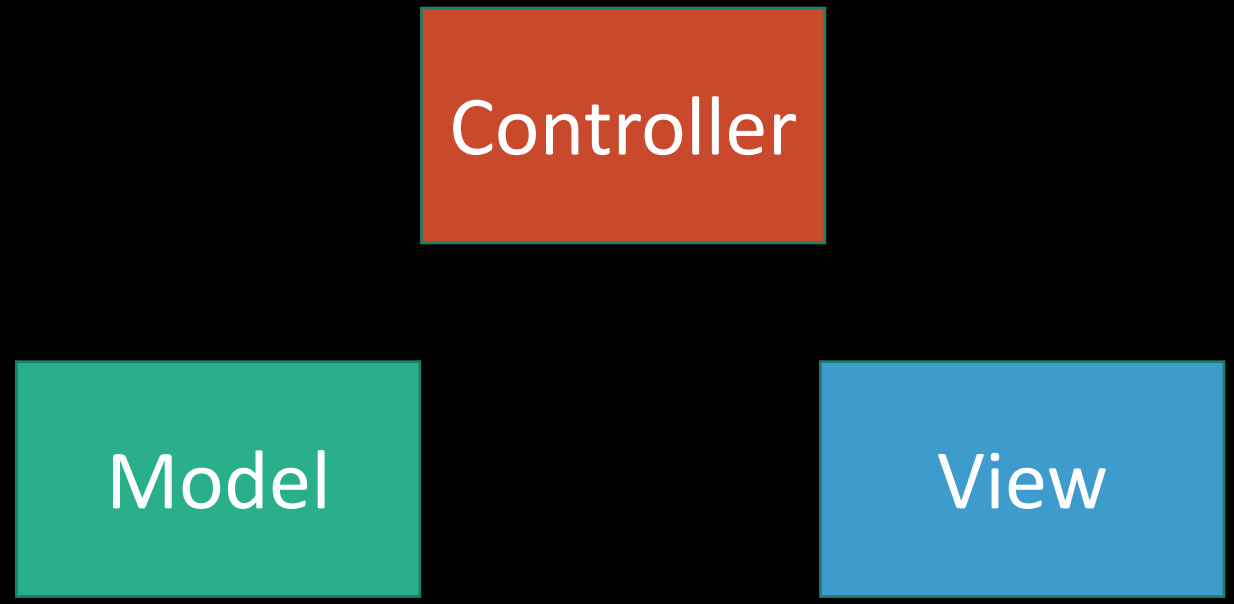
Chap 09
스프링 MVC 시작하기

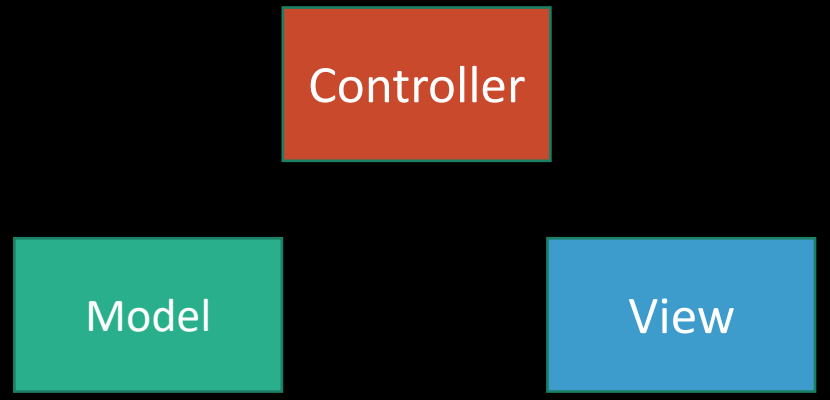
스프링 MVC

- 스프링은 MVC(Model-view-controller) 프레임워크를 지원한다.



MVC 디자인 패턴

- Model: 애플리케이션의 데이터와 그 가공을 책임지는 컴포넌트로, 백그라운드에서 동작하는 로직을 처리한다.
- View: 사용자들이 볼 수 있는 화면을 출력한다.
- Controller: Model과 View를 연결시켜주는 다리 역할을 하며, 사용자의 입력처리와 흐름 제어를 담당한다.



pom.xml 설정

- 서블릿/JSP로 웹 어플리케이션을 개발하려면 <packaging>의 값으로 war를 줘야 한다. (web application archive)
- 또한, 스프링을 이용해서 개발하므로 spring-webmvc 모듈 의존을 추가한다.

