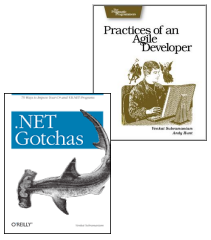
OSGi:
A Well Kept Secret

spkr.name = 'Venkat Subramaniam'
spkr.company = 'Agile Developer, Inc.'
spkr.credentials = %w{Programmer Trainer Author}
spkr.blog = 'agiledeveloper.com/blog'
spkr.email = 'venkats@agiledeveloper.com'



Agile Developer

Abstract

- In this presentation we will introduce OSGI and discuss how it can help modularize and version your enterprise Java applications.
- We will delve into OSDI fundamentals, modularization and versioning, developing and deploying components, OSGI implementations, and Spring integration.

Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

Component Management

- How do you
  - install a component on a server?
  - update it?
  - uninstall it?
  - start and stop these component services?
  - manage versions of component?
  - run multiple versions of a component at the same time?
- No clear solution in the Java Space
- Enter OSGi

OSGi

- One of the well kept secrets in the Java community!
  - Started in 1999 as Open Services Gateway initiative for embedded Java and network devices
  - Now, OSGi is simply a trademark—not an acronym
- Lightweight Dynamic Module System for Java
  - in–VM SOA
- Remotely managed service platform
  - API for managing services
  - Allows software components/bundles to be detected and used

OSGi to your rescue

- OSGi helps you cope with classpath hell
- You can easily reload a class
- You can load multiple versions of a class
- Global classpath does not affect your application

OSGi

- Defines
  - application life cycle model
  - service registry
- Currently at Release 4
- Reminds you of JINI and JMX?
- OSGi is much more a universal middleware
- JSR 277 tries to bring this into Java 7
  - Aims for only static module system [See Peter Kriens's comment at http://www.osgi.org/blog/2006/10/jsr-277-review.html]
- JSR 291: Dynamic Component Support for Java tries to bring this into current Java SE Environments.

Agenda

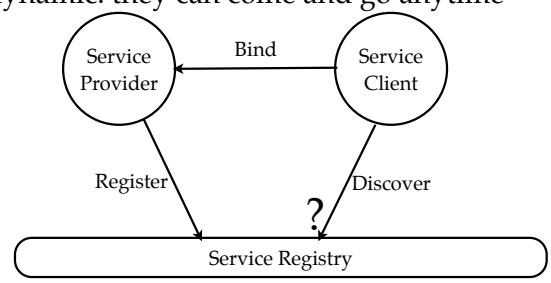
- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

Bundles

- Component (bundle) is a JAR with Java classes + manifest file + resources
- Bundles indicate their dependence on other bundles by using import statements
  - allows bundle to be wired to proper bundle
- Require-Bundle header
  - implementation specific dependencies
- You can specify the version (or range of versions) for dependency

