Tips and Tricks with Spring and Sakai

Cris J. Holdorph

holdorph@uncion.net

Agenda

- Concepts
  - Spring Configuration Files
  - Spring AOP – ProxyFactoryBean
- Tips and Tricks
  - Using JMX to manage your Application
  - Timing your Application
  - Using a Method Cache

Spring Configuration Files

- Beans are defined in Spring configuration files
- Beans reference other Beans
- References are most often made against Interfaces rather then implementation classes

Sample Spring Configuration

```
<beans>
    <bean id="MyService"
class="org.myapp.services.MyService">
        <property name="enforceAuthorization" value="true" />
        <property name="sourceDirectory" value="/opt/myapp" />
        <property name="backupDirectory" value="/tmp" />
        <property name="maximum" value="100" />
    </bean>
</beans>
```

Spring AOP

- AOP allows you to add code to your application without modifying the original code
- Spring AOP prefers to accomplish this with *Proxy* objects
- *Proxy* objects use a Decorator Pattern to Wrap the original *Target* object and add code
- The *Proxy* is configured to implement one or more interfaces of the original *Target* object

ProxyFactoryBean

- Use the Spring class *ProxyFactoryBean* to create Spring AOP *Proxy* objects
- Rename the original *Target* object
- Configure the *Proxy* object
  - Specify the original *Target* object
  - Specify the interfaces the *Proxy* should implement
  - Name the *Proxy* object the name of the original *Target* object

Sample Proxy Configuration

```
<beans>
    <bean id="MyServiceTarget" class="org.myapp.services.MyService">
        <property name="enforceAuthorization" value="true" />
        <property name="sourceDirectory" value="/opt/myapp" />
        <property name="backupDirectory" value="/tmp" />
        <property name="maximum" value="100" />
    </bean>
    <bean id="MyService"
class="org.springframework.aop.framework.ProxyFactoryBean">
        <property name="proxyInterfaces">
            <value>org.myapp.services.MyService</value>
        </property>
        <property name="target" ref="MyServiceTarget"/>
        <property name="interceptorNames">
            <list><value>advisor</value></list>
        </property>
    </bean>
</beans>
```

Using JMX to Manage Sakai

- Define your *original bean*
- Define a *MBeanExporter*
  - Optionally limit the methods/attributes exposed
- Add JMX JVM args to *CATALINA_OPTS*
- Start Tomcat
- Start JConsole
- Connect to Tomcat PID

