Introduction to OSGi

University of Colorado, Boulder

CSCI 5828 Foundations of Software Engineering

Spring 2012 – Professor Kenneth Anderson

Presentation by:

Brendan Greene

Software Developer

Northrop Grumman Corporation

Introduction to OSGi

My background

- Software developer for the last 3.5 years in the defense industry.
- Spent 2.5 years overseas in the Marshall Islands developing missile defense software.
- Currently doing agile development using multiple Java open source frameworks such as Hibernate, Spring, Camel and OSGi.
- Studied at the University of California, Irvine.
- Bachelor of Science in Computer Science.
- Bachelor of Science in Biological Sciences.

Introduction to OSGi

Introduction to OSGi

What is OSGi?

- First appeared in 1999.
- Once called the Open Services Gateway Initiative, the OSGi Alliance now refers to the framework specification as OSGi or the OSGi Service Platform.
- Originally intended to bring Java into embedded systems.
- OSGi is a modularity layer for the Java platform.
- In this context we're referring to the traditional computer science definition of modularity:
  - The logical decomposition of a large system into smaller collaborating pieces.
- OSGi provides a service oriented development model allowing for SoA within a virtual machine.

Introduction to OSGi

Why OSGi?

Java's modularity shortcomings:

- Provides encapsulation through access modifiers:
  - public, protected, private
  - Java also provides the package
- Shortcoming: In order for classes or methods from one package to be available to another they must be declared as public.
- Problematic when package A needs code from package C, now packages B, D, E and F can see that code.
- OSGi provides package level encapsulation.

Introduction to OSGi

Why OSGi?

Java's modularity shortcomings:

- The Java class path.
- Does not provide a dynamic way to handle versioning and dependencies.
- Often create applications and add JARs until the "great JVM" is appeased.
- OSGi offers an elegant solution to handling dependencies by requiring dependency declarations within units of modularity.

Introduction to OSGi
Why OSGi?

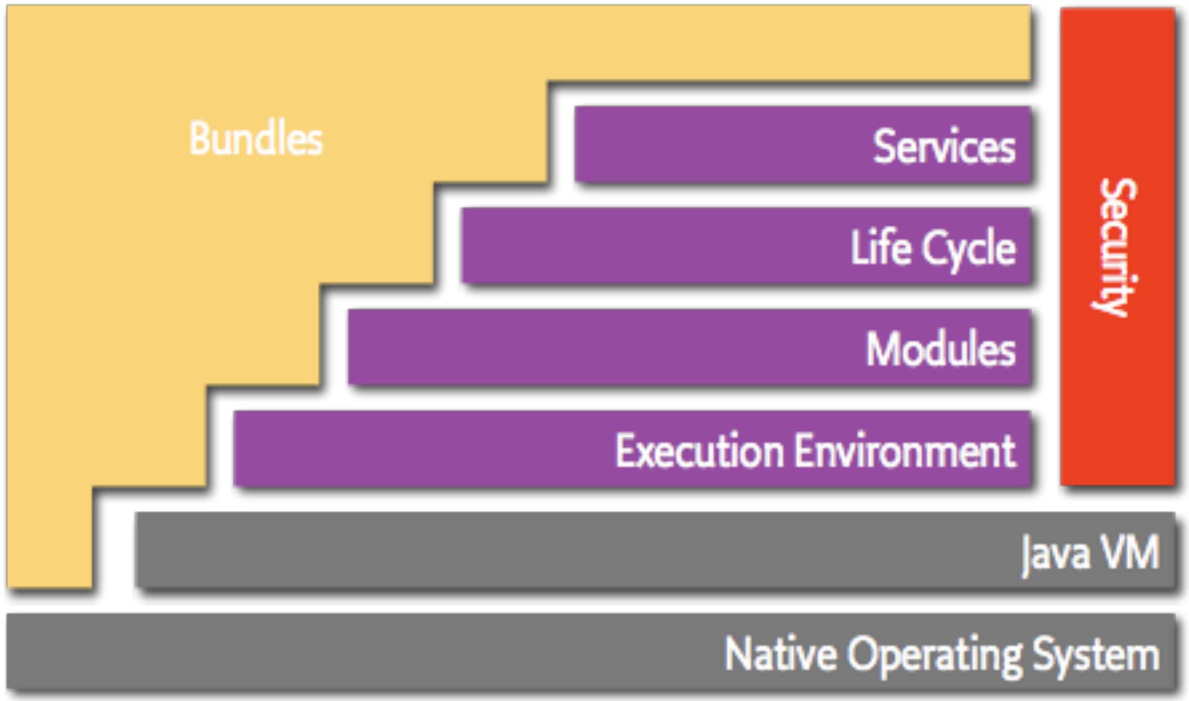
Java's modularity shortcomings:

- Large Java applications can be difficult to deploy and manage.
- Example of this would be a monolithic JEE application.
- In order to update a deployment the system/ servers need to be cycled and the application built and deployed causing system outages.
- OSGi provides an isolated module cycling/ updating capability in order to increase availability.