OSGi and Bundles

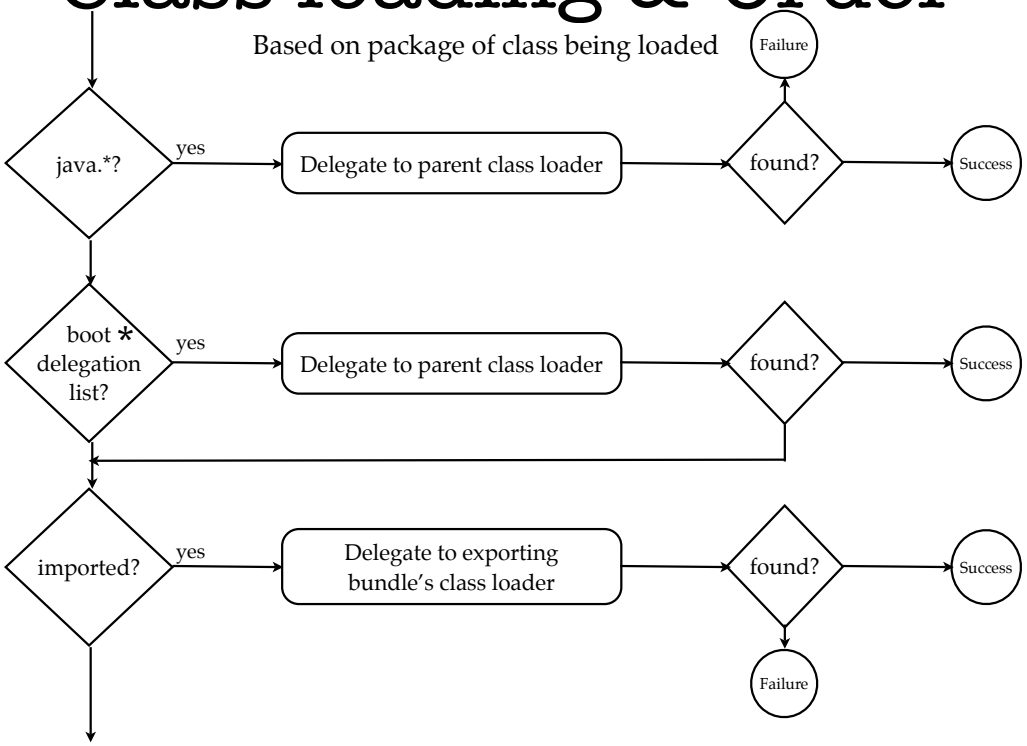
- OSGi resolves dependencies between modules
- Within OSGi, Bundle is a black box
  - Protected - you can't introspect into it
  - You explicitly decide what to expose (using Export-Package)
- Bundles publish services dynamically
- Service Registry helps find and bind to services
- Services are dynamic: they can come and go anytime



Bundles and Conflicts

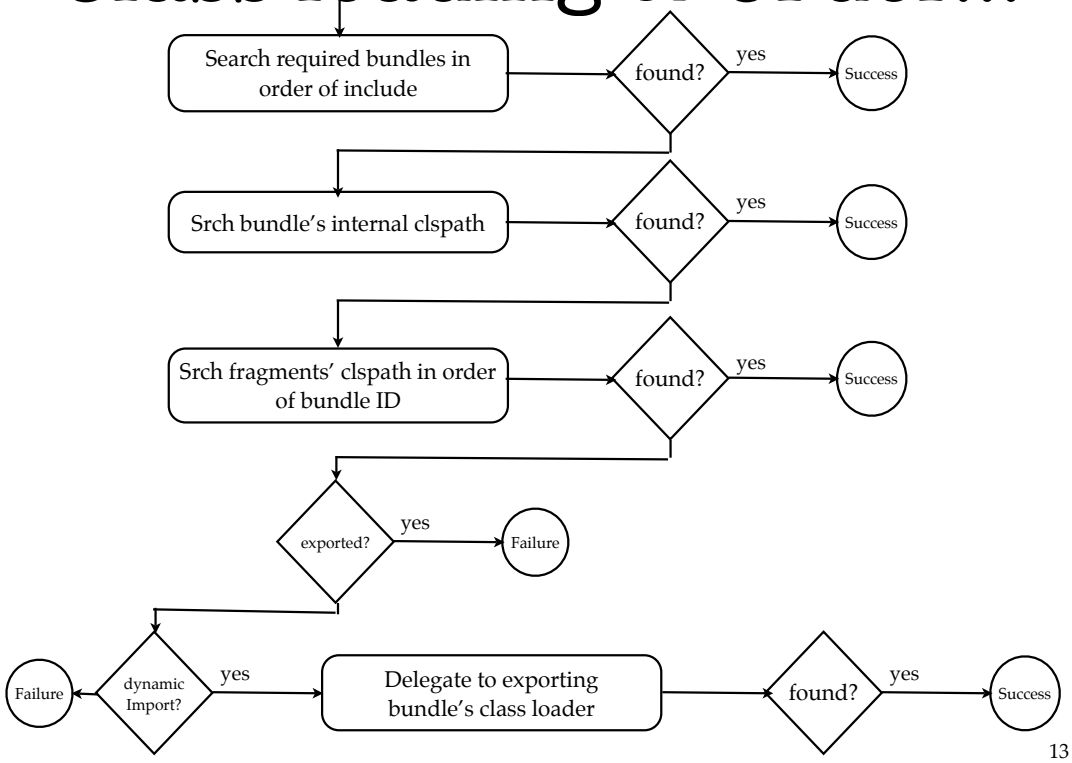
- What if multiple bundles share a class?
- What if they have different versions of same class?
- What if different bundles ask for different versions of same class?
- OSGi takes care of all these concerns deterministically
  - Package is always exported with unique version
  - Importer can specify range of versions to accept
  - OSGi supports, but minimizes, number of versions of same class in concurrent use
  - Framework verifies use of correct version of bundle
  - If a bundle is uninstalled, importers are resolved to bind to new exporter
  - Each bundle has it's own classloader

