

Gerd Wütherich ♦ Martin Lippert

OSGi Service Platform by example

Die OSGi Service Platform – Das Buch



- » Detaillierte Einführung in OSGi-Technologie
- » April 2008, dpunkt.verlag
- » ISBN 978-3-89864-457-0
- » Website: www.osgibook.org

Agenda

- » OSGi-Technologie im Überblick
- » OSGi-Technologie im Detail
 - » Bundles
 - » Package-Abhängigkeiten zwischen Bundles
 - » Bundle-Lebenszyklus
 - » OSGi Services
- » Zusammenfassung

Wir erinnern uns...

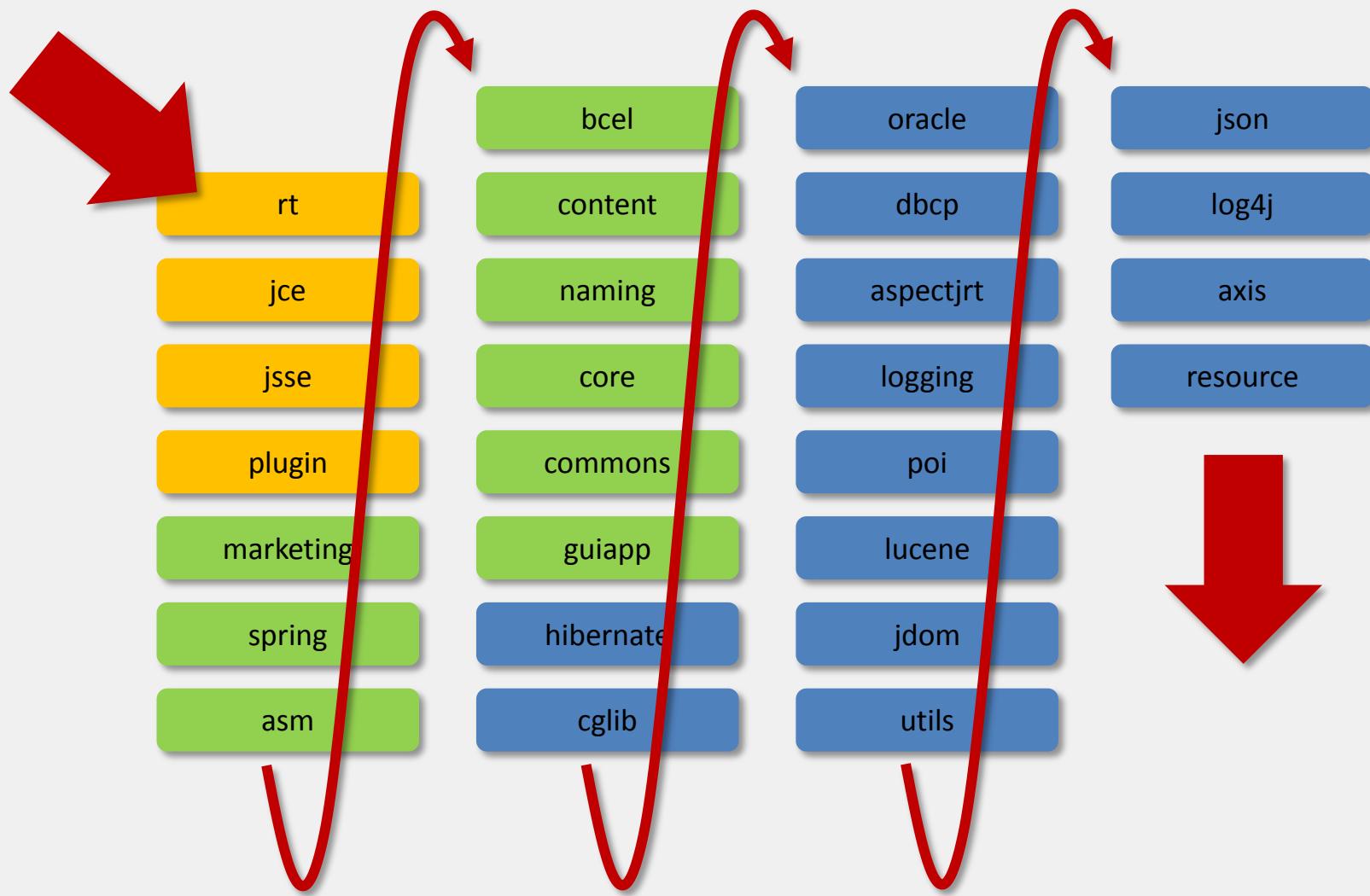


- » Kein OO
- » Kein Information Hiding
- » Nur globale Variablen
- » ...

Und heute...



JAR-Hell



Was fehlt uns?