Introduction to OSGi

Exploring OSGi

OSGi's Layered Architecture:



Source: OSGi Alliance Website. < http://www.osgi.org/About/WhatIsOSGi>

Introduction to OSGi
Exploring OSGi

Focus is on the following OSGi Layers

- Module Layer
  - The Bundle
  - Package Level Encapsulation
- Lifecycle Layer
  - The Container
  - Managing an OSGi application
- Service Layer
  - The nervous system
  - How OSGi bundles communicate

Introduction to OSGi

Exploring OSGi: Module Layer

Modularity:

- OSGi module definition is the bundle, also known as OSGi's unit of modularity and deployment.
    - Standard Java provides monolithic deployment units such as JAR, WAR and EAR.
- Modules promote code reuse.
- Modules promote high cohesion.
- Modules enforce logical boundaries for code location.
- Similar concept to object-orientation in that they both provide separation of concerns.

Introduction to OSGi

Exploring OSGi: Module Layer

OSGi's core component is the ***Bundle***

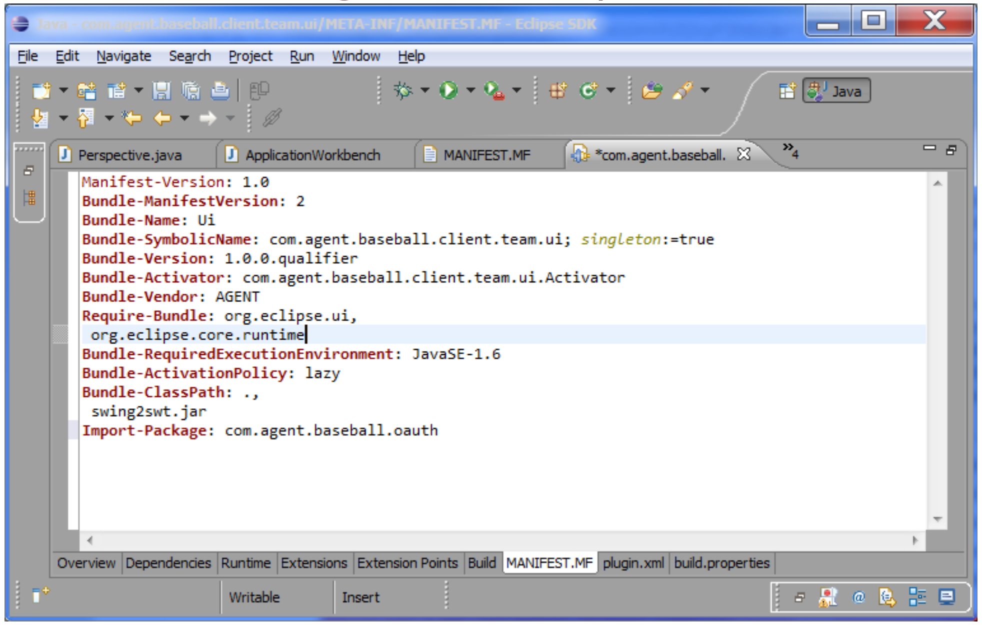
- Structurally similar to a JAR with additional metadata.
- A JAR and Bundle differ in that a Bundle is not a fully featured application packaged up, but rather a set of highly cohesive classes.
- A Bundle can contain:
  - Any text or configuration files found in a typical JAR
  - A MANIFEST.MF file located in the META-INF directory
  - Embedded JAR files that can be used by just the bundle by explicitly placing them on the Bundle-Classpath header in the MANIFEST.MF

Introduction to OSGi

Exploring OSGi: Module Layer

Bundle Configuration

- Configuration is facilitated through the MANIFEST.MF file.
- OSGi specification defines a large set of headers for bundle configuration (this set is still growing)


Java - com.agent.baseball.client.team.ui/META-INF/MANIFEST.MF - Eclipse SDK
File Edit Navigate Search Project Run Window Help
Java
Perspective.java
ApplicationWorkbench
MANIFEST.MF
*com.agent.baseball.
Overview Dependencies Runtime Extensions Extension Points Build MANIFEST.MF plugin.xml build.properties
Writable
Insert

