OSGi with Apache Karaf

Planning

- OSGi in a nutshell
- Apache Karaf
  (with exercise)
- Blueprint
  (with exercise)

Planning

- OSGi in a nutshell
- Apache Karaf
  (with exercise)
- Blueprint
  (with exercise)

OSGi in a Nutshell

- What is OSGi?
- OSGi Core
  - OSGi Bundles
    - Sidetrack: maven-bundle-plugin
  - OSGi Service Registry
- OSGi Compendium

What is OSGi?

- What is OSGi?
  - OSGi Alliance (http://www.osgi.org)
    - initially focused on embedded/networked devices
    - standard for service-oriented, component-based Java applications
  - Latest spec is R6
    - Core specification
    - Compendium specification
    - Enterprise specification
    - Residential specification

Bundles

- What is (in) an OSGi bundle?
  - JAR file containing classes and resources
  - Extra manifest headers
    - Identification and description
    - Classloading
    - Activation

Bundles

- OSGi bundle manifest headers
  - Identification and description
    - Bundle-SymbolicName
    - Bundle-Name
    - Bundle-Description

```
Apache Karaf :: Kitchen :: Mexican (39)
--------------------------------
...
Bundle-ManifestVersion = 2
Bundle-Name = Apache Karaf :: Kitchen :: Mexican
Bundle-SymbolicName = be.anova.course.karaf.mexican
Bundle-Version = 1.0
...
```

Bundles

- Bundle classloading
  - Every bundle has own classloader
    - java.* and boot delegation
    - Imported packages
    - Required bundles
    - Bundle classes
    - Fragments
    - Dynamic imports

Bundles

- Classloading headers in MANIFEST.MF
  - Export-Package
  - Required-Bundle
  - Import-Package
  - DynamicImport-Package

```
TSSJS :: Kitchen :: Mexican (39)
--------------------------------
Export-Package = be.anova.course.karaf.mexican
Import-Package = be.anova.course.karaf,be.anova.course.karaf.mexican
```

Bundles

- Access classes exported by other bundles
  - Import-Package
    - Specifies list of packages to import
  - Require-Bundle
    - Imports all packages from the required bundle
  - DynamicImport-Package
    - Dynamically add imports when required

Bundles

- Versions and version ranges
  - minimum version
  - version range
    - include version with [and]
    - exclude version with (and)

Examples

[1.2.3, 4.5.6)	1.2.3 <= x < 4.5.6
[1.2.3, 4.5.6]	1.2.3 <= x <= 4.5.6
(1.2.3, 4.5.6)	1.2.3 < x < 4.5.6
(1.2.3, 4.5.6]	1.2.3 < x <= 4.5.6
1.2.3	1.2.3 <= x

Bundle

- System bundle (0)
  - JRE classes
  - OSGi spec classes
  - Some Karaf-specific classes
  - Configuration
    - etc/config.properties
    - etc/jre.properties