- » **Gröbere Einheiten**

- » Mehr als Klassen oder Packages

- » **Klare Abhängigkeiten**

- » zwischen den Einheiten/Modulen

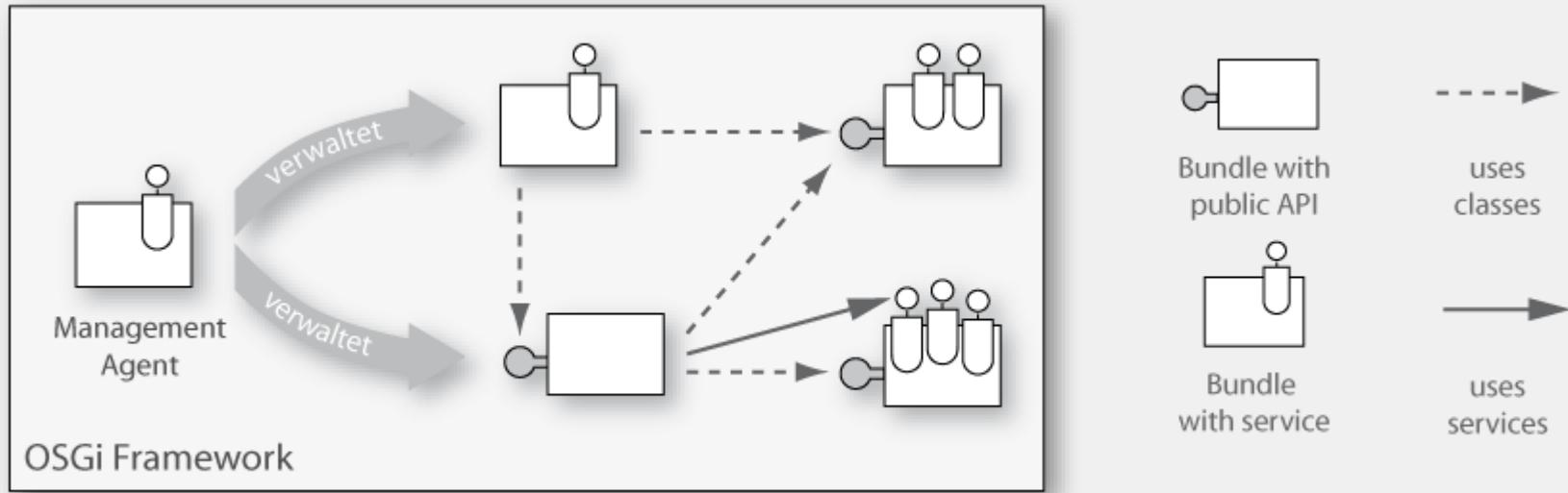
- » **Klare Sichtbarkeiten**

- » Zwischen den Einheiten/Modulen

OSG – was?

- » Die OSGi Service Platform...
 - » ... ist ein *dynamisches Modulsystem für Java*.
 - » ... ermöglicht die dynamische Integration und das Management von Softwarekomponenten (*Bundles*) und Diensten (*Services*).
- » Bundles und Services können zur Laufzeit in der Plattform *installiert, gestartet, gestoppt* und *deinstalliert* werden.
- » Besteht aus:
 - » OSGi Framework (Container für Bundles und Services)
 - » OSGi Standard Services (verschiedene, horizontale Services)

Das OSGi Framework



- » Basiskomponente der OSGi Service Platform
- » Erlaubt die Installation und Verwaltung von Bundles und Services
- » Verwaltet Anhängigkeiten zwischen Bundles
- » Kann über Management Agents „von außen“ administriert werden

Welche Vorteile bietet die OSGi Service Platform?

- » Modularisierung und Versionierung
- » Abhängigkeitsmanagement (zur Laufzeit)
- » Hot deployment
- » (Fern-) Management des laufenden Frameworks über sog. Management Agents
- » Serviceorientiertes Programmiermodell

SOA-Konfusion:

- » Web Services publizieren und nutzen Dienste über das Netz
- » OSGi Services publizieren und nutzen Dienste innerhalb einer virtuellen Maschine

Wo wird die OSGi Service Platform eingesetzt?

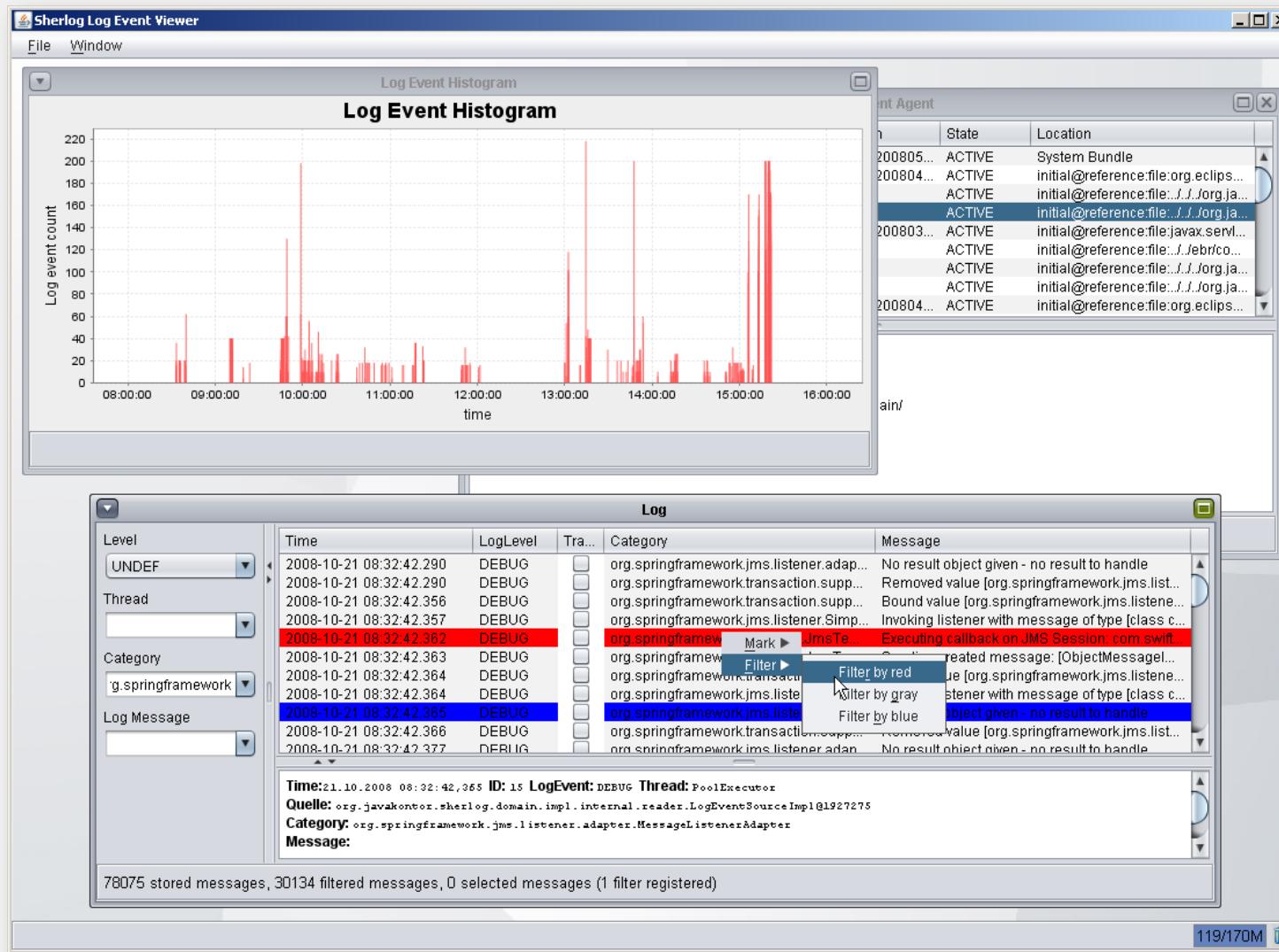
Einige Beispiele:

