OSGi

Dynamisches Komponenten Modell von Java

https://docs.osgi.org/specification/osgi.core/7.0.0/
https://github.com/jwausle/osgi-schulung

OSGi Agenda

1. Einführung, Geschichte
2. Bundle
3. Bundle MANIFEST.MF (Bundle-Header)
4. Bundle Version
5. Bundle Class Loading (Demo + Übung)
6. Bundle State/Lifecycle (Demo + Übung)
7. Service (Demo + Übung)
8. ServiceTracker (Demo + Übung)
9. Declarative Service (Demo + Übung)
10. Configuration Admin (Demo + Übung)

Was nicht?

- Kein Eclipse RCP
- Kein Maven Tycho
- Kein Maven Bnd
- Kein Bnd deep dive
- Kein Gogo deep dive

Timeline

- 09 - 12 Vormittags
- 12 - 13 Mittagspause
- 13 - 15/16 Nachmittags
- Pausen je nach Bedarf

https://docs.osgi.org/specification/osgi.core/7.0.0/

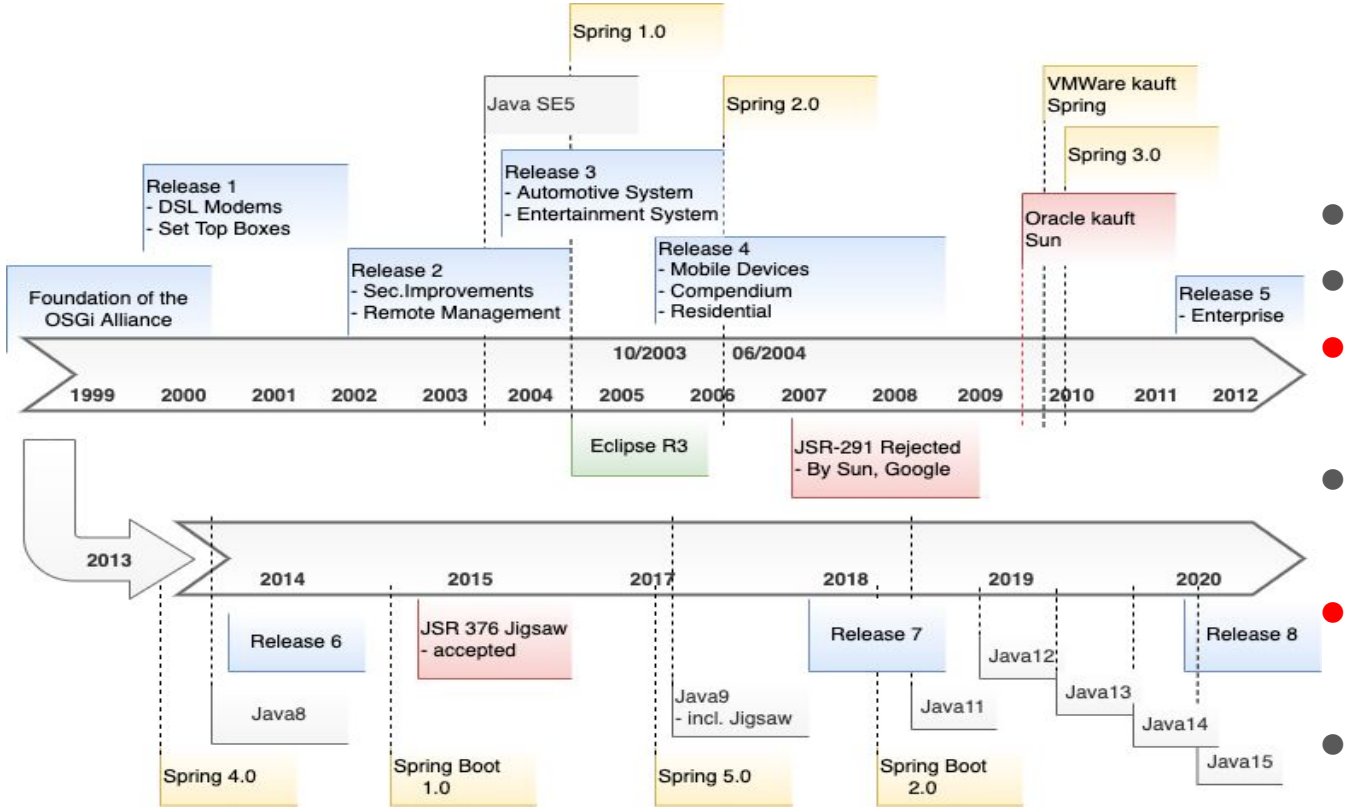
Jan Winter

- 47 Jahre, lebe in Leipzig
- Seit ~ 20 Jahren Software Entwickler
  - Seit 6 Jahren Freiberufler
  - 7 Jahre Berater bei der Itemis AG
    - 3 Jahre bei Telekom/Qivicon ~ Eclipse-Smarthome/Openhub
  - 5 Jahre in einem Startup
  - Freiberuflich während des Studiums
- Seit ~ 15 Jahren OSGi Entwickler
- Seit ~ 10 Jahren Trainer (Nebentätigkeit)
  - Git, Gitlab CI, Bitbucket CI
  - Kubernetes/Docker, GitOps/FluxCD
  - OSGi

OSGi vs. JEE vs Spring

	OSGi	J2EE/JEE/Jakarta	Spring
Spezifikation/ Framework	Spezifikation	Spezifikation	Framework
Classpath	Hierarchisch, Dynamisch	Hierarchisch, Dynamisch	Klassisch, Statisch
Services/CDI	Dynamisch	Dynamisch	Statisch
Einstiegshürde	Hoch	Hoch	**Niedrig**
Organisation	Alliance/NGO	Sun/Oracle	VMWare
Implementierungen	Apache Felix/Equinox	Wildfly/Glassfish	Spring/-Boot

OSGi Geschichte



- 1999 Founded
- 2004 Eclipse R3
- 2007 JSR-291 rejected
- 2009 Oracle kauft Sun
- 2014 JSR-376 Jigsaw accepted
- 2017 Java 9

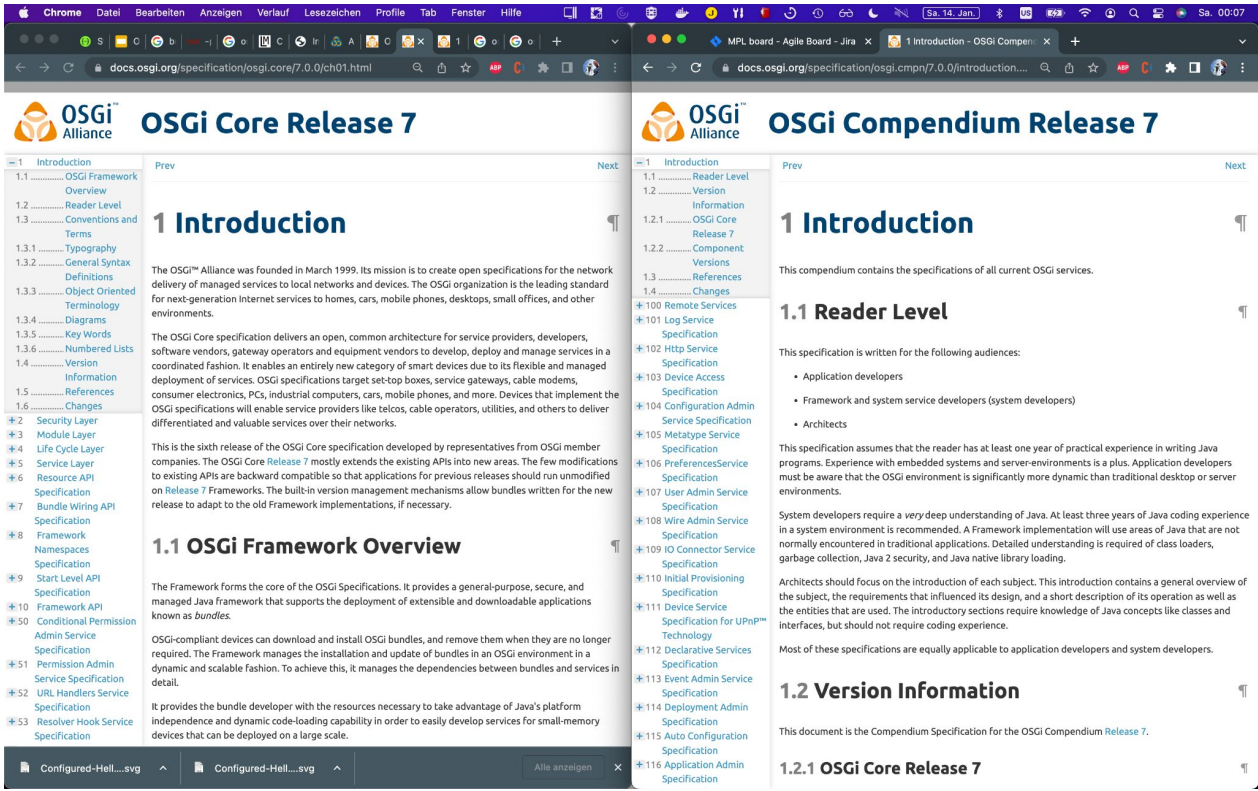
OSGi - Spec

Core Spec - Basis

- Bundle
- Bundle Lifecycle
- Start level
- Service
- Security
- …

Compendium

- Declarative Service
- Configuration Admin
- Http Service Spec
- Http Whiteboard
- …



https://docs.osgi.org/specification/#release-7

OSGi - Jetzt geht’s los

- Part 1: Bundle - Classpath
  - MANIFEST.MF
  - Version
  - Class Loading
  - Lifecycle
- Part 2: Services
  - Service Reference
  - Service Tracker
  - Declarative Services
  - Config Admin

