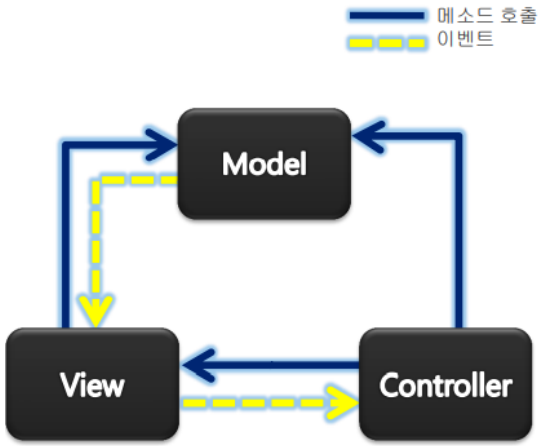
제9장 스프링 웹 MVC 기초

- 스프링 MVC 기초 개념
- 스프링 MVC 설정 과정
- 메이븐(Maven) 프로젝트 생성

MVC 패턴

- MVC(Model-View-Controller)
  - Model
    - 애플리케이션 상태의 캡슐화
    - 상태 쿼리에 대한 응답
    - 애플리케이션의 기능 표현
    - 변경을 뷰에 통지
  - View
    - 모델을 화면에 시각적으로 표현
    - 모델에게 업데이트 요청
    - 사용자의 입력을 컨트롤러에 전달
    - 컨트롤러가 뷰를 선택하도록 허용
  - Controller
    - 애플리케이션의 행위 정의
    - 사용자 액션을 모델 업데이트와 매핑
    - 응답에 대한 뷰 선택