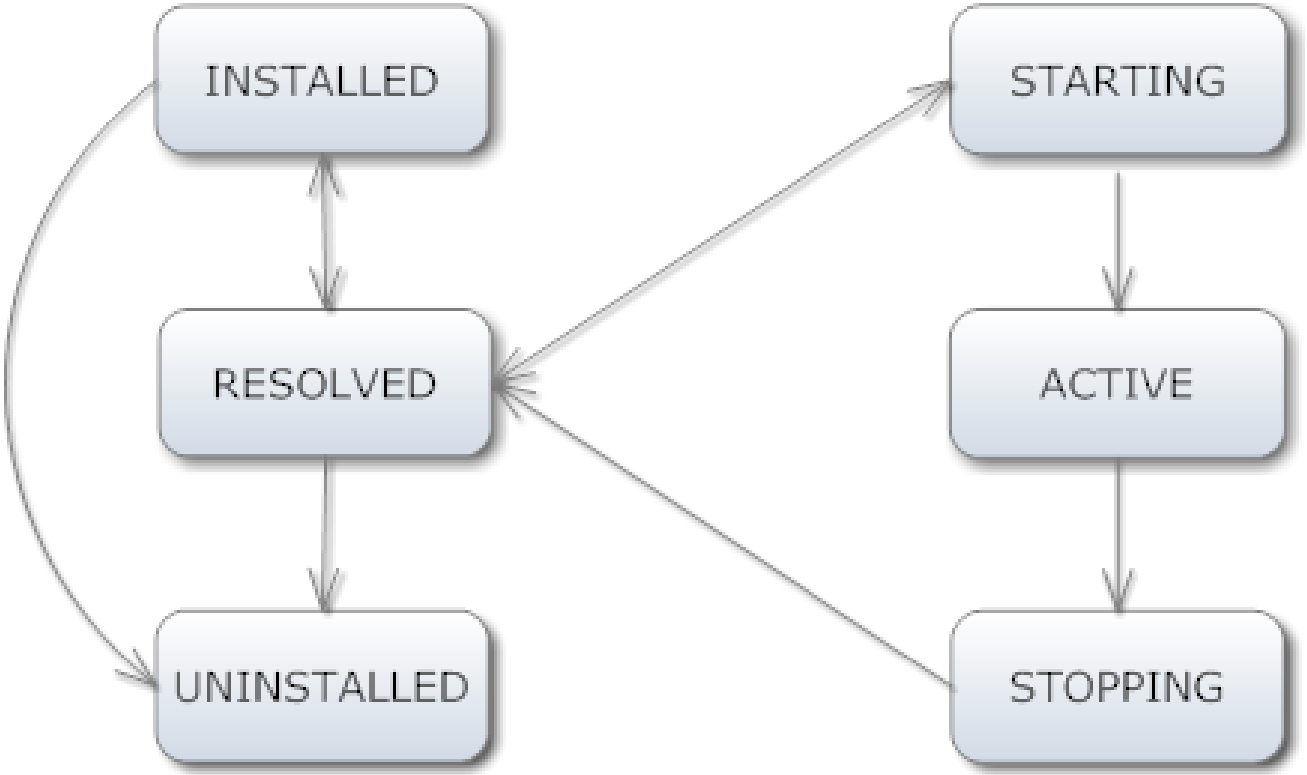
Bundles

- Example: manifest information

```
camel-ognl (195)
----------------
Bundle-ManifestVersion = 2
Bundle-Name = camel-ognl
Bundle-SymbolicName = org.apache.camel.camel-ognl
Export-Service =
    org.apache.camel.spi.LanguageResolver;language=ognl
Export-Package =
  org.apache.camel.language.ognl;uses:="org.apache.camel.language,
  org.apache.camel,ognl,org.apache.camel.impl,
  org.apache.camel.spi";version="2.16.3"
Ignore-Package = org.apache.camel.language.ognl
Import-Package =
  ognl,org.apache.camel;version="[2.16,2.17)",
  org.apache.camel.impl;version="[2.16,2.17)",
  org.apache.camel.language;version="[2.16,2.17)",
  org.apache.camel.spi;version="[2.16,2.17)"
```

Bundles

- OSGi Bundle states



Sidetrack: maven-bundle-plugin

- Create OSGi bundles with Maven
  - Packaging bundle
  - Plugin maven-bundle-plugin
    - Based on bnd.bndtools.org

Sidetrack: maven-bundle-plugin

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0">
    <!-- SNIP -->
    <packaging>bundle</packaging>

    <!-- SNIP -->
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.felix</groupId>
                <artifactId>maven-bundle-plugin</artifactId>
                <version>3.2.0</version>
                <extensions>true</extensions>
                <configuration>
                    <instructions>
                       <!--  more about this on the next slide -->
                    </instructions>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

Sidetrack: maven-bundle-plugin

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0">
  <!-- SNIP -->
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.felix</groupId>
        <artifactId>maven-bundle-plugin</artifactId>
        <configuration>
          <instructions>
            <Import-Package>
              com.fasterxml.jackson.dataformat*;resolution:=optional,
              akka.dispatch,
              *
            </Import-Package>
            <Private-Package>be.anova.smx.impl.*</Private-Package>
            <Export-Package>be.anova.smx</Export-Package>
          </instructions>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