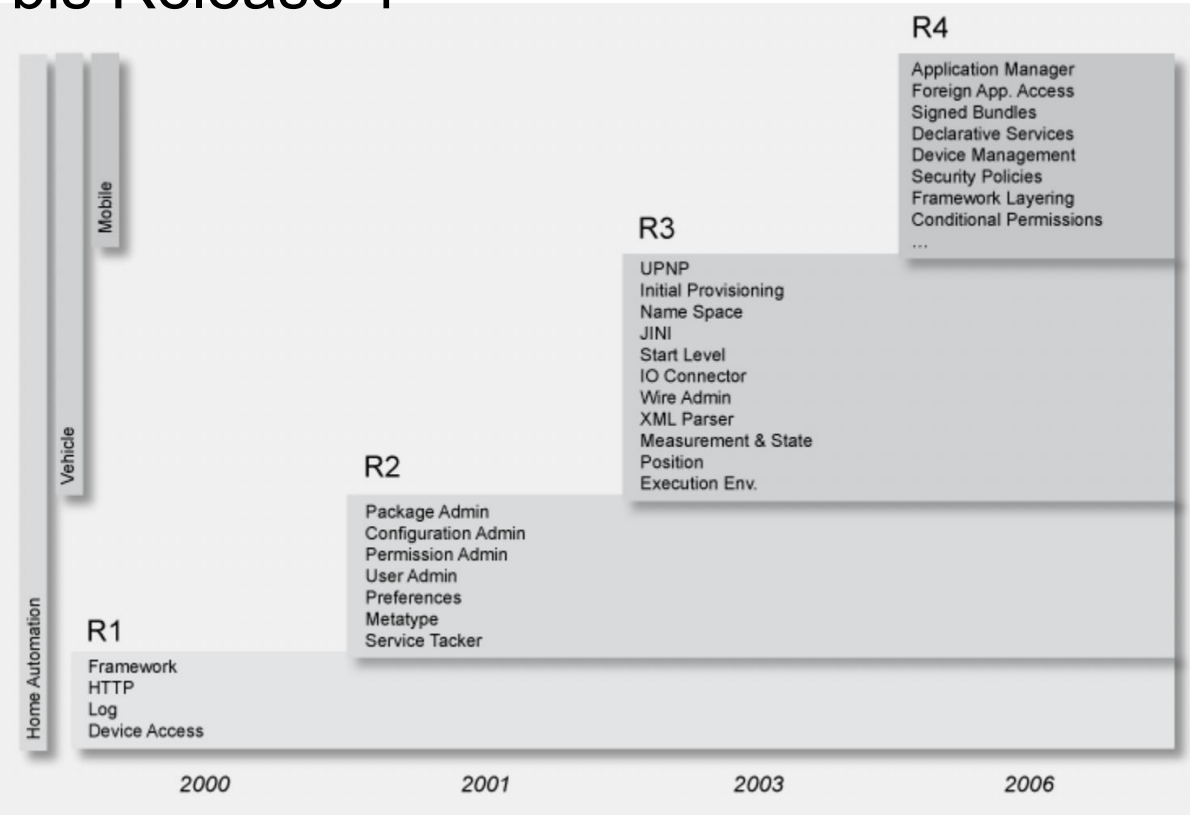
OSGi - Spec History bis Release 4

R1 - Release 1

R2 - Release 2

R3 - Release 3

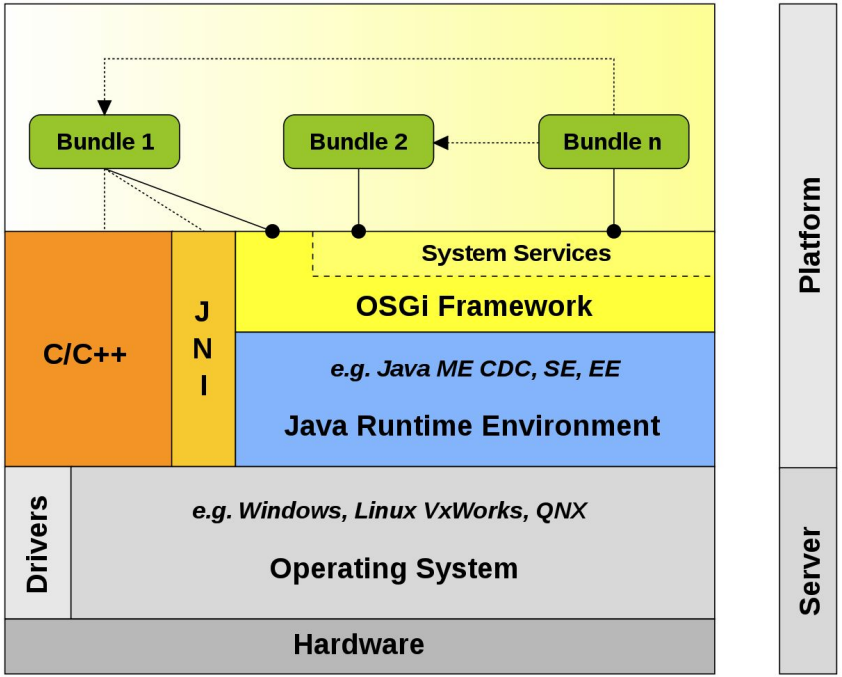
R4 - Release 4



https://docs.osgi.org/specification/#release-7

OSGi Bundle

- **Bundle** ~ Module/JAR
- **Bundle** ~ Package Provider
- **Package** ~ API/Service/Component Provider
- **API** ~ Interface
- **Service** ~ API Instance
- **Component** ~ Class Instance
- **Version**ierbare Bundle, Package, API, Service



Bundle - META-INF/MANIFEST.MF

Bundle-SymbolicName: com.acme.daffy.jan

Bundle-Version: 1.1

Export-Package: com.acme.daffy.tracker;version=1.4

Import-Package: org.osgi.util.tracker;version=1.4

DynamicImport-Package: com.acme.plugin.*

Require-Bundle: com.acme.chess,com.acme.chess.2

Bundle-ClassPath: /provided-lib.jar,.

Fragment-Host: org.eclipse.swt

Bundle-Activator: com.acme.fw.Activator

Bundle-RequiredExecutionEnvironment:
CDC-1.0/Foundation-1.0

Bundle-ContactAddress: Leipzig

Bundle-Copyright: OSGi (c) 2022

Bundle-Description: Network Firewall

Bundle-Developers: Jan Winter

Bundle-Icon: /icons/acme-logo.png;size=64

Bundle-DocURL: http://www.example.com/doc

Bundle-License: Apache-2.0

Bundle-ManifestVersion: 2

Bundle-Name: Firewall

Bundle-Vendor: OSGi Alliance

https://docs.osgi.org/specification/osgi.core/7.0.0/framework.module.html#i2654895

Bundle - META-INF/MANIFEST.MF (Sample)

Manifest-Version: 1.0

Bundle-ManifestVersion: 2

Bundle-Name: de.jwausle.osgi.api.consumer.v1

Bundle-SymbolicName: de.jwausle.osgi.api.consumer.v1

Bundle-Version: 1.0.0.202204241327

Import-Package: de.jwausle.osgi.api.provider;version="[1.0,2)"

Export-Package: de.jwausle.osgi.api.consumer.v1;version="1.0.0"

Private-Package: de.jwausle.osgi.api.consumer.v1.internal

Require-Capability: osgi.ee;filter:="(&(osgi.ee=JavaSE)(version=12))"

Bundle - Version

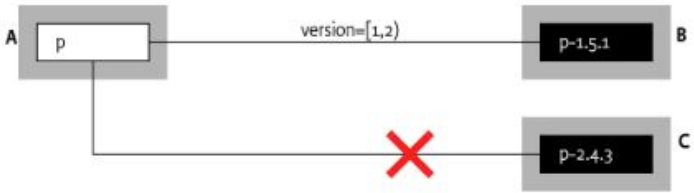
1.5.1 ~ {major}.{minor}.{micro}.rest

- **major** - Changes for an incompatible update for both a consumer and a provider of an API
- **minor** - Changes for a backward compatible update for a consumer but not for a provider.
- **micro** - A change that does not affect the API, for example, a typo in a comment or a bug fix in an implementation.

```
A: Import-Package: p; version="[1,2)"
B: Export-Package: p; version=1.5.1
```

Figure 3.9 graphically shows how a constraint can exclude an exporter.

