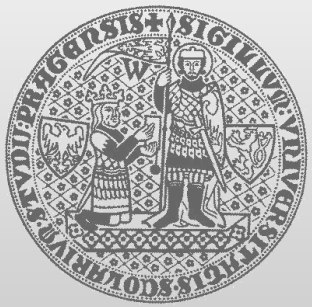
NPRG044: OSGi framework

http://d3s.mff.cuni.cz

michal malohlava

michal.malohlava@d3s.mff.cuni.cz



CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Step #1:
Download Eclipse RCP
http://www.eclipse.org/downloads/

Do you use JARs?

JAR advantages and disadvantages

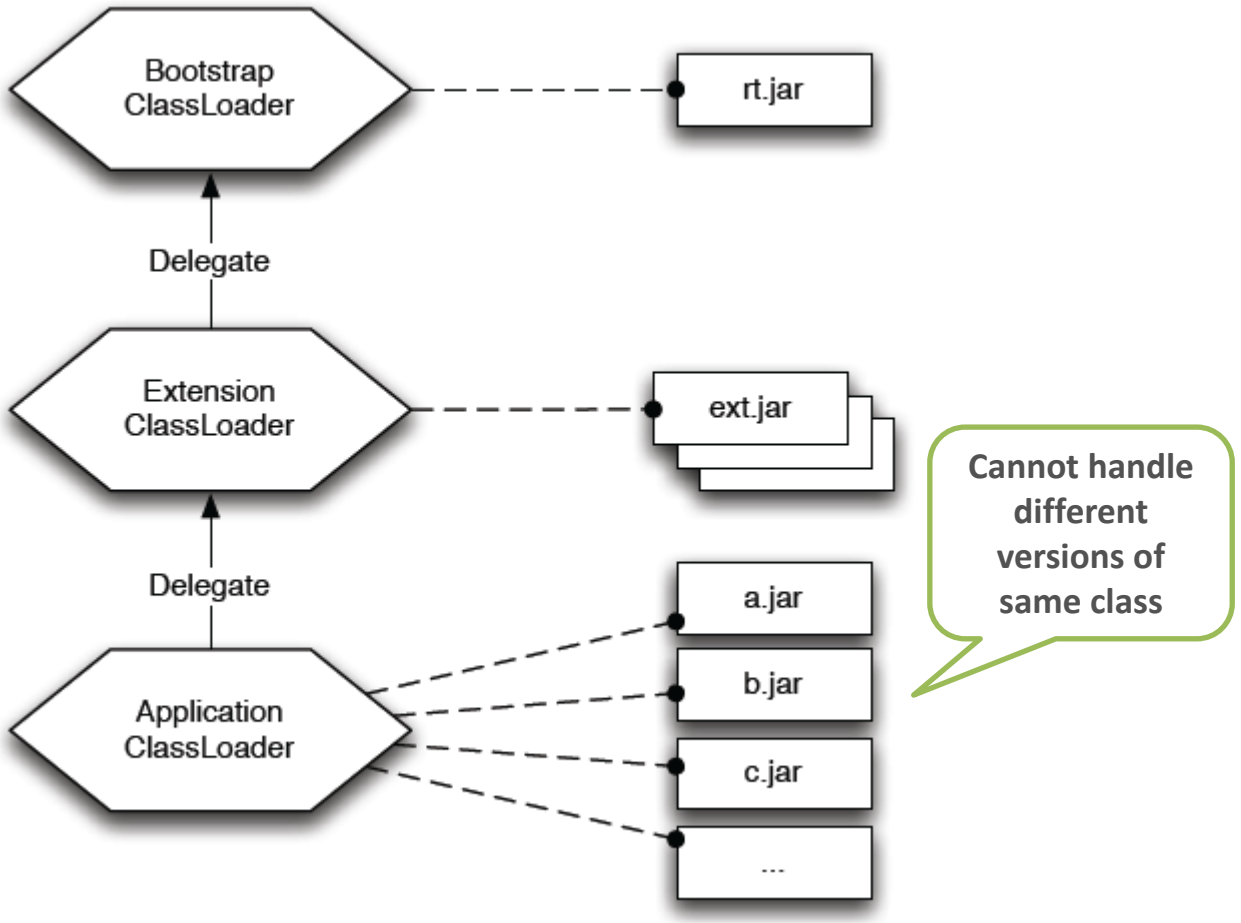
+ Includes class files and additional resources
+ Deployment

- No information hiding
- No dynamicity
- Cannot specify required JARs
- No versioning