Sidetrack: maven-bundle-plugin

- Common pattern
  - Define plugin in parent POM
  - Define properties to configure plugin
  - Configure plugin through properties in all other modules

OSGi Service Registry

- Allows connecting bundles together with POJO
  - Specified as Java Interface
    - Optional: extra properties
  - Provider bundle
    - implement interface
    - register service
  - Client bundle
    - find in registry
    - react when registered/unregistered

OSGi Service Registry

- Core interfaces
  - ServiceRegistration
  - ServiceReference
  - ServiceTracker
- Often registered/used using
  - Spring DM
  - Declarative Services
  - Blueprint

OSGi Service Registry API

- Register a service

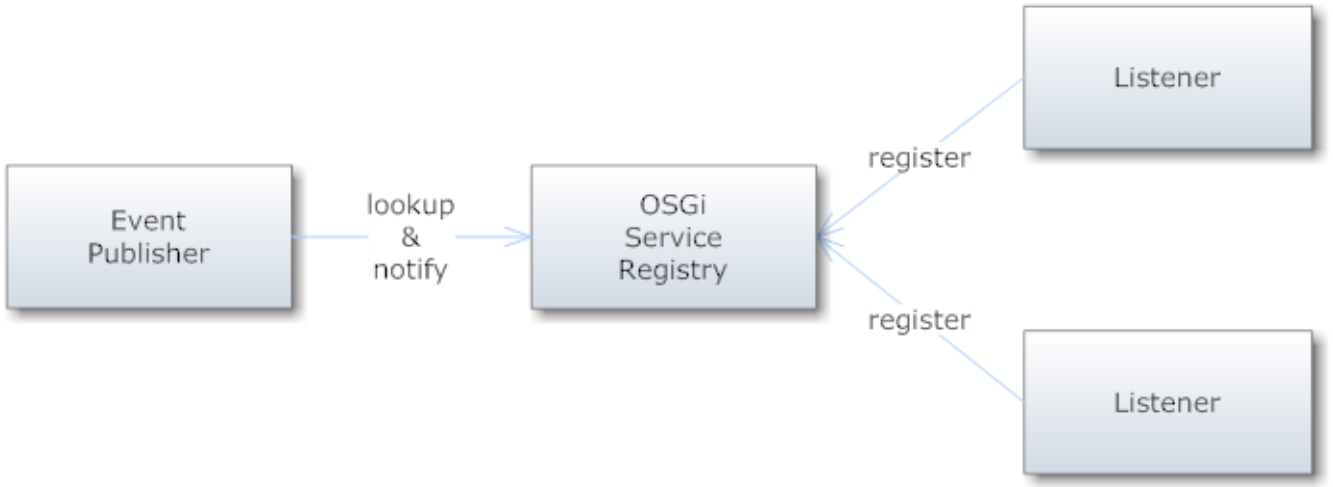
```
public class PublishServiceActivator implements BundleActivator {

  private ServiceRegistration registration;

  public void start(BundleContext context) throws Exception {
    SomeService service = new SomeServiceImpl();
    Dictionary props = new Hashtable();
    props.put("info", "extra service info");

    registration =
      context.registerService(SomeService.class, service, props);
  }

  public void stop(BundleContext context) throws Exception {
    if (registration != null) {
      registration.unregister();
    }
  }
}
```

- Get a service from the registry

```
public class ConsumeServiceActivator implements BundleActivator {

  private ServiceReference<SomeService> reference;

  public void start(BundleContext context) throws Exception {
    reference = context.getServiceReference(SomeService.class);
    SomeService service = context.getService(reference);
  }

  public void stop(BundleContext context) throws Exception {
    if (reference != null) {
      context.ungetService(reference);
    }
  }

}
```

OSGi Service Registry

- Example: Whiteboard pattern
  - all event listeners register themselves
  - event publisher tracks registrations to send event



OSGi Compendium

- Specification with additional services
  - Log Service
  - HTTP Service
  - Configuration Admin Service
  - Declarative Services
  - Event Admin Service
  - Blueprint Container
  - ...