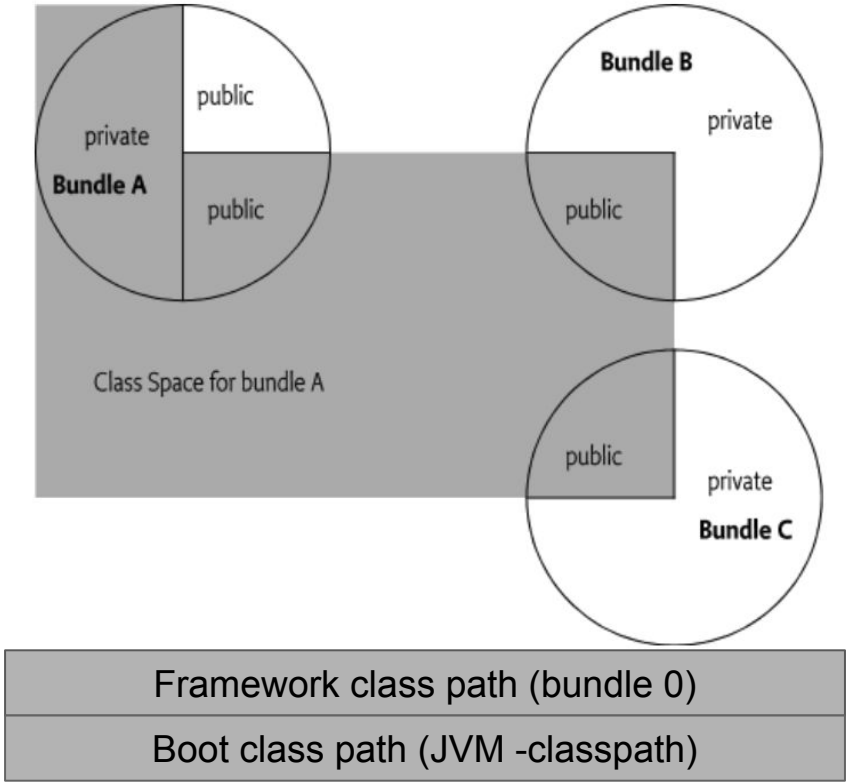
Figure 3.9 Version Constrained



https://docs.osgi.org/specification/osgi.core/7.0.0/framework.module.html#d0e3894

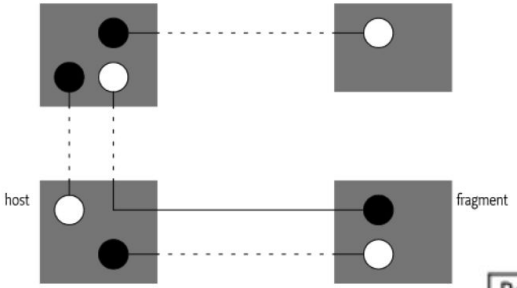
Bundles - Class Loading Architecture

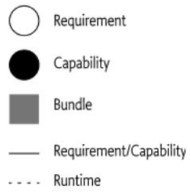
- *Boot class path* - The boot class path contains the java.* packages and its implementation packages.
- *Framework class path* - The Framework usually has a separate class loader for the Framework implementation classes as well as key service interface classes.
- *Bundle Space* - The bundle space consists of the JAR file that is associated with the bundle, plus any additional JAR that are closely tied to the bundle, like *fragments*



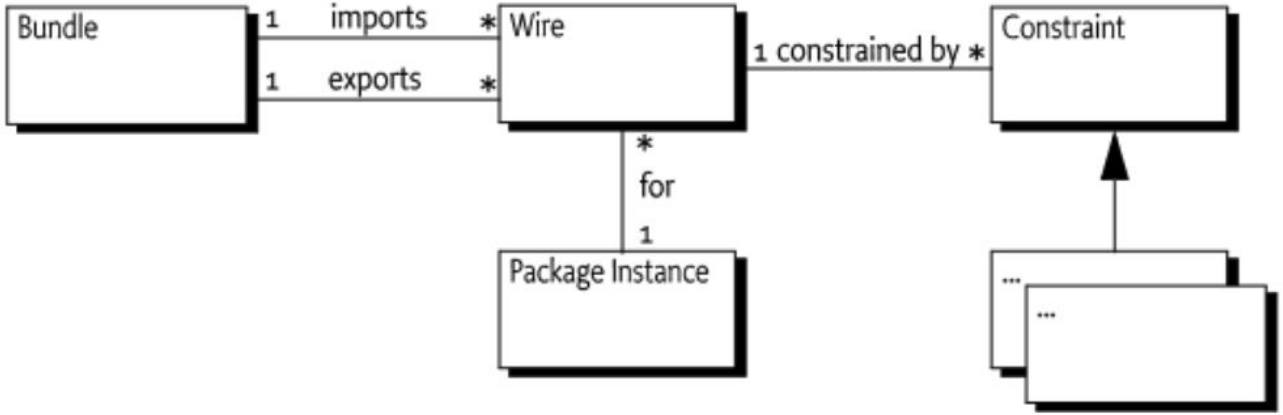
https://docs.osgi.org/specification/osgi.core/7.0.0/framework.module.html#i3174728

