Spring

This article will describe/discuss the Spring Framework.

Application-context.xml

The application-context.xml file is the starting point for your spring application. It allows you to define your imports and beans. Here is an example application-context.xml file:
application-context.xml

```
<?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.xsd">

    <import resource="classpath:spring/vault-commons.xml" />
    <import resource="classpath:spring/vault-security.xml" />
    <import resource="classpath:spring/dao/storage.xml" />
    <import resource="classpath:spring/datasource/amazon-storage.xml" />
    <import resource="classpath:spring/datasource/postgres-datasource.xml" />
    <import resource="classpath:spring/rotation-service.xml" />

</beans>
```

Above we are including the rotation-service.xml file which is as follows:
rotation-service.xml

```
<?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.xsd">

    <bean id="tenantKeyRotationService" class="com.irdeto.placodermi.vault.rotation.service.
TenantKeyRotationService" >
        <constructor-arg name="tenantKeyDao" ref="tenantKeyDao" />

    </bean>

</beans>
```

In the above we have defined a bean with a name of tenantKeyRotationService which is of class com.irdeto.placodermi.vault.rotation.service.TenantKeyRotationService. We are also performing Constructor injection, injecting other beans which are defined in other application-context xml files. Here we are injecting tenantKeyDao into our tenantKeyRotationService class.

Loading Application Context programatically

Here we can see how to programatically load our spring application-context.xml file. In the following example, we are loading Beans.xml which defines our bean texteditor.

main class

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {
   public static void main(String[] args) {

      ApplicationContext context = new ClassPathXmlApplicationContext("Beans.xml");

      TextEditor te = (TextEditor) context.getBean("textEditor");
      te.spellCheck();
   }
}
```

Defining Profiles

We can define profiles for which beans would get instantiated.
application-context.xml

```
<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.xsd"
        >
    ...


    <beans profile="postgres">
        <bean id="postgresInmemoryProcess" class="com.irdeto.placodermi.vault.test.helper.
LocalPostgresTestHelper">
            <constructor-arg value="${test.inmemory.postgres.port}" />
        </bean>
        <import resource="classpath:spring/test-postgres-init.xml"/>
        <import resource="classpath:spring/datasource/postgres-datasource.xml"/>
    </beans>


    ...


</beans>
```

Above you should notice that we specify the beans tag and specify a profile called postgres. We can now tell our application or test to run using a list of profiles by adding the following env variable:

-Dspring.profiles.active=postgres,kms

Configuring Variables

We can define variables to use within our application by specifying the **org.springframework.beans.factory.config.PropertyPlaceholderConfigurer** bean in our application-context.xml file.

application-context.xml

```
<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.xsd">

    ...
    <bean class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
        <property name="locations">
            <list>
                <value>classpath:config/test-vault.properties</value>
            </list>
        </property>
    </bean>
</beans>
```

In the above, we have defined a property file which would be found in resources/config/test-vault.properties and would look like the following:

```
signingkey.default.lifetime.minutes=262974
accesstoken.default.lifetime.minutes=262974
vault.java.options=-server -Xms1g -Xmx1g
```

Variables defined this way can be referenced in other application-context.xml files using the following syntax: ${property}. For example:

```
<?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.xsd">

   <bean id="dateUtils" class="com.irdeto.placodermi.vault.commons.date.DateUtils" />

   <bean id="vaultConfiguration" class="com.irdeto.placodermi.vault.commons.configuration.VaultConfiguration">
      <property name="defaultSigningKeyLifetime" value="${signingkey.default.lifetime.minutes}" />
      <property name="defaultAccessTokenLifetime" value="${accesstoken.default.lifetime.minutes}" />
   </bean>

</beans>
```