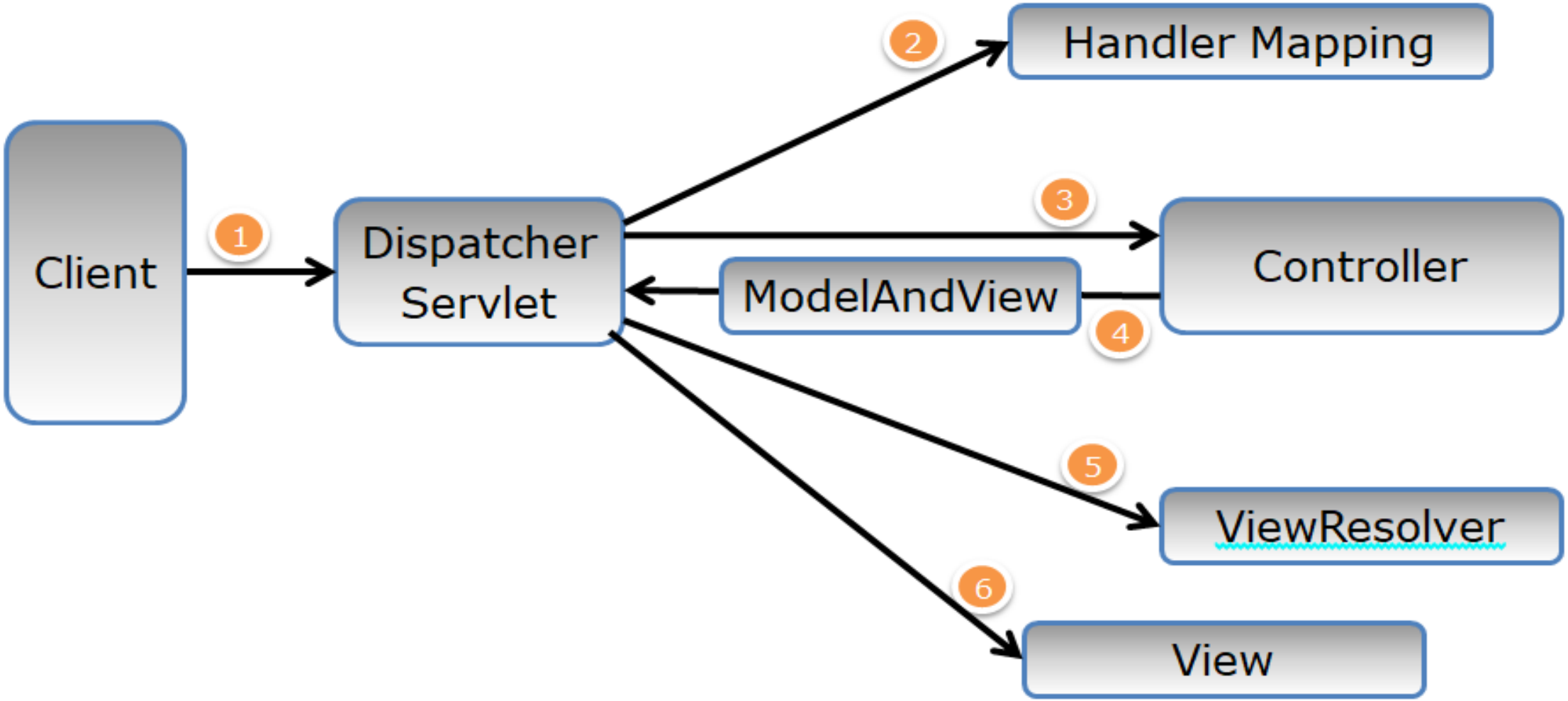
Spring MVC

안 화 수

Spring MVC 흐름도 (1/2)



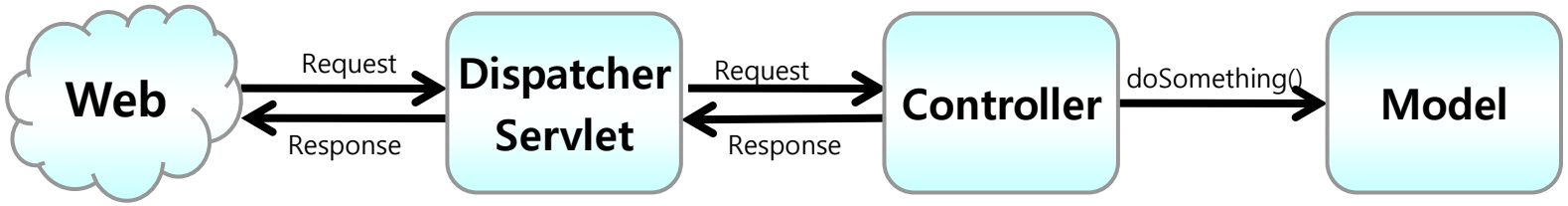
Spring MVC 흐름도 (2/2)

