Spring ContextLoaderListener And DispatcherServlet Concepts

In Spring Web Applications, there are two types of container, each of which is configured and initialized differently. One is the "Application Context" and the other is the "Web Application Context". Lets first talk about the "Application Context".

Application Context is the container initialized by a **ContextLoaderListener** or **ContextLoaderServlet** defined in the **web.xml** and the configuration would look something like this:

```
<listener>
     <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>

<context-param>
     <param-name>contextConfigLocation</param-name>
     <param-value>classpath:*-context.xml</param-value>
</context-param>
```

In the above configuration, I am asking spring to load all files from the classpath that match *-context.xml and create an Application Context from it. This context might, for instance, contain components such as middle-tier transactional services, data access objects, or other objects that you might want to use (and re-use) across the application. There will be one application context per application.
The other context is the **"WebApplicationContext"** which is the child context of the application context. Each **DispatcherServlet** defined in a Spring web application will have an associated **WebApplicationContext.** The initialization of the **WebApplicationContext** happens like this:

```
<servlet>
      <servlet-name>platform-services</servlet-name>
      <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
      <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:platform-services-servlet.xml</param-value>
      </init-param>
      <load-on-startup>1</load-on-startup>
</servlet>
```

You provide the name of the spring configuration file as a servlet initialization parameter. What is important to remember here is that the name of the XML must be of the form

```
<servlet name>-servlet. xml.
```

In our example, the name of the servlet is `platform-services` therefore the name of our XML must be `platform-services-servlet.xml` .

Whatever beans are available in the ApplicationContext can be referred to from each WebApplicationContext. It is a best practice to keep a clear separation between middle-tier services such as business logic components and data access classes (that are typically defined in the ApplicationContext) and web-related components such as controllers and view resolvers (that are defined in the WebApplicationContext per Dispatcher Servlet).

There is a list to understand ApplicationContexts and WebApplicationContexts

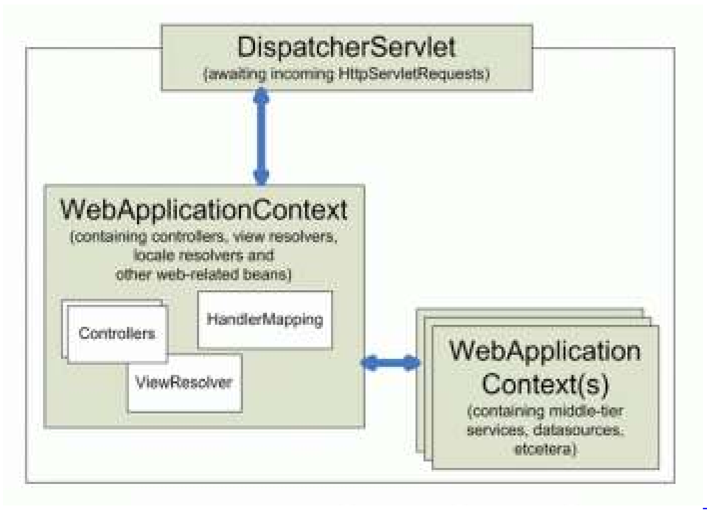
a. **Application-Contexts** are hierarchial and so are **WebApplicationContexts.** Refer documentation here (http://static.springsource.org/spring/docs/3.1.x /javadoc-api/org/springframework/web/context/ContextLoader.html)

b. **ContextLoaderListener** creates a root web-application-context for the web-application and puts it in the **ServletContext.** This context can be used to load and unload the spring-managed beans ir-respective of what technology is being used in the controller layer(Struts or Spring **MVC).**

c. **DispatcherServlet** creates its own **WebApplicationContext** and the handlers/controllers/view-resolvers are managed by this context.

d. When **ContextLoaderListener** is used in tandem with **DispatcherServlet,** a root web-application-context is created first as said earlier and a child-context is also created by **DispatcherSerlvet** and is attached to the root application-context. Refer documentation here (http://static.springsource.org/spring /docs/2.5.5/api/org/springframework/web/servlet/DispatcherServlet.html) .

Refer to the diagram below from the Spring documentation (http://static.springsource.org/spring/docs/2.5.x/reference/mvc.html#mvc-servlet) .


DispatcherServlet
(awaiting incoming HttpServletRequests)
WebApplicationContext
(containing controllers, view resolvers, locale resolvers and other web-related beans)
Controllers
HandlerMapping
ViewResolver
WebApplication Context(s)
(containing middle-tier services, datasources, etcetera)
(http://schoudari.files.wordpress.com/2012/07/mvc-context-reference.gif)

QUESTION

In official doc of ContextLoaderListener says it is to start WebApplicationContext . Regarding WebApplicationContext , api says **Interface to provide configuration for a web application .** But i am not able to understand what i am achieving with ContextLoaderListener which internally init the WebApplicationContext ?

As per my understanding, ContextLoaderListener reads the spring configuration file (with value given against contextConfigLocation in web.xml), parse it and loads the singleton bean defined in that config file. Similarly when we want to load prototype bean, we will use same webapplication context to load it. So we initialize the webapplication with ContextLoaderListener so that we read/parse/validate the config file in advance and whenever we wan to inject dependency we can straightaway do it without any delay. Is this understanding correct?