- » Eclipse Platform
 - » Eclipse SDKs (IDEs), RCP, eRCP, ...
- » IBM
 - » Websphere App Server (basiert auf OSGi)
 - » Lotus (basiert auf Eclipse-RCP, damit auch OSGi)
 - » Jazz (basiert auf Server-Side-Eclipse)
- » BEA/Oracle
- » SpringSource Application Platform / dm Server
- » Adobe
- » ...

Implementierungen der OSGi Service Platform

- » Open Source Implementierungen:
 - » Eclipse Equinox (<http://www.eclipse.org/equinox/>)
 - » Apache Felix (<http://cwiki.apache.org/FELIX/index.html>)
 - » Knopflerfish (<http://www.knopflerfish.org/>)
 - » ProSyst mBedded Server Equinox Edition
(http://www.prosyst.com/products/osgi_se_equi_ed.html)
- » Kommerzielle Implementierungen:
 - » ProSyst (<http://www.prosyst.com>)
 - » Knopflerfish Pro (<http://www.makewave.com>)
 - » ...

OSGi-Demo: Sherlog - Log-File-Analyser



Bundles



- » ... sind die Modularisierungseinheiten innerhalb des OSGi Frameworks
- » ... enthalten zusammengehörige Klassen und Ressourcen
- » ... können unabhängig im OSGi Framework deployed werden
- » ... sind JAR-Dateien
- » ... enthalten ein Bundle Manifest, das das Bundle beschreibt

Das Bundle Manifest

- » ... ist Teil des Bundles in der Datei META-INF/MANIFEST.MF
- » ... enthält Informationen, die das Bundle beschreiben, z.B.
 - » den eindeutigen Namen und die Version
 - » die öffentliche Schnittstelle/API
 - » Package-Abhängigkeiten

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-Name: Histogram Bundle
Bundle-SymbolicName: org.javakontor.sherlog.ui.histogram
Bundle-Version: 1.0.0
Bundle-Activator: org.javakontor.sherlog.ui.histogram.Activator
Import-Package: org.osgi.framework;version="1.4.0"
Bundle-ClassPath: .
```

Bundle Aktivierung

- » Jedes Bundle kann im Bundle-Manifest einen Bundle-Aktivator definieren

```
Bundle-Activator: org.javakontor. ... .Activator
```

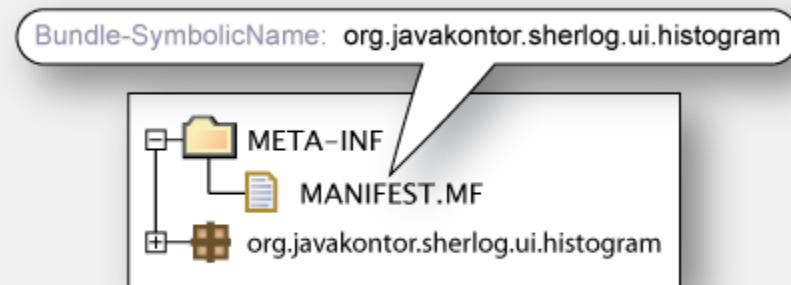
- » Der Bundle-Aktivator muss das Interface *BundleActivator* implementieren

```
package org.osgi.framework;