OSGi Enterprise

- Specifications
  - JTA
  - JMX
  - JDBC
  - JNDI
  - JPA
  - ...
- ServiceMix uses Apache Aries implementations

OSGi Residential

- Specifications
  - UPnP Device Service
  - EnOcean Device Service
  - Device Abstraction Layer Service
  - USB Information Device Category Service
  - Serial Device Service
  - Resource Monitoring Service

Planning

- OSGi in a nutshell
- Apache Karaf
- Blueprint
  (with exercise)

Apache Karaf

- Introduction
- Command shell
- Admin service
- Feature descriptors
- Hot-deployment
- Web console

Introduction

- Apache Karaf
  - A flexible OSGi-based server runtime
  - Choice of OSGi Framework implementation:
    - Equinox
    - Apache Felix
  - Manage the container using
    - Command shell
    - Web console
    - JMX

Introduction

- Some other features
  - Provisioning through feature descriptors
  - Applications
    - Spring DM and Blueprint
  - Hot-deployment
  - Manage child instances
  - Failover using file or JDBC lock

Command shell

- Based on Apache Felix Gogo
  - Implementation of OSGi RFC-147
  - Uses a <group>:<command> syntax
- Command shell can be accessed
  - Directly when starting the container
  - Using an SSH client
  - From the web console
- Unix-like TAB-completion, |, grep, cat, ...

Command shell – system

- Commands to interact with OSGi Framework/Apache Karaf container
  - system:shutdown to restart/stop container
  - system:framework to debug OSGi Framework
  - system:start-level to get/set start level
  - system:name and system:version
  - system:property to set/set system property

Command shell – bundle

- Commands to interact with bundles
  - bundle:install
  - bundle:start, bundle:stop
  - bundle:update
  - bundle:uninstall
- Install from URL
  - file:, http:, mvn:

Command shell – bundle

```
karaf@root> bundle:install mvn:commons-pool/commons-pool/1.5.2
Bundle ID: 40

karaf@root> bundle:list
START LEVEL 100
   ID   State         Blueprint      Level  Name
[   0] [Active     ] [            ] [    0] System Bundle (2.0.4)
...
[  37] [Active     ] [            ] [   60] Commons DBCP (1.4)
[  40] [Installed  ] [            ] [   60] Commons Pool (1.5.2)

karaf@root> bundle:sta<TAB>
bundle:start          bundle:start-level

karaf@root> bundle:start 40

karaf@root> bundle:headers 40

Export-Package = org.apache.commons.pool;version="1.5.2"
Import-Package = org.apache.commons.pool;version="1.5.2"
...

karaf@root> bundle:uninstall 40
```

Command shell – bundle

- Other useful commands
  - bundle:headers
  - bundle:capabilities and bundle:requirements
  - bundle:classes and bundle:find-class
  - bundle:dynamic-import
  - bundle:services
  - bundle:tree-show
  - bundle:watch

Sidetrack: mvn: URLs

- Syntax:
  - mvn:<g>/<a>/<v>/<type>/<classifier>

```
mvn:commons-io/commons-io/2.5
mvn:org.duracloud/durastore-web/4.1.5/war
mvn:org.apache.shiro/shiro-feature/1.3.2/xml/features
```

- Repositories:
  - system folder
  - etc/org.ops4j.pax.url.cfg

Command shell – config

- Interact with ConfigAdmin service
  - config:list
  - for changing the runtime config
    - config:edit
    - config:property-set, config:property-delete, config:property-append and config:property-list
    - config:update or config:cancel

Command shell – config

```
karaf@root> config:edit org.apache.karaf.shell.ssh

karaf@root> config:property-set sshPort 8102

karaf@root> config:update

karaf@root> config:list
...
Pid:            org.apache.karaf.shell.ssh
BundleLocation: null
Properties:
   org.apache.felix.karaf.features.configKey = org.apache.felix.karaf.shell.ssh
   service.pid = org.apache.felix.karaf.shell.ssh
   sshPort = 8102
   sshRealm = karaf
   felix.fileinstall.filename = org.apache.felix.karaf.shell.ssh.cfg
...
```