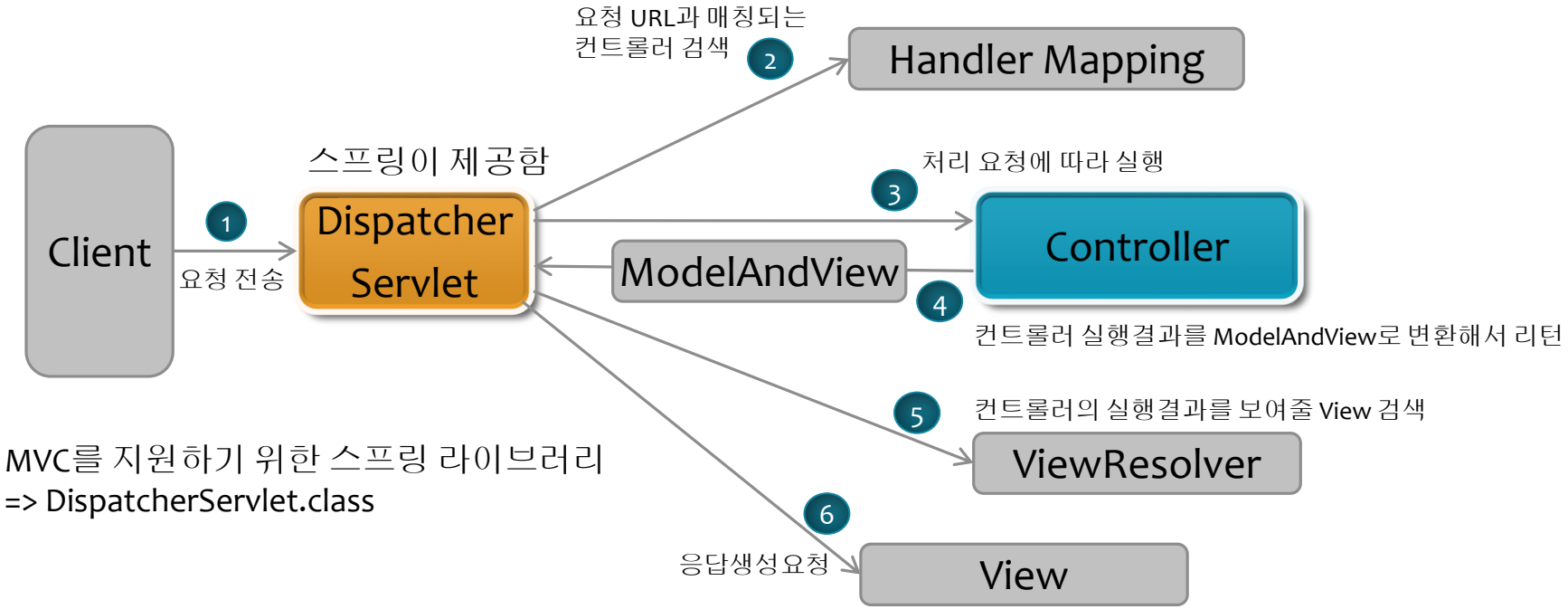


MVC 패턴

- MVC(Model-View-Controller)패턴
  - 애플리케이션의 확장을 위해 Model, View, Controller 세가지 영역으로 분리
  - 컴포넌트의 변경이 다른 영역 컴포넌트에 영향을 미치지 않음
  - 컴포넌트간의 결합성이 낮아 프로그램 수정이 용이
  - 장점
    - 화면과 비즈니스 로직을 분리해서 작업 가능
    - 영역별 개발로 인하여 확장성이 뛰어남
    - 표준화된 코드를 사용하므로 공동작업이 용이하고 유지보수성이 좋음
  - 단점
    - 개발 과정이 복잡해 초기 개발속도가 늦음
    - 초보자가 이해하고 개발하기에 다소 어려움

스프링 MVC의 흐름

- MVC패턴 기반 웹 개발 프레임워크(스프링 MVC웹 요청 처리과정)



스프링 MVC의 흐름

- 스프링 MVC 의 주요 구성 요소

구성요소	설명
DispatcherServlet	클라이언트의 요청을 전달받는다. 컨트롤러에게 클라이언트의 요청을 전달하고, 컨트롤러가 리턴한 결과값을 View에 전달하여 알맞은 응답을 생성하도록 한다.
HandlerMapping	클라이언트의 요청 URL을 어떤 컨트롤러가 처리할지를 결정한다.
HandlerAdapter	DispatcherServlet의 처리 요청을 변환해서 컨트롤러에게 전달하고 , 컨트롤러의 응답 결과를 DispatcherServlet이 요구하는 형식으로 변환한다. 웹브라우저 캐시 등의 설정도 담당한다.
Controller	클라이언트의 요청을 처리한 뒤 결과를 리턴한다. 응답 결과에서 보여줄 데이터를 모델에 담아 전달한다.
ModelAndView	컨트롤러가 처리한 결과 정보 및 뷰 선택에 필요한 정보를 담는다.
ViewResolver	컨트롤러의 처리결과를 보여줄 뷰를 결정한다.
View	컨트롤러의 처리결과 화면을 생성한다. JSP나 Velocity템플릿 파일 등을 이용해서 클라이언트에 응답 결과를 전송한다.

스프링 MVC설정 과정

- 스프링 MVC를 사용하기 위한 기본 설정 과정
  1. Web.xml에 DispatcherServlet설정
  2. Web.xml에 character 인코딩 처리 위한 필터 설정
  3. 스프링 MVC 설정
     - HandlerMapping 설정
     - HandlerAdapter 설정
     - ViewResolver 설정

DispatcherServlet 설정(1)

- web.xml에 등록
- 스프링의 중심이 되는 서블릿 클래스
- 스프링 설정파일:"<servlet-name>-servlet.xml" 에서 애플리케이션 정보를 로드, WEB-INF\아래 추가
- <url-pattern>은 DispatcherServlet이 처리하는 URL 매핑 패턴을 정의

```
<servlet>
   <servlet-name>dispatcher</servlet-name>
   <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
   </servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>dispatcher</servlet-name>
   <url-pattern>*.htm</url-pattern>
</servlet-mapping>
```

DispatcherServlet 설정(2)

- 한 개 이상의 설정파일을 사용해야 하거나 이름이 <이름>-servlet.xml형식 아닌 파일을 사용해야 할 경우
- contextConfigLocation 초기화 파라미터로 설정 파일 목록을 지정

```
<servlet>
   <servlet-name>dispatcher</servlet-name>
   <servlet-class>
            org.springframework.web.servlet.DispatcherServlet
   </servlet-class>
</servlet>
<init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>
            /WEB-INF/main.xml
            /WEB-INF/bbs.xml
            classpath:/common.xml     //클래스 패스에 위치한 파일 사용할 경우
      </param-value>
</init-param>
</servlet>
```

DispatcherServlet 설정(3)