OK, it's not completely true, but JARs classpath is almost entirely useless

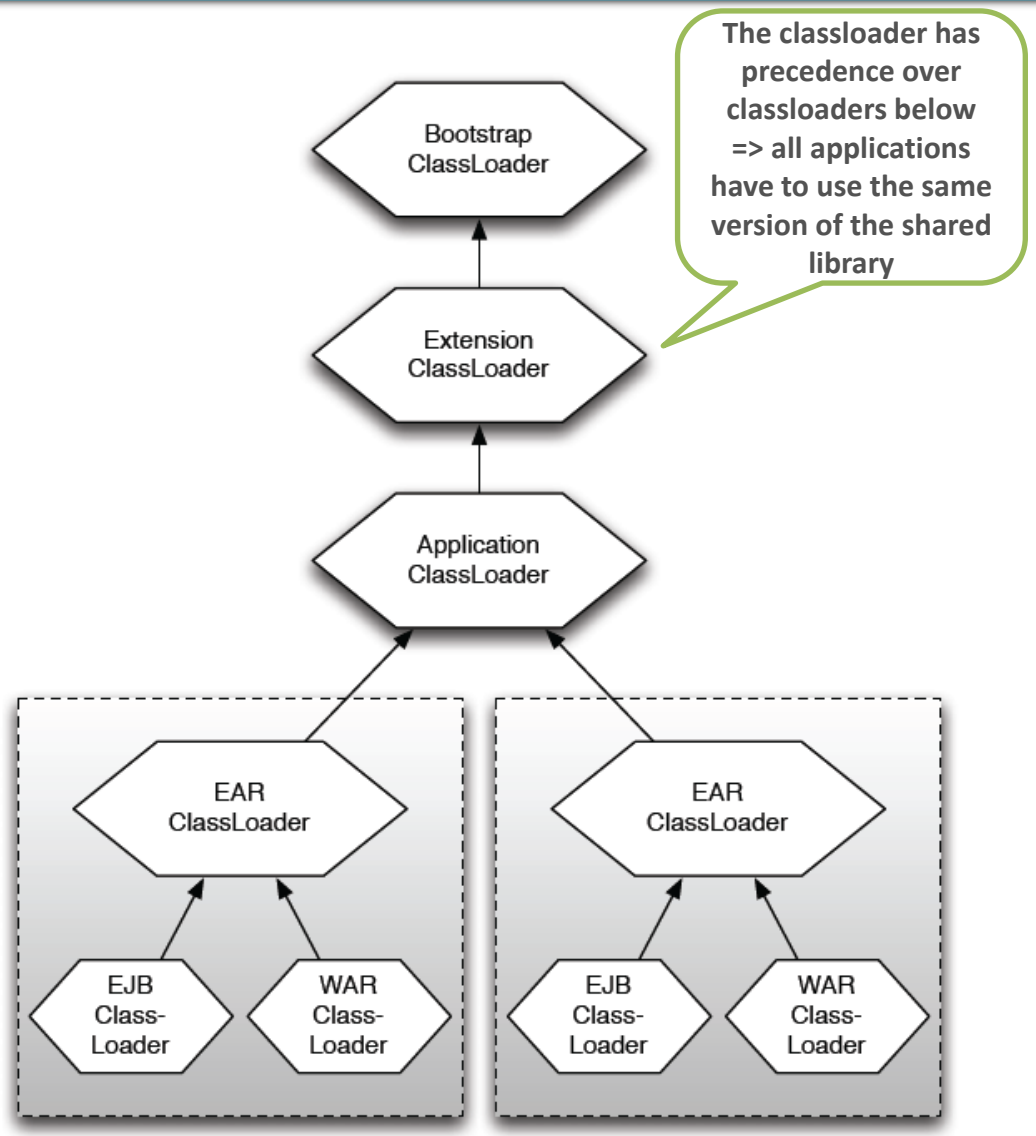
Department of Distributed and Dependable Systems D3S

Common Java classloading



The picture was taken from the book "OSGi in Practice" written by Neil Bartlett
See http://njbartlett.name/osgibook.html

J2EE classloading



The picture was taken from the book "OSGi in Practice" written by Neil Bartlett
See http://njbartlett.name/osgibook.html

OSGi

- OSGi is a specification
  - Open Service Gateway Initiative
  - Current version R4.3 (see http://www.osgi.org/)
  - 3 parts + Java API + execution environment specification
    - Core
    - Compendium
    - Enterprise
- Specifies
  - Framework
  - Modules
  - Provided services
  - Additional functionality

OSGi framework

- Framework to build modular applications
  - "LEGO principle"
  - Fine-grained modules which are
    - Reusable
    - Scalable
    - Isolated
    - Bringing separation of concepts
      - Modules should be "easily" testable, manageable, maintainable, repairable, exchangeable
    - Bringing abstraction

