



OSGI BASICS

SYSTEM INTEGRATION WITH JBOSS

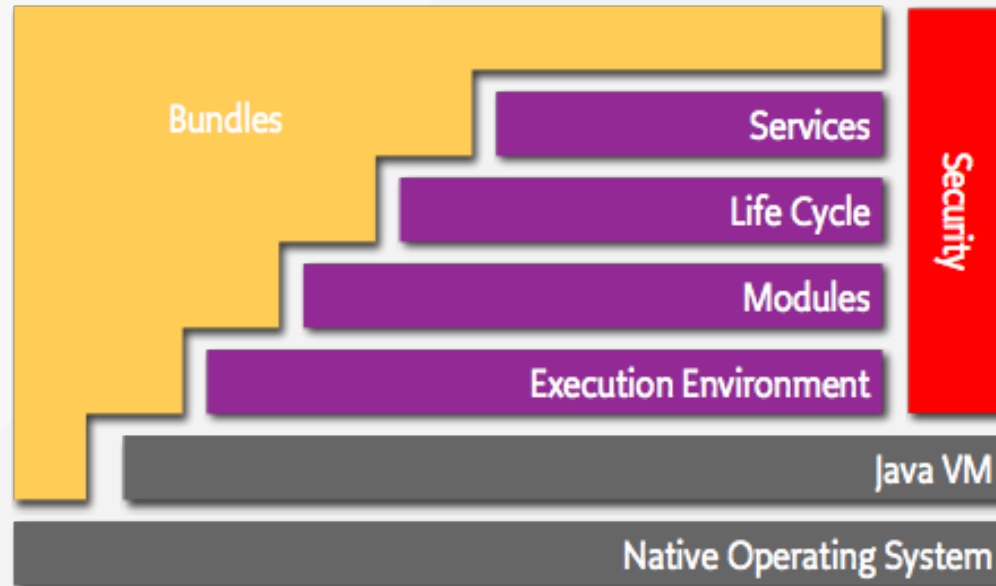
Jiří Pechanec

OSGI TECHNOLOGY

- **O**pen **S**ervices **G**ateway initiative (obsolete)
- Set of specifications (last v6, 2014)
- Managed by [OSGi Alliance](#)
- **Modular system** and a **service platform** for the Java programming language
- Implements a complete and **dynamic component model**
- Applications or components, coming in the form of **bundles**
- Bundles have **lifecycle**
- "Service Oriented Architecture" in JVM
- **Enterprise** ready system (web bundles)

OSGI ARCHITECTURE

- Key layers
 - Modules
 - Life Cycle
 - Services
- Execution Environment
 - Defines what methods and classes are available in a specific platform.



OSGI USAGE

- Main implementations
 - [Apache Felix](#)
 - [Eclipse Equinox](#) (reference implementation)
 - [Knopflerfish](#)
- Usage
 - Middleware products
 - **Apache Karaf**
 - **JBoss Fuse**
 - **GlassFish** (v3) - Application server for Java EE
 - IDE (Integrated Development Environment)
 - **Eclipse**