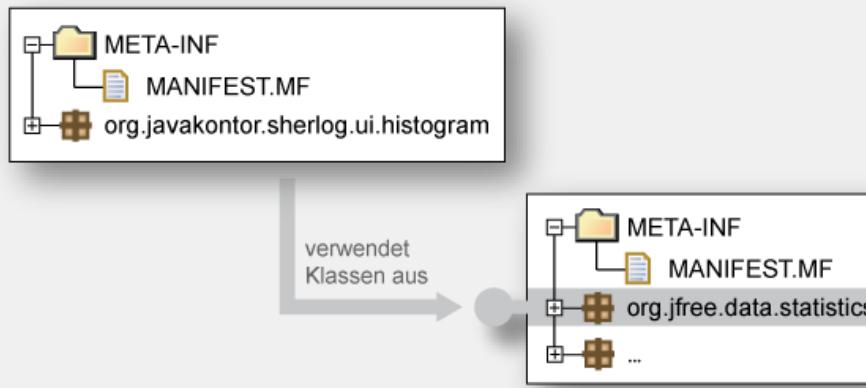
public interface BundleActivator {

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

Demo 1: Ein Histogram-Bundle

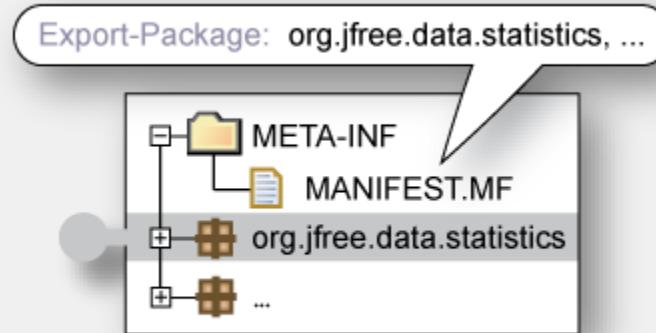


Package-Abhangigkeiten



- » Package-Abhangigkeiten mussen explizit angegeben werden:
 - » Packages mussen **exportiert** werden, um sichtbar fur andere Bundles zu sein.
 - » Packages mussen **importiert** werden, um im Bundle genutzt werden zu konnen.
- » Das OSGi Framework ist verantwortlich fur das Auflosen der importierten und exportierten Packages.

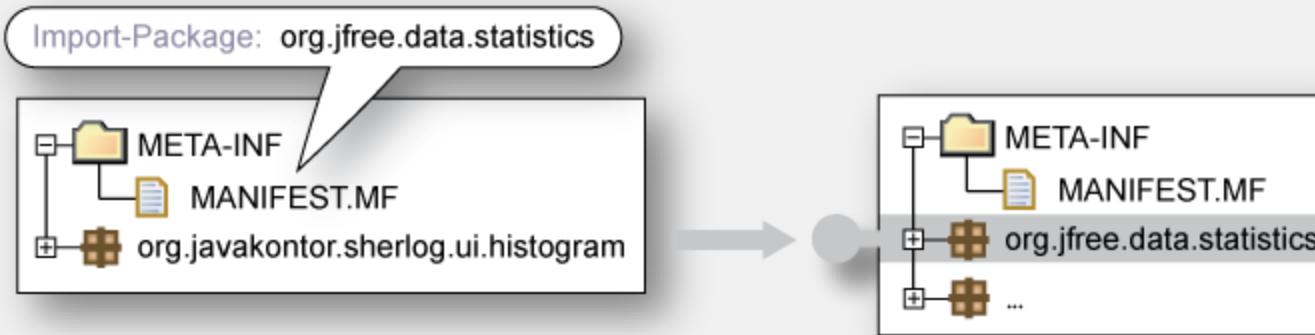
Packages exportieren



- » Nur die öffentliche API ist für andere Bundles sichtbar
- » Manifest-Header “Export-Package”: kommaseparierte Auflistung aller exportierten Packages

Export-Package: org.jfree.data.statistics, ...

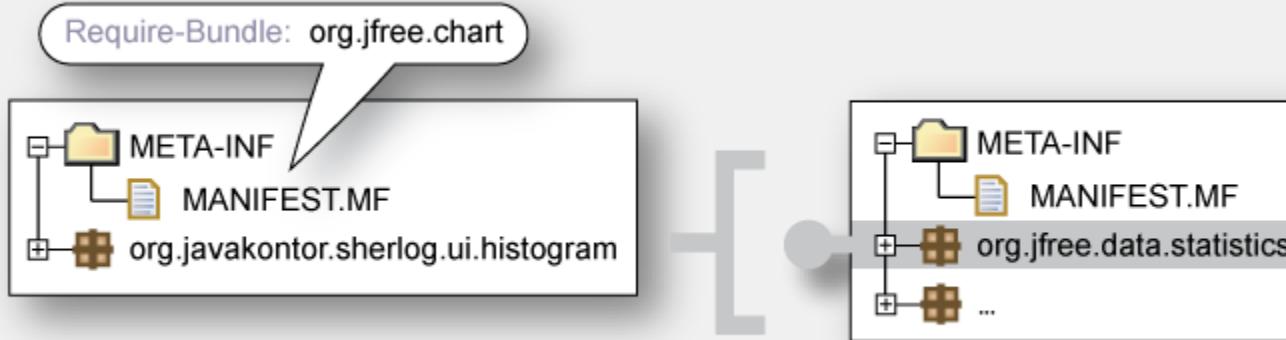
Importieren von Packages mit Import-Package



- » Der Manifest-Header “Import-Package” gibt Package-Abhängigkeiten an
- » Importierte Packages werden durch Komma getrennt aufgelistet