Bundle - Resolving





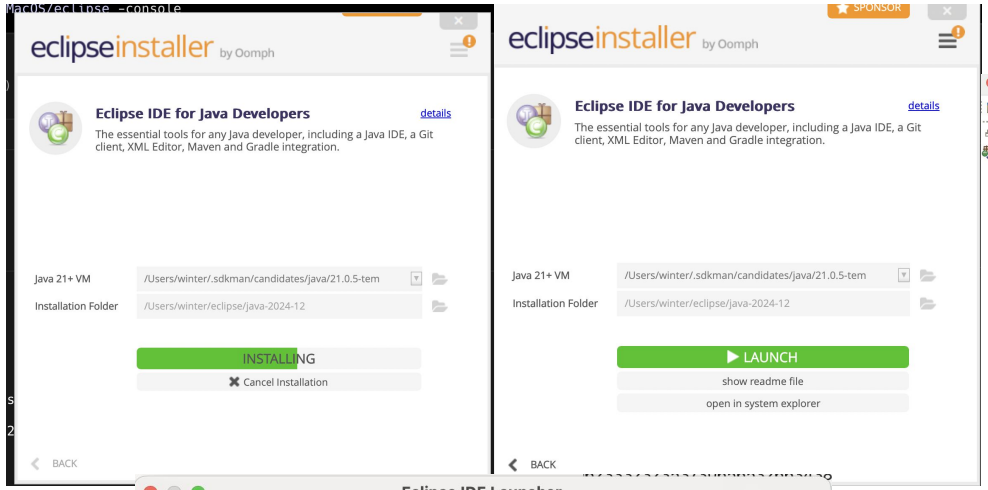
osgi.wiring.package

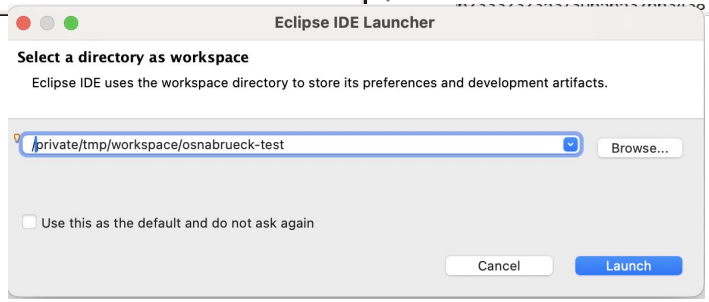


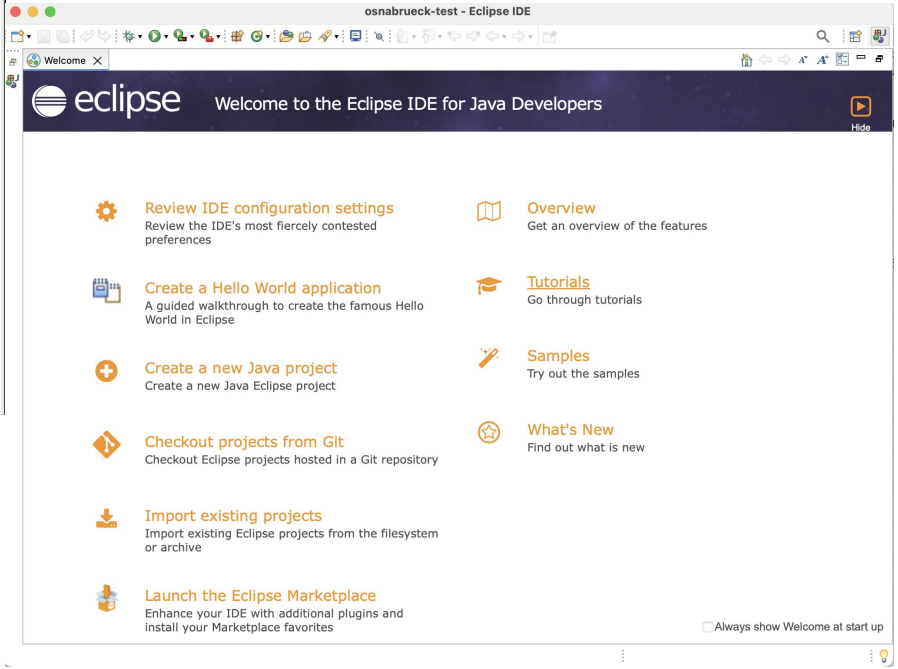
Setup Eclipse

de.jwausle.osgi.classpath

Setup Eclipse - Install Eclipse

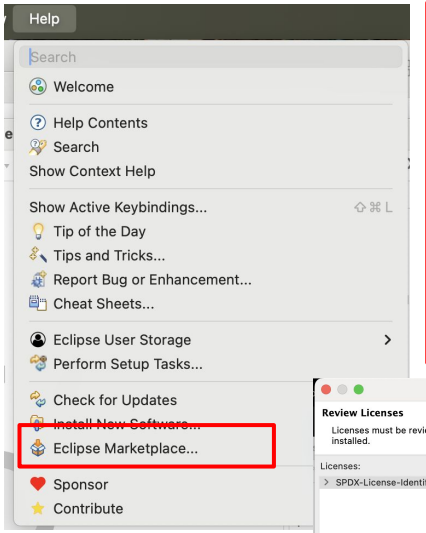


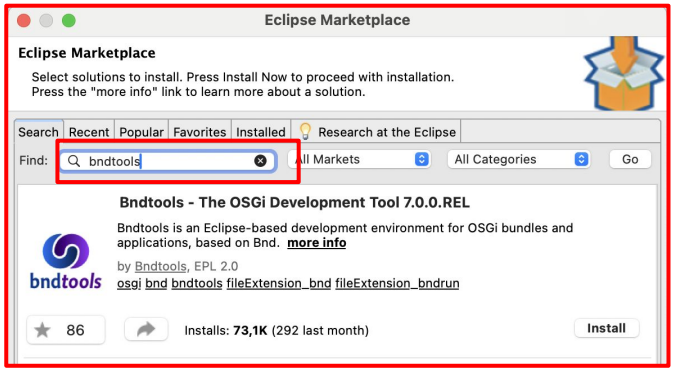


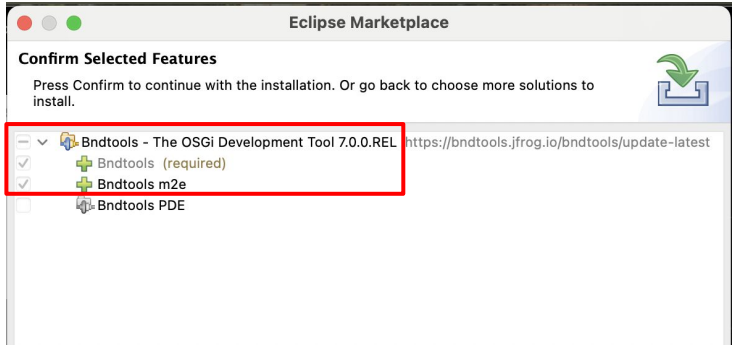


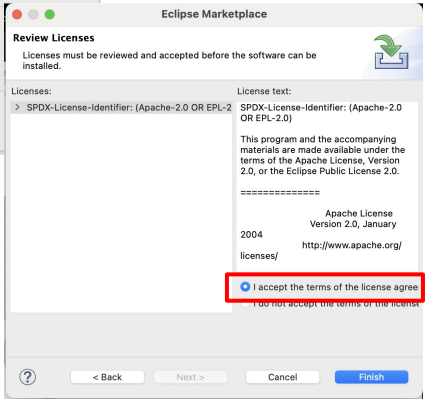
https://www.eclipse.org/downloads/

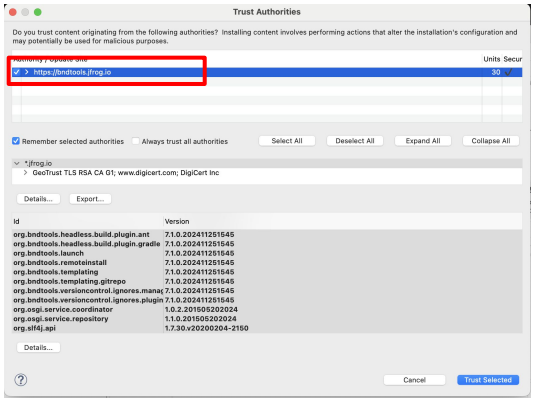
Setup Eclipse - Install Bnd Tools Plugin

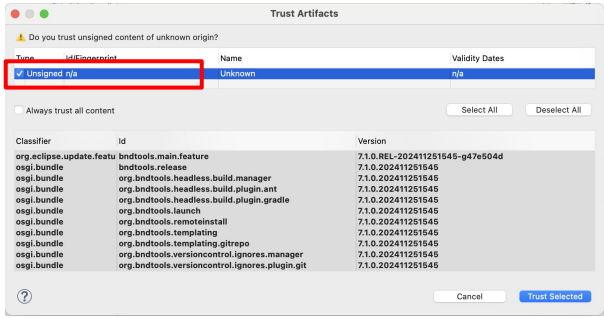








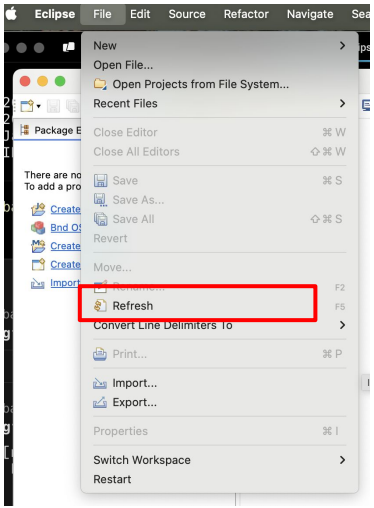


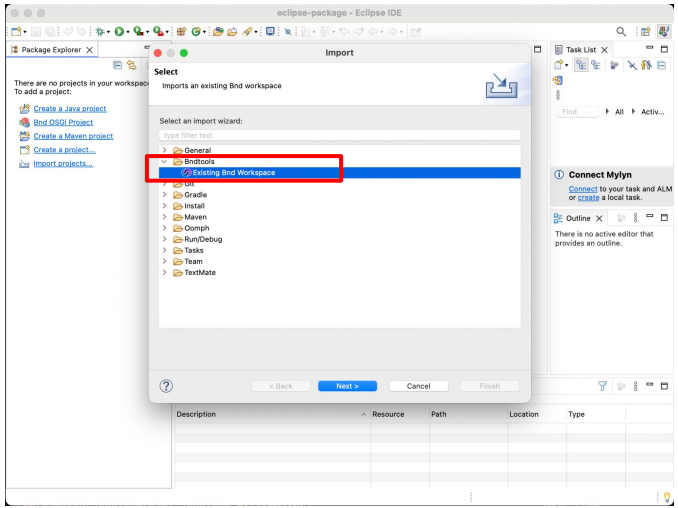


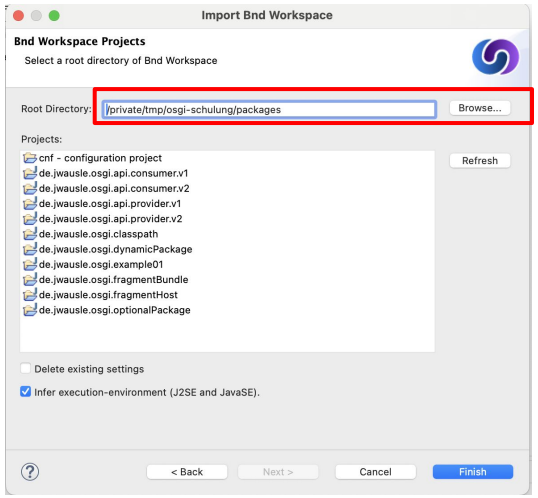
https://bndtools.org/installation.html

Setup Eclipse - Import Package Workspace

```
git clone git@github.com:jwausle/osgi-schulung.git
```







https://bndtools.org/installation.html

Example - Classpath

de.jwausle.osgi.classpath

Demo - Tools

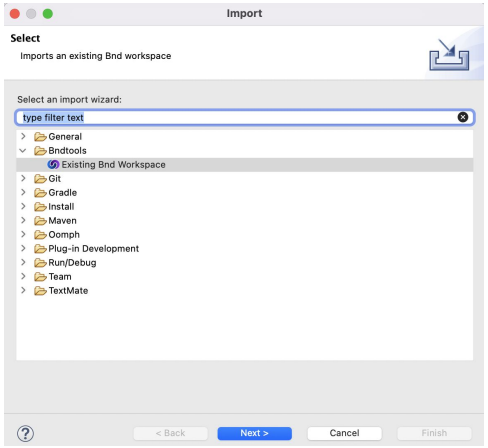
Eclipse IDE 2024-09 - https://www.eclipse.org/downloads/ (installed + started)

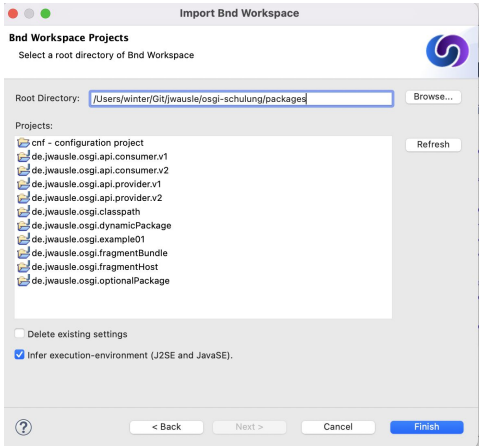
Eclipse Bndtools Plugin - https://.../bndtools-osgi-development-tool (installed)

Git - https://git-scm.com/downloads (installed)

Osgi-Schulung - https://github.com/jwausle/osgi-schulung (Cloned)