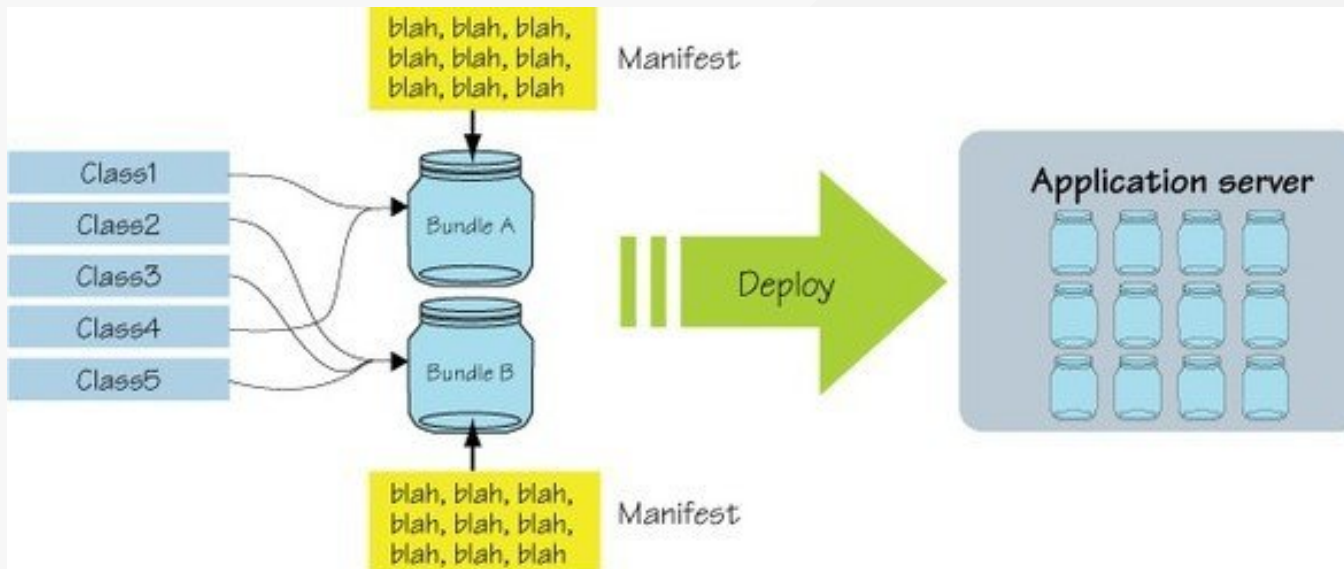


APACHE KARAF CONTAINER

- Small OSGi based runtime
- Lightweight container
- Various components and applications can be deployed (jar, war)
- Based on Apache Felix
- Features:
 - **Hot deployment** ([home]/deploy)
 - **Dynamic configuration** ([home]/etc)
 - **Logging System** (centralised, Log4J)
 - **Provisioning** (mvn, http, file etc.)
 - **Extensible Shell console**
 - **Remote access** (ssh)
 - **Security framework** (JAAS)
 - **Managing multiple instances**

BUNDLE

- Basic module in OSGi container
- = JAR archive + OSGi metadata
- OSGi metadata
 - Properties in META-INF/MANIFEST.MF file
 - Bundle unique static identification
 - Symbolic name + version



MANIFEST HEADERS

- Non-OSGi headers
 - Manifest-Version, Build-Jdk, Build-By, Main-Class
- **OSGi headers**
 - Mandatory
 - **Bundle-SymbolicName**
 - **Bundle-Version**
 - **Bundle-ManifestVersion** (the only value is "2")
 - Optional
 - Human readable
 - Bundle-Name
 - Bundle-Description
 - Bundle-Vendor
 - Lifecycle
 - **Bundle-Activator**
 - Modules
 - **Export-Package**
 - **Import-Package**
 - Bundle-ClassPath (default value is ".")

MANIFEST EXAMPLE

```
Bundle-Name: Hello World  
Bundle-SymbolicName: org.wikipedia.helloworld  
Bundle-Description: A Hello World bundle  
Bundle-ManifestVersion: 2  
Bundle-Version: 1.0.0  
Bundle-Activator: org.wikipedia.Activator  
Export-Package: org.wikipedia.helloworld;version="1.0.0"  
Import-Package: org.osgi.framework;version="1.3.0"
```

OSGI PACKAGES

- **Export package**

- Provided package for other bundles
- Property "version" - **single value**

- **Import package**

- Missing package
- Property "version" - **interval** (always!)
 - *no version* → $0.0.0 \leq x \leq \text{max}$
 - "1.2" → $1.2.0 \leq x \leq \text{max}$
 - "[2.1, 3)" → $2.1 \leq x < 3$



Question: Is it possible to require only single version?