Import-Package: org.jfree.data.statistics

Packages importieren mit Require-Bundle

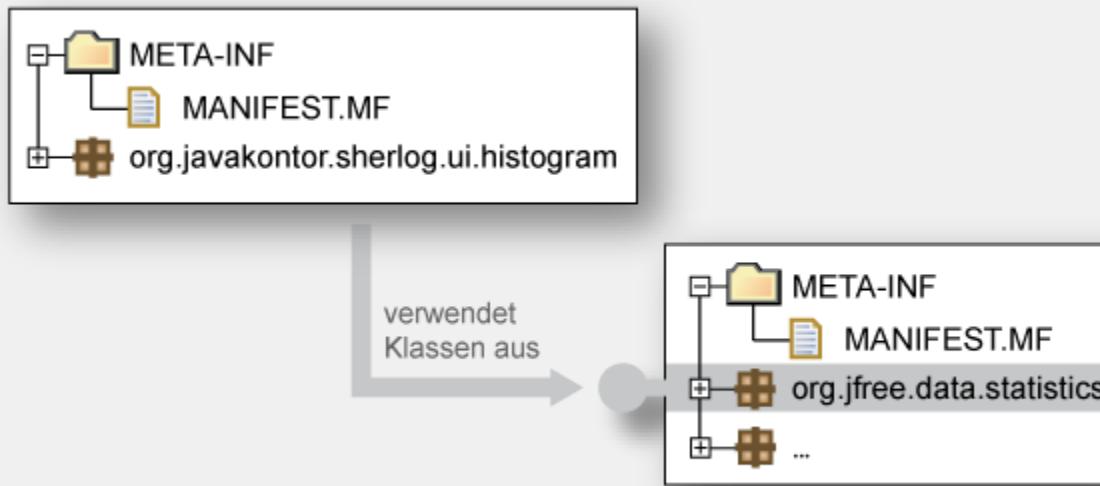


- » Referenziert ein bestimmtes Bundle
- » Bundles werden durch Komma getrennt aufgeführt

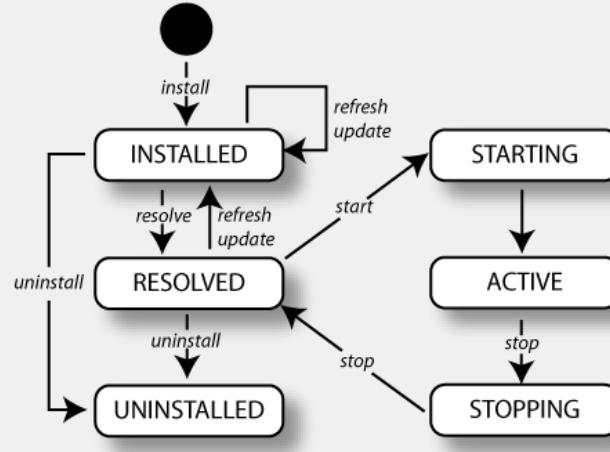
Require-Bundle: org.jfree.chart

- » Alle exportierten Packages der benötigten Bundles werden vom Bundle importiert

Demo 2: Package Dependencies

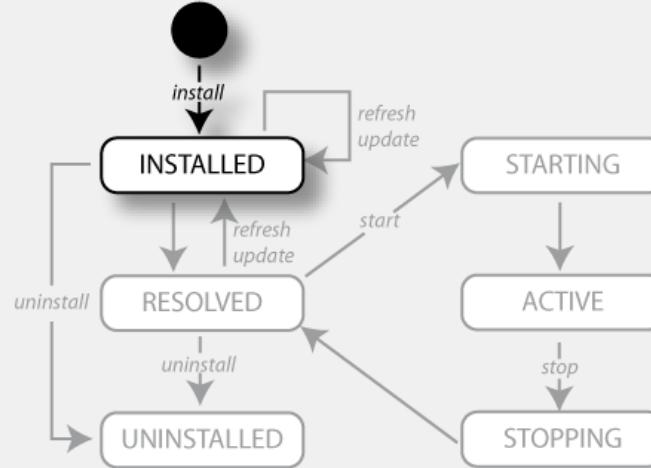


Der Bundle Lifecycle



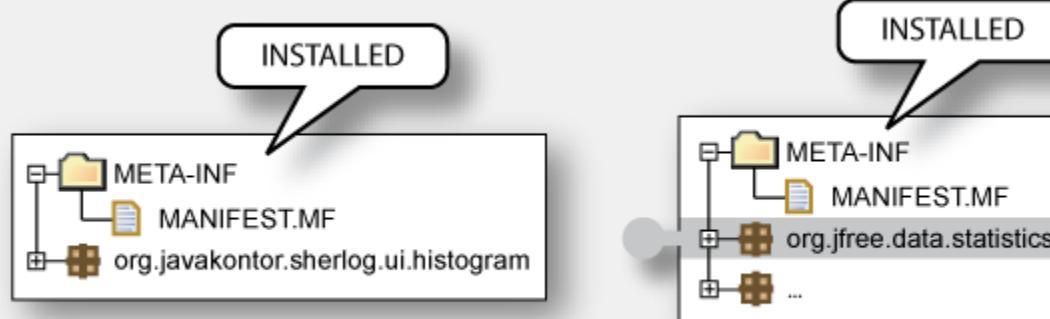
- » Bundles haben einen definierten Lebenszyklus
- » Zustandsänderungen können programmatisch oder durch einen Management Agent getriggert werden

Installieren von Bundles I



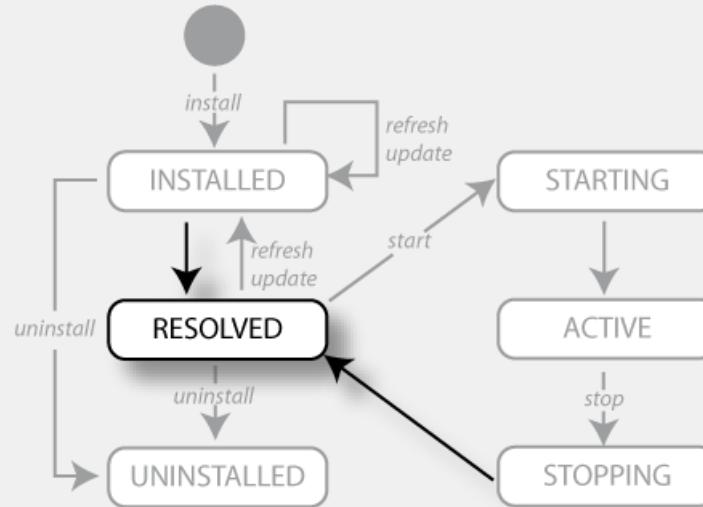
- » Das Bundle wird (persistent) im OSGi Framework verfügbar gemacht
- » Der Bundle-Zustand wird auf **INSTALLED** gesetzt