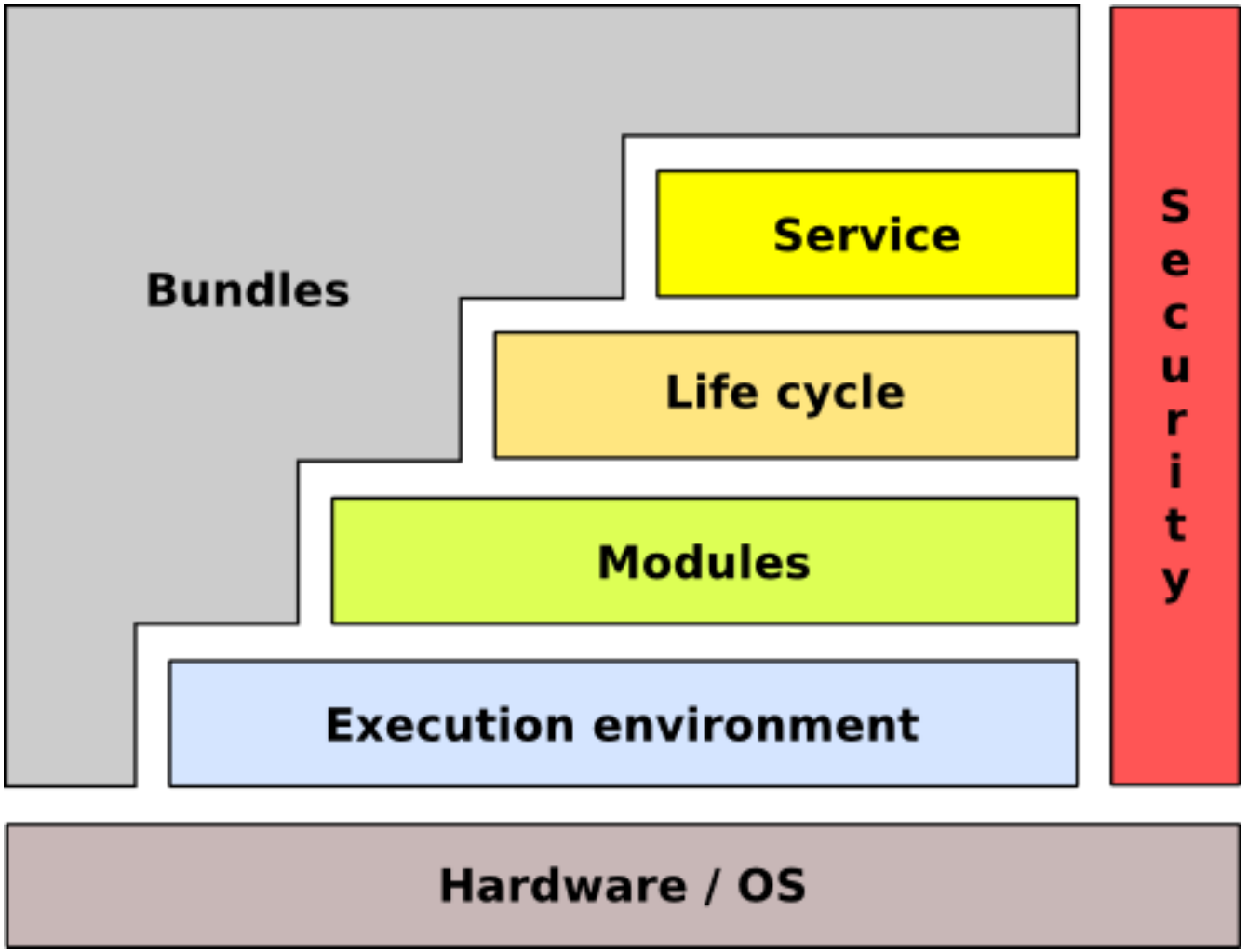
OSGi applications

- Existing applications
  - BMW service platform
  - Eclipse
  - Virgo server (Spring dm Server)
  - GlassFish J2EE application server
  - IBM WebSphere J2EE application server
  - Newton
  - JBoss, JOnAS
  - Apache Karaf
- Users
  - Bombardier, Volvo, Siemens, BMW, IBM, Red Hat, Siemens AG, NEC, Oracle

OSGi implementations

- Open source
  - ***Eclipse Equinox***
    - Many extensions of OSGi, e.g. bundle aspects, extension points
  - *Apache Felix*
    - Based on Oscar (implementation of OSGi R3)
    - Compliant to OSGi specification R4.2
  - *Knopflerfish*
  - *Concierge*
    - Implementation of OSGi R3, optimized for embedded devices
- Commercial
  - ProSyst, Knopflerfish Pro

OSGi framework conceptual architecture



The picture was taken from the book “OSGi in Practice” written by Neil Bartlett
See http://njbartlett.name/osgibook.html

OSGi Basic concepts

- **Bundle**
  - Module
  - Unit of deployment
- **Service**
  - Communication between components