- Xml 설정 파일이 아닌 @Configuration 클래스를 이용해서 설정 정보를 작성했을 경우
- contextClass 를 추가로 설정

```
<servlet>
   <servlet-name>dispatcherConfig</servlet-name>
   <servlet-class> org.springframework.web.servlet.DispatcherServlet   </servlet-class>
</servlet>
<init-param>
     <param-name>contextClass</param-name> // 미지정시 XmlWebApplicationContext를 사용함
     <param-value>
          org.springframework.web.context.support.AnnotationConfigWebApplicationContext
     </param-value>
</init-param>
```

```
<init-param>
     <param-name>contextConfigLocation</param-name>
     <param-value> k2.MvcStartConfig     </param-value>  //@Configuration 자바클래스의 풀네임 지정
</init-param>
</servlet>

<servlet-mapping>
   <servlet-name>ispatcherConfig</servlet-name>
   <url-pattern>/config/*</url-pattern>
</servlet-mapping>
```

DispatcherServlet 설정(4)

- 캐릭터 인코딩 필터 설정

```
<web-app..>
..중간생략…
     <filter>
          <filter-name>encodingFilter</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>
     </filter>
     <filter-mapping>
          <fitler-name>encodingFilter</filter-name>
          <url-pattern>/*</url-pattern>
     </filter-mapping>
</web-app>
```

스프링 MVC 설정 기초(1)

- HandlerMapping 구현 객체, HandlerAdapter 구현 객체 ,ViewResolver 구현 객체
- HandlerMapping 과 HandlerAdapter는 <mvc:annotation-driven>태그를 이용하면 아래 두 클래스를 빈으로 등록 해 줌. @Controller로 적용된 클래스를 컨트롤러로 사용할 수 있도록 함
  - RequestMappingHandlerMapping
  - RequestMappingHandlerAdapter
- ViewResolver만 추가로 설정하면 됨

```
<?xml version= .. >
<beans xmlns= …
       xmlns:mvc=http://www.springframework.org/schema/mvc
        ..
       xsi:schemaLocation="http=……
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc.xsd">

        <mvc:annotation-driven/>
        <bean id ="viewResolver"  class="org.springframework.web.servlet.vire.InternalResourceViewReslover">
          <property name="prefix" value="/WEB-INF/view/" />
          <property name="suffix value=".jsp" />
        </bean>
```

스프링 MVC 설정 기초(2)

- 애너테이션 적용시

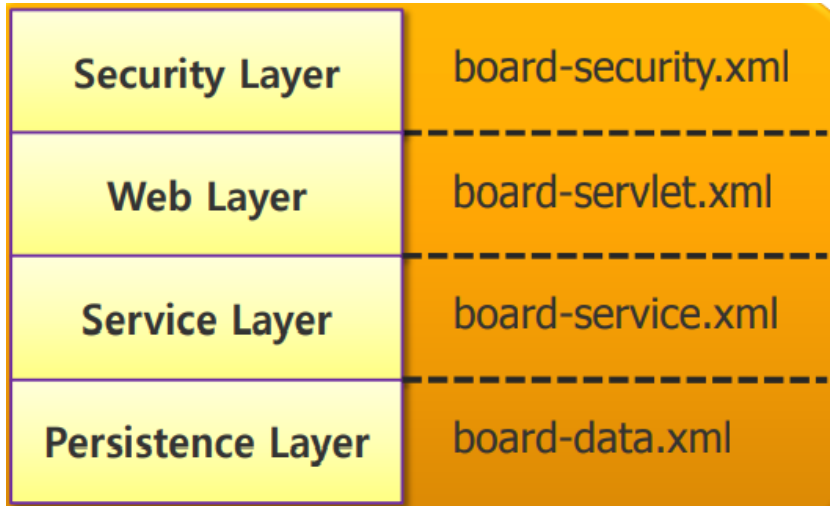
```
import org.springframework.web.servlet.config.annotation.EnablewebMvc;

@Configuration
@EnableWebMvc          //스프링 MVC설정을 활성화 함
public class MvcStartConfig {
    @Bean
    public InternalResourceViewResolver viewReslover(){
       InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
       viewResolver.setPrefix("/WEB-INF/view/" );
       viewResolver.setSuffix(".jsp");
       return viewResolver;
     }
```

```
…
        <mvc:annotation-driven/>
        <bean id ="viewResolver"  class="org.springframework.web.servlet.vire.InternalResourceViewReslover">
            <property name="prefix" value="/WEB-INF/view/" />
            <property name="suffix value=".jsp" />
        </bean>
```

애플리케이션 컨텍스트 분리

- 애플리케이션 컨텍스트 분리
  - 애플리케이션 레이어에 따라 애플리케이션 컨텍스트 분리



컨트롤러 구현 과정(1)

- @Controller/ @RequestMapping /Model 을 이용한 컨트롤러 구현

1. @Controller 애노테이션 클래스에 적용
2. @RequestMapping 애노테이션을 이용해서 처리할 요청 경로를 지정한다
3. 웹 브라우저의 요청을 처리할 메소드를 구현하고 뷰이름을 리턴한다.