Class loading & Order



*– org.osgi.framework.bootdelegation

Class loading & Order...



Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

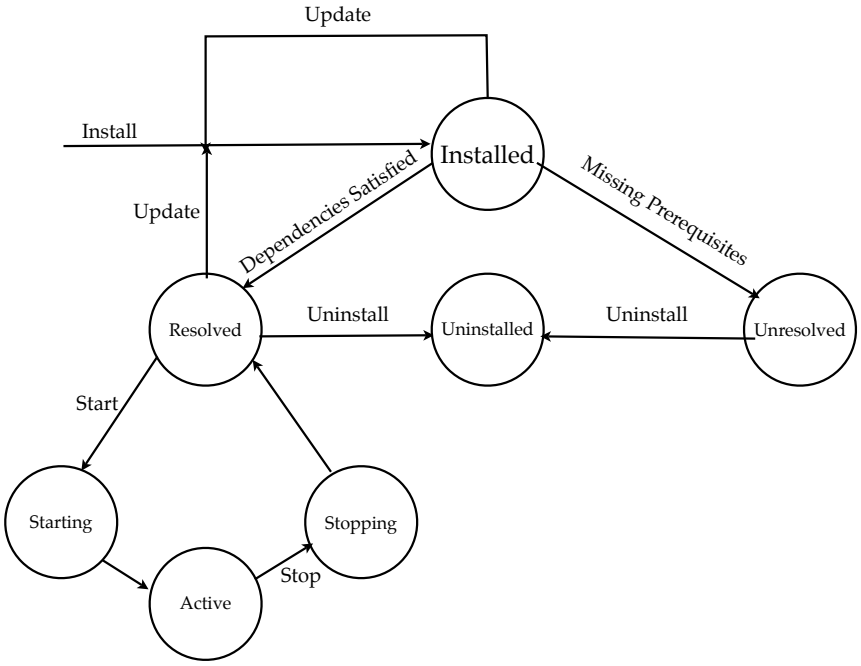
OSGi and Applications

- Publishes a service interface and service properties
- Different services may implement same interface but provide different properties
- Only published part of bundle is visible outside the bundle
- Only exported packages are visible
- Service Registry may be queried using LDAP syntax
- Service clients may register for change notification from registry

OSGi Facilities

- Applications can be installed, started, stopped, updated, and uninstalled, all while JVM is still running
- Applications restore state when JVM restarted
- Very secure so applications can't harm or interfere with the environment or other applications
- Dynamic discovery and binding
- Remote management architecture