Bundle

- **Unit of deployment**
  - Classical JAR with meta-information
    - Class files,
    - Additional resources (images, videos, source code, …)
    - META-INF, OSG-INF directories containing meta-information
- **Bundle is versioned**
  - Major, minor, micro, qualifier (1.0.3_rc2)
  - Multiple versions at runtime are allowed
- **Bundle can export/hide packages**
  - *Recommended practice: "*Exposing only API not implementation"
  - Declarative dependencies
    - Bundles
    - Packages
      - Range of version [1.0, 2.0)

Bundle meta-information

- Manifest META-INF/MANIFEST.MF

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: LogTargetBundle
Bundle-Activator: LogTargetActivator
Bundle-SymbolicName:
cz.cuni.mff.d3s.LogTargetBundle
Bundle-Version: 1.0.0.qualifier
Bundle-Vendor: D3S MFF UK
Bundle-RequiredExecutionEnvironment: JavaSE-1.6
Import-Package:
cz.mff.cuni.d3s.nprg044.tut1.test01.api,
org.osgi.framework;version="1.5.0",
org.osgi.service.component;version="1.1.0",
org.osgi.service.log;version="1.3.0"
Service-Component: OSGI-INF/componentOne.xml, OSGI-
INF/factory.xml
```

Each line is limited 72bytes by design of JVM

Bundle dependencies

```
Export-Package: cz.*;
exclude="*Impl"
```

- Export packages
  - List all of packages + versions + attributes
  - Fine grained package filtering
    - Exclude,include, parameters

```
Import-Package: cz.mff.*;
version="[1.0,1.3.1)";res
olution=optional
```

- Import package
  - Require specific version(s)
    - e.g. [1.0, 2.0)
  - Resolution: **optional/mandatory**