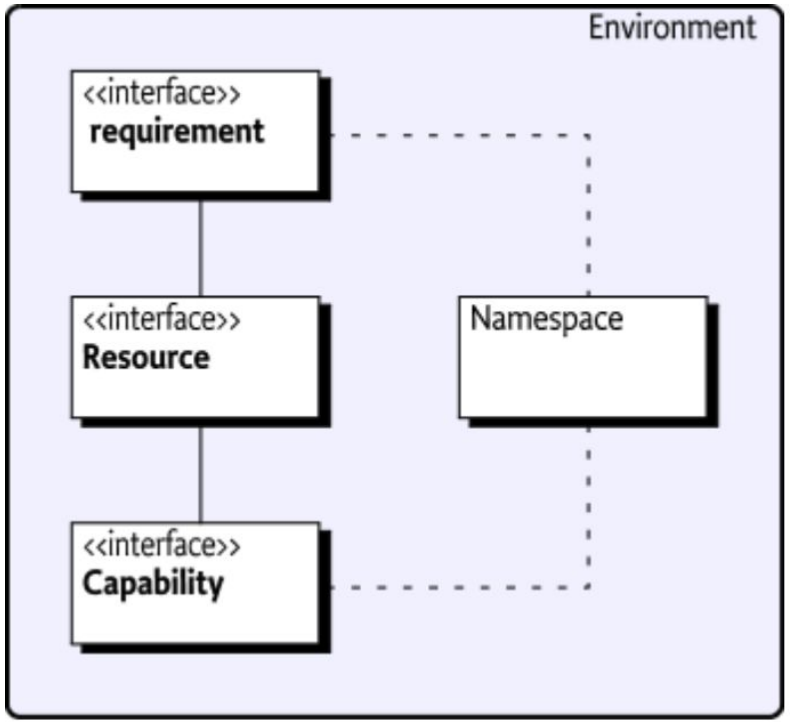
Import Existing Bnd Workplace





Dependencies - Requirement/Capability model

- *Resource* - An abstraction for an artifact that needs to become installed in some way to provide its intended function. A Bundle is modeled by a Resource but for example a display or secure USB key store can also be Resources.
- *Namespace* - Defines what it means for the Environment when a requirement and capability match in a given Namespace.
- *Capability* - Describing a feature or function of the Resource when installed in the Environment. A capability has attributes and directives.
- *Requirement* - An assertion on the availability of a capability in the Environment. A requirement has attributes and directives. The filter directive contains the filter to assert the attributes of the capability in the same Namespace.



https://docs.osgi.org/specification/osgi.core/7.0.0/framework.module.html#framework.module.dependencies

Framework - Requirement/Capability

- **osgi.ee** Namespace (Bundle-RequiredExecutionEnvironment)
- **osgi.wiring.package** Namespace (Import-Package, Export-Package)
- **osgi.wiring.bundle** Namespace (Required-Bundle)
- **osgi.wiring.host** Namespace (Fragment-Host)
- **osgi.identity** Namespace (Bundle-SymbolicName, Bundle-Version)
- **osgi.native** Namespace

https://docs.osgi.org/specification/osgi.core/7.0.0/framework.namespaces.html

Example - Optional/Dynamic

de.jwausle.osgi.example01,
de.jwausle.osgi.optional,
de.jwausle.osgi.dynamic

OSGi - Dynamic vs Optional Package

	Optional Package	**Dynamic Package**
Consumer - Import-Package	...package;resolution:=optional	...package;resolution:=dynamic
Consumer - Package Wiring	On Bundle Resolve After Refresh	On Bundle Resolve Dynamic without Refresh
Provider - Package Wiring	On Bundle Resolve	On Bundle Resole
Use cases	LogAppender	Plugin Architecture (Core)

Example - FragmentHost

de.jwausle.osgi.fragmentHost,
de.jwausle.osgi.fragmentBundle

Example - Packages and Versions

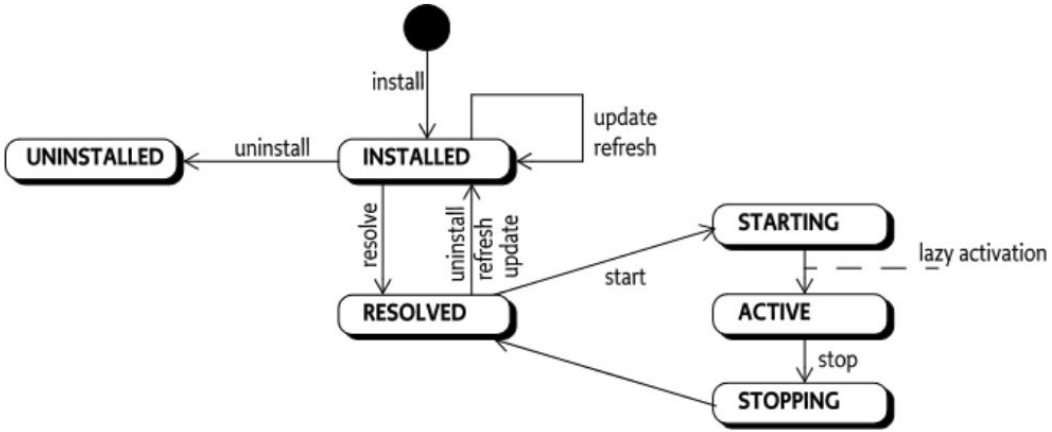
consumer.v2
consumer.v2 with provider.v1|v2

OSGi - StartLevel

https://bnd.bndtools.org/chapters/300-launching.html

Bundle - State

- INSTALLED - The bundle has been successfully installed.
- RESOLVED - All Java classes that the bundle needs are available. This state indicates that the bundle is either ready to be started or has stopped.
- STARTING - The bundle is being started, the BundleActivator.start method will be called
- ACTIVE - The bundle has been successfully activated and is running
- STOPPING - The bundle is being stopped
- UNINSTALLED - The bundle has been uninstalled. It cannot move into another state.

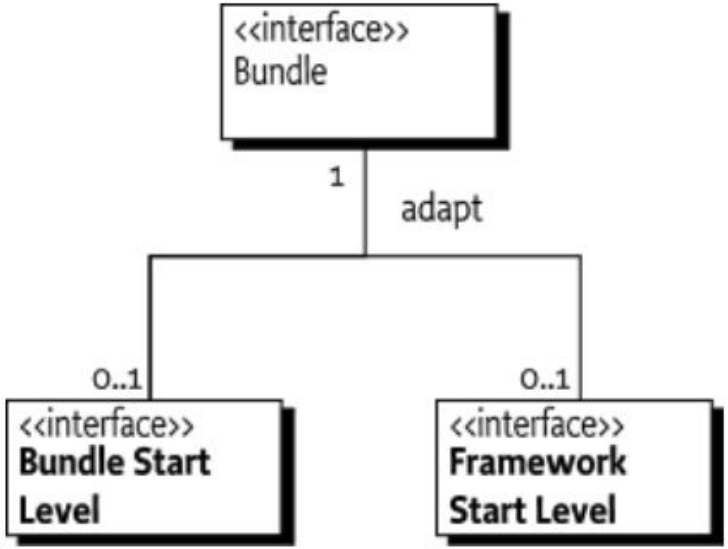


https://docs.osgi.org/specification/osgi.core/7.0.0/framework.lifecycle.html#d0e9143

Bundle - StartLevel

Control the bundle start sequence

- *BundleStartLevel* - Used to get and set the start level on a specific bundle
- *FrameworkStartLevel* - Used to get and control the framework start level.



- The Framework has an active start level that is used to decide which bundles can be started
- All bundles must be assigned a bundle start level
- When a bundle is installed, it is initially assigned the bundle start level

Example - StartLevel

consumer.v1

https://bnd.bndtools.org/chapters/300-launching.html

Bundle - API

METADATA	Bundle.getHeaders Bundle.getLocation
RESOURCE	Bundle.getResource Bundle.getResources Bundle.getEntry Bundle.getEntryPaths Bundle.findEntries Bundle resource/entry URL creation
CLASS	Bundle.loadClass
LIFECYCLE	BundleContext.installBundle Bundle.update Bundle.uninstall
EXECUTE	Bundle.start Bundle.stop BundleStartLevel.setBundleStartLevel
LISTENER	BundleContext.addBundleListener for SynchronousBundleListener BundleContext.removeBundleListener for SynchronousBundleListener
EXTENSIONLIFECYLE	BundleContext.installBundle for extension bundles Bundle.update for extension bundles Bundle.uninstall for extension bundles
RESOLVE	FrameworkWiring.refreshBundles FrameworkWiring.resolveBundles
STARTLEVEL	FrameworkStartLevel.setStartLevel FrameworkStartLevel.setInitialBundleStartLevel
CONTEXT	Bundle.getBundleContext
WEAVE	WovenClass.setBytes WovenClass.getDynamicImports