Installieren von Bundles



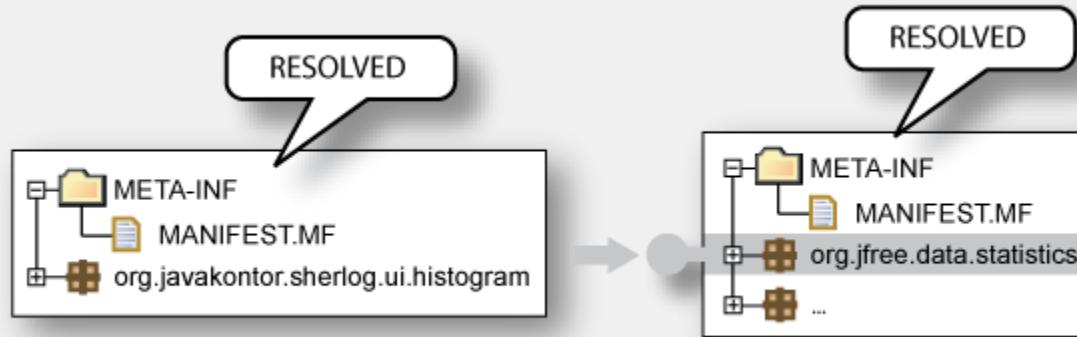
- » Ein Bundle im Zustand INSTALLED kann (noch) nicht genutzt werden:
 - » Das Bundle kann nicht gestartet werden
 - » Packages werden nicht exportiert

Resolving



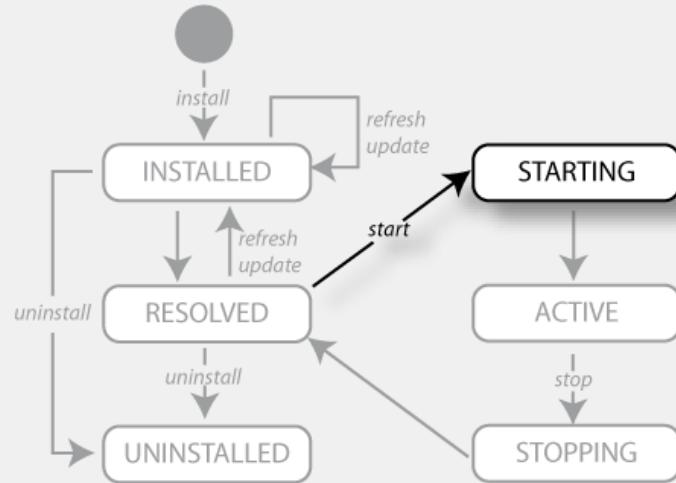
- » Importierten Packages werden exportierte Packages zugeordnet
- » Ist dies erfolgreich, wird der Zustand wird auf RESOLVED gesetzt

Resolving



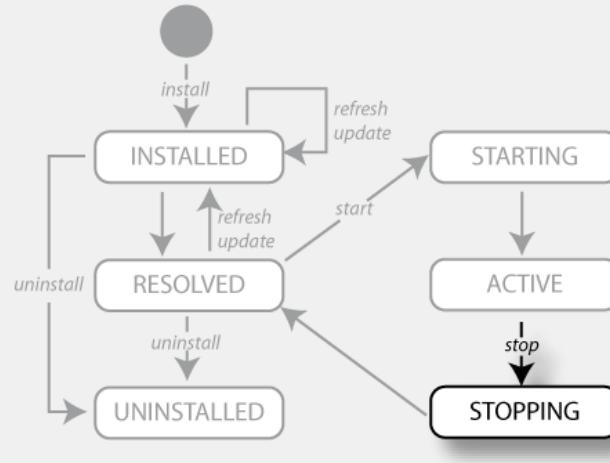
- » Ein Bundle im Zustand RESOLVED kann gestartet werden
- » Exportierte Packages können von anderen Bundes importiert werden

Starten von Bundles



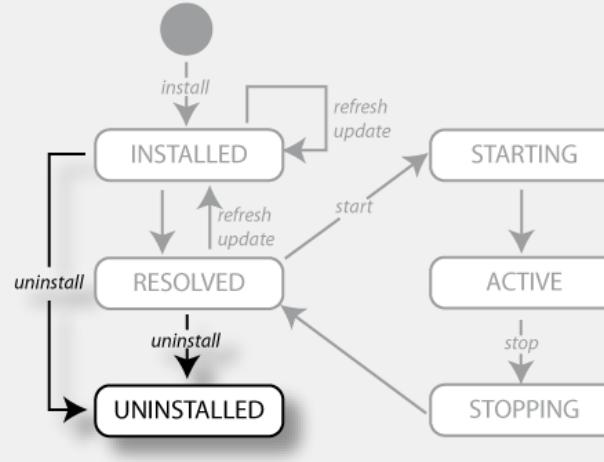
- » Das Bundle wird in den Zustand **STARTING** gesetzt
- » Der Bundle-Aktivator wird instantiiert und die `start()`-Methode aufgerufen