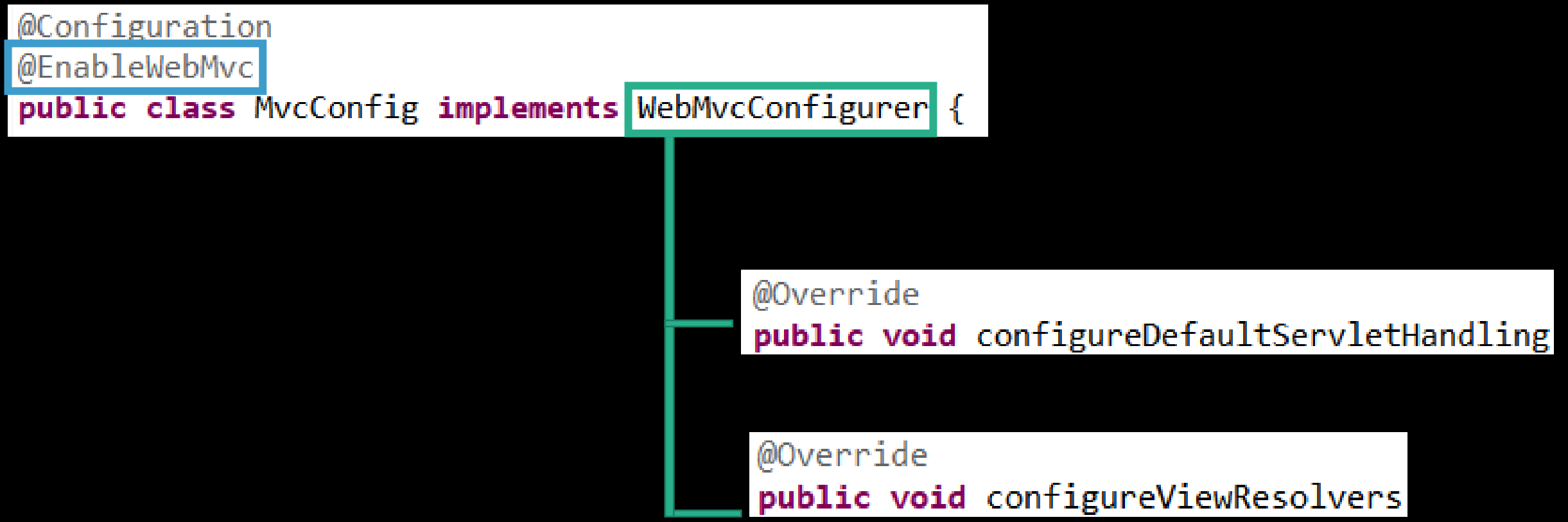
```
<packaging>war</packaging>
```

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.0.2.RELEASE</version>
</dependency>
```

스프링 MVC 설정

- 1) 스프링 MVC 설정 클래스

- @EnableWebMvc 어노테이션을 사용하면 MVC를 쓰기 위해 필요한 복잡한 빈 설정들을 알아서 처리해준다.
- 스프링 MVC의 개별 설정을 조정할 때 WebMvcConfigurer 인터페이스를 구현한다. 이 인터페이스 내부의 메소드를 오버라이드하여 설정을 조정한다.


@Configuration
@EnableWebMvc
public class MvcConfig implements WebMvcConfigurer {
@Override
public void configureDefaultServletHandling
@Override
public void configureViewResolvers

스프링 MVC 설정

- 2) DispatcherServlet 설정

- 스프링 MVC가 웹 요청을 처리하려면 DispatcherServlet이 필요하다. 이를 src/main/webapp/WEB-INF/web.xml에 등록한다.

```
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
        <param-name>contextClass</param-name>
        <param-value>
            org.springframework.web.context.support.AnnotationConfigWebApplicationContext
        </param-value>
    </init-param>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            config.MvcConfig
            config.ControllerConfig
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

스프링 MVC 구현

- 1) 컨트롤러 구현

```java
@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";
    }
}
```

- 웹 요청을 처리하고 그 결과를 뷰에 전달하는 빈 객체이다.
- 스프링 컨트롤러로 사용할 클래스는 @Controller 어노테이션을 붙여야 하고, @GetMapping이나 @PostMapping과 같은 요청 매핑 어노테이션을 이용해서 처리할 경로를 지정해줘야 한다.

스프링 MVC 구현

- 1) 컨트롤러 구현 - @GetMapping 과 요청 URL의 관계

컨텍스트 경로

localhost:8080/sp5-chap09/hello?name=재인 요청 URL

```
@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";
    }
}
```

매개변수 전달

뷰에서 사용하는 속성 이름

결과를 보여줄 뷰

HelloController.java
(스프링 컨트롤러)

스프링 MVC 구현

- 2) 뷰 구현

```jsp
<%@ page contentType="text/html; charset=utf-8" %>
<!DOCTYPE html>
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
  인사말: ${greeting}
  </body>
</html>
```

- 뷰에서는 컨트롤러가 생성한 결과를 보여준다.
- 뷰 코드는 JSP를 이용해서 구현한다.

스프링 MVC 구현

- 2) 뷰 구현 – 뷰 이름과 JSP 파일의 연결

```java
@Configuration
@EnableWebMvc
public class MvcConfig implements WebMvcConfigurer {
```

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

MvcConfig.java
(MVC 설정 클래스)

```jsp
<%@ page contentType="text/html; charset=utf-8" %>
<!DOCTYPE html>
<html>
  <head>
    <title>Hello</title>
  </head>
  <body>
  인사말: ${greeting}
  </body>
</html>
```

/WEB-INF/view/hello.jsp
(뷰 JSP파일)