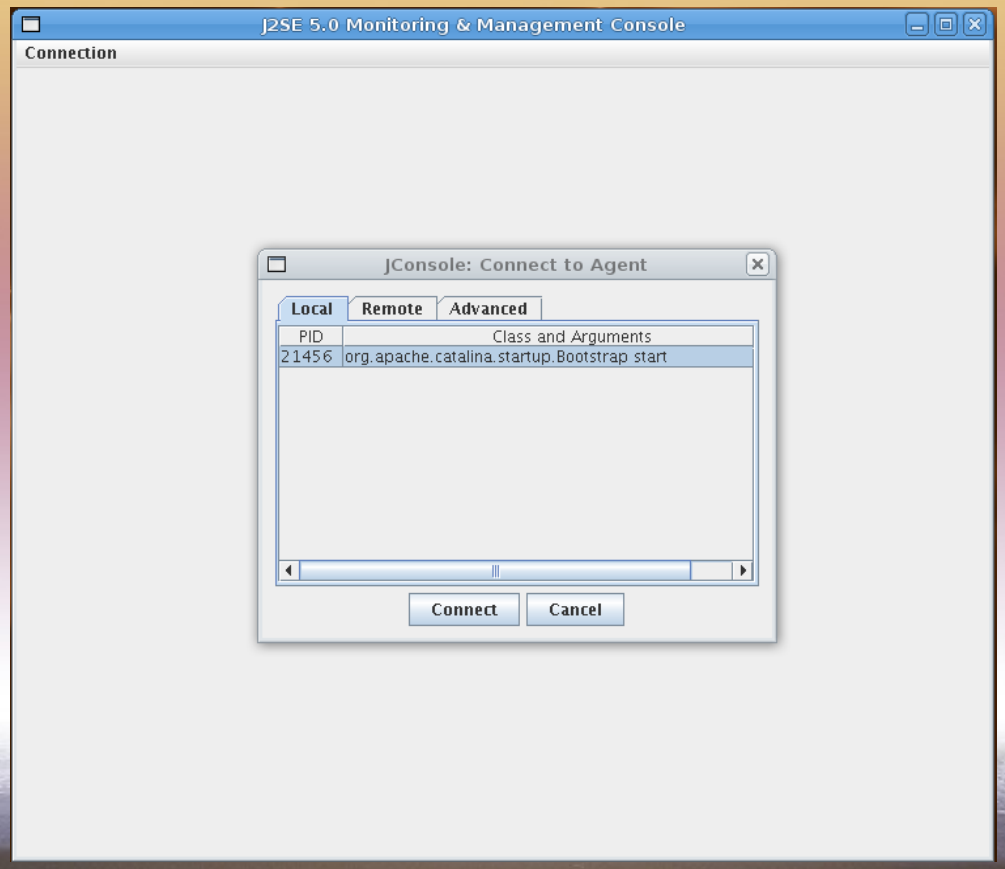
JMX Spring Configuration

```
<beans>
    <bean id="MyService" class="org.myapp.services.MyService">
        <property name="minimum" value="20" />
        <property name="maximum" value="100" />
    </bean>
    <bean id="exporter" class="org.springframework.jmx.export.MBeanExporter">
        <property name="beans">
            <map><entry key="bean:name=myService" value-ref="MyService"/></map>
        </property>
        <property name="assembler"> <bean
class="org.springframework.jmx.export.assembler.MethodNameBasedMBeanInfoAssembler">
                <property name="managedMethods">
                    <value>getMaximum,setMaximum</value>
                </property>
        </bean></property>
    </bean>
</beans>
```