Stoppen von Bundles



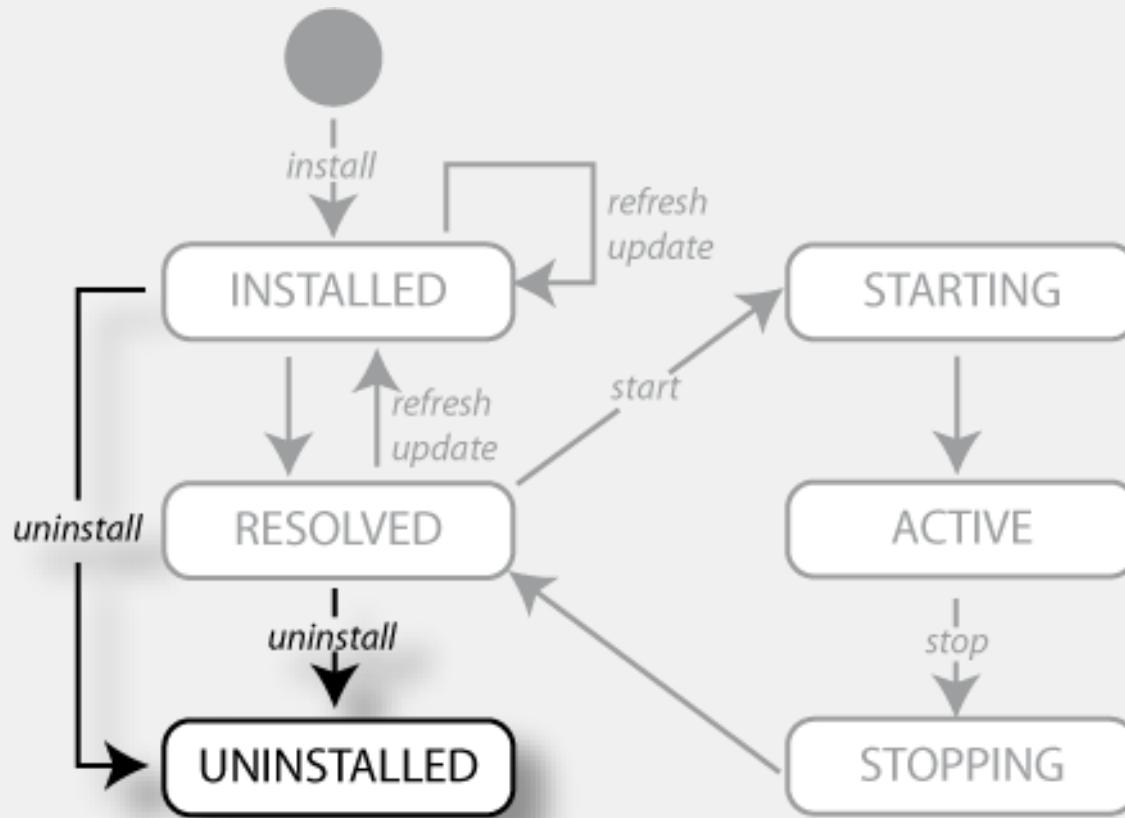
- » Das Bundle wird in den Zustand **STOPPING** gesetzt
- » Falls ein Aktivator deklariert ist, wird `BundleActivator.stop()` aufgerufen

Deinstallieren von Bundles

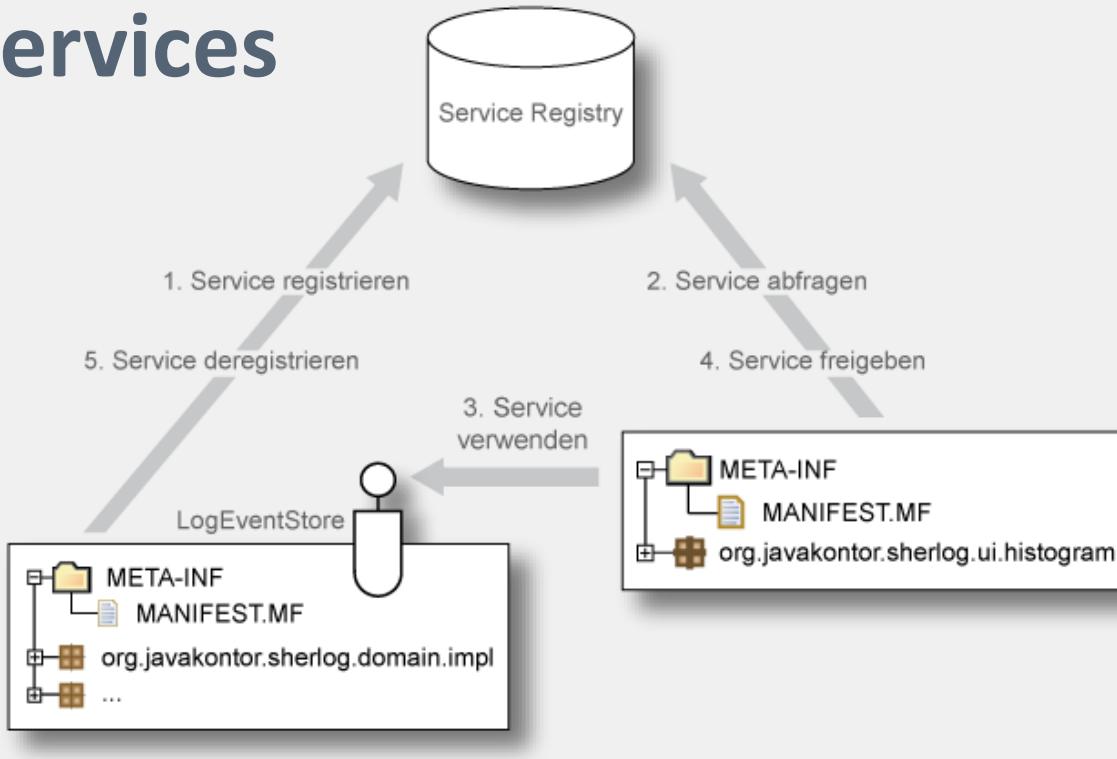


- » Entfernt Bundles aus dem OSGi Framework
- » Das Bunde wird in den Zustand UNINSTALLED gesetzt

Demo 3: Bundle-Lebenszyklus



OSGi Services



Ein OSGi Service...