```
Require-Bundle: logger-api-bundle
```

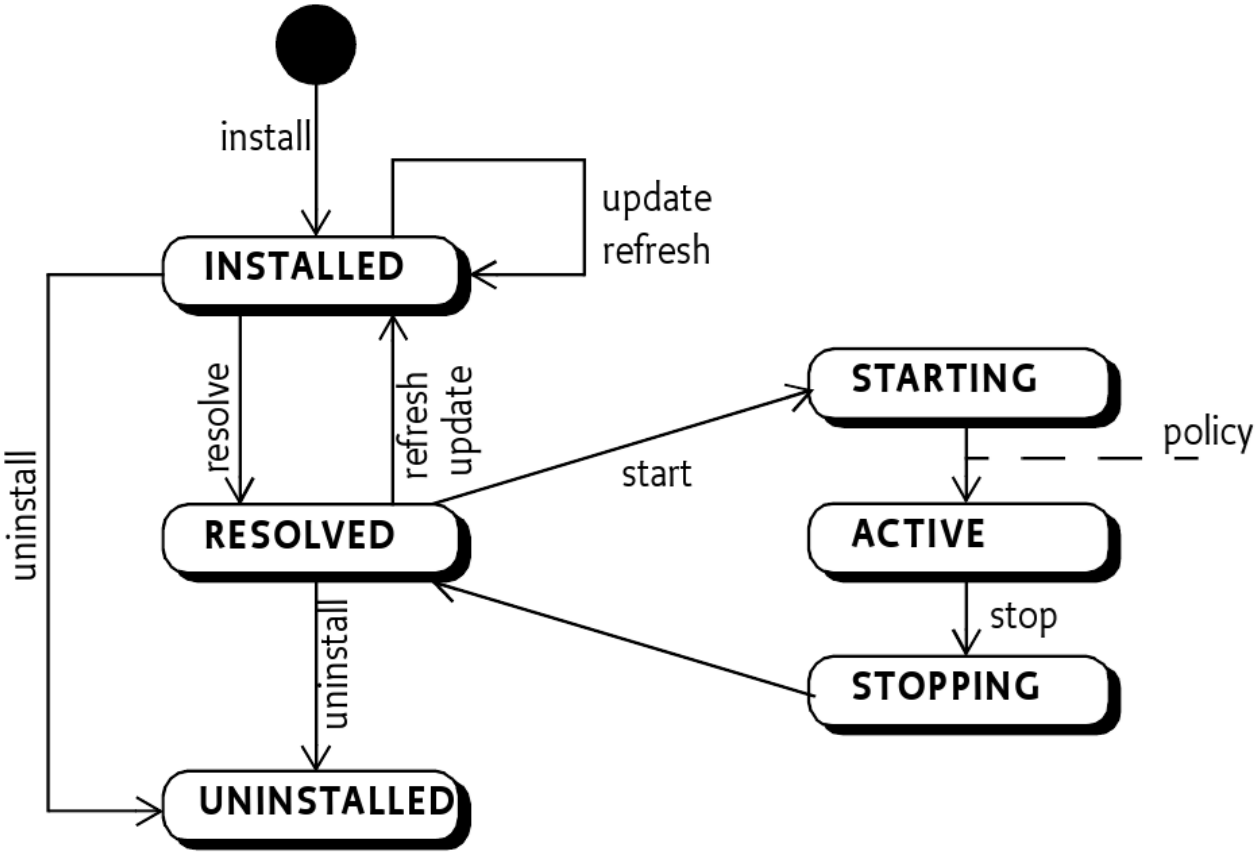
- Require bundle
  - Not recommended because it restricts further changes in API

Department of Distributed and Dependable Systems D3S

MANIFEST.FM headers

- Bundle-ClassPath
  - Way to bundle 3rd party JAR libraries
- Bundle-Activator
  - Name of class implementing BundleActivator
  - The class is called when the bundle is activated
- Bundle-SymbolicName
  - Bundle ID
- Bundle-Version
  - 1.0.3.qualifier (*qualifier* corresponds to timestamp)
- Bundle-RequiredExecutionEnvironment
  - Enforces the execution context
- DynamicImport-Package
  - On-the-fly import
- Bundle-NativeCode
  - Import .so, .dll

Bundle lifecycle



The picture was taken from the book "OSGi in Practice" written by Neil Bartlett
See http://njbartlett.name/osgibook.html

Bundle lifecycle - activation

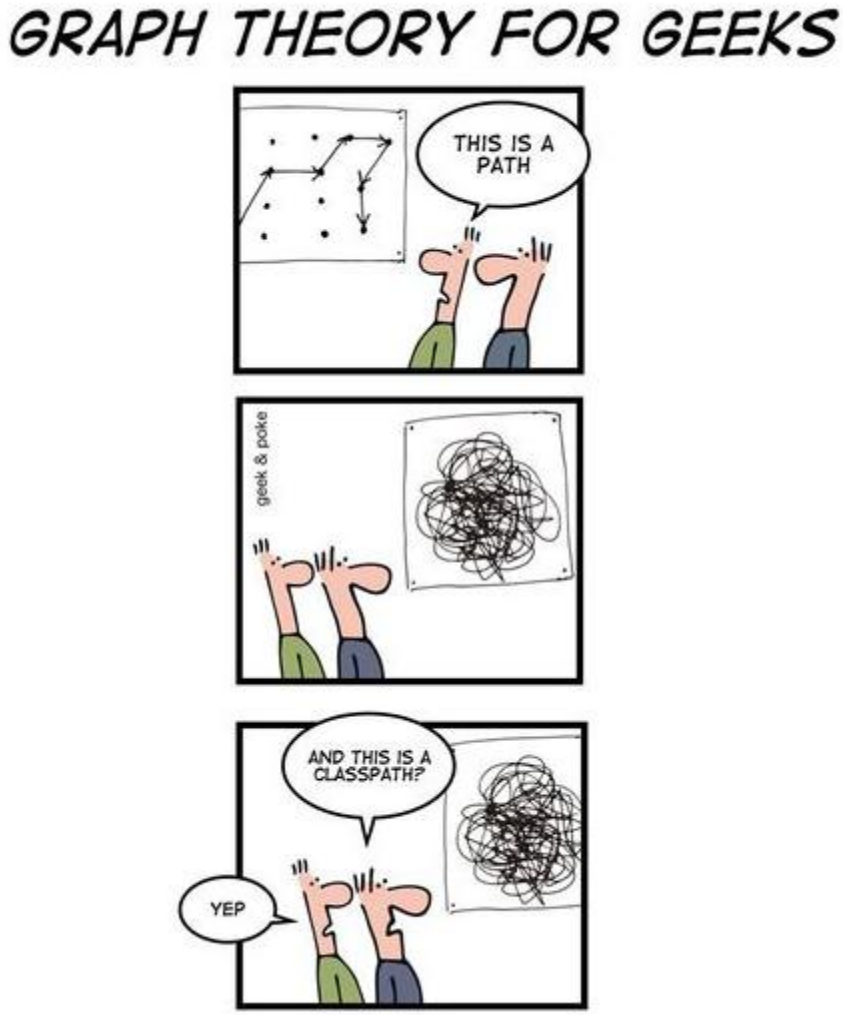
- Manage bundle lifecycle
- class ***BundleActivator***
  - ***void start(BundleContext ctx)***
    - Register service, listeners, look for services
  - ***void stop(BundleContext)***
    - Stop trackers, listeners, …
- class ***BundleContext***
  - Properties
  - Services
  - Bundles
  - Filters
  - Listeners

```
public class SimpleLogTargetActivator
                    extends Activator {

