OSGi with Apache Felix Karaf

Planning

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

Planning

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

OSGi in a Nutshell

- What is OSGi?
- OSGi Core
  - OSGi Bundles
  - 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 R4.2
    - Core specification
    - Compendium specification
    - Enterprise 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 Felix Karaf :: Kitchen :: Mexican (39)
-------------------------------
...
Bundle-ManifestVersion = 2
Bundle-Name = Apache Felix Karaf :: Kitchen :: Mexican
Bundle-SymbolicName = be.anova.course.karaf.mexican
Bundle-Version = 1.0
...
```

Bundles

- Bundle classloading
  - System bundle classloader
    - java.* classes
    - OSGi framework classes
  - Every bundle has own classloader
    - Imported packages
    - Required bundles
    - Fragments
    - Bundle classes

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
```

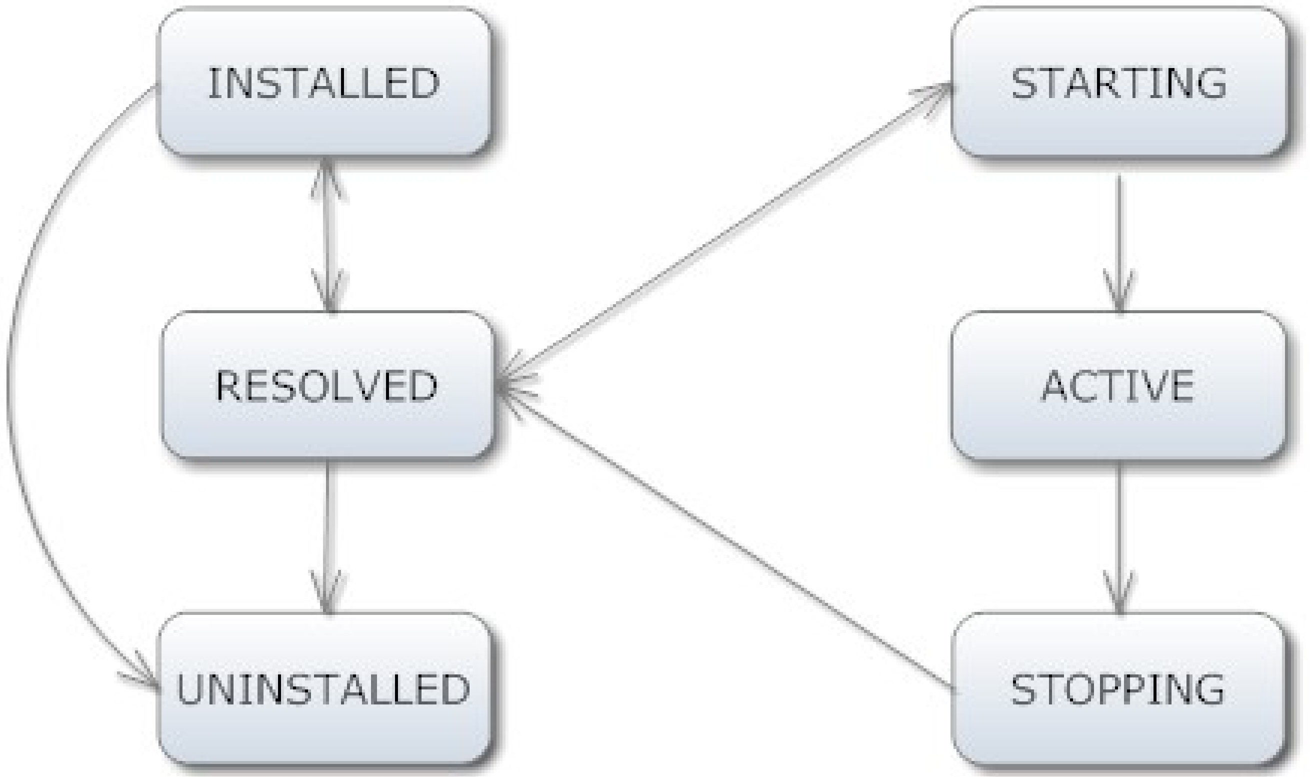
Bundles

- Example: manifest information

```
camel-ognl (195)
----------------
Bundle-ManifestVersion = 2
Bundle-Name = camel-ognl
Bundle-SymbolicName = org.apache.camel.camel-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="2.2.0"
Ignore-Package = org.apache.camel.language.ognl
Import-Package =
  ognl;version="[2.7,3)",org.apache.camel;version="[2.2,2.3)",
  org.apache.camel.impl;version="[2.2,2.3)",
  org.apache.camel.language;version="[2.2,2.3)",
  org.apache.camel.spi;version="[2.2,2.3)"
```