ANSWER

Your understanding is correct. The `ApplicationContext` is where your Spring beans live. The purpose of the `ContextLoaderListener` is two-fold:

1) to tie the lifecycle of the `ApplicationContext` to the lifecycle of the `ServletContext`
2) to automate the creation of the `ApplicationContext` , so you don't have to write explicit code to do create it - it's a convenience function

Another convenient thing about the `ServletContextListener` is that it creates a `WebApplicationContext` and `WebApplicationContext` provides access to the `ServletContext` `ServletContextAware` beans and the `getServletContext` method.

QUESTION

Difference between applicationContext.xml and spring-servlet.xml in Spring

Are `applicationContext.xml` and `spring-servlet.xml` related anyhow in spring framework? Will the properties files declared in `applicationContext.xml` be available to `DispatcherServlet` ? On a related note, why do I need a `*-servlet.xml` at all ? Why is `applicationContext.xml` alone insufficient?

ANSWER

Spring lets you define multiple contexts in a parent-child hierarchy.

The `applicationContext.xml` defines the beans for the "root webapp context", i.e. the context associated with the webapp.

The `spring-servlet.xml` (or whatever else you call it) defines the beans for one servlet's app context. There can be many of these in a webapp, one per Spring servlet (e.g. `spring1-servlet.xml` for servlet `spring1` , `spring2-servlet.xml` for servlet `spring2`).
Beans in `spring-servlet.xml` can reference beans in `applicationContext.xml` , but not vice versa.

All Spring MVC controllers must go in the `spring-servlet.xml` context.

In most simple cases, the `applicationContext.xml` context is unnecessary. It is generally used to contain beans that are shared between all servlets in a webapp. If you only have one servlet, then there's not really much point, unless you have a specific use for it.

QUESTION

What is the difference between ApplicationContext and WebApplicationContext in Spring MVC?

What is the difference between Application Context and Web Application Context?

I am aware that `WebApplicationContext` is used for Spring MVC architecture oriented applications?

I want to know what is the use of `ApplicationContext` in MVC applications? And what kind of beans are defined in `ApplicationContext` ?

ANSWER

```
public interface WebApplicationContext extends ApplicationContext {
    ServletContext getServletContext();
}
```

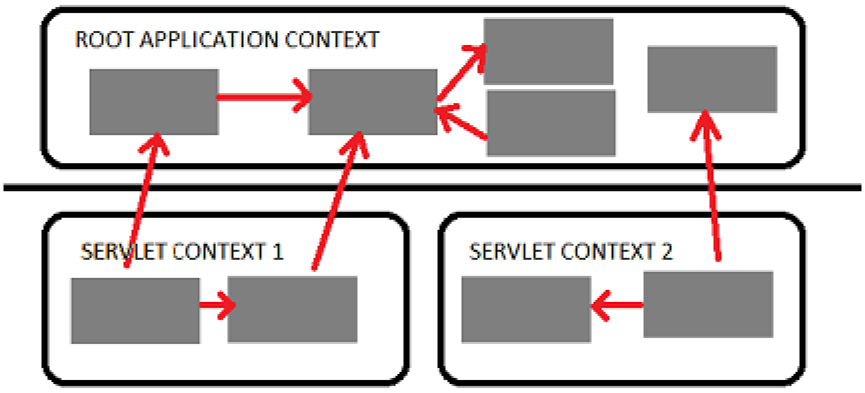
Beans, instantiated in **WebApplicationContext** will also be able to use **ServletContext** if they implement **ServletContextAware interface**

```
package org.springframework.web.context;
public interface ServletContextAware extends Aware {
     void setServletContext(ServletContext servletContext);
}
```

There many things possible to do with the **ServletContext** instance, for example accessing **WEB-INF** resources(xml configs and etc.) by calling the **getResourceAsStream()** method.Typically all application contexts defined in **web.xml** in a servlet Spring application are Web Application contexts, this goes both to the root webapp context and the servlet's app context.

Also, depending on web application context capabilities may make your application a little harder to test, and you may need to use MockServletContext (http://static.springsource.org/spring/docs/1.1.4/api/org/springframework/mock/web/MockServletContext.html) class for testing.

Difference between servlet and root context Spring allows you to build multilevel application context hierarchies, so the required bean will be fetched from the parent context if it's not present in the current aplication context. In web apps as default there are two hierarchy levels, root and servlet contexts:



such thing allows you to run some services as the singletons for the entire application(Spring Security beans and basic database access services typically reside here) and another as separated services in the corresponding servlets to avoid name clashes between beans. For example one servlet context will be serving the web pages and another will be implementing a stateless web service.
This two level separation comes out of the box when you use the spring servlet classes: to configure the root application context you should use *context-param* tag in your web.xml

```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/root-context.xml
            /WEB-INF/applicationContext-security.xml
    </param-value>
</context-param>
```

(the root application context is created by ContextLoaderListener (http://static.springsource.org/spring/docs/3.0.x/api/org/springframework/web/context

/ContextLoaderListener.html) which is declared in web.xml)

```
<listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
 </listener>
```

and servlet tag for the sevlet application contexts

```
<servlet>
   <servlet-name>myservlet</servlet-name>
   <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
   <init-param>
      <param-name>contextConfigLocation</param-name>
      <param-value>app-servlet.xml</param-value>
   </init-param>
</servlet>
```

please note that if init-param will be omitted, then spring will use myservlet-servlet.xml in this example.