Application Life Cycle



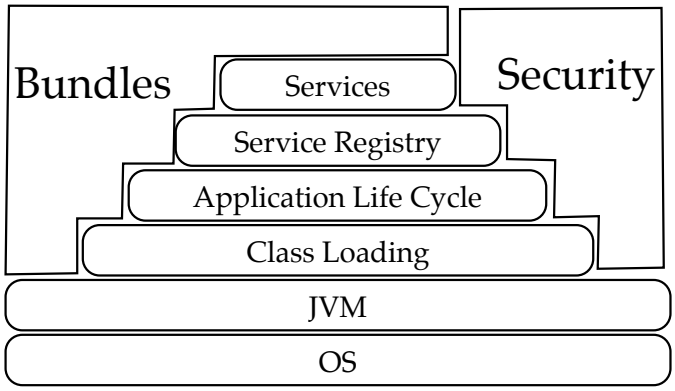
OSGi Security

- JVM offers first level of defense
- Java language access mechanism comes in next
- Java's code based security may play a role
- OSGi separates bundles—they need permission to access other bundles

Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

OSGi Architecture

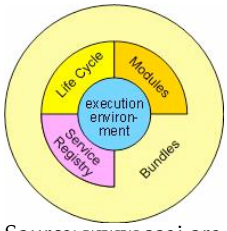


Service Platform Capabilities

- Software Component (bundle) Life Cycle Management
- Services may be managed remotely
- Enhanced ubiquitous multi-level security

OSGi Framework

- Execution Environment: Underlying Java environment configurations and Profiles
- Modules layer defines class loading policies
  - In addition to Java's mechanism, adds modules
  - Adds private classes for a module
  - Provides linking between modules



Source: www.osgi.org

- Life Cycle layer adds bundles that can be installed, started, stopped, updated, and uninstalled dynamically
- Service Registry layer allows discovering and sharing objects between bundles.
- Service providers implement Java interfaces and register
- Service clients can be notified as services appear and disappear

Framework Services

- Permission Admin
  - Useful to manipulate permission of bundles
- Package Admin
  - As bundles evolve, this allows the dependency of classes and resources to be kept in synch
- Start Level
  - By allowing you to set a start level for a bundle, manages bundles that must be run together or initialized before others are started
- URL Handler
  - Allows components to provide additional URL handlers

System Services

Most commonly needed services across different applications

- Application Admin
  - Allows you to manage start and stop of applications
- Component Runtime
  - Handles XML based dependency declaration
- Configuration Admin Service
  - Eases setting and getting configuration information
- Deployment Admin
  - OSGi primarily supports Jar format for bundle deployment
  - Provides secondary format, the deployment package, to combine bundles with arbitrary resources
- Device Access Service
  - Comes into action during plug and play for automatically downloading bundle with driver for a device

System Services

- Event Admin
  - Provides a generic, topic based event mechanism
- IO Connector Service
  - Implements CDC/CLDC javax.microedition.io package as a service; allows protocol schemes
- Log Service
  - Allows for logging warnings, etc. and dispatches to subscriber bundles
- Preferences Service
  - Provides access to hierarchical properties like Java Preferences
- User Admin Service
  - For authentication and authorization

Protocol Services

- Http Service
  - A servlet runner for hosting servlets dynamically and may be administered remotely
- UPnP Service
  - Universal Plug and Play maps devices on UPnP network to registry and vice-versa
- DMT Admin
  - Defines how Device Management Tree can be accessed/extended

Miscellaneous Services

- Wire Admin
  - Configuration based wiring (resolving dependencies) bundles
- XML Parser
  - Helps locate a JAXP parser

Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

Implementations

- To mention a few...
  - Equinox OSGi (Eclipse)
  - Apache Felix
  - Knopflerfish OSGi
  - ...
  - Spring OSGi

Piero Campenelli's Comparison
http://tinyurl.com/2m3t2m
http://tinyurl.com/246t4r

Usage

- To mention a few...
  - Newton (http://newton.codecauldron.org)—dynamically moves code around network on demand
  - Eclipse
  - IBM WebShere
  - JOnAS
  - Auto manufacturers GST specifications
  - Philips iPronto—advanced universal remote control
  - ...

Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

Eclipse and OSGi

- Eclipse Rich Client Platform (RCP) uses OSGi as runtime for plug-in architecture
- Eclipse is based on plug-in architecture–it's more about exploring and using plug-in than Java or IDE
- Eclipse needs dynamic Plug-ins
- Equinox, at the core of Eclipse 3.0's, is an implementation of OSGi
- What Eclipse calls as Plugin is OSGi bundle
  - Eclipse provides plugin project to ease creation

Eclipse OSGi

```
> java -jar org.eclipse.osgi_3.2.2.jar -console

osgi> ss

Framework is launched.

id      State       Bundle
0       ACTIVE      system.bundle_3.2.2.R32x_v20070118
```


Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

Creating and Deploying a Bundle

- Bundle is a Jar
- `Bundle-SymbolicName`
  - Only *required* header in the manifest

Creating Bundle

```
package com.agiledeveloper;

public class FirstBundle
{
}
```

MANIFEST.MF

```
Bundle-SymbolicName: com.agiledeveloper.FirstBundle
```

Deploying

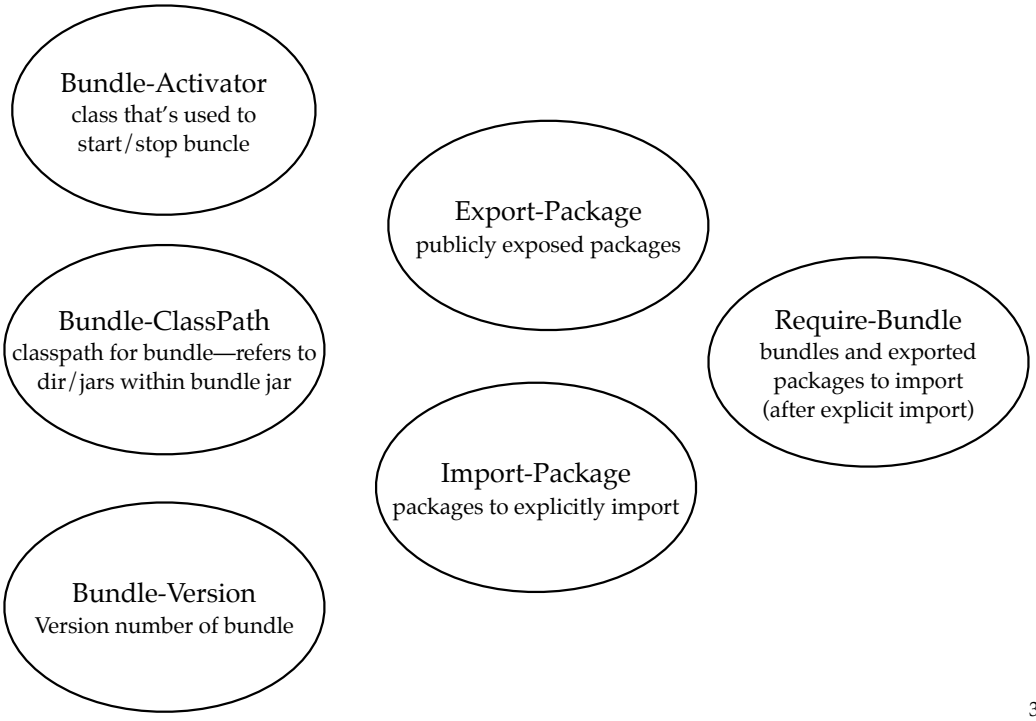
```
> java -jar org.eclipse.osgi_3.2.2.jar -console

osgi> install file:./firstbundle.jar
Bundle id is 1

osgi> ss

Framework is launched.

id      State       Bundle
0       ACTIVE      system.bundle_3.2.2.R32x_v20070118
1       INSTALLED   com.agiledeveloper.FirstBundle_0.0.0

osgi>
```