Bundles

- OSGi Bundle states



OSGi Service Registry

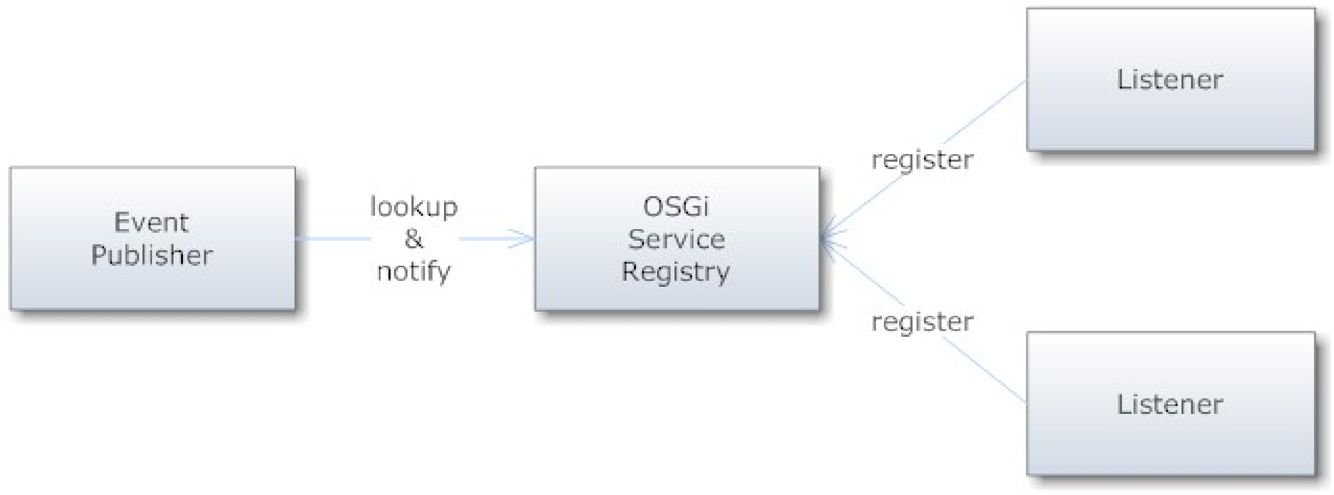
- Allows connecting bundles together with POJO
  - Specified as Java Interface
  - Provider bundle
    - implement interface
    - register service
  - Client bundle
    - find in registry
    - react when registered/unregistered

OSGi Service Registry

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

OSGi Service Registry

- 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
  - ConfigAdmin Service
  - Declarative Services
  - Blueprint Container
  - ...

Planning

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

Apache Felix Karaf

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

Introduction

- Apache Felix 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 – osgi

- Commands to interact with OSGi Framework
  - osgi:shutdown to stop container
  - osgi:list to show bundles
  - osgi:headers to show bundle metadata
  - osgi:ls <id> to show bundle services

Command shell – osgi

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

Command shell – osgi

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

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

karaf@root> osgi:start 40

karaf@root> osgi:headers 40

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

karaf@root> osgi:uninstall 40
```

Command shell – packages

- Allows interacting with OSGi PackageAdmin
  - packages:exports shows lists of exported packages
  - packages:imports shows
    - wired imports
    - bundles providing the matching export
- Difference between osgi:headers and packages:

Command shell – packages