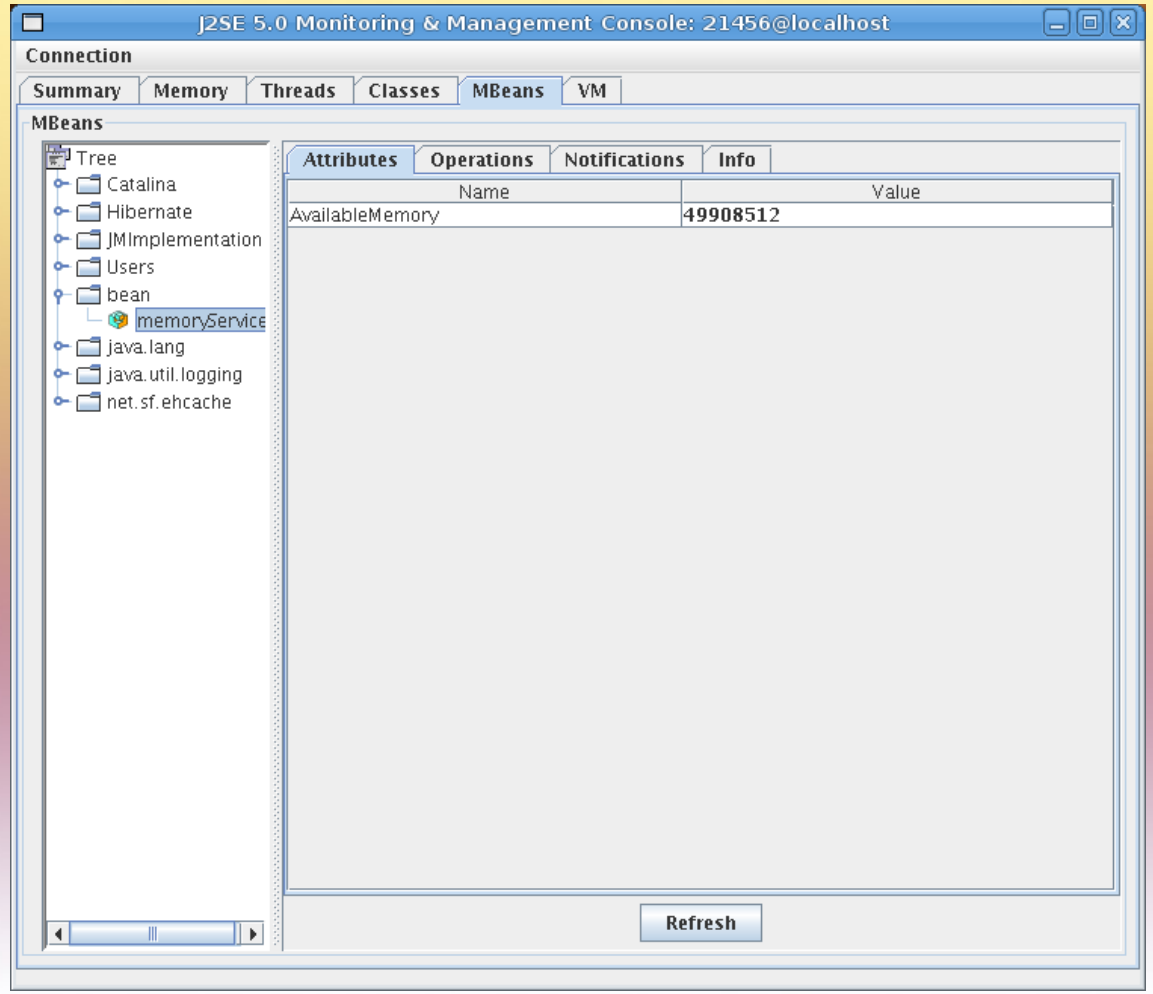
JMX Spring Configuration

```
$ export CATALINA_OPTS="-Dcom.sun.management.jmxremote.port=18080
-Dcom.sun.management.jmxremote.authenticate=false"


$ jconsole &
```



JConsole


J2SE 5.0 Monitoring & Management Console: 21456@localhost
Connection
Summary
Memory
Threads
Classes
MBeans
VM
MBeans
Tree
Catalina
Hibernate
JMImplementation
Users
bean
memoryService
java.lang
java.util.logging
net.sf.ehcache
Attributes
Operations
Notifications
Info
Name
Value
AvailableMemory
49908512
Refresh

Timing Your Application

- Define your *original bean*
- Define a *PerformanceMonitorInterceptor*
- Define a *RegexpMethodPointcutAdvisor*
- Define a *ProxyFactoryBean* to proxy your *original bean* and apply your *Advisor*
- Set the Log level for the *PerformanceMonitorInterceptor* to **TRACE**

```
log4j.logger.org.springframework.aop.interceptor.Perfor
   manceMonitorInterceptor=TRACE
```

Timing Configuration

```
<beans>
    <bean id="MyServiceTarget" class="org.myapp.services.MyService">
        <property name="enforceAuthorization" value="true" />
        <property name="sourceDirectory" value="/opt/myapp" />
        <property name="backupDirectory" value="/tmp" />
        <property name="maximum" value="100" />
    </bean>

    <bean id="timingLogger"
class="org.springframework.aop.interceptor.PerformanceMonitorInterceptor"/>

    <bean id="timingAdvisor"
class="org.springframework.aop.support.RegexpMethodPointcutAdvisor">
        <property name="advice" ref="timingLogger"/>
        <property name="patterns">
            <list><value>.*</value></list>
        </property>
    </bean>
```

Timing Configuration (cont.)

```
    <bean id="MyService"
class="org.springframework.aop.framework.ProxyFactoryBean">
        <property name="proxyInterfaces">
            <value>org.myapp.services.MyService</value>
        </property>
        <property name="target"><ref local="MyServiceTarget"/></property>
        <property name="interceptorNames">
            <list>
                <value>timingAdvisor</value>
            </list>
        </property>
    </bean>
</beans>
```

