Spring MVC : 기본구조

생성일	@2022년 8월 18일 오전 11:59
태그	

1. 프로젝트 구동의 시작?

web.xml에서 시작한다.

-. 프로젝트 구동 시, 가장 먼저 실행되는 ContextLoaderListener가 등록되어있는데, 실행 시 **WebApplicationContext**를 만든다.

→ WebApplicationContext는 contextConfigLocation (즉 root-context.xml)에서 정의한 스프링의 빈 객체들이 관리 될 공간이다.

-. root-context.xml에 정의 된 객체(Bean)들이 WebApplicationContext안에 생성되서, 객체들 간의 의존성 처리가 된다.

```
<!-- 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-cl
ass>
  </listener>
```

2. DispatcherServlet 서블릿 관련 설정 동작

-. DispatcherServlet에서 XmlWebApplicationContext를 이용해서, servlet-context.xml 파일을 로딩하고 읽어온다. 이 때, servlet-context.xml에 등록되어 있는 객체(Bean)은 WebApplicationContext에 같이 저장된다. (root-context.xml 에서 생성 된 Bean 객체들과 동일한 장소에서 관리 된다.)

```
<!-- Processes application requests -->
  <!-- DispatcherServlet은 Controller역할을 하고 있다. 웹브라우저에서 온 요청을 위임하는 역할을한다.
-->
  <servlet>
    <servlet-name>appServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

  <!-- 두번째 설정파일 -->
    <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>/WEB-INF/spring/appServlet/servlet-context.xml</param-value>
    </init-param>

    <!-- Handler : 웹브라우저에서 온 요청을 실제 처리하는 컨트롤러의 메소드를 의미 -->
    <!-- NoHandlerFoundException 예외 발생 -->
    <init-param>
      <param-name>throwExceptionIfNoHandlerFound</param-name>
      <param-value>true</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>
```

root-context과 servlet-context 차이는 ?

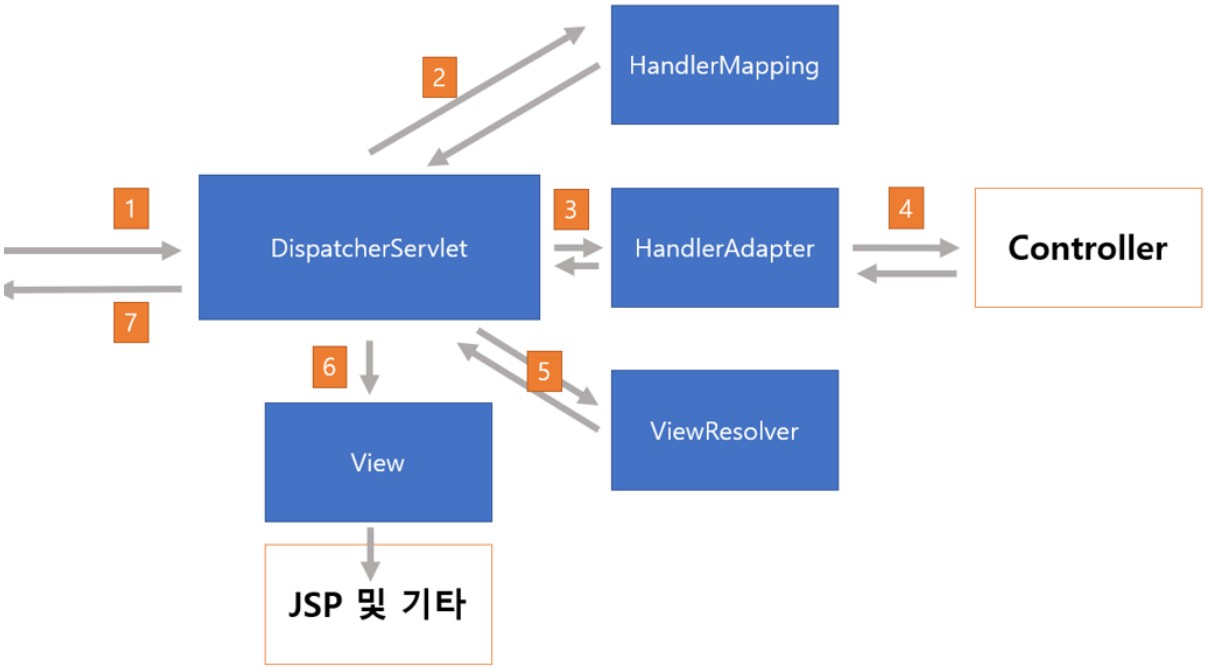
-. 결국에 등록 된 Bean 객체들은 같은 WebApplicationContext에서 관리가 된다.

-. root-context.xml은 최상위 부모 root이다. 즉, root-context에 등록 된 Bean들은 모든 자식 context에서 사용이 가능하다는 의미이다. (servlet-context는 root-context의 자식)

-. 반면에, servlet-context에 등록 된 Bean들은 해당 context에서만 사용이 가능하다.

-. 보통 DispathcerServlet에 관련 된 Bean을 등록하는데 사용한다.

2. Spring MVC의 기본 흐름



(1) Client가 요청하는 모든 Request는 DispatcherServlet에 의해 처리 된다.

(2) HandlerMapping은 Request가 어떤 컨트롤러에서 처리를 할 껀지 스캔하기 하는 역할을 한다. @RequestMapping 어노테이션 기준으로 판단하며, 처리할 컨트롤러를 찾았다면 다시, DispatcherServlet에게 리턴한다.

(3) DispatcherSerlvlet은 HandlerAdapter이용하여, 해당 컨트롤러를 실행 시킨다.

(4) Controller는 개발자의 코드 작성 공간으로 Request를 처리하는 비지니스 로직을 작성한다. 비니니스 로직의 산출물은 View에 전달해야하기 때문에 Model 객체에 담아서 전달된다.

(5) ViewResolver는 컨트롤러에서 반환 된 결과를 어떤 View에서 처리할까 결정하는 역할을 한다. 보통 이러한 설정은 servlet-context.xml에 정리를 한다.

```
<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>
```

(6) DispatcherServlet에 의해 View 역할을 하는 JSP를 생성하고,

(7) 만들어진 응답은 Client에게 보내어 진다.