```
karaf@root> packages:exports 15
Apache Felix Karaf :: Features Core (15):
                              org.apache.felix.karaf.features; version=1.4.0

karaf@root> packages:imports 15
System Bundle (0): org.osgi.framework; version=1.5.0
System Bundle (0): org.osgi.service.packageadmin; version=1.2.0
System Bundle (0): org.osgi.service.startlevel; version=1.1.0
System Bundle (0): javax.management; version=0.0.0
System Bundle (0): javax.management.loading; version=0.0.0
System Bundle (0): javax.xml.parsers; version=0.0.0
System Bundle (0): org.w3c.dom; version=0.0.0
System Bundle (0): org.xml.sax; version=0.0.0
OPS4J Pax Logging - API (3): org.slf4j; version=1.5.6
OPS4J Pax Logging - API (3): org.slf4j.helpers; version=1.5.6
Apache Felix Configuration Admin Service (5): org.osgi.service.cm; version=1.3.0
Apache Felix Preferences Service (6): org.osgi.service.prefs; version=1.1.0
Apache Felix Gogo Shell Runtime (25): org.osgi.service.command; version=0.2.2
Apache Felix Karaf :: Shell Console (27):
                              org.apache.felix.gogo.commands; version=0.2.2
Apache Felix Karaf :: Shell Console (27):
                              org.apache.felix.karaf.shell.console; version=1.4.0
```

Command shell – config

- Interact with ConfigAdmin service
  - config:list
  - for changing the runtime config
    - config:edit
    - config:propset, config:propdel, config:propappend
    - config:update or config:cancel

Command shell – config

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

karaf@root> config:propset sshPort 8102

karaf@root> config:update

karaf@root> config:list
...
Pid:            org.apache.felix.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
  - dev: shell holds some developer tools
  - log: shell interacts with log service
  - ssh: shell to work with SSH client and server

Command shell

```
karaf@root> dev:show-tree 37
Bundle org.apache.commons.dbcp [37] is currently ACTIVE

org.apache.commons.dbcp [37]
+- org.apache.geronimo.specs.geronimo-jta_1.1_spec [35]
+- org.apache.commons.pool [36]

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
  - admin: command shell
  - web console

Admin Service

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

Admin Service

```
karaf@root> admin:start test

karaf@root> admin:list
  Port   State       Pid  Name
[ 8103] [Starting] [ 2758] test
[ 8101] [Started ] [ 2517] root

karaf@root> admin:connect test
Connecting to host localhost on port 8103
Connected
        __ __                  ____
       / //_/____ __________ _/ __/
      / ,<  / __ `/ ___/ __ `/ /_
     / /| |/ /_/ / /  / /_/ / __/
    /_/ |_|\__,_/_/   \__,_/_/

  Apache Felix Karaf (1.4.0)

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

karaf@test> ^D
^D
```

Feature Descriptors

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

Feature Descriptors

- Example: http and webconsole features

```
<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>
</features>
```

Feature Descriptors

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

```
karaf@root> features:list
State          Version        Name       Repository
[installed  ] [2.5.6.SEC01] spring     karaf-1.4.0
[uninstalled] [1.2.0      ] spring-dm  karaf-1.4.0
[uninstalled] [1.4.0      ] wrapper    karaf-1.4.0
[uninstalled] [1.4.0      ] obr        karaf-1.4.0
...

karaf@root> features:install obr
karaf@root> features:uninstall spring
```

Feature Descriptors

- What's available?
  - Karaf provides a few basic features
    - wrapper, webconsole, spring, spring-dm, ...
  - ServiceMix 4.2.0 comes with
    - NMR/JBI support and JBI components
    - features for ActiveMQ, CXF, Pax Web, ...
  - Some project provide their own descriptors
    - Apache Camel: EIP-based integration framework
    - Apache Sling: Content-driven web framework

Feature Descriptors

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

```
karaf@root> features:addUrl
                     mvn:org.apache.camel.karaf/apache-camel/2.2.0/xml/features

karaf@root> features:list
State         Version              Name                         Repository
...
[uninstalled] [2.2.0             ] camel                        repo-0
[uninstalled] [2.2.0             ] camel-ftp                    repo-0
...

karaf@root> features:install camel
karaf@root> features:install camel-ftp 2.2.0
```