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;


@Controller
public class EventController{
     @RequestMapping("/event/list") // 메소드가 처리할 요청 경로 지정, "/event/list" 요청경로를  list()메소드가 처리하도록 설정
      public String list(){       //웹 요청을 처리할 메소드
          return "event/list";    //처리결과를 보여 줄 뷰 이름 리턴
}
```

컨트롤러 구현 과정(2)

- 컨트롤러는 다양한 방식으로 모델(Model)에 데이터를 담을 수 있음
- Model을 파라미터로 추가하고, Model 파라미터에 데이터를 추가함

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class EventController{
     @RequestMapping("/event/list") // 메소드가 처리할 요청 경로 지정, "/event/list" 요청경로를  list()메소드가 처리하도록 설정
      public String list(Model model){      //웹 요청을 처리할 메소드, 또는 public String list(ModelMap model) 동일함
          List<Event> eventList=getOpenedEventList();
          model.addAttribute("eventList", eventList); //addAttribute(모델 데이터 이름, 입력받는 값);
          return "event/list";   //처리결과를 보여 줄 뷰 이름 리턴
}
```

컨트롤러 구현 과정(3)

- Model에 모델 데이터를 추가하는데 사용되는 메소드
  - Model addAttribute(String attrName, Object attrValue)
    - 이름이 attrName이고 값이 attrValue인 모델 속성을 추가한다.
  - Model addAllAttributes(Map<String, ?> attributes)
    - 맵의 <키,값> 쌍들을 모델 속성 이름과 값으로 추가한다.
  - Boolean containsAttribute(String attrName)
    - 이름이 attrname인 모델 속성을 포함할 경우 true를 리턴한다.

컨트롤러 구현 과정(4)

- ModelAndView 를 사용한 모델/뷰 처리
  - 모델 설정과 뷰 이름을 합쳐 놓은 것
  - 컨트롤러에서 처리결과를 보여줄 View와 View에 전달할 값(모델)을 저장하는 용도로 사용

```
import org.springframework.web.servlet.ModelAndView;

@Controller
public class EventController{
     @RequestMapping("/event/list2") // 메소드가 처리할 요청 경로 지정, "/event/list2" 요청경로를  list2()메소드가 처리하도록 설정
      public ModelAndView list2(Model model){
          List<Event> eventList=getOpenedEventList();
          ModelAndView modelView = new ModelAndView();
          modelView.setViewName("event/list");  // 뷰 이름 지정 setViewName("viewName");
          modelView.addObject("eventList",eventList); //addObject(String name, Object value);

          return modelView
}
```

컨트롤러 구현 과정(5)

- @RequestMapping을 이용한 요청 매핑
  - 클라이언트는 URL로 요청을 전송하는데 이요청 URL을 어떤 메소드가 처리할지 여부를 결정하는 것이 @RequestMapping애노테이션임
  - 여러 속성 지정 가능

```
@Controller
public class EventController{

     @RequestMapping(value="/event/create", method=RequestMethod.POST)
     public String create(…) {
          …
}
```

  - 여러 경로를 한 메소드에서 처리하고 싶을 때..
    - @RequestMapping({"/main", "/index" })
  - 클래스(공통경로)와 메소드에 적용 가능

컨트롤러 구현 과정(6)

- @ModelAttribute을 이용한 모델 데이터 처리
- 커맨드 객체의 모델 이름 변경 가능
- 공통 모델 처리 가능
  - 코드를 중복하지 않으면서 함께 사용되는 데이터를 설정할 수 있음

ViewResolver 설정

- 컨트롤러는 결과를 생성할 뷰 이름만 지정
- 컨트롤러가 지정한 뷰 이름으로부터 응답 결과 화면을 생성하는 View객체를 구할 때 사용되는 것이 ViewResolver임
- 스프링이 제공하는 주요 ViewResolver 구현 클래스