  @Override
  public void start(BundleContext context){
   /* ... */
  }

  @Override
  public void stop(BundleContext context) {
   /* ... */
  }
}
```

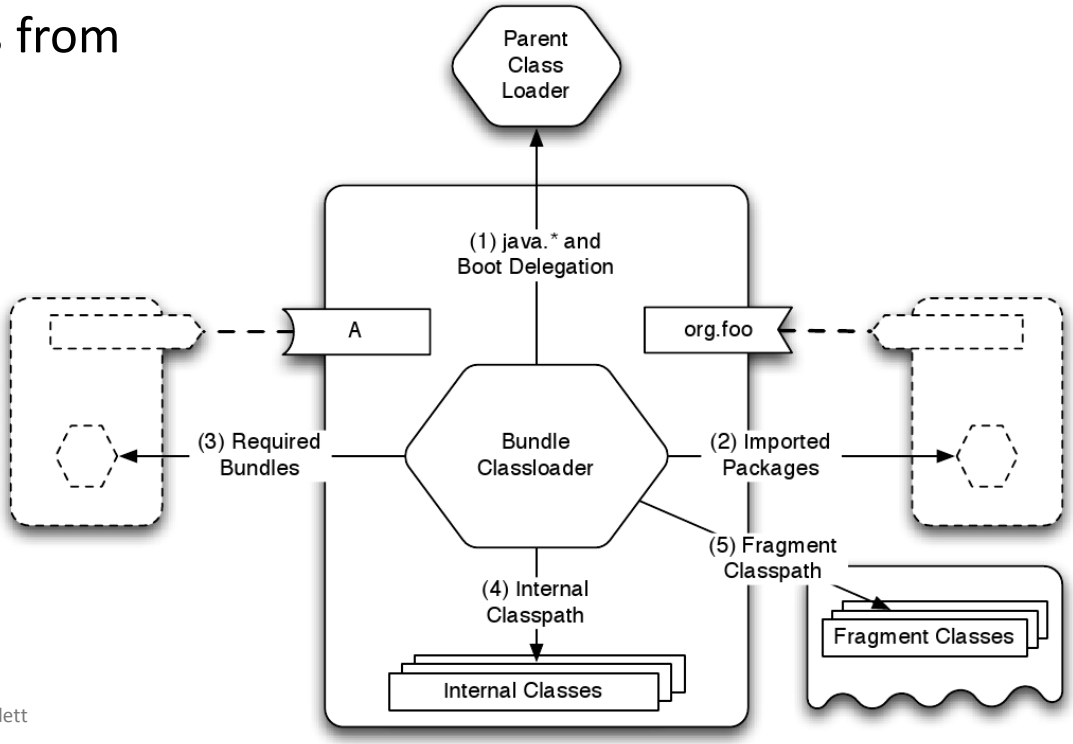
OSGi classloading



The picture was taken from the OSGi wiki
See http://wiki.osgi.org/wiki/Main_Page

OSGi classloading

- Separated class loader per bundle
  - Classloaders do not compose a tree, but a general graph
- Lookup order
  - Parent
    - only for classes from java.* package
  - Imported packages
  - Required bundles
  - Local bundle classpath



The picture was taken from the book "OSGi in Practice" written by Neil Bartlett
See http://njbartlett.name/osgibook.html

Bundle classpath

- Bundle classpath is composed of classes from
  - Imported packages
  - Provisions of required bundles
  - Local classpath specified via *Bundle-Classpath*

Demo #01

- Download Eclipse RCP
  - http://www.eclipse.org/downloads/
- Create a simple bundle with activator
  - Via wizard in *"New > Project > …"*
- Bundle the bundle
- Deploy bundle
  - Create new OSGi launch configuration & launch it
    - Package *org.eclipse.osgi* is required to be selected
    - Use "Add required bundles"
- Observe its state in the console

OSGi Console

- Important commands
  - *help*
  - *ss*
    - Display installed bundles
  - *services*
    - Display published services
  - *status*
  - *exit*
    - Shutdown OSGi framework
  - *start/stop <bundle-id>*
  - *Update <bundle-id>*
  - *packages*
    - *Shows exported packages*
  - *diag*
    - *Run diagnostic*

Service

- **Communication layer for bundles**
  - Well-defined communication points
  - Inherent dynamic nature
    - Can appear/disappear any time at runtime
  - Multiple providers can provide the same services
    - The service has additional properties which can for example specify its priority
- **Service**
  - Service is an object registered by a bundle in a *ServiceRegistry*
    - Programmatically
    - Declaratively
  - Service has associated properties
    - E.g., *service.ranking*

Registering service (1)

- Programmatically in BundleActivator

```
public void start(BundleContext context) {
  SimpleLogTargetImpl logTargetImpl = new SimpleLogTargetImpl();

  registration = context.registerService(
              ILogTarget.class.getName(),
              logTargetImpl,
              null);
}
```

- Problems
  - Code
    - Services semantics is spread over code
      - dependencies, properties, implementation v. provided interface

Registering service (2)

- Declarative services (DS)

