eclipseCON™ 2008

How To Distribute Spring Dynamic Modules For OSGi Using The Newton Project

Spring Onto Newton

Spring Onto Newton

- Spring is one of the most advanced POJO frameworks available in the market place today

Spring Onto Newton

- Spring is one of the most advanced POJO frameworks available in the market place today
- Spring Dynamic Modules adds dynamicity to the Spring life-cycle

Spring Onto Newton

- Spring is one of the most advanced POJO frameworks available in the market place today
- Spring Dynamic Modules adds dynamicity to the Spring life-cycle
- Spring Dynamic Modules are automatically OSGi enabled bundles

Spring Onto Newton

- Spring is one of the most advanced POJO frameworks available in the market place today
- Spring Dynamic Modules adds dynamicity to the Spring life-cycle
- Spring Dynamic Modules are automatically OSGi enabled bundles
- Newton can seamlessly deploy Spring (and any other) OSGi enabled bundle across an enterprise environment.

How Spring bundles are interpreted by Newton

```
bundle.jar
`-- META-INF
    |-- MANIFEST.MF
    |-- newton
    |   `-- sca.template
    `-- spring
        |-- bean-osgi.xml
        `-- bean.xml
```

How Spring bundles are interpreted by Newton

bundleA.jar -> bean.xml

```
<beans>
  <bean name="BeanA" class="example.Bean" />
</beans>
```

bundleB.jar -> bean.xml

```
<beans>
  <bean name="BeanB" class="example.Bean" />
</beans>
```

How Spring bundles are interpreted by Newton

bundleA.jar -> bean-osgi.xml

```
<beans>
  <osgi:reference
    id="nextBean"
    interface="example.Bean"
    filter="(tag=b)"
    cardinality="0..1">

    <osgi:listener
      bind-method="bindBean"
      unbind-method="unbindBean"
      ref="BeanA"/>
  </osgi:reference>

  <osgi:service ref="BeanA" interface="example.Bean">
    <service-properties>
      <entry key="tag" value="a"/>
    </service-properties>
  </osgi:service>
</beans>
```

How Spring bundles are interpreted by Newton

```
bundle(A|B).jar -> sca.template
```

```
<composite name="bean">
  <component name="bean">
    <description>A Spring Bean</description>
    <implementation.spring />
  </component>
</composite>
```

So What Does Newton Add?

How to configure Spring bundles

bean.xml

```
<beans>
  <property-placeholder persistent-id="example.bean"/>
  <bean name="Bean" class="example.Bean" >
    <property name="beanName" value="${beanName}"/>
  </bean>
</beans>
```

How to configure Spring bundles

bean.xml

```
<composite name="bean">
  <component name="bean">
    <property name="service.pid" value="example.bean”/>
    <property name="beanName" value="${system#user.name}"/>
    <implementation.spring />
  </component>
</composite>
```

How to promote OSGi level services and references for remote discovery/communication

```
<osgi:reference
    id="nextBean"
    interface="example.Bean"
    filter="(newton.sca.reference=nextBean)"
    cardinality="0..1" />

<service-properties>
  <entry key="newton.sca.service" value="springBean"/>
</service-properties>
```

How to promote OSGi level services and references

```
<composite name="bean">
  <reference name="nextBean" >
    <interface.java interface="example.Bean" />
    <binding.rmi filter="(tag=b)" />
  </reference>
  <service name="bean">
    <interface.java interface="example.Bean" />
    <binding.rmi >
      <attribute name="tag" value="a" type="string" />
    </binding.rmi>
  </service>
  <component name="bean">
    <description>A Spring Bean</description>
    <reference name="nextLink" />
    <service name="link" />
    <implementation.spring />
  </component>
</composite>
```

How to promote OSGi level services and references

```
<composite name="bean">
  <reference name="nextBean" >
    <interface.java interface="example.Bean" />
    <binding.rmi filter="(tag=b)" />
  </reference>
  <service name="bean">
    <interface.java interface="example.Bean" />
    <binding.rmi >
      <attribute name="tag" value="a" type="string" />
    </binding.rmi>
  </service>
   <component name="bean">
     <description>A Spring Bean</description>
     <reference name="nextLink" />
     <service name="link" />
     <implementation.spring />
   </component>
</composite>
```