Command shell

- Some other examples
  - `jaas:` shell to manage user credentials
  - `log:` shell interacts with log service
  - `ssh:` shell to work with SSH client and server

Command shell

```
karaf@root> jaas:realm-manage --realm karaf

karaf@root> jaas:user-list
User Name | Group      | Role
-------------------------------------
smx       | admingroup | admin
smx       | admingroup | manager
smx       | admingroup | viewer
smx       | admingroup | webconsole
smx       | admingroup | systembundles
karaf     | admingroup | admin
karaf     | admingroup | manager
karaf     | admingroup | viewer
karaf     | admingroup | webconsole
karaf     | admingroup | systembundles

karaf@root> log:set DEBUG

karaf@root> log:display

karaf@root> ssh:ssh localhost
Connecting to host localhost on port 22
Login:
```

Admin Service

- Karaf allows creating child instances
  - share the system directory (with the base bundles)
  - each has own etc, hotdeploy, data, ...
  - automatically assigned a new ssh port
- Can be administered through
  - instance: command shell
  - web console

Admin Service

- instance: command shell
  - instance:create <name>
  - instance:start, instance:stop
  - instance:destroy
  - instance:list
  - instance:connect

Admin Service

```
karaf@root> instance:create test

karaf@root> instance:start test

karaf@root> instance:list
SSH Port | SSH Host | RMI Registry | RMI Registry Host | RMI Server | RMI Server Host
| State   | PID   | Name
-------------------------------------------------------------------------------
-----------------------
    8101 |  0.0.0.0 |         1099 |           0.0.0.0 |      44444 |         0.0.0.0
| Started | 9378  | root
    8102 |  0.0.0.0 |         1100 |           0.0.0.0 |      44445 |         0.0.0.0
| Started | 13734 | test
```

Admin Service

```
karaf@root> instance:connect test
Connecting to host localhost on port 8102
Password:  *****
Connected

 ____                  _          __  __ _
/ ___|  ___ _ ____   _(_) ___ ___|  \/  (_)_  __
\___ \ / _ \ '__\ \ / / |/ __/ _ \ |\/| | \ \/ /
 ___) |  __/ |   \ V /| | (_|  __/ |  | | |>  <
|____/ \___|_|    \_/ |_|\___\___|_|  |_|_/_/\_\

  Apache ServiceMix (7.0.0.M2)

Hit '<tab>' for a list of available commands
and '[cmd] --help' for help on a specific command.
Hit '<ctrl-d>' or 'system:shutdown' to shutdown ServiceMix.

karaf@test>^D
```

Feature Descriptors

- Default Karaf provisioning mechanism
- XML descriptor
  - list of bundles to install
  - configuration information
  - dependencies between features

Feature Descriptors

- Example

```
<features name="karaf-1.4.0">
    <feature name="http" version="1.4.0">
        <config name="org.ops4j.pax.web">
          org.osgi.service.http.port=8181
        </config>
        <bundle>mvn:org.ops4j.pax.web/pax-web-api/0.7.2</bundle>
        <bundle>mvn:org.ops4j.pax.web/pax-web-spi/0.7.2</bundle>
        ...
    </feature>
    <feature name="webconsole" version="1.4.0">
        <feature version="1.4.0">http</feature>
        <bundle>mvn:org.apache.felix/org.apache.felix.webconsole/2.0.6</bundle>
        ...
    </feature>
    <feature name="my-feature" version="1.0.0">
        <configfile finalname="${karaf.etc}/my.own.application.cfg">
          mvn:my.own/application/1.0.0/cfg
        </config>
        ...
    </feature>
</features>
```

Feature Descriptors

- A feature can be installed
  - using the feature: command shell
  - using the web console
  - using JMX

```
karaf@root> feature:info --bundle camel-snmp
Feature camel-snmp 2.16.3
Feature contains followed bundles:
  mvn:o.a.s.bundles/o.a.s.bundles.snmp4j/2.3.4_1
  mvn:org.apache.camel/camel-snmp/2.16.3

karaf@root> feature:install obr

karaf@root> feature:uninstall spring
```