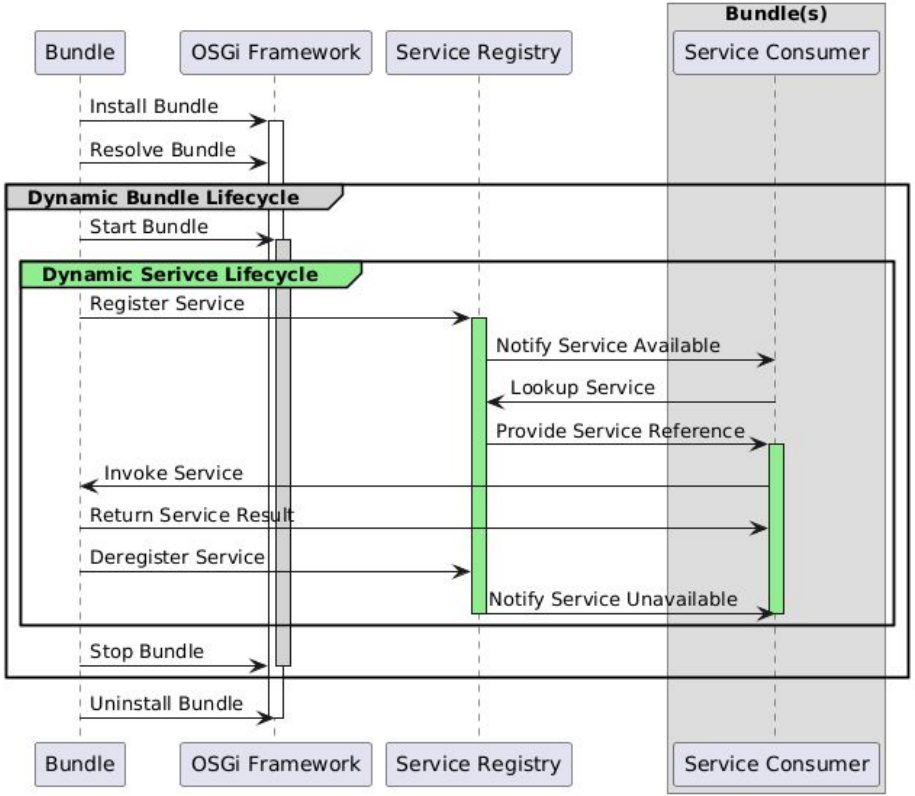
https://docs.osgi.org/specification/osgi.core/7.0.0/framework.lifecycle.html#framework.lifecycle.adminpermission

OSGi Services

What is a Service

OSGi	Dynamic object instance which can be used by other services for communication. The lifecycle is bound to a started bundle.
J2EE/JEE - Bean	Dynamic object instance which can be used by other parties for communication. The lifecycle is bound to application/jvm process.
Spring - Bean	Constant shared object instance which can be used by other services for communication. The lifecycle is bound to the jvm process.
OS - Windows/Linux/ Daemon	Background process to provide constant functionality on top of the OS abstraction. The lifecycle is bound to a running OS.
Other	● Network services (Internet Stack- IP/DNS/DHCP/TCP/HTTP/…) ● Web/Rest services (Architektur/Communication pattern) ● Microservices (Architektur pattern) ● Cloud Services (Software as a Service)

Service (OSGi) Lifecycle

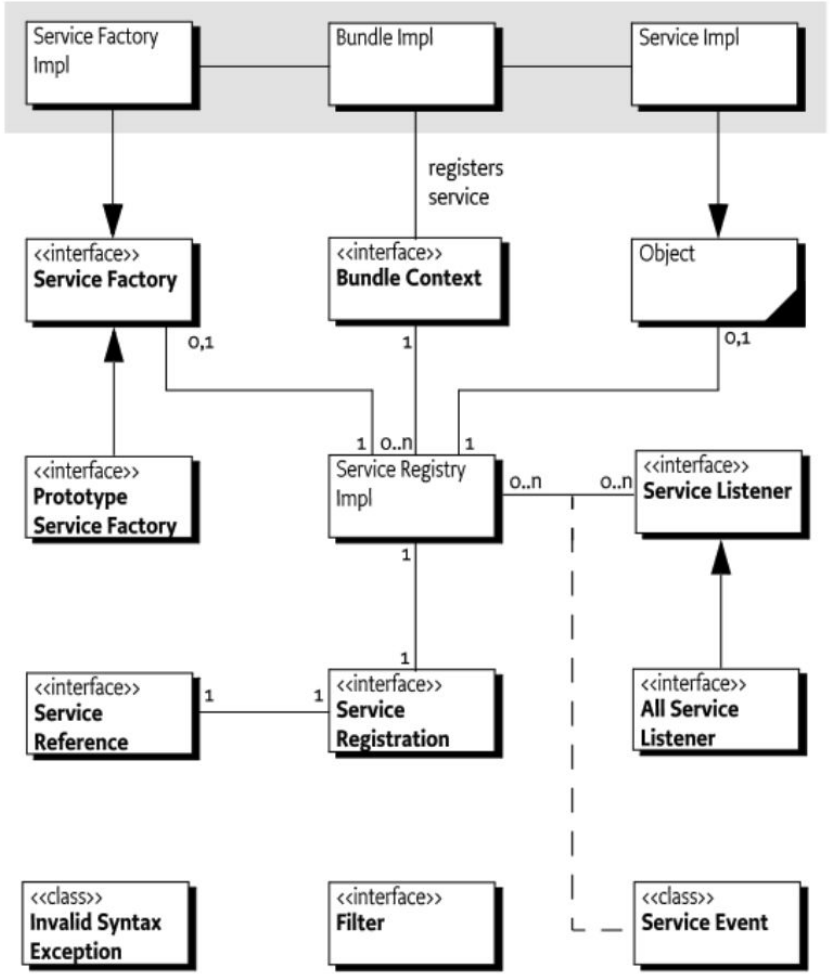


- Service Lifecycle is bound to the **active** Bundle Lifecycle (Relates to StartLevel)
- Service Instances will be created on Bundle Start
- Service Instances *should* be destroyed on Bundle Stop
- Service Dependencies will be notified about Service (de)registration

https://docs.osgi.org/specification/osgi.core/7.0.0/framework.service.html#d0e11540

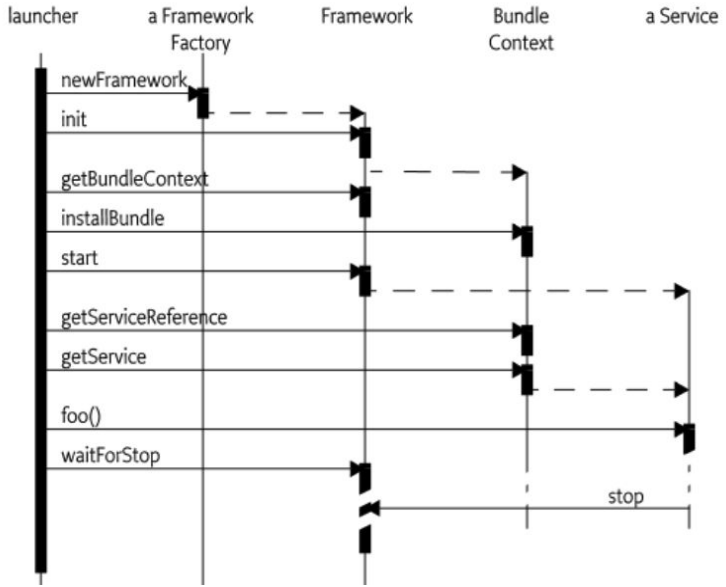
Services (OSGi)

- *Service* - An object registered with the service registry under one or more interfaces together with properties. The service can be discovered and used by bundles.
- *Service Registry* - Holds the service registrations.
- *Service Reference* - A reference to a service. Provides access to the service's properties but not the actual service object. The service object must be acquired through a bundle's Bundle Context.
- *Service Registration* - The receipt provided when a service is registered. The service registration allows the update of the service properties and the unregistration of the service.

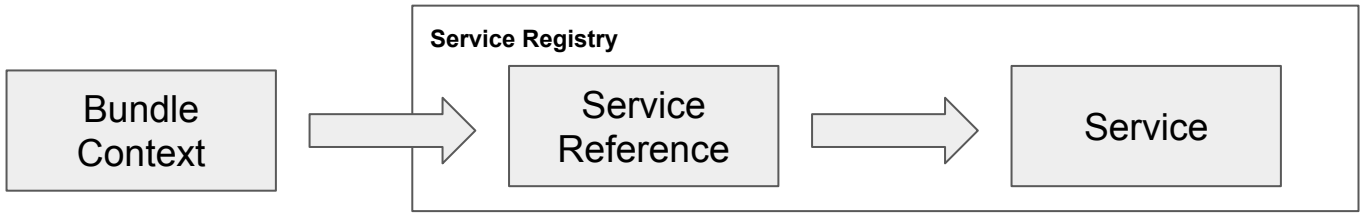


https://docs.osgi.org/specification/osgi.core/7.0.0/framework.service.html#d0e11540

Service Registration



1. Bundle register Service
2. Bundle register Service over the bundleContext
3. Service reference is the key for Service instance
4. BundleContext get Service instance by Service reference
5. When Service instance exist