How to promote OSGi level services and references

```
<composite name="bean">
  <reference name="nextBean" >
    <interface.java interface="example.Bean" />
    <binding.rmi filter="(tag=b)" />
  </reference>
   <service name="bean">
     <interface.java interface="example.Bean" />
     <binding.rmi >
       <attribute name="tag" value="a" type="string" />
     </binding.rmi>
   </service>
  <component name="bean">
    <description>A Spring Bean</description>
    <reference name="nextLink" />
    <service name="link" />
    <implementation.spring />
  </component>
</composite>
```

How to promote OSGi level services and references

```
<composite name="bean">
  <reference name="nextBean" >
    <interface.java interface="example.Bean" />
    <binding.rmi filter="(tag=b)" />
  </reference>
  <service name="bean">
    <interface.java interface="example.Bean" />
    <binding.rmi >
      <attribute name="tag" value="a" type="string" />
    </binding.rmi>
  </service>
  <component name="bean">
    <description>A Spring Bean</description>
    <reference name="nextLink" />
    <service name="link" />
    <implementation.spring />
  </component>
</composite>
```

Stretching Spring Across The Fabric

```
<system name="beans" boundary="fabric">
  <description>spring DM for OSGi</description>
  <system.composite name="beanA"
                    bundle="example.bundleA"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <system.composite name="beanB"
                    bundle="example.bundleB"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <replication.handler name="scale"
     type="org.cauldron.newton.system.replication.ScalableReplicationHandler">
    <property name="scaleFactor" value="0.5" type="float" />
  </replication.handler>
</system>
```

Deploying Spring To The Fabric

```
<system name="beans" boundary="fabric">
  <description>spring DM for OSGi</description>
  <system.composite name="beanA"
                    bundle="example.bundleA"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <system.composite name="beanB"
                    bundle="example.bundleB"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <replication.handler name="scale"
     type="org.cauldron.newton.system.replication.ScalableReplicationHandler">
    <property name="scaleFactor" value="0.5" type="float" />
  </replication.handler>
</system>
```

Deploying Spring To The Fabric

```
<system name="beans" boundary="fabric">
  <description>spring DM for OSGi</description>
   <system.composite name="beanA"
                     bundle="example.bundleA"
                     template="bean"
                     version="1.0" >
      <replicator name="scale50" />
    </system.composite>

  <system.composite name="beanB"
                    bundle="example.bundleB"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <replication.handler name="scale"
     type="org.cauldron.newton.system.replication.ScalableReplicationHandler">
    <property name="scaleFactor" value="0.5" type="float" />
  </replication.handler>
</system>
```


 |

Deploying Spring To The Fabric

```
<system name="beans" boundary="fabric">
  <description>spring DM for OSGi</description>
  <system.composite name="beanA"
                    bundle="example.bundleA"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <system.composite name="beanB"
                    bundle="example.bundleB"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <replication.handler name="scale"
     type="org.cauldron.newton.system.replication.ScalableReplicationHandler">
    <property name="scaleFactor" value="0.5" type="float" />
  </replication.handler>
</system>
```


 |

Deploying Spring To The Fabric

```
<system name="beans" boundary="fabric">
  <description>spring DM for OSGi</description>
  <system.composite name="beanA"
                    bundle="example.bundleA"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <system.composite name="beanB"
                    bundle="example.bundleB"
                    template="bean"
                    version="1.0" >
    <replicator name="scale50" />
  </system.composite>

  <replication.handler name="scale"
      type="org.cauldron.newton.system.replication.ScalableReplicationHandler">
    <property name="scaleFactor" value="0.5" type="float" />
  </replication.handler>
</system>
```

Advanced Features

- Contracts
- Features
- Assessment
- Failure recovery
- Presence
- Pluggable Bindings
- Pluggable Repliction Strategies
- ...and much more...

Summary

Summary

- Spring applications are easy to build and quick to test due to their POJO nature

Summary

- Spring applications are easy to build and quick to test due to their POJO nature
- Newton adds the ability to dynamically deploy and configure Spring applications

Summary

- Spring applications are easy to build and quick to test due to their POJO nature
- Newton adds the ability to dynamically deploy and configure Spring applications
- Newton can seamlessly add remote discovery and marshaling to Spring-DM components

Summary

- Spring applications are easy to build and quick to test due to their POJO nature
- Newton adds the ability to dynamically deploy and configure Spring applications
- Newton can seamlessly add remote discovery and marshaling to Spring-DM components
- Newton can dynamically scale your Spring application and monitor the nodes to which it is deployed for failures.

Questions?

- Contact info:
  - dave.savage@paremus.com
  - http://newton.codecauldron.org
  - http://www.paremus.com