MANIFEST GENERATOR

- [BND](#) tool
 - Simplifies bundle creation
 - Generates OSGi metadata semi-automatically
 - Syntactic sugars
 - `Import-Package: com.library.*; version = 1.21`
- [Maven bundle plugin](#)
 - Uses BND tool internally

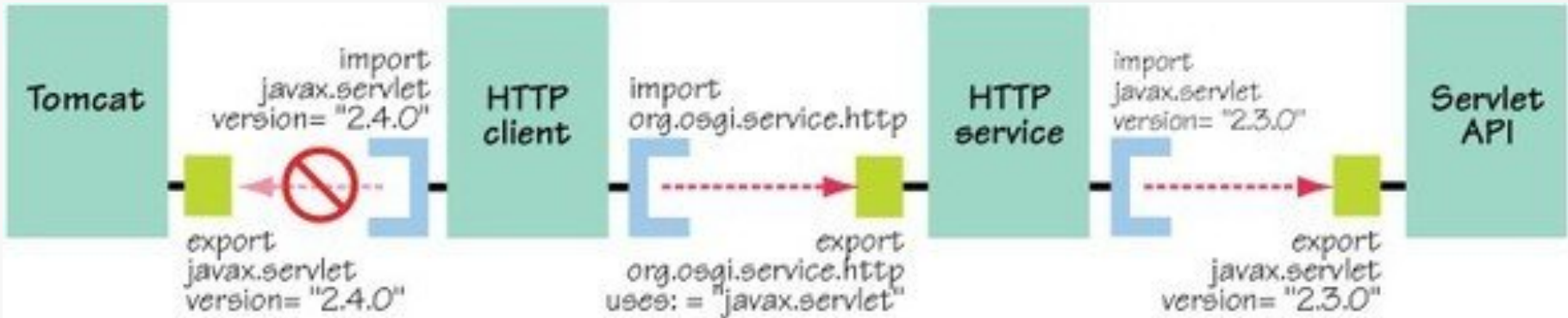
```
<plugin>
  <groupId>org.apache.felix</groupId>
  <artifactId>maven-bundle-plugin</artifactId>
  <extensions>>true</extensions>
  <configuration>
    <instructions>
      <Bundle-Activator>org.wikipedia.Activator</Bundle-Activator>
    </instructions>
  </configuration>
</plugin>
```

IMPORT PACK. PRIORITIES

- RESOLVED bundle state has precedence before INSTALLED
- Greater versions have precedence
- Property match (e.g. vendor="sun")
- Directive "uses:=<package-name>"



Question: Why HTTP client will not be resolved if other bundles are resolved?