- » ...ist ein einfaches Java-Objekt (POJO),
- » ...wird an der zentralen Service Registry registriert,
- » ...kann von der zentralen Service Registry abgefragt werden.

Services registrieren

» Registrieren über den BundleContext:

```
public class Activator implements BundleActivator {  
  
    public void start(BundleContextcontext) throws Exception{  
  
        context.registerService(ViewContribution.class.getName() ,  
            newHistogramViewContribution() , null);  
  
        [...]  
    }  
  
    [...]  
}
```

Services benutzen

» Abfragen über den BundleContext:

```
ServiceReference serviceReference = context
    .getServiceReference(LogEventStore.class.getName()) ;

if(serviceReference!= null) {

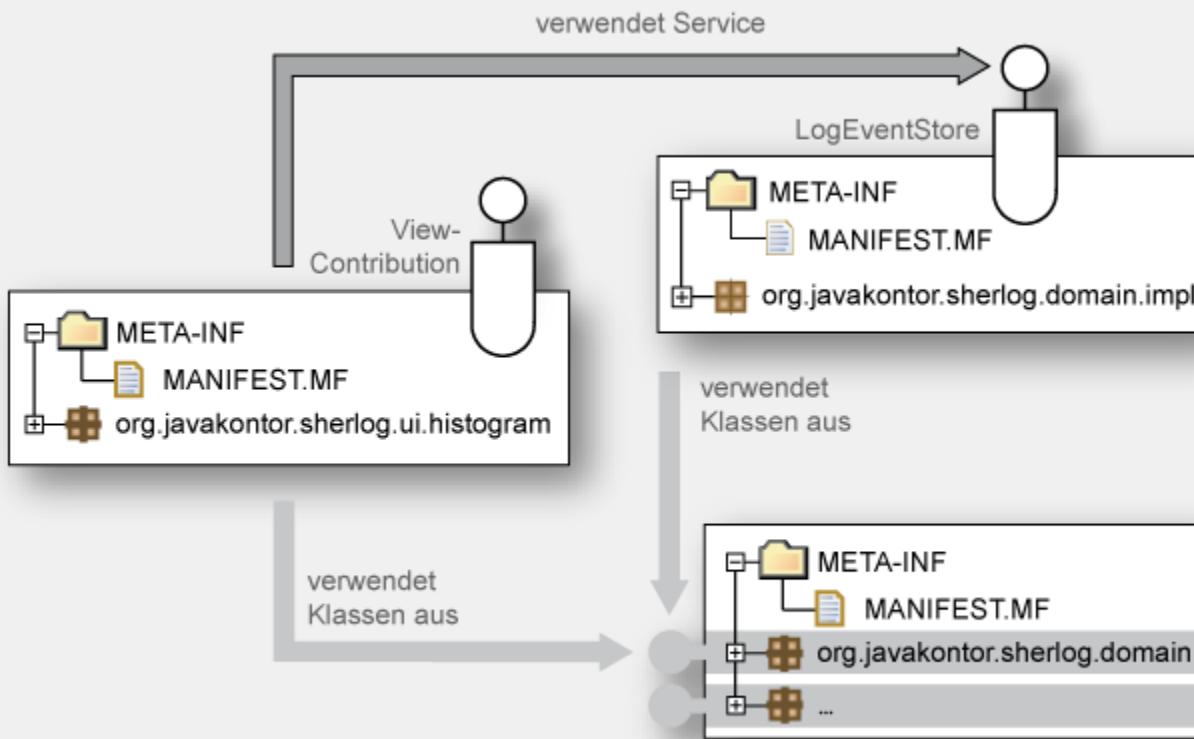
    LogEventStore logEventStore=
        (LogEventStore) context.getService(serviceReference) ;

    if(logEventStore!= null) {

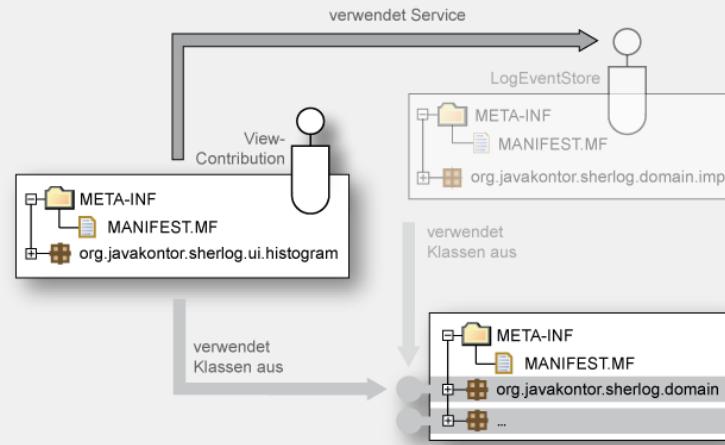