```
<scr:component name="logger-component"
            enable="true" activate="activate">
        <implementation class="cz.cuni...LoggerImpl"/>
        <service>
                <provide interface="cz...ILogger"/>
        </service>
</scr:component>
```

- Declaratively
  - Services provided by *components*
  - Automated service management by DS framework
    - Dependency injection of required services
    - Life-cycle management

Department of Distributed and Dependable Systems D3S

Service components

- Component is a normal Java class contained within a bundle
- Defined in separated XML file in *OSGI-INF* directory
  - *MANIFEST.FM*
    - Has to contain component file reference: *Service-Component: OSGI-INF/component.xml*

```
<scr:component name="logger-component"
           activate="activate">
        <implementation class="cz.cuni...LoggerImpl"/>
        <service>
                <provide interface="cz...ILogger"/>
        </service>
</scr:component>
```

- Activation
  - Declared method
    - ComponentContext, BundleContext, Map as parameters
- Service provider
  - Specify name of provided service

Demo #02

- Implement a bundle with API
- Implement two bundles implementing API
- Register API services
  - Programmatically
  - Declaratively
- Launch configuration has to contain *'org.eclipse.equinox.ds'* bundle
- Observe provided services in console (command *services)*

Department of Distributed and Dependable Systems D3S

Service consumption (1)

- Bundle can search for service which implements specific interface
- Several bad solution
  - context.getService(...)
    - Nasty code with active waiting
  - Service registry listeners

```
Service reference ref =
context.getServiceRef("cz.bar");
if (ref!=null) {
  Bar bar = (Bar) context.getService(ref);
  if (bar != null) {
   ...
   context.ungetService(ref)
  }
}
```

- Recommended solutions (thread-safe)
  - **Service tracker**
  - **Components**

Service tracker – white board pattern

- Services dependencies
  - Content provider v. consumers
    - e.g., consume a new service if and only if the specified service appears
  - *“Don't look for content providers, let them to register as services and track for the services”*
  - *ServiceTracker* capture service life-cycle
    - via *ServiceTrackerCustomizer*
      - Capture process of adding/removing/modifying service

Service tracker

- **Service Tracker**
  - Tracking for service
    - Filters (name, id, property, owning bundle, ...)
      - LDAP syntax (e.g. *(&(objectName=”foo”)(property1=”Xyz”))*)

```
//In Bundle Activator - start
tracker = new ServiceTracker(context,
  ILogger.class.getName(), null);

tracker.open();

// get the service(s)
ILogger log = (ILogger)tracker.getService();
ILogger log = (Ilogger) tracker.waitForService(1000);

// stop tracking
tracker.close();
```

Service tracker

- Construction determines attributes

```
ServiceTracker(
        BundleContext context,
        java.lang.String clazz,
        ServiceTrackerCustomizer customizer)
```

- ServiceTracker methods
  - *open()/close()* – start/stop tracking for a service
  - *getService()*
  - *addingService/removedService/modifiedService(ServiceReference rf)*
    - Interface ***ServiceTrackerCustomizer***
    - Can be overridden by user

Service consumption (2)

- Declaratively via **service components**

- Service reference
  - Name
  - Interfaces
  - Bind/unbind method
  - Target
  - Policy: static/dynamic
  - Cardinality: M..N
    - 1..1 – if multiple services is accessible then the one with the highest *service.ranking* is used

```
<scr:component name="getServiceComp">
 <implementation class="GetLoggerService">
 <reference name="log"
   interface="org.osgi...LogService"
   bind="setLog"
   unbind="unsetLog"
</scr:component>
```

Services lookup

- Lookup strategy
  - Look for service during component activation

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component name="example.listen"
  xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0">
 <implementation class="com.acme.LogLookupImpl"/>
 <reference name="LOG"
    interface="org.osgi.service.log.LogService"/>
</scr:component>
```

```
public class LogLookupImpl {
 private void activate(ComponentContext ctxt) {
  LogService log = (LogService)
            ctxt.locateService("LOG");
  }
}
```

- Event strategy
  - Let the DS framework to inject the service via defined methods
  - Bind/unbind attributes of reference declaration