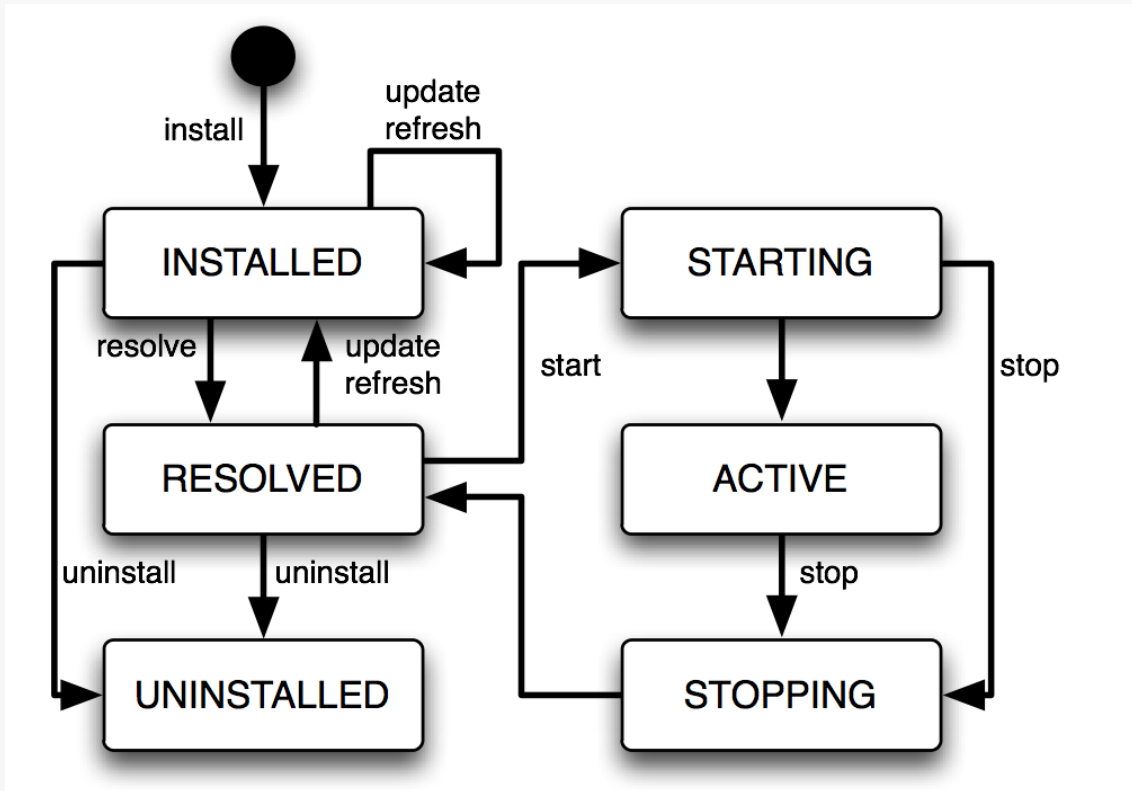


OSGI CLASSLOADING

- Every bundle has its own classloader
- Classloader's structure is Map
- One item for each imported package
 - key - Package name
 - value - Another classloader
- There is no delegation of requests to parent classloaders
- Class searching
 - 1 java.*
 - 2 Import packages (Import-Package)
 - 3 Own packages (Bundle-ClassPath)

What is [home]/etc/jre.properties good for?

OSGI LIFECYCLE



Demo 1 : Bundle events

BUNDLE ACTIVATORS

- Connected with states **STARTING** and **STOPPING**
- Activators implement interface `org.osgi.framework.BundleActivator`
- OSGi header "Bundle-Activator" in MANIFEST.MF
- Often used for OSGi services registrations

```
public interface BundleActivator {  
    public void start(BundleContext context) throws Exception;  
    public void stop(BundleContext context) throws Exception;  
}
```

OSGI SERVICES

- "SOA" in JVM
- Advantages
 - Less coupling
 - Multiple implementations
- Every service has contract (one or more interfaces)
- Service registration
 - Activator
 - Component framework
- Service discovery
 - Contract + service properties (LDAP syntax)
- Service priority
 - 1 Attribute "service.ranking"
 - 2 Attribute "service.id"

SERVICE REGISTRATION

```
public class Activator implements BundleActivator {  
  
    private ServiceRegistration registration;  
  
    public void start(BundleContext ctx) {  
        Dictionary props = new Properties();  
        props.put("service.ranking", 1);  
        registration = ctx.registerService(Foo.class.getName(),  
            new FooImpl(), props);  
    }  
  
    public void stop(BundleContext ctx) {  
        registration.unregister();  
    }  
}
```

LISTENERS

- **Bundle listener**

- Tracks bundle events (STARTED, STOPPED, UPDATED, ...)
- Used in BundleTracker (advanced bundle monitoring)

```
public interface BundleListener extends EventListener {  
    public void bundleChanged(BundleEvent event);  
}
```

- **Service listener**

- Tracks service events (REGISTERED, MODIFIED, ...)
- Used in ServiceTracker (advanced service monitoring)

```
public interface ServiceListener extends EventListener {  
    public void serviceChanged(ServiceEvent event);  
}
```



Demo 2 : Paint application (OSGi in Action)

COMPONENT FRAMEWORKS

- Simplifies creation of logical components
 - (e.g. OSGi service, java beans, ...)
- Frameworks are based on OSGi listeners
- Component definition
 - XML descriptor
 - Declarative Services, **Blueprint**, **Spring DS**
 - Java annotations
 - [Apache Felix iPojo](#)