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
    - JBI artifacts with JBI feature installed
    - WAR files with web feature installed

Hot-deployment

- Example: hot-deploy a Camel route

```xml
<?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

Planning

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

Blueprint

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

Introduction

- OSGi standard for IoC/DI
  - Inspired by Spring DM (is also the RI)
  - We use Geronimo/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
  - Lifecycle states available in osgi:list

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

```xml
<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">
        <ref-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

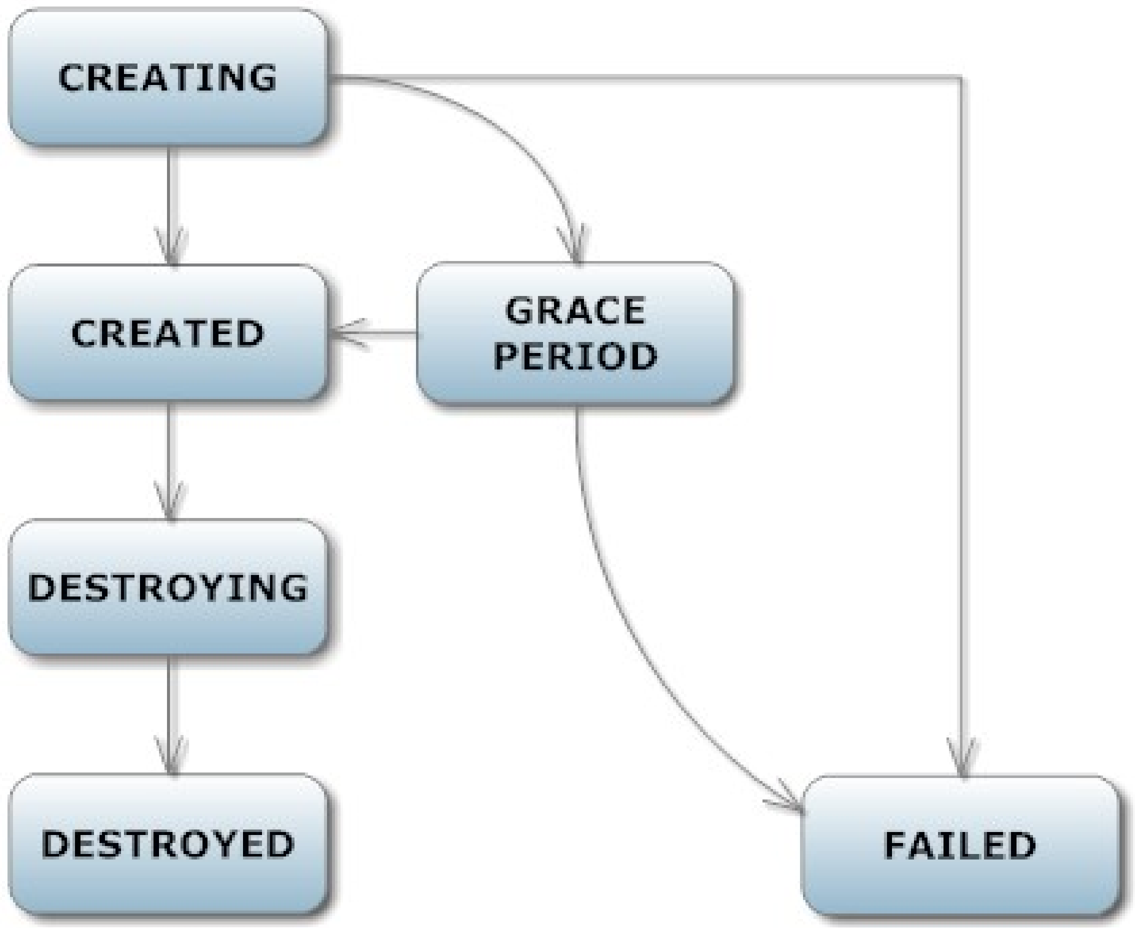
```
<bean id="kitchen" class="be.anova.course.blueprint.Kitchen" />

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

OSGi Service Registry

- 3 options to code bind/unbind method

```java
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) { }

}
```

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

Planning

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