Timing Configuration (Spring 2)

```
<beans "insert xsd declarations here" >
    <bean id="MyService" class="org.myapp.services.MyService">
        <property name="enforceAuthorization" value="true" />
        <property name="sourceDirectory" value="/opt/myapp" />
        <property name="backupDirectory" value="/tmp" />
        <property name="maximum" value="100" />
    </bean>

    <bean id="timingAdvice"
class="org.springframework.aop.interceptor.PerformanceMonitorInterceptor"/>

    <aop:config>
        <aop:advisor pointcut="execution(* org.myapp.services.MyService.*(..))"
                     advice-ref="timingAdvice"/>
    </aop:config>
</beans>
```

Using a Method Cache

- Write a MethodInterceptor (Java code)
- Define your original bean
- Define the ehcache components
- Define the method cache interceptor using the ehcache components you defined
- Define the regex pointcut advisor referencing the interceptor you defined
- Define the proxy factory bean pointing to the original bean and the regex pointcut advisor you defined

(HashMap) MethodInterceptor

```
public class MethodCacheInterceptor implements MethodInterceptor {
    private Map<String,Object> cache = new HashMap<String,Object>();

    public Object invoke(MethodInvocation invocation) throws Throwable {
        String targetName = invocation.getThis().getClass().getName();
        String methodName = invocation.getMethod().getName();
        Object[] arguments = invocation.getArguments();
        String cacheKey = targetName + methodName + arguments;
        Object result = cache.get(cacheKey);
        if (result == null) {
            result = invocation.proceed();
            cache.put(cacheKey,result);
        }
        return result;
    }
}
```

Ehcache Spring Configuration

```
<beans>
    <bean id="cacheManager"
class="org.springframework.cache.ehcache.EhCacheManagerFactoryBean">
        <property name="configLocation">
            <value>classpath:ehcache.xml</value>
        </property>
    </bean>

    <bean id="methodCache"
class="org.springframework.cache.ehcache.EhCacheFactoryBean">
        <property name="cacheManager">
            <ref local="cacheManager" />
        </property>
        <property name="cacheName">
            <value>org.myapp.services.MyService.METHOD_CACHE</value>
        </property>
    </bean>
</beans>
```

Ehcache.xml

```
<ehcache>
    <diskStore path="java.io.tmpdir"/>

    <cache name="org.myapp.services.MyService.METHOD_CACHE"
        maxElementsInMemory="300"
        eternal="false"
        timeToIdleSeconds="500"
        timeToLiveSeconds="500"
        overflowToDisk="false“ />

</ehcache>
```

MethodInterceptor Spring conf.

```
<beans>
        <bean id="methodCacheInterceptor"
class="org.myapp.services.MethodCacheInterceptor">
          <property name="cache"><ref local="methodCache" /></property>
          <property name="caching" value="true"/>
        </bean>

        <bean id="methodCacheAdvisor"
class="org.springframework.aop.support.RegexpMethodPointcutAdvisor">
          <property name="advice">
            <ref local="methodCacheInterceptor" />
          </property>
          <property name="patterns">
            <list><value>.*getSomeValues.*</value></list>
          </property>
        </bean>
</beans>
```

ProxyBean Configuration

```
    <bean id="MyService"
class="org.springframework.aop.framework.ProxyFactoryBean">
        <property name="proxyInterfaces">
            <value>org.myapp.services.MyService</value>
        </property>
        <property name="target"><ref local="MyServiceTarget"/></property>
        <property name="interceptorNames">
            <list>
                <value>methodCacheAdvisor</value>
            </list>
        </property>
    </bean>
</beans>
```

Resources

- My Unicon Blog
  - (JMX) http://www.unicon.net/node/614
  - (Timing) http://www.unicon.net/node/765
  - (Method cache) *Coming Soon!*
- Pro Spring, Spring in Action Books
- Spring Forums
  - http://forum.springframework.org/

Questions?