BLUEPRINT

- Specification of OSGi component framework
- Specification managed by OSGi Alliance
- Uses dependency injection
- Designed to deal with the dynamic nature of OSGi
 - Services can become available and unavailable at any time.
 - Uses proxy objects
- Each bundle has own Blueprint container
- Uses OSGi extender pattern
 - Extension definition
 - 1 OSGI-INF/blueprint/*.xml
 - 2 "Bundle-Blueprint" property in MANIFEST.MF
- Implementations
 - [Apache Aries](#)
 - [Eclipse Gemini](#) (reference implementation)

BLUEPRINT DESCRIPTOR

- XML elements
 - Main
 - <bean>, <service>, <reference>, <reference-list>
 - Other
 - <value>, <ref>, <idref>, <map>, <props>, <list>, <array>, ...

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.osgi.org/

    <reference id="bazRef" interface="Baz"/>

    <bean id="barImplRef" class="BarImpl">
      <property name="baz" ref="bazRef"/>
    </bean>

    <service interface="Foo" ref="barImplRef" ranking="1">
      <service-properties>
        <entry key="prop" value="foobar"/>
      </service-properties>
    </service>
  </blueprint>
```

(SPRING DM)

- [Spring DM](#) = Spring Dynamic Modules
- An open source project in the Spring portfolio
- Allows to implement Spring Applications on top of an OSGi framework
- Connects the benefits of both technologies
 - Spring - Component management, AOP, Dependency Injection, ...
 - OSGi - Dynamic environment
- Each bundle has own Spring container
- Spring beans can be exported as OSGi services
- Uses OSGi extender pattern
 - Extension definition
 - META-INF/spring/*.xml
 - "Spring-Context" property in MANIFEST.MF

Spring DM is discontinued and last version is for spring 3.x.

SPRING DESCRIPTOR

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:osgi="http://www.springframework.org/schema/osgi">

  <osgi:reference id="bazRef" interface="Baz"/>

  <bean id="barImplRef" class="BarImpl">
    <property name="baz" ref="bazRef"/>
  </bean>

  <osgi:service interface="Foo" ref="simpleService" ranking="3">
    <osgi:service-properties>
      <entry key="prop" value="foobaz"/>
    </osgi:service-properties>
  </osgi:service>

</beans>
```

OSGI ENTERPRISE

- **Web applications**

- web.xml
- Web-ContextPath in MANIFEST.MF

- **Dependency Injection**

- Blueprint
- Spring

- **JNDI**

- `osgi:service/<interface>[/<properties>]`
- Service property "osgi.jndi.service.name" for custom JNDI names
 - `osgi:service/<jndi-name>`

- **JPA, JDBC**

- problems with dynamic classpath
- solution: OSGi services (Datasources, EntityManagers)
- persistence.xml ("Meta-Persistence" in MANIFEST.MF)
- container/application managed

- **JTA**

- container/application managed

OSGI ENTERPRISE (2)

- Blueprint + JPA + Transactions

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

  <bean id="inventory" class="fancyfoods.persistence.InventoryImpl">
    <tx:transaction method="*" value="Required" />
    <jpa:context property="entityManager" unitname="fancyfoods" />
  </bean>

  <service ref="inventory" interface="fancyfoods.food.Inventory" />
</blueprint>
```

OSGI ENTERPRISE (3)

- **Enterprise archives**

- Java EE
 - EAR archives (Enterprise archives)
- OSGi
 - ESA
 - Enterprise Subsystem Archives
 - no application.xml but SUBSYSTEM.MF
 - Dependencies do not have to be contained in archive
 - EBA
 - Enterprise Bundle Archive
 - similar to ESA (from Aries)
 - **Features**
 - XML descriptor (from Apache)



redhat.®

THANK YOU!

REFERENCES

- **OSGi in Action** - Richard S. Hall, Karl Pauls, Stuart McCulloch, David Savage
- **Enterprise OSGi in Action** - Holly Cummins, Timothy Ward
- **Spring in Action** - Craig Walls
- **Spring Dynamic Modules in Action** - Arnaud Cogoluegnes, Thierry Templier, Andy Piper
- <http://www.osgi.org/Specifications>