구현 클래스	설명
InternalResourceViewResolver	뷰이름으로부터 JSP나 Tiles연동을 위한 View객체를 리턴한다
VelocityViewResolver	뷰이름으로 부터 Velocity연동을 위한 View객체를 리턴한다
VelocityLayoutViewResolver	VelocityViewResolver와 동일한 기능을 제공하며 추가로 Velocity의 레이아웃 기능을 제공한다
BeanNameViewResolver	뷰 이름과 동일한 이름을 갖는 빈 객체를 View객체로 사용한다

ViewResolver 인터페이스

- ViewResolver는 뷰 이름과 지역화를 위한 Locale을 파라미터로 전달 받으며, 매핑되는 View객체를 리턴
- 매핑되는 View 객체가 존해하지 않으면 null을 리턴함

```
pacakage org.springframework.web.servlet;
import java.util.Locale;

public interface ViewResolver{
        View resolveViewName(String viewName, Locale locale) throws Exception;
}
```

컨트롤러 메소드의 리턴 타입

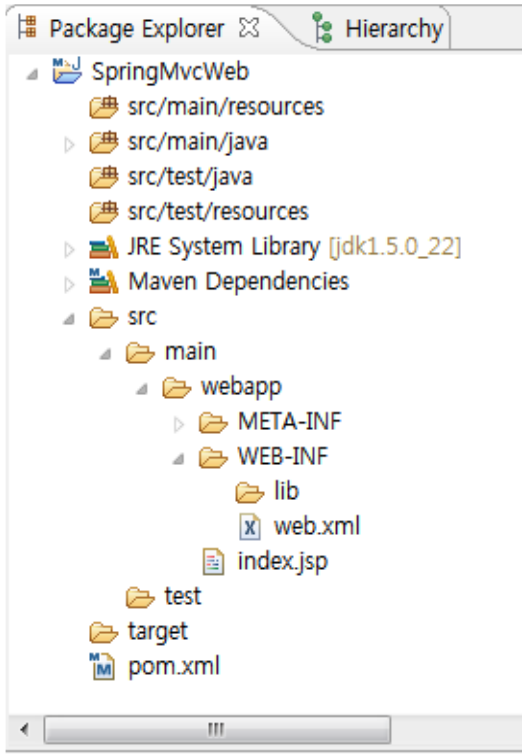
리턴 타입	설명
ModelAndView	뷰 정보 및 모델 정보를 담고 있는 ModelAndView 객체
Model	뷰에 전달할 객체 정보를 담고 있는 Model을 리턴한다. 이때 뷰 이름은 요청 URL로부터 결정된다.(RequestToViewNameTranslator)
Map	뷰에 전달할 객체 정보를 담고 있는 Map을 리턴한다. 이때 뷰 이름은 요청 URL로부터 결정된다.(RequestToViewNameTranslator)
String	뷰 이름을 리턴한다.
View 객체	View 객체를 직접 리턴, 해당 View 객체를 이용해서 뷰를 생성한다.
void	메서드가 ServletResponse나 HttpServelResponse 타입의 파라미터를 갖는 경우 메서드가 직접 응답을 처리한다고 가정한다. 그렇지 않을 경우 요청 URL로부터 결정된 뷰를 보여준다.(RequestToViewNameTranslator)
@ResponseBody 어노테이션 적용	메소드에서 @ResponseBody 어노테이션이 적용된 경우, 리턴 객체를 HTTP 응답으로 전송한다. HttpMessageConverter를 이용해서 객체를 HTTP응답 스트림으로 변환한다.

View객체

- 응답 결과를 생성하는 역할
- 모든 뷰 클래스는 View인터페이스를 구현하고 있음

메이븐 프로젝트 생성

- 메이븐(Maven)
  - 빌드 툴로서 빌드, 배포, 테스트, 의존성 관리, 프로젝트 모듈화 등을 지원하는 도구
- 프로젝트 구조
  - src/main/java : 배포시에 필요한 소스
  - src/main/resources : 배포시에 필요 자원
  - src/test/java : 테스트 소스
  - src/test/resources : 테스트시에 필요한 자원
  - src/main/webapp : 컨텍스트 루트
  - src/target : 빌드 결과물 생성 폴더



메이븐 프로젝트 생성

- 메이븐 작업 컨텍스트 설정
  - pom.xml
- 서블릿 설정 및 컨텍스트 설정
  - /WEB-INF/web.xml