Managing Configurations using Spring

If you use commons-configuration in several places within your application, particularly when you reuse the same files backing the configurations, you will may notice that you have the same code chunks sprinkled around your code. Being the good Java developer you are, you want to reduce copy-and-pasted code, and refactor it to provide a standard way at getting a Configuration you build. Being the good Java developer, you also are using the wonderful Spring framework.

Without further ado, here is a way of using Spring to manage your configurations. In this example, we want to use a few PropertiesConfigurations, as well as a MapConfiguration, and use these together to build a CompositeConfiguration.

In your spring context, you would have the following beans:

```
    <bean id="buildProperties" class="org.apache.commons.configuration.PropertiesConfiguration">
        <constructor-arg index="0"><value>build.properties</value></constructor-arg>
    </bean>

    <bean id="projectProperties" class="org.apache.commons.configuration.PropertiesConfiguration">
        <constructor-arg index="0"><value>project.properties</value></constructor-arg>
    </bean>

    <bean id="defaultProperties" class="org.apache.commons.configuration.MapConfiguration">
            <constructor-arg index="0">
                    <map>
                        <entry key="fooServiceBean" value="mockFooService"/>
                        <entry key="barServiceBean" value="mockBarService"/>
                        <entry key="dbUsername" value="baz"/>
                        <entry key="dbPassword" value="aieeeeeeeeeeee"/>
                        <entry key="supportEmail" value="baz@booyah.com"/>
                    </map>
            </constructor-arg>
    </bean>

    <bean id="compositeProperties" class="org.apache.commons.configuration.CompositeConfiguration">
        <constructor-arg index="0">
            <!-- The order that these beans are defined is important when properties are defined in multiple
Configurations -->
            <list>
                    <ref bean="defaultProperties"/>
                <ref bean="projectProperties"/>
                <ref bean="buildProperties"/>
            </list>
        </constructor-arg>
    </bean>
```

So now, you can inject this configuration into your other beans, or get at the CompositeConfiguration directly using an ApplicationContext.

Spring Modules

An alternative to integrate Commons Configuration with Spring is to use the addon from the Spring Modules project, see :

https://springmodules.dev.java.net/docs/reference/0.8/html_single/#commons