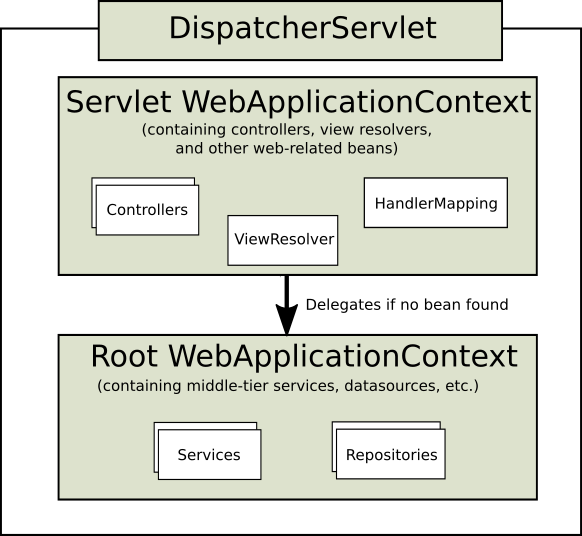
**컨텍스트 선언하기**

스프링에서의 컨텍스트(Context) 란?

스프링을 접해본 사람이라면 컨텍스트라는 단어를 들어봤을 것이다.  
무척이나 생소한 용어이고 쉽사리 이해가 가지 않는 말이기도 하다.  
이에 대한 디테일한 정의는 검색하여 찾아보는 것으로 하고,  
여기선 간략하게 "필요한 정보를 포함하고 있는 설정 파일" 정도로 넘어가겠다.

이번 글의 작성 목적은 지난번 web.xml 정리 편에서 간략하게 넘어갔던  
컨텍스트 간의 계층적 관계에 대해 더 깊이 적어보려고 한다.



스프링에서 컨텍스트의 계층 관계는 흔히 부모-자식 관계로 이루어진다고 한다.  
위 그림처럼 서블릿(자식) 컨텍스트가 루트(부모) 컨텍스트를 참조하는 관계이고,  
그 반대의 경우는 불가능한 구조이다.

**Root WebApplicationContext**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7 | <context-param>      <param-name>contextConfigLocation</param-name>      <param-value>/WEB-INF/conf/root-context.xml</param-value>  </context-param>  <listener>      <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

ContextLoaderListener를 통해 Root WebApplicationContext를 생성하는데,  
context-param Element를 통해 선언했기 때문에 Application의 전역에서 사용 가능한  
WebApplicationContext가 되는 것이다.

**Servlet WebApplicationContext**

|  |  |  |
| --- | --- | --- |
| 1  2  3  4  5  6  7  8  9 | <servlet>      <servlet-name>dispatcherServlet</servlet-name>      <servler-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>      <init-param>          <param-name>contextConfigLocation</param-name>          <param-value>/WEB-INF/spring-servlet.xml</param-value>      </init-param>      <load-on-startup>1</load-on-startup>  </servlet>  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

DispatcherServlet을 통해 Servlet WebApplicationContext를 생성하는데,  
이는 servlet Element를 통해 선언했기 때문에 해당 Servlet에서만 사용 가능한  
WebApplicationContext가 되는 것이다.

**계층 관계 컨텍스트**

위의 두 WebApplicationContext 선언 코드를 살펴보면 공통적인 부분이 존재한다.  
두 컨텍스트 모두 param-name이 contextConfigLocation이라는 것을 볼 수 있는데,  
이 부분이 바로 스프링에서 사용되는 컨텍스트 간의 계층 관계를 연결해주는 부분이다.

앞서, context-param Element를 통해 선언한 변수는 전역에서 사용된다고 하였는데,  
그렇기 때문에 Application의 전역에는 Root WebApplicationContext이 영향을 미치고 있다.  
이때, Servlet에서 Root WebApplicationContext와 같은 param-name의 Servlet WebApplicationContext을  
선언하게 된다면, 전역변수와 지역변수의 관계처럼 덮어씌워지는 것이 아닌,  
클래스의 상속 관계와 같이 계층 관계로 연결이 되는 것이다.

때문에, 계층 관계로 연결된 Root와 Servlet WebApplicationContext는 Java의 상속관계처럼  
Servlet WebApplicationContext에서는 Root WebApplicationContext를 참조 가능하지만,  
그 반대는 불가능한 구조로 구현이 가능한 것이다.

명확하게 구분되는 것은 아니지만, 이러한 이유 때문에, 주로 다음과 같이 구성된다.

* Servlet WebApplicationContext: 주로 각각의 Servlet에서 사용하는 View 자원을 구성할 때.
* Root WebApplicationContext: 주로 View 자원 이외에 공통적으로 사용이 가능한 Infra, Log 자원 등을 구성할 때.

보통 스프링 설정 파일이 최소한 두 개이상 있을 겁니다. xxx-servlet.xml 과 나머지로 나눌 수 있습니다. 그중에서 xxx-servlet.xml은 DispatcherServlet이 WebApplicationContext를 만들 때 사용하고, 나머지는 ContextLoaderListener 또는 ContextLoaderServlet이 일반적인 ApplicationContext를 만들 때 사용합니다.

이게 끝이 아닙니다. WebApplicationContext는 바로 이 ApplicationContext를 상속받아서 여러 서블릿들이 공통으로 사용하는 빈들을 사용할 수 있게 되는 겁니다. 따라서 만들어지는 순서도 중요한데, Listener가 아니라 ContextLoaderServlet을 사용했을 때는 load 머시기 설정 값에 1을 줘서 DispatcherServlet보다 먼저 만들게 해야 합니다. 그래서 WebApplicationContext를 만들 때 해당 ApplicationContext를 상속받아서 그 안에 있는 빈들을 사용할 수 있게 되겠죠.

이런 구조로 설계한 건, DispatcherServlet이 하나의 웹 애플리케이션에서 여러 개일 수 있기 때문입니다. 여러 개의 DispatcherServlet에서 공통으로 사용할 빈들을 상위에 있는 ApplicationContext에 선언해두고 공유할 수 있게 하는 거죠.

xxx-servlet.xml에는 컨트롤러 설정, 뷰 리졸버, 핸들러 맵퍼 등의 설정이 되어 있고, applicationContext.xml에는 서비스, DAO 등의 설정이 들어있습니다.