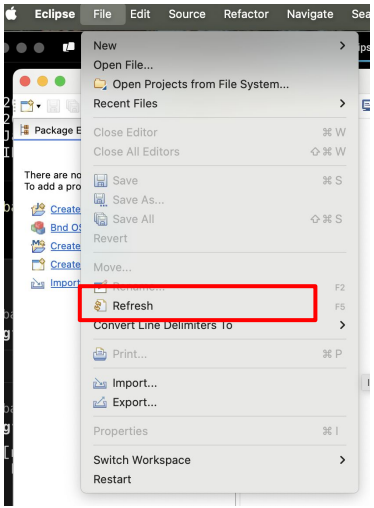
Service(Registration) Properties

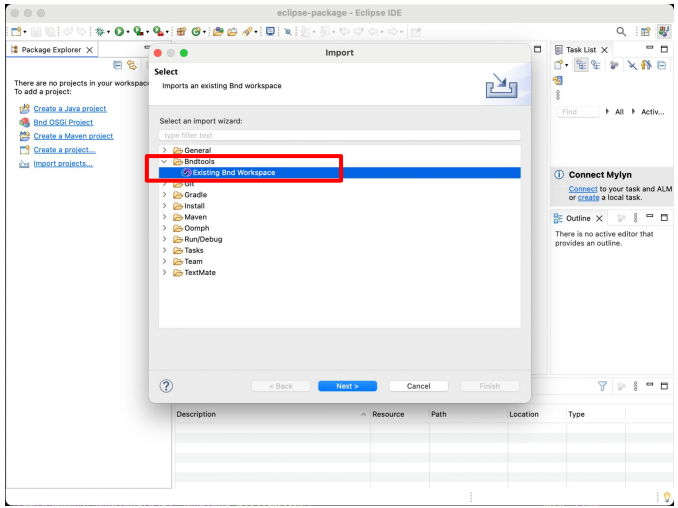
Property Key	Type	Constants	Property Description
`objectClass`†	`String[]`	OBJECTCLASS	The `objectClass` property contains the set of interface names under which a service object is registered with the Framework. The Framework must set this property automatically. The Framework must guarantee that when a service object is retrieved with getService(ServiceReference), it can be cast to any of the interface names.
`service.bundleid`†	`Long`	SERVICE_BUNDLEID	The `service.bundleid` property identifies the bundle registering the service. The Framework must set this property automatically with the value of the bundle id of the registering bundle.
`service.description`	`String`	SERVICE_DESCRIPTION	The `service.description` property is intended to be used as documentation and is optional. Frameworks and bundles can use this property to provide a short description of a registered service object. The purpose is mainly for debugging because there is no support for localization.
`service.id`†	`Long`	SERVICE_ID	Every registered service object is assigned a unique, non-negative `service.id` by the Framework. This number is added to the service's properties. The Framework assigns a unique, non-negative value to every registered service object that is larger than values provided to all previously registered service objects.
`service.pid`	`String+`	SERVICE_PID	The `service.pid` property optionally identifies a persistent, unique identifier for the service object. See Persistent Identifier (PID).
`service.scope`†	`String`	SERVICE_SCOPE	The `service.scope` property identifies the service's scope. The Framework must set this property automatically. If the registered service object implements PrototypeServiceFactory, then the value will be prototype. Otherwise, if the registered service object implements ServiceFactory, then the value will be bundle. Otherwise, the value will be singleton. See Service Scope.
`service.ranking`	`Integer`	SERVICE_RANKING	See Service Ranking Order.
`service.vendor`	`String`	SERVICE_VENDOR	This optional property can be used by the bundle registering the service object to indicate the vendor.

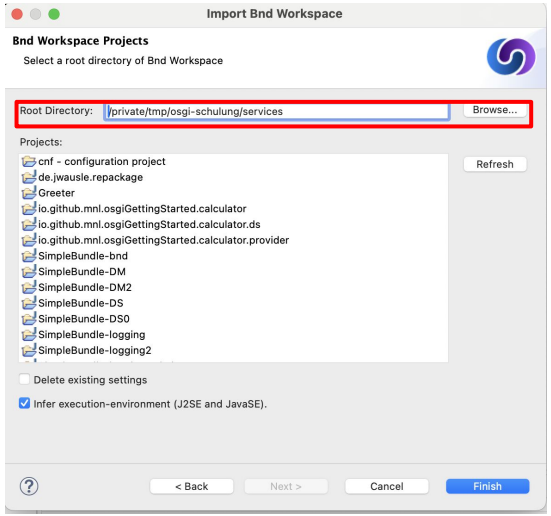
https://docs.osgi.org/specification/osgi.core/7.0.0/framework.service.html#framework.service.serviceproperties

Setup Eclipse - Import Service Workspace

```
git clone git@github.com:jwausle/osgi-schulung.git
```







https://bndtools.org/installation.html

Example - ServiceReference

SimpleBundle-logging

Service - Reference

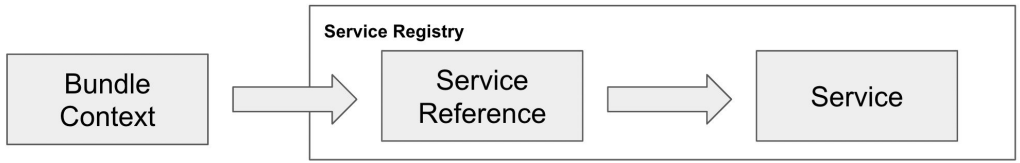
```
public class HelloWorldActivator implements BundleActivator {

   private ServiceRegistration<HelloWorld> registerService;
   private HelloWorld helloWorld;

   @Override
   public void start(BundleContext context) throws Exception {
         ServiceReference<LogService> reference = context.getServiceReference(LogService.class);
          LogService result = context.getService(reference);
          this.helloWorld = new HelloWorld(result);
          this.registerService = context.registerService(HelloWorld.class, this.helloWorld , null);
   }


   @Override
   public void stop(BundleContext context) throws Exception {
      this.helloWorld = null;
      context.ungetService(registerService.getReference());
   }
}
```

On Bundle Start

1. Context get references
2. Init Service
3. Register Service

On Bundle Stop

4. Unregister service



https://docs.osgi.org/specification/osgi.core/7.0.0/util.tracker.html

Example - Service Tracker

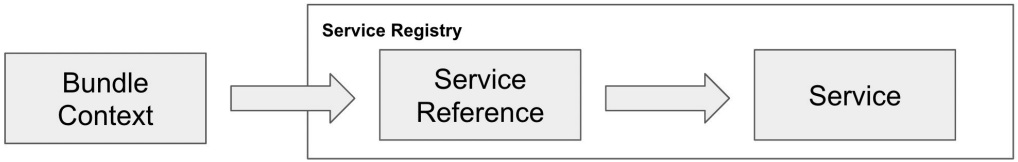
SimpleBundle-logging2

Service - Tracker

```
public class HelloWorldActivator implements BundleActivator {
   private ServiceTracker<LogService, LogService> logServiceTracker;
   private HelloWorld helloWorld;
   @Override
   public void start(BundleContext context) throws Exception {
      if (logServiceTracker == null) {
         logServiceTracker = new ServiceTracker<LogService, LogService>(context, LogService.class, null) {
            @Override
            public LogService addingService(ServiceReference<LogService> reference) {
               LogService result = context.getService(reference); // super.addingService(reference)
               helloWorld = new HelloWorld(result);
               System.out.println("Hello World started.");
               return result;
            }
            @Override
            public void removedService(ServiceReference<LogService> reference, LogService service) {
               super.removedService(reference, service);
               helloWorld = null;
               System.out.println("Hello World stopped.");
            }
         };
      }
      logServiceTracker.open();
   }
```

1. ServiceTracker init
2. ServiceTracker open

3. ServiceTracker get informed when Service instance created
4. ServiceTracker get informed when Service instance destroyed

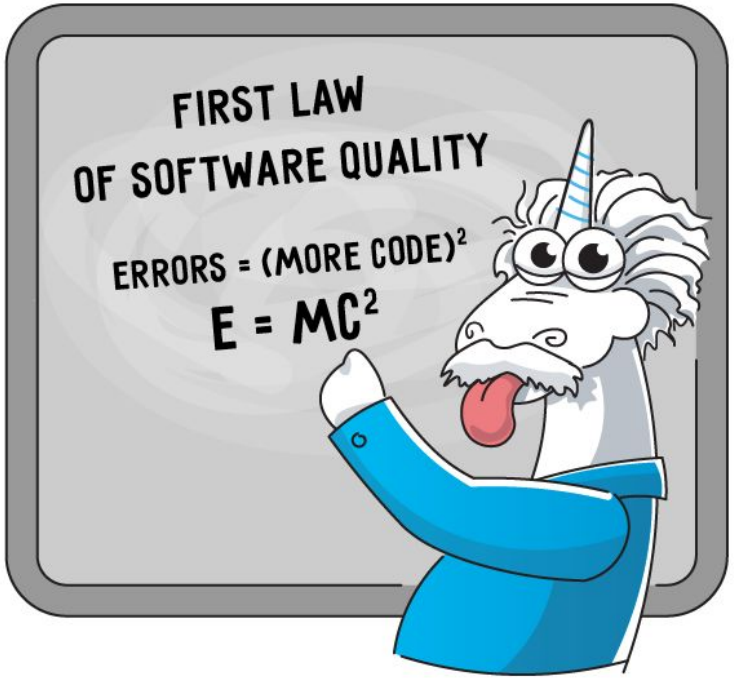


https://docs.osgi.org/specification/osgi.core/7.0.0/util.tracker.html

Service - Tracker (Problem)

1. Service Tracker must be implemented for each Service Dependency

2. Service Tracker implementations are Boilerplate-Code



https://docs.osgi.org/specification/osgi.core/7.0.0/util.tracker.html

Apache Felix Dependency Manager

- Library to simplify to service declaration and registration
- Plus Declarative Service Builder
- Alternative to Declarative Service (DS)