Version

```
Bundle-SymbolicName: com.agiledeveloper.FirstBundle
Bundle-Version: 1.0.0
```

```
osgi> update 1

osgi> ss

Framework is launched.

id      State       Bundle
0       ACTIVE      system.bundle_3.2.2.R32x_v20070118
1       INSTALLED   com.agiledeveloper.FirstBundle_1.0.0

osgi>
```

Manifest Options



Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

API

- Bundles implement BundleActivator
  - Provides hook to respond to start, stop
- BundleContext is API for framework
  - Provided to bundle when it starts
- installBundle method
  - Takes a URL of bundle to install

BundleActivator

```
Bundle-SymbolicName: com.agiledeveloper.FirstBundle
Bundle-Version: 1.0.1
Bundle-Activator: com.agiledeveloper.FirstBundl
```

```
osgi> update 1

osgi> start 1
Starting Bundle

osgi> update 1
Stoping Bundle
Starting Bundle

osgi>
```

```
package com.agiledeveloper;

import org.osgi.framework.BundleActivator;
import org.osgi.framework.BundleContext;

public class FirstBundle implements BundleActivator
{
    public void start(BundleContext bundleContext)
    {
        System.out.println("Starting Bundle");
    }

    public void stop(BundleContext bundleContext)
    {
        System.out.println("Stoping Bundle");
    }
}
```

Creating a Service

```
package com.agiledeveloper;

public interface TimeService
{
    public String getTime();
}
```

```
package com.agiledeveloper;

import java.util.Date;

public class TimeServiceImpl implements TimeService
{
    public String getTime()
    {
        return new Date().toString();
    }
}
```


Registering Service

```
public class FirstBundle implements BundleActivator
{
    private ServiceRegistration _serviceRegistration;

    public void start(BundleContext bundleContext)
    {
        System.out.println("Starting Bundle");

        _serviceRegistration = bundleContext.registerService(
                    TimeService.class.getName(),
                        new TimeServiceImpl(), null);
    }

    public void stop(BundleContext bundleContext)
    {
        System.out.println("Stoping Bundle");

        _serviceRegistration.unregister();
    }
}
```

```
Bundle-SymbolicName: com.agiledeveloper.FirstBundle
Bundle-Version: 1.0.2
Bundle-Activator: com.agiledeveloper.FirstBundle
Import-Package: org.osgi.framework
Export-Package: com.agiledeveloper
Export-Service: com.agiledeveloper.TimeService
```

Deploying Service

```
osgi> status
Framework is launched.

id      Bundle Location
State       Bundle File Name
0       System Bundle
  ACTIVE      org.eclipse.osgi_3.2.2.R32x_v20070118
1       file:./firstbundle.jar
  ACTIVE      com.agiledeveloper.FirstBundle_1.0.2
Registered Services
{org.osgi.service.packageadmin.PackageAdmin}={service.ranking=2147483647,
  service.pid=0.org.eclipse.osgi.framework.internal.core.PackageAdminImpl,
  service.vendor=Eclipse.org, service.id=1}
...
...
...
{com.agiledeveloper.TimeService}={service.id=21}

osgi>
```


Service Listener

```
public class TestServiceListener
        implements BundleActivator, ServiceListener
{
    public void start(BundleContext bundleContext)
    {
        bundleContext.addServiceListener(this);
    }

    public void stop(BundleContext bundleContext)
    {
        bundleContext.removeServiceListener(this);
    }

    public void serviceChanged(ServiceEvent serviceEvent)
    {
        String event[] = {
                "INVALID", "Registered",
                "Modified", "INVALID", "UnRegistered"};

        System.out.println(
            event[serviceEvent.getType()] + " : " +
            serviceEvent.getServiceReference());
    }
}
```