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Ui
Bundle-SymbolicName: com.agent.baseball.client.team.ui; singleton:=true
Bundle-Version: 1.0.0.qualifier
Bundle-Activator: com.agent.baseball.client.team.ui.Activator
Bundle-Vendor: AGENT
Require-Bundle: org.eclipse.ui,
 org.eclipse.core.runtime
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Bundle-ActivationPolicy: lazy
Bundle-ClassPath: .,
 swing2swt.jar
Import-Package: com.agent.baseball.oauth
```

Introduction to OSGi

Exploring OSGi: Module Layer

OSGi Bundle and package level encapsulation:

- Can manage package exporting and importing through headers in the bundle MANIFEST.MF
- Headers used:
- Export-Package
  - Declare which packages are visible to other bundles in essence adding an additional access modifier to Java
- Import-Package
  - Declare which "exported-package(s)" the bundle depends on for functionality.
- Require-Bundle
  - Include an entire Bundle as a dependency (legacy).

Introduction to OSGi

Exploring OSGi: Module Layer

OSGi Bundle and package level encapsulation:

- Bundle dependencies are not transitive.
  - If A imports B and B imports C, A does not have access to code in C.
- Bundles importing a package do not have access to nested packages.
  - Bundles must explicitly declare a dependency on a package, and that package must be explicitly exported.
- Additionally, you can use modifiers to restrict which classes within a package are exposed when exporting.

Introduction to OSGi:

Exploring OSGi: Lifecycle Layer

Lifecycle Layer

- Provides the ability to dynamically install and manage bundles in the OSGi framework.
  - The OSGi specification defines the following lifecycle operations: install, update, start, stop and uninstall.
- Provides the interaction between bundles by allowing for bundles to register with the framework and gain access to the execution environment.
- Provides flexibility to architect a system with the knowledge that components can be updated, removed or plugged-in without experiencing a system outage.

Introduction to OSGi:
Exploring OSGi: Lifecycle Layer

OSGi containers implement the lifecyle specification

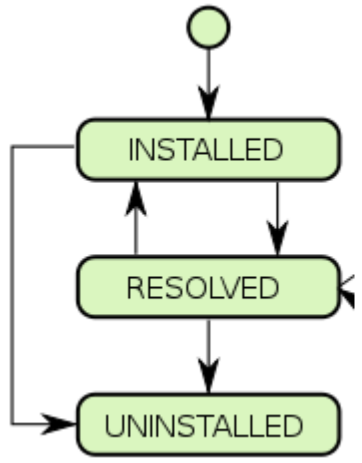
- Three popular open source containers are Eclipse Equinox, Apache Felix and Knoplerfish
  - Eclipse Equinox: http://www.eclipse.org/equinox/
  - Apache Felix: http://felix.apache.org/site/index.html
  - Knoplerfish: http://www.knoplerfish.org/
- Eclipse Equinox is the run-time for the popular Eclipse IDE and is the reason the IDE is able to implement a plug-in model (plug-in = bundle)
- Containers provide command line and web based interfaces for administration of the container

Introduction to OSGi:

Exploring OSGi: Lifecycle Layer

Installing Bundles

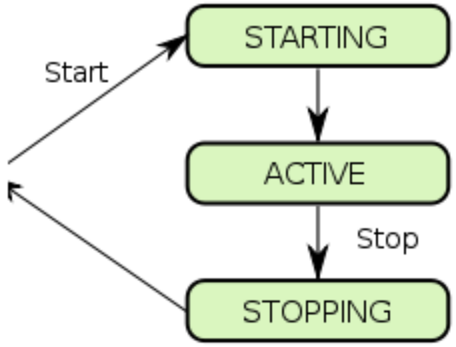
- Bundle installed
- Bundle dependencies are resolved (remain in installed state if missing dependencies)
- Bundle can be refreshed
- Bundle uninstalled



Introduction to OSGi:
Exploring OSGi: Lifecycle Layer

Starting Bundles

- Once a bundle has resolved its dependencies it is ready to start.
- A Bundle will be "activated" and transition to an active state.
- Important to know for troubleshooting.
- An installed bundle cannot become active if it depends on another bundle that cannot resolve.



Introduction to OSGi:
Exploring OSGi: Lifecycle Layer

OSGi Classloading

- A key to OSGi having such a dynamic runtime.
- No more monolithic classloader.
- Each bundle has its own classloader.
    - Bundle classloader first scans its embedded packages and then loads any package imports.
- If a Bundle can't resolve, the system can still start and function while troubleshooting the module occurs.

Introduction to OSGi:

Exploring OSGi: Service Layer

OSGi Services and the Service Registry

- OSGi services enable a single-JVM Service Oriented Architecture
- OSGi services follow a publish, find and bind paradigm.
  - Services are published to the service registry.
  - Clients search the registry for needed/available services depending on the need.