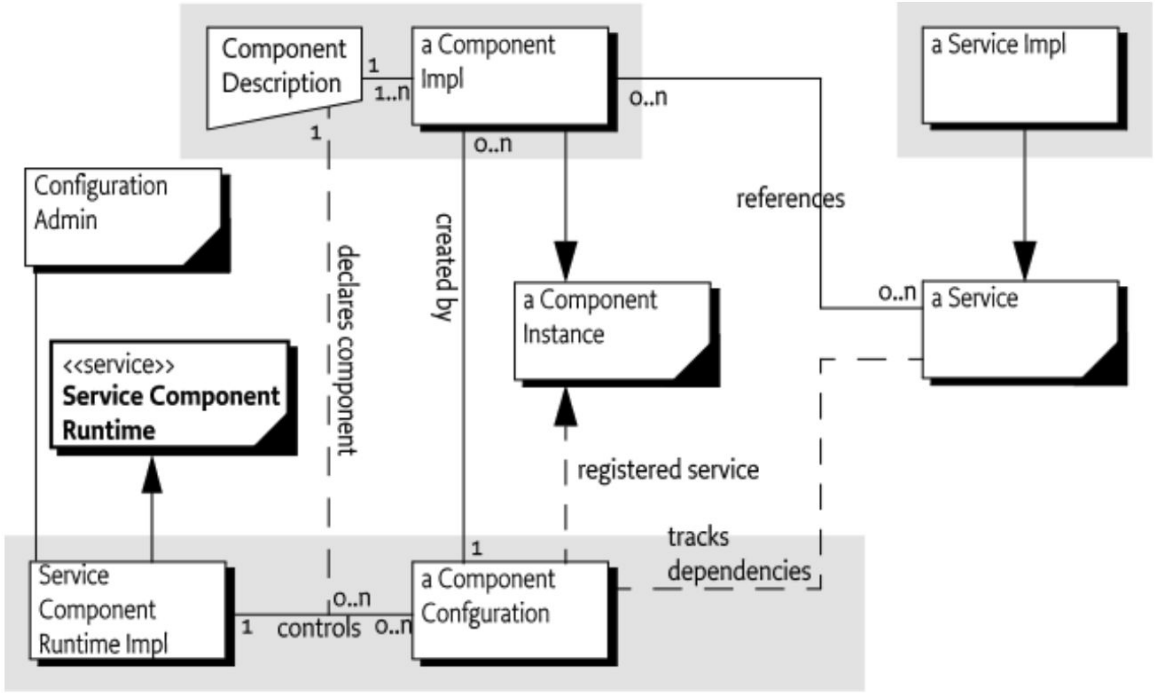
https://felix.apache.org/documentation/subprojects/apache-felix-dependency-manager.html

Example - DM

SimpleBundle-DM

Declarative Services

- Declarative Service Registration Config
- To publishing, finding and binding OSGi Service
- Init and process all ServiceTracker
- Reflect Service instance when all depend Service References fulfilled



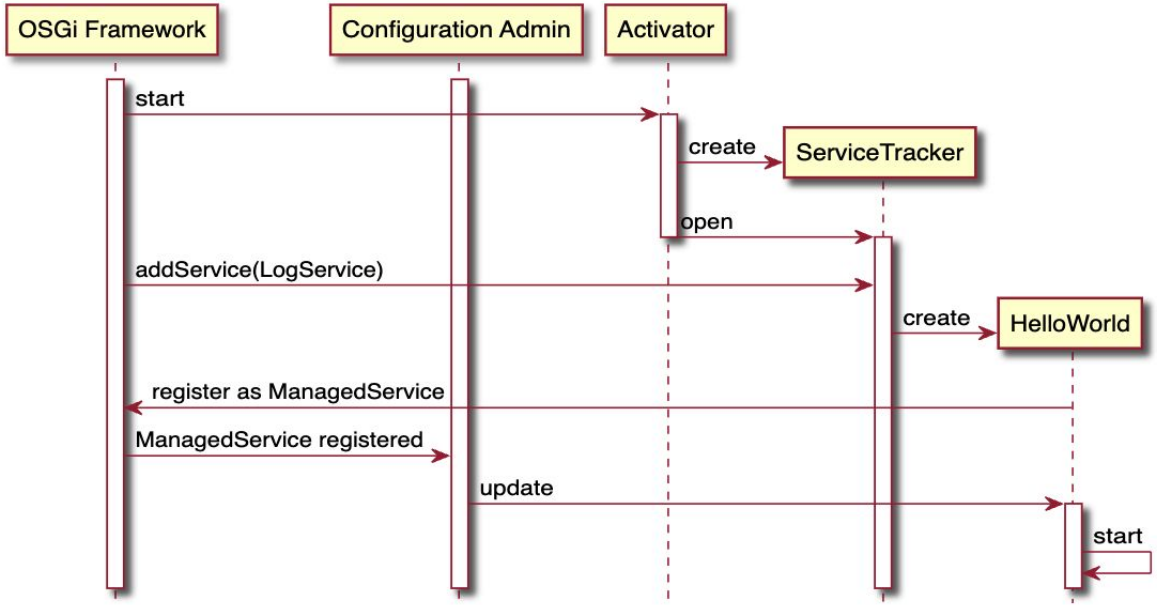
https://docs.osgi.org/specification/osgi.cmpn/7.0.0/service.component.html

Example - DS

SimpleBundle-DS0 (XML)
SimpleBundle-DS (@)

OSGi Configuration Admin

- Whiteboard Pattern

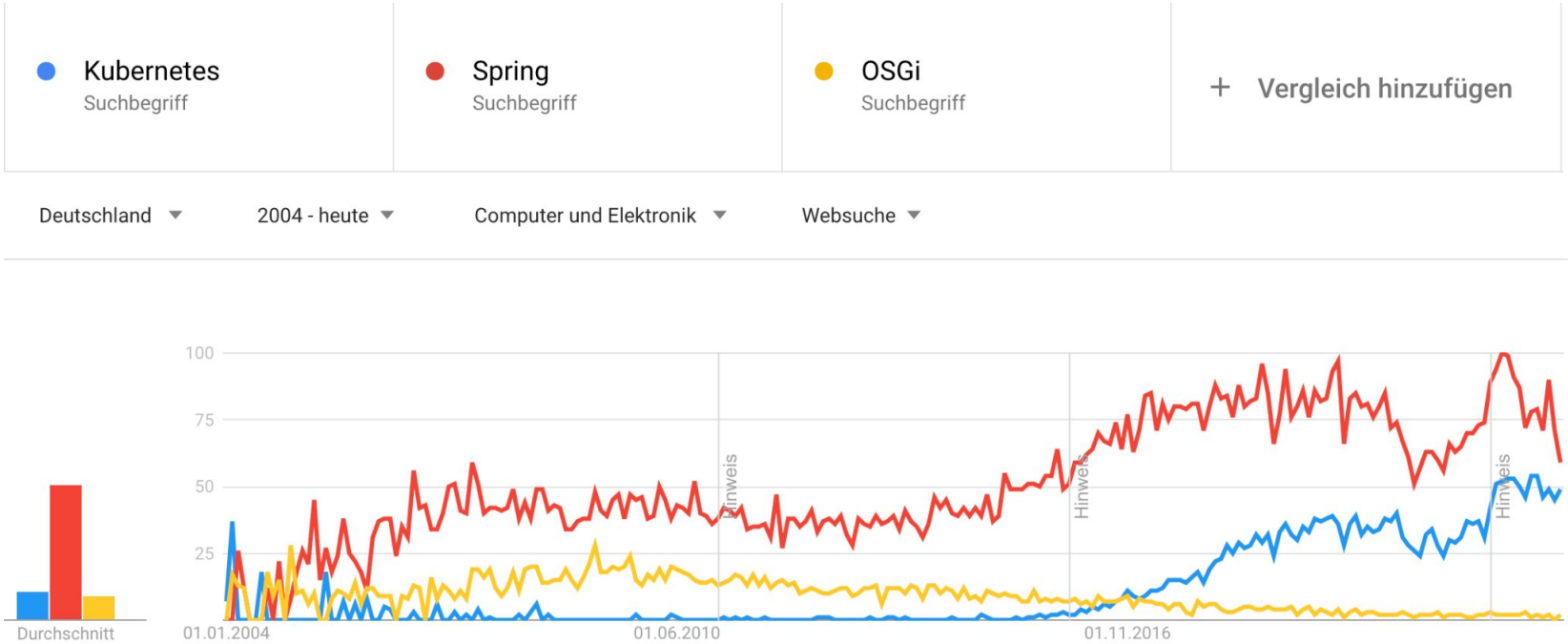


https://docs.osgi.org/specification/osgi.cmpn/7.0.0/service.cm.html#service.cm

Example ConfigAdmin

logging-admin (LogLevel)
logging-admin2 (HelloWorld)

OSGi vs Spring vs Kubernetes



https://trends.google.de/trends/explore?cat=5&date=all&geo=DE&q=OSGi,Kubernetes,Spring

OSGi Links

https://docs.osgi.org/specification/

https://bnd.bndtools.org/

https://bndtools.org/

winter@jwausle.de