Feature Descriptors

- What's available?
  - Karaf provides a few basic features
    - wrapper, webconsole, spring, spring-dm, ...
  - Some project provide their own descriptors
    - Apache Camel: EIP-based integration framework
    - Apache CXF: web services framework
    - Apache Sling: Content-driven web framework
  - ServiceMix comes with
    - Activiti, Akka, Drools

Feature Descriptors

- Example: how-to turn Karaf into a Camel container?

```
karaf@root> feature:repo-add
                    mvn:org.apache.camel.karaf/apache-camel/2.16.3/xml/features

karaf@root> feature:list
Name                | Version          | Required | State       | Repository    |
-------------------------------------------------------------------------------
...
camel               | 2.16.3           |          | Uninstalled | camel-2.16.3  |
camel-ftp           | 2.16.3           |          | Uninstalled | camel-2.16.3  |
...

karaf@root> feature:install camel
karaf@root> feature:install camel-ftp/2.16.3
```

Hot-deployment

- Hot-deployment based on Felix FileInstall
  - Karaf supports deployment of
    - Bundles
    - Expanded bundles
    - XML files (Blueprint and Features)
  - An extensible mechanism
    - Spring XML files with Spring feature installed
    - WAR files with web feature installed
    - BPMN files with Activiti feature installed

Hot-deployment

- Example: hot-deploy a Camel route

```
<?xml version="1.0"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:camel="http://camel.apache.org/schema/spring"
       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
        http://camel.apache.org/schema/spring
          http://camel.apache.org/schema/spring/camel-spring.xsd">

  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
      <from uri="timer:camel-on-karaf?period=3000" />
      <to uri="log:camel-on-karaf"/>
    </route>
  </camelContext>

</beans>
```

Web console

- Installable as an optional feature
- Based on Apache Felix Web Console
- Extra plugins available for
  - administration of instances
  - working with features
  - access the shell

Web console: hawtio

- Modular web console for Java stuff
  - Plugins for OSGi, Camel, Karaf, ...
  - Uses Jolokia
- Installable as an optional feature

Exercise

- Start with a plain Karaf instance
- Add the webconsole feature
- Add Camel feature descriptor and feature
- Add a simple Camel route in a Spring file (using hot-deploy)
- Add hawtio feature descriptor and feature

Planning

- OSGi in a nutshell
- Apache Karaf
- Blueprint
  (with exercise)

Blueprint

- Introduction
- Working with beans
- Working with OSGi Service Registry
- Namespace handlers
- Lifecycle
- Exercise

Introduction

- OSGi standard for IoC/DI
  - Inspired by Spring DM
  - Karaf uses Aries implementation
  - Features
    - XML configuration files
    - Register beans as services in OSGi Service Registry
    - Reference other services in OSGi Service Registry
    - Extensible through custom namespaces
    - (Custom) Converters

Introduction

- Blueprint is a first-class citizen in Karaf
  - Installed by default
  - Used internally for Karaf/ServiceMix
  - Hot-deployment
    - Plain XML configuration file
    - OSGI-INF/blueprint/*.xml in bundles

Introduction

- Getting started with Blueprint
  - <blueprint /> root element with namespace http://www.osgi.org/xmlns/blueprint/v1.0.0

```
<?xml version="1.0" encoding="utf-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">

    <!-- add beans, services and references here -->

</blueprint>
```

Beans

- Bean with a default constructor
  - property set with value...
  - ... or with reference to another bean

```
<bean id="restaurant"
                class="be.anova.course.blueprint.Restaurant">

   <property name="stars" value="***"/>
   <property name="kitchen" ref="continental"/>

</bean>
```

Beans

- Bean with constructor arguments
  - argument set with value...
  - ... or with reference to another bean

```
<bean id="restaurant"
                    class="be.anova.course.blueprint.Restaurant">

    <argument value="***"/>
    <argument ref="continental"/>

</bean>
```

Beans

- Bean creation with static factory
  - class refers to static factory class
  - arguments for the factory method

```
<bean class="be.anova.course.blueprint.Kitchen"
      factory-method="createMenu">

    <argument value="chicken fajitas"/>

</bean>
```