- Promotes interface driven development
- A similar concept to Dependency Injection

Introduction to OSGi:

Exploring OSGi: Service Layer

What exactly is a service?

- A service embodies a contract between a provider and a caller.
- The caller doesn't need to know anything about the implementation of the service, or even where it is implemented.
- Development of services focuses on deciding what other components can do the work needed.
- Promotes loose coupling and code reuse.
- In OSGi a service can be an interface or a superclass, but the preferred methodology is to code to an interface.

Introduction to OSGi:
Exploring OSGi: Service Layer

Role of services in OSGi

- Services provide a reliable way for bundles to communicate.

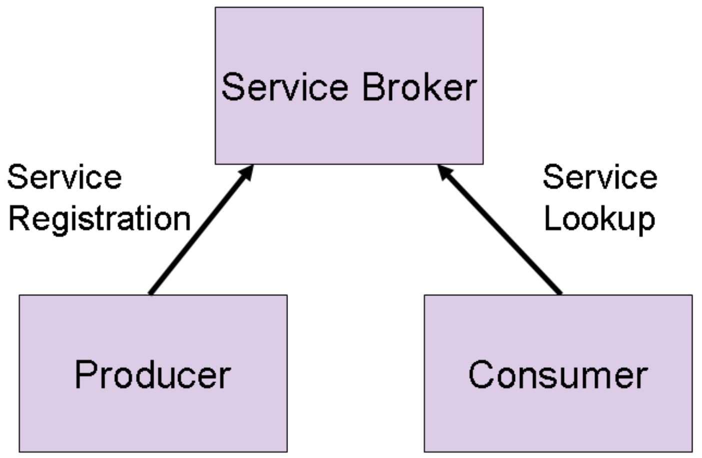


Figure 1 – Simple service architecture

Source: Eclipse Website < http://www.eclipse.org/>

Introduction to OSGi:

Exploring OSGi: Service Layer

Publishing/Registering Services

- Services are implemented as POJO (Plain old java objects) with no dependency on the OSGi APIs.
- Described and published through interfaces.
- Services are registered by a bundle in order to make the API of the service available.
    - Registration done through a simple method call.
- Best practice dictates that a service interface reside in a different package than its implementation.
- When using the Export-Package manifest header, only the interfaces are exposed to the container.

Introduction to OSGi:
Exploring OSGi: Service Layer

Finding and Using Services

- OSGi provides the ServiceReferences interface that defines two methods:
  - getRegisteredServices: Returns a list of services
  - getServicesInUse: Gets services being utilized by other bundles
- Even better OSGi provides the ServiceTracker class.
- This is a dynamic class that gets and ungets services based on events during runtime.

Introduction to OSGi:
Exploring OSGi: Service Layer

Some useful services that come with OSGi

- Event Admin Service
  - This service allows you to design your application as an event driven application.
  - Handled through messaging in a manner very similar to the Java Messaging Service (JMS).
- Configuration Management Service
  - Allows for configuring a bundle that will manage config files and even scan other bundles for config files.
  - Can alter configuration of the application through the OSGi container's administration command line.

Introduction to OSGi

Future of OSGi

- Although it is an older framework, it is starting to get a lot of visibility.
- Parts of OSGi are being incorporated into Java 1.8 to satisfy modularity requirements.
- Many open-source products are refactoring code in order to support OSGi.

Introduction to OSGi:
OSGi in the world

OSGi is everywhere

- Runtime for the Eclipse IDE
- IBM Websphere Application Server
- Oracle Glassfish AS
- Jboss AS
- Spring Roo
- All Spring Framework Jars (are bundles)
- Parts of OSGi are being incorporated into Java 1.8 as a JSR to support modularity

Introduction to OSGi:
OSGi in the world

OSGi Roadblocks

- It has a terrible name.
- OSGi has a steep learning curve.
- OSGi is not yet fully compliant with JEE, however the Enterprise OSGi specification is available and several implementations exist.
- Difficult to migrate legacy systems to OSGi since a lot of effort would need to go into Bundling up all the system’s JARs.

Introduction to OSGi

Summary

- OSGi provides Java the fuel to get past just being object-oriented and embrace module granularity.
- OSGi promotes interface based programming as well as module cohesion and loose coupling between components.
- OSGi provides a dynamic container for managing and updating systems while reducing outages.
- OSGi Rocks!

Introduction to OSGi

Reference Material for OSGi

- OSGi specification website
  - http://www.osgi.org
- OSGi in Action
  - http://www.manning.com/hall/
- OSGi in Depth
  - http://www.manning.com/alves/
- Tutorials
  - http://felix.apache.org/site/apache-felix-osgi-tutorial.html