- 요청 처리 순서
  1. DispatcherServlet이 요청을 수신
     - 단일 front controller servlet
     - 요청을 수신하여 처리를 다른 컴포넌트에 위임
     - 어느 컨트롤러에 요청을 전송할지 결정
  2. DispatcherServlet은 HandlerMapping에 어느 컨트롤러를 사용할 것인지 요청
     - 지금은 @RequestMapping 을 이용해서 요청을 받는다.
  3. DispatcherServlet은 요청을 컨트롤러에게 전송하고 컨트롤러는 요청을 처리한 후 결과 리턴
     - 비즈니스 로직 수행 후 결과 정보(Model)가 생성되어 JSP와 같은 뷰에서 사용됨
  4. ModelAndView 오브젝트에 수행결과가 포함되어 DispatcherServlet에 리턴
  5. ModelAndView는 실제 JSP정보를 갖고 있지 않으며, ViewResolver가 논리적 이름을 실제 JSP이름으로 변환
  6. View는 결과정보를 사용하여 화면을 표현함.

Spring MVC 구현 Step

❖ Spring MVC를 이용한 애플리케이션 작성 스텝

1. web.xml에 DispacherServlet 등록 및 Spring설정파일 등록
2. 컨트롤러 구현 및 Spring 설정파일에 등록
3. 컨트롤러와 JSP의 연결 위해 View Resolver 설정
4. JSP 코드 작성



web.xml (1/3)

- **web.xml에 DispatcherServlet 설정**
- DispatcherServlet 을 등록하고 servlet mapping을 설정한다.
- Spring의 환경 설정 파일을 등록한다.

```
<servlet>
        <servlet-name>appServlet</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <init-param>
                <param-name>contextConfigLocation</param-name>
                <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
        <servlet-name>appServlet</servlet-name>
        <url-pattern>/</url-pattern>
</servlet-mapping>
```

web.xml (2/3)

- **web.xml에 Spring의 환경 설정 파일 등록**
- Spring의 환경 설정 파일을 등록한다.
- DataBase 접속과 같이 공통적인 내용을 root-context.xml 파일에 등록한다.

```
<!-- The definition of the Root Spring Container shared by all Servlets and Filters -->
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring/root-context.xml</param-value>
</context-param>
<!-- Creates the Spring Container shared by all Servlets and Filters -->
<listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

web.xml (3/3)

- **web.xml에 한글 인코딩을 처리하기 위한 필터 등록**
  - 한글값이 POST 방식으로 전송될때 한글이 깨지지 않도록 인코딩을 처리하기 위한 필터를 등록한다.

```
<filter>
        <filter-name>CharacterEncodingFilter</filter-name>
        <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
        <init-param>
                <param-name>encoding</param-name>
                <param-value>UTF-8</param-value>
        </init-param>
        <init-param>
                <param-name>forceEncoding</param-name>
                <param-value>true</param-value>
        </init-param>
</filter>
<filter-mapping>
        <filter-name>CharacterEncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
</filter-mapping>
```

Spring의 환경 설정 파일(servlet-context.xml)

❖ Spring의 환경 설정 파일 : servlet-context.xml

- Spring의 환경 설정 파일 ViewResolver 를 등록한다.
  1. view 파일을 저장할 최상위 디렉토리(prefix) : /WEB-INF/views
  2. view 파일의 확장자(suffix) : .jsp

- Spring의 환경 설정 파일에 base-package 를 등록한다.
  1. base-package 하위 클래스를 스캔한다는 의미를 가진다.
  2. base-package 하위 클래스에 @Controller, @Service, @Repository 어노테이션이 붙어있는 클래스는 @Autowired 어노테이션을 이용해서 필요한 빈 객체를 setter 메소드 없이 자동으로 주입을 받는다.