Beans

- Bean creation with instance factory
  - factory-ref refers to factory instance
  - arguments for the factory method

```
<bean factory-ref="kitchen"
      factory-method="createMenu">

    <argument value="chicken fajitas"/>

</bean>
```

Beans

- Some other ways to specify values
  - <ref/>
  - <null/>
  - <list/>, <set/>, <array/>
  - <map/>, <props/>
  - <value/>

Beans

- Example

```
<bean class="be.anova.course.blueprint.Kitchen">
  <property name="menus">
    <list>
      <value>chicken fajitas</value>
      <ref component-id="fajitasBean"/>
    </list>
  </property>
  <property name="suggestion">
    <map>
      <entry key="main" value="burrito"/>
      <entry key="dessert" ref="capirotadaBean"/>
    </map>
  </property>
</bean>
```

OSGi Service Registry

- Interact with the OSGi Service Registry
  - register a service
  - reference a single service
  - reference a list of services
  - service reference listener

OSGi Service Registry

- Register a service
  - specify service interface for registration
  - create/refer to service implementation bean

```
<service interface="be.anova.course.blueprint.Restaurant">

   <bean class="be.anova.course.blueprint.RestaurantImpl">
       <property name="stars" value="***"/>
   </bean>

</service>
```

OSGi Service Registry

- Reference a single service
  - specify service interface
  - optionally specify filter, mandatory/optional, ...
  - can also be used as a value directly

```
<reference id="kitchen" interface="be.anova.course.blueprint.Kitchen"/>

<bean id="restaurant" class="be.anova.course.blueprint.Restaurant">
    <property name="stars" value="***"/>
    <property name="kitchen" ref="kitchen"/>
</bean>
```

OSGi Service Registry

- Reference multiple service
  - specify service interface
  - optionally specify filter, mandatory/optional, ...
  - add id to be able to reference it from other beans

```
<bean id="kitchen" class="be.anova.course.blueprint.Kitchen">

    <property name="menus">
        <reference-list interface="be.anova.course.blueprint.Menu" />
    </property>

</bean>
```

OSGi Service Registry

- Listen for service references
  - invoke listener when service registered/unregistered
  - specify listener bean and bind/unbind methods

```
<bean id="kitchen" class="be.anova.course.blueprint.Kitchen" />

<reference-list interface="be.anova.course.blueprint.Menu" >
    <reference-listener ref="kitchen"
                        bind-method="addMenu"
                        unbind-method="removeMenu"/>
</reference-list>
```

OSGi Service Registry

- 3 options to code bind/unbind method

```
public class Kitchen {

    // option 1 : service object
    public void addMenu(Menu menu) { }

    // option 2 : service object with properties
    public void addMenu(Menu menu, Map props) { }

    // option 3 : OSGi ServiceReference
    public void addMenu(ServiceReference reference) { }

}
```

Namespace handlers

- Extra functionalitity through namespace handlers
  - ConfigAdmin
  - JPA, JTA
  - Camel, CXF, ActiveMQ, ...

