# OSGI Blueprint

那么这个OSGI service是从哪里来的呢?下面我们介绍一下Blueprint如何注册一个OSGI service。

首先我们在java代码中定义一个interface,然后用一个类去实现它，就和面向接口开发一样。最后，我们就通过Blueprint的service节点发布这个OSGI service。

1. <?xml version="1.0" encoding="UTF-8"?>
2. <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0 http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd" **default**-timeout="0">
3. <!--最简单的osgi服务注册 -->
4. <bean id="coder" **class**="com.ponder.CoderImpl"/>
5. <service id="CoderService" ref="coder" **interface**="com.ponder.ICoder"/>
6. <!--Embeded bean的osgi服务注册 -->
7. <service id="CoderService2" **interface**="com.ponder.ICoder">
8. <bean **class**="com.ponder.CoderImpl"/>
9. </service>
10. <!--带osgi服务属性的服务-->
11. <service id="CoderService3" **interface**="com.ponder.ICoder">
12. <service-properties>
13. <entry key="param01" value="val01"/>
14. <entry key="param02" value="val02"/>
15. <entry key="param03">
16. <array value-type="java.lang.String">
17. <value>val03-1</value>
18. <value>val03-2</value>
19. <value>val03-3</value>
20. </array>
21. </entry>
22. </service-properties>
23. <bean **class**="com.ponder.CoderImpl"/>
24. </service>
25. <!-- 没将接口抽离出来的osgi服务,会将com.ponder.CoderImpl里的所有**public**方法都作为服务的方法 -->
26. <service id="CoderService3"  **interface**="com.ponder.CoderImpl">
27. <bean **class**="com.ponder.CoderImpl"/>
28. </service>
29. </blueprint>

<!--最简单的osgi服务注册 -->

<bean id="coder" class="com.ponder.CoderImpl"/>

<service id="CoderService" ref="coder" interface="com.ponder.ICoder"/>

Service ID可以不指定，会自动生成，规则是接口名称的小写，如“icode”、

ref 为内部引用。如<service id="CoderService" ref="coder" interface="com.ponder.ICoder"/>

reference为外部引用。如

<**reference** **id**="dataBroker"

**interface**="org.opendaylight.controller.md.sal.binding.api.DataBroker"

**odl**:**type**="default" />

在上例中，我们举了三个发布OSGI service的例子，最简单的例子就先用bean节点定义那个服务的实现类，并给这个bean定义一个id（“coder”），然后在service节点用interface属性指明对应的接口，并用ref属性来引用刚才定义的那个bean。这样Blueprint container就可以往OSGI framework里注册发布这个服务了。

第二个例子中，则是将bean节点直接嵌入service节点中，而不需用bean的id和service节点属性ref了。

第三个例子则给service添加了service-property,这个service-property和bean节点的property不同，它不出现在java代码里，只是在Blueprint往OSGI framework里注册这个服务时，以map的形式附带上这些service-property。这些service-property可以在服务被引用时，作为filter的条件。

服务被发布后，在Blueprint里，可以用reference节点来引用用这些服务。

1. <?xml version="1.0" encoding="UTF-8"?>
2. <blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.osgi.org/xmlns/blueprint/v1.0.0 http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd" **default**-timeout="0">
3. <!--引用osgi服务,并注入bean(com.ponder.Processor)里 -->
4. <reference id="coderService" filter=”（param01=val02）” **interface**="com.ponder.ICoder" timeout="0"/>
5. <bean id="processor" **class**="com.ponder.Processor">
6. <!--与这里对应，类com.ponder.Processor里应定义有以下属性：
7. **private** com.ponder.ICoder coder;
8. 并包含其setter。
9. -->
10. <property name="coder" ref="coderService"/>
11. </bean>
13. </blueprint>

使用reference引用发布的service服务，并通过property或者argument引用到类中。

<**bean** **id**="arpHandlerProvider"

**class**="org.opendaylight.l2switch.arphandler.core.ArpHandlerProvider"

**init-method**="init" **destroy-method**="close">

<**argument** **ref**="dataBroker" />

<**argument** **ref**="notificationProviderService" />

<**argument** **ref**="salFlowService" />

<**argument** **ref**="packetHandlerService" />

<**argument** **ref**="arpHandlerConfig" />