```
<beans:bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <beans:property name="prefix" value="/WEB-INF/views/" />
        <beans:property name="suffix" value=".jsp" />
</beans:bean>

<context:component-scan base-package="com.myhome.springtest" />
```

Controller 클래스 구현

❖ Controller 클래스 구현 : HomeController.java

1. 컨트롤러 클래스 위에는 @Controller 을 붙인다.
2. 클라이언트의 요청을 받기 위해서 받기 위해서는 @RequestMapping 으로 요청을 받는다.
3. 컨트롤러에서 view 페이지로 값을 가지고 갈 경우에는 Model 이나 ModelAndView 클래스에 값을 저장해서 이동한다.
4. return 할 경우에는 prefix와 suffix 를 제외한 나머지 경로 파일명만 기술한다.

```
@Controller
public class HomeController {
	private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {
		logger.info("Welcome home! The client locale is {}.", locale);

		Date date = new Date();
		DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);
		String formattedDate = dateFormat.format(date);
		model.addAttribute("serverTime", formattedDate );

		return "home";
	}
}
```

View 페이지 구현

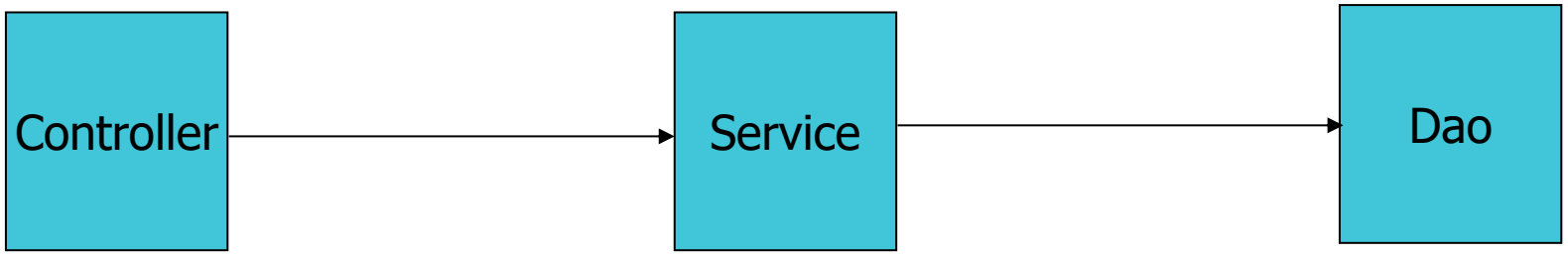
View 페이지 구현 : home.jsp

- 컨트롤러 클래스에서 Model 에 저장해서 넘어온 값을 표현언어(Expression Language)로 출력 한다.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ page session="false" %>
<html>
<head>
	<title>Home</title>
</head>
<body>
<h1>
	Hello world!
</h1>
<P>  The time on the server is ${serverTime}. </P>
</body>
</html>
```

어노테이션(Anotation)

- **Anotation**



@Controller
BoardController

@Autowired
private BoardService service

@Service
BoardService

@Autowired
private BoardDao dao;

@Repository
BoardDao

@Autowired
private SqlSession session

어노테이션(Anotation)

❖ @Controller

- @Controller 어노테이션은 Controller 기능을 수행 할 수 있도록 만들어 주는 역할을 한다.

```
@Controller
public class HomeController {

	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {

		Date date = new Date();
		DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);
		String formattedDate = dateFormat.format(date);
		model.addAttribute("serverTime", formattedDate );

		return "home";
	}
}
```

어노테이션(Anotation)

❖ @RequestMapping

- @RequestMapping 어노테이션은 클라이언트의 요청을 받을때 사용하는 어노테이션이다.

```
@Controller
public class HomeController {

	@RequestMapping(value = "/", method = RequestMethod.GET)
	public String home(Locale locale, Model model) {

		Date date = new Date();
		DateFormat dateFormat = DateFormat.getDateTimeInstance(DateFormat.LONG, DateFormat.LONG, locale);
		String formattedDate = dateFormat.format(date);
		model.addAttribute("serverTime", formattedDate );

		return "home";
	}
}
```

어노테이션(Anotation)

❖ @RequestParam

- @RequestParam 어노테이션은 name 으로 값을 받을때 사용하는 어노테이션이다.
- @RequestParam("name") 어노테이션은 request.getParameter("name")와 같은 역할을 한다.

```
@Controller
public class HomeController {

        @RequestMapping(value = "/", method = RequestMethod.GET)
        public String home(@RequestParam("name")  String name, Model model) {

        }
}
```

어노테이션(Anotation)

❖ @ModelAttribute

- @ModelAttribute 어노테이션은 DTO클래스로 값을 받을때 사용하는 어노테이션이다.

```
@Controller
public class HomeController {

        @RequestMapping(value = "/", method = RequestMethod.GET)
        public String home(@ModelAttribute  BoardDTO board, Model model) {

        }
}
```