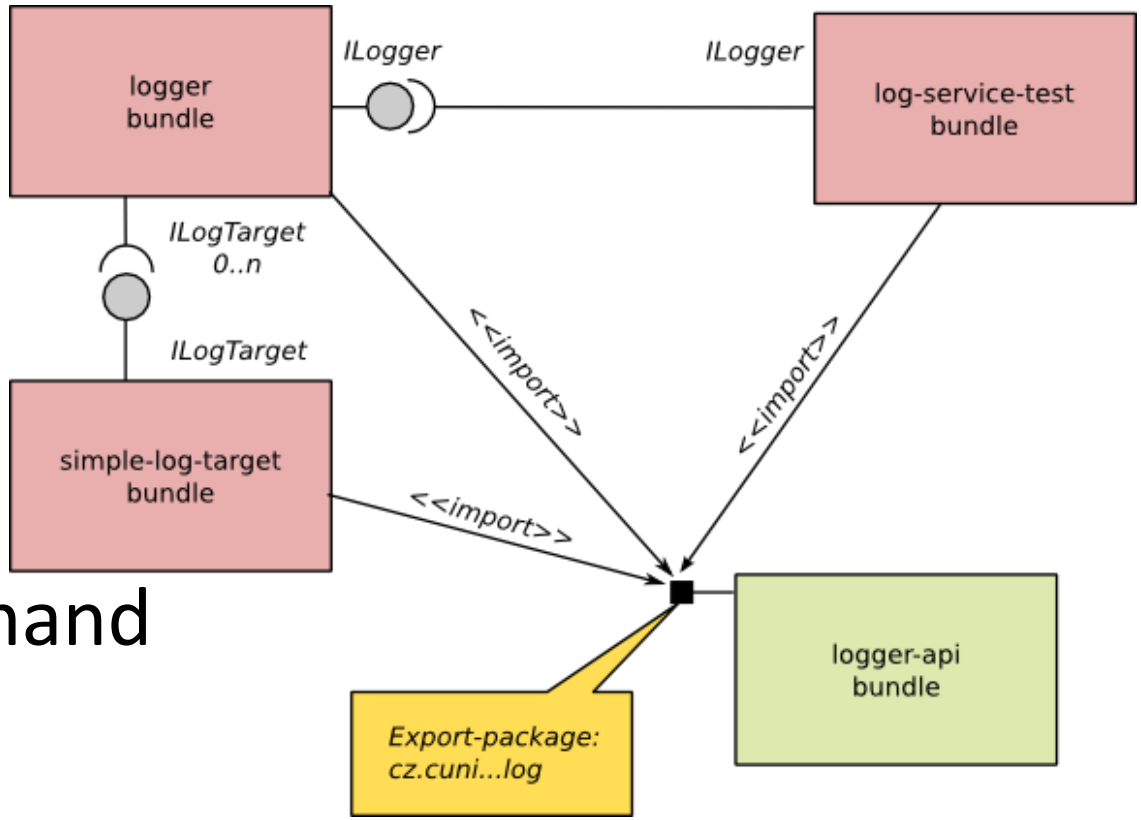
Department of Distributed and Dependable Systems D3S

Component factories

- A component can be declared as a factory
  - ***ComponentFactory*** service is registered
    - ***newInstance(Dictionary d)*** method
  - The user tracks for Component Factory service and create a new instance
- A component factory can provide a service
  - Registered for each created instance

Demo #03

- Write a new bundle with a tester component consuming declared services in demo #02
- Possible scenario
- Try to call *update* command in console



OSGi services

- Logging (*LogService*)
- Http (*HttpService*)
  - Exposing registered servlets
- Event
  - Messaging Producer <-> Consumer
- Device manager
- Diagnostics/Monitoring
  - JMX
- Application manager
  - Application package – set of resource (bundles, data,...)
    - Can be deployed/install
- Location/measurement services
- Remote services

Demo #04

- Use HttpService as an additional implementation of ILogTarget
  - Introduce a new bundle with a component exposing ILogTarget and requiring HttpService
- Register HTTP servlet
- Launch configuration has to introduce web-server bundles (e.g., Jetty). Don't forget on
  - javax.servlet
  - org.eclipse.equinox.http.servlet
  - org.eclipse.equinox.http.jetty
- Specify JVM property:
  -Dorg.osgi.service.http.port=8080

OSGi 4.2 features

- Framework launching
- Remote services
- Blueprint services
- Bundle tracker
- Service hooks
- Conditional permissions

- Enterprise features – bundling (WAR), JPA, JNDI, JDBC integration

OSGi 4.3 features

- Introduction of generics into OSGi API
- Capabilities
- Requirements
- Adapt concept
  - Bundle can be adapted to another type
- Weaving hook
  - Bytecode modification
- Resolver, bundle, service events hooks

Bundles repositories

- OBR
  - http://bundles.osgi.org
  - OSGi compendium implementation
- Spring
  - http://sigil.codecauldron.org/spring-external.obr
  - http://sigil.codecauldron.org/spring-release.obr
- Knopflerfish
  - http://www.knopflerfish.org/repo/bindex.xml

Department of Distributed and Dependable Systems

Resources

- OSGi specification
  - http://www.osgi.org/
- OSGi tooling
  - http://en.wikipedia.org/wiki/OSGi-Tooling
- NPRG044 source code
  - http://code.google.com/a/eclipselabs.org/p/nprg044-eclipse-platform/

Department of Distributed and Dependable Systems D3S

OSGi tools

- Pax
- Bnd
- Scribids