Namespace handlers

- Example:
  - ConfigAdmin

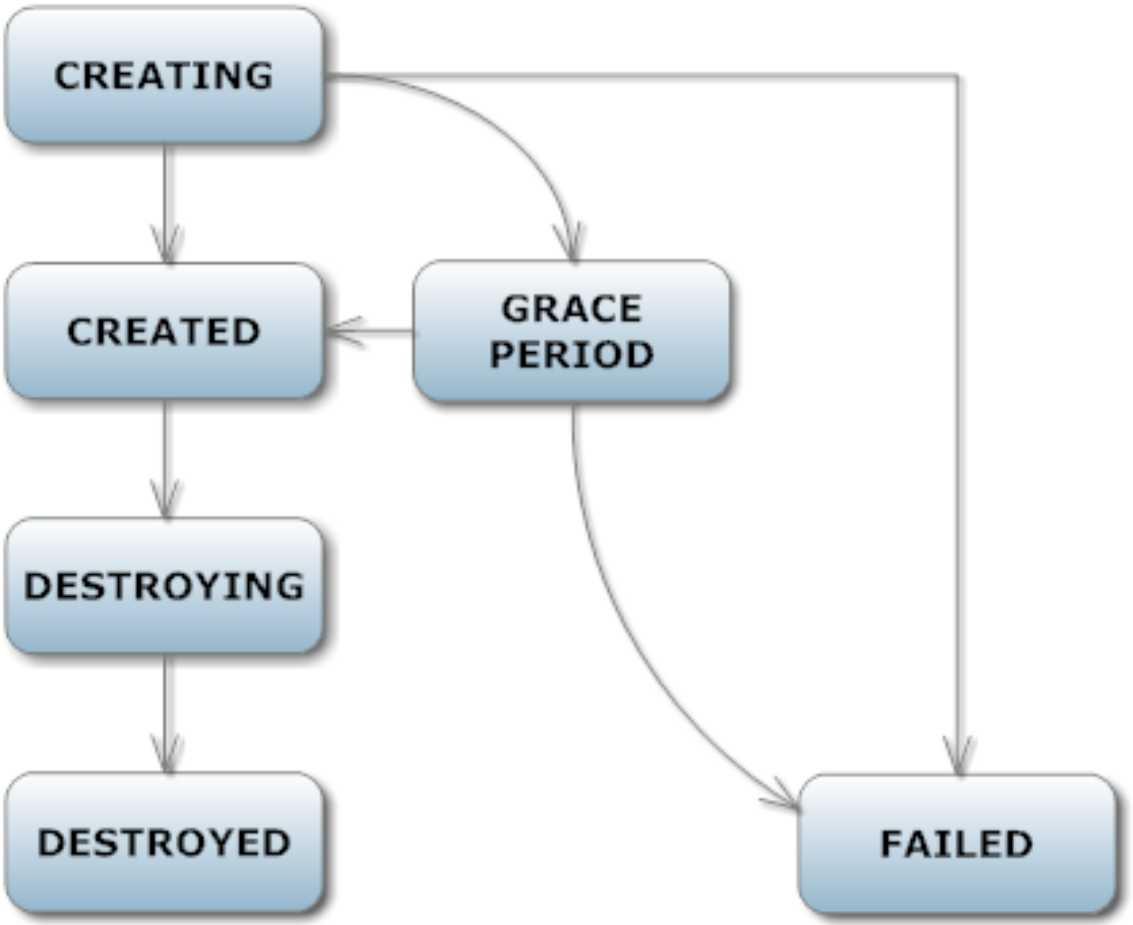
```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
     xmlns:cm="http://aries.apache.org/blueprint/xmlns/blueprint-cm/v1.1.0">

    <cm:property-placeholder persistent-id="b.a.smx.cm"
                                                update-strategy="reload">
        <cm:default-properties>
            <cm:property name="amount" value="30"/>
            <cm:property name="currency" value="EUR"/>
        </cm:default-properties>
    </cm:property-placeholder>

    <bean id="myBean" class="b.a.smx.cm.MyBean" destroy-method="close">
        <property name="amount" value="${amount}" />
        <property name="currency" value="${currency}"/>
    </bean>
</blueprint>
```

Namespace handlers

- Example:
  - Aries JPA and JTA support
  - Uses JPA's `@PersistenceContext`

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
           xmlns:tx="http://aries.apache.org/xmlns/transactions/v1.2.0"
           xmlns:jpa="http://aries.apache.org/xmlns/jpa/v2.0.0">

    <jpa:enable />

    <service ref="orderDao" interface="b.a.smx.persistence.OrderDAO" />

    <bean id="orderDao" class="be.a.smx.persistence.impl.OrderDAOImpl">
        <tx:transaction method="*"/>
    </bean>

</blueprint>
```

Lifecycle


CREATING
CREATED
GRACE PERIOD
DESTROYING
DESTROYED
FAILED

Exercise

- Make a cuisine bundle
  - provides a kitchen and a cook (staff member)
- Make a staffing bundle
  - provides staff members
- In the restaurant bundle
  - add reference to the kitchen
  - dynamically keep track of list of staff members