Listening To Service

```
> java -jar org.eclipse.osgi_3.2.2.jar -console

osgi> install file:./testservicelistener.jar
Bundle id is 1

osgi> install file:./firstbundle.jar
Bundle id is 2

osgi> start 1

osgi> start 2
Starting Bundle
Registered : {com.agiledeveloper.TimeService}={service.id=21}

osgi> stop 2
Stoping Bundle
UnRegistered : {com.agiledeveloper.TimeService}={service.id=21}

osgi>
```


Using Service

```
public void start(BundleContext bundleContext) throws Exception
{
    _serviceTracker = new ServiceTracker(bundleContext,
                      TimeService.class.getName(),
                      null);

    _serviceTracker.open();

    _timeService = (TimeService) _serviceTracker.getService();

    new Thread(new Runnable()
    {
        public void run()
        {
            useService();
        }
    }).start();
}
```

How you use it is up to you...

```
public void stop(BundleContext bundleContext) throws Exception
{
    _keepRunning = false;
    _serviceTracker.close();
}
```

```
Bundle-SymbolicName: com.agiledeveloper.user.BundleUser
Bundle-Version: 1.0.3
Bundle-Activator: com.agiledeveloper.user.BundleUser
Import-Package: org.osgi.framework, org.osgi.util.tracker, com.agiledeveloper
Import-Service: com.agiledeveloper.TimeService
```

Version Constraints

- Import-Package can specify version constraints
- `Import-Package package-name;version=version-range`

The syntax of a version range is:

```
version-range ::= interval | atleast
interval ::= ( '[' | '(' ) floor ',' ceiling ( ']' | ')' )
atleast ::= version
floor ::= version
ceiling ::= version
```

OSGi Service Platform Core Specification

- [and] indicate inclusive range
- (and) indicate exclusive range
- For example, version="[1.2.3, 2.0.0)" includes 1.2.3, 1.2.4, ..., 1.9.9, but not 2.0.0

Version Constraints

```
Bundle-SymbolicName: com.agiledeveloper.FirstBundle
Bundle-Version: 1.0.4
Bundle-Activator: com.agiledeveloper.FirstBundle
Import-Package: org.osgi.framework
Export-Package: com.agiledeveloper;version=1.0.4
Export-Service: com.agiledeveloper.TimeService
```

Exporting Bundle's Manifest

```
Bundle-SymbolicName: com.agiledeveloper.user.BundleUser
Bundle-Version: 1.0.0
Bundle-Activator: com.agiledeveloper.user.BundleUser
Import-Package: org.osgi.framework, org.osgi.util.tracker,
                com.agiledeveloper;version="[1.0.2, 1.0.5)"
Import-Service: com.agiledeveloper.TimeService
```

Importing Bundle's Manifest

Will use latest version between 1.0.2 to 1.0.4

Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

Spring OSGi Goals

- Spring modules are packaged as OSGi bundles
- Allows you to expose POJOs as OSGi service
- OSGi apps can import and use Spring packages and services
- Through Spring-OSGi project
- Stronger tie in Spring 2.1

OsgiBundleXmlApplicationContext

- Spring context based on OSGi bundle
- Allows you to resolve bundle entries
- Programmatic or declarative configuration

ConfigurableBundleCreatorTests

- Starts OSGi container
- Installs bundles
- Runs tests
- Provides access to BundleContext

Agenda

- What's OSGi?
- OSGi Bundles
- Applications
- Services
- Real World OSGi
- Eclipse
- Developing Bundles
- Service interaction
- Spring and OSGi
- Conclusion

Quiz Time

References

- http://www.osgi.org
- http://www.osgi.org/blog/
- http://www.knopflerfish.org
- http://www.eclipse.org/equinox/
- http://www.eclipse.org/osgi/
- http://cwiki.apache.org/FELIX/index.html
- http://www.springframework.org/osgi
- Campanelli's comparison and EcoSystem article: http://tinyurl.com/2m3t2m and http://tinyurl.com/246t4r
- "About the OSGi Service Platform," Technical Whitepaper, OSGi Alliance, Nov. 2005.

Thank You!

http://www.agiledeveloper.com — download