DZone

Consider Replacing Spring XML Configuration with JavaConfig

by Nicolas Frankel MVB · Mar. 10, 13 · Java Zone

Learn how to troubleshoot and diagnose some of the most common performance issues in Java today. Brought to you in partnership with AppDynamics.

Spring articles are becoming a trend on this blog, I should probably apply for a SpringSource position  Colleagues of mine sometimes curse me for my stubbornness in using XML configuration for Spring. Yes, it seems so 2000's but XML has definite advantages:

1. Configuration is centralized, it's not scattered among all different components so you can have a nice overview of beans and their wirings in a single place
2. If you need to split your files, no problem, Spring let you do that. It then reassembles them at runtime through internal `<import>` tags or external context files aggregation
3. Only XML configuration allows for explicit wiring – as opposed to autowiring. Sometimes, the latter is a bit too magical for my own taste. Its apparent simplicity hides real complexity: not only do we need to switch between by-type and by-name autowiring, but more importantly, the strategy for choosing the relevant bean among all eligible ones escapes but the more seasoned Spring developers. Profiles seem to make this easier, but is relatively new and is known to few
4. Last but not least, XML is completely orthogonal to the Java file: there's no coupling between the 2 so that the class can be used in more than one context with different configurations

The sole problem with XML is that you have to wait until runtime to discover typos in a bean or some other stupid boo-boo. On the other side, using Spring IDE plugin (or the integrated Spring Tools Suite) definitely can help you there.

An interesting alternative to both XML and direct annotations on bean classes is JavaConfig, a former separate project embedded into Spring itself since v3.0. It merges XML decoupling advantage with Java compile-time checks. JavaConfig can be seen as the XML file equivalent, only written in Java. The whole documentation is of course available online, but this article will just let you kickstart using JavaConfig. As an example, let us migrate from the following XML file to a JavaConfig

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.2.xsd">
```

```
    <bean id="button" class="javax.swing.JButton">
        <constructor-arg value="Hello World" />
    </bean>

    <bean id="anotherButton" class="javax.swing.JButton">
        <property name="icon" ref="icon" />
    </bean>

    <bean id="icon" class="javax.swing.ImageIcon">
        <constructor-arg>
            <bean class="java.net.URL">
              <constructor-arg value="http://morevaadin.com/assets/images/learning_vaadin_cover.png"
            </bean>
        </constructor-arg>
    </bean>
</beans>
```

The equivalent file is the following:

```
import java.net.MalformedURLException;
import java.net.URL;

import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class MigratedConfiguration {

    @Bean
    public JButton button() {

        return new JButton("Hello World");
    }

    @Bean
    public JButton anotherButton() {

        return new JButton(icon());
    }

    @Bean
    public Icon icon() throws MalformedURLException {

        URL url = new URL("http://morevaadin.com/assets/images/learning_vaadin_cover.png");

        return new ImageIcon(url);
    }
}
```

Usage is simpler than simple: annotate the main class with `@Configuration` and individual producer methods with `@Bean` . ~~The only drawback, IMHO, is that it uses autowiring. Apart from that,~~ It just

works.

Note that in a Web environment, the web deployment descriptor should be updated with the following lines:

```
<context-param>
    <param-name>contextClass</param-name>
    <param-value>org.springframework.web.context.support.AnnotationConfigWebApplicationContext</para
</context-param>
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>com.packtpub.learnvaadin.springintegration.SpringIntegrationConfiguration</param-va
</context-param>
```

Sources for this article are available in Maven/Eclipse format here.

To go further:

- Java-based container configuration documentation
- `AnnotationConfigWebApplicationContext` JavaDoc
- `@ContextConfiguration` JavaDoc (to configure Spring Test to use JavaConfig)

Like This Article? Read More From DZone

SpringMVC4 + Spring Data JPA + SpringSecurity configuration using JavaConfig

This Week in Spring: Announcements Galore, Spring Cloud Contracts, and Integrating the Database

This Week in Spring

Free DZone Refcard
Getting Started With Vaadin Framework 8

Topics: JAVA , SPRING , JAVACONFIG

Java Partner Resources

Top 10 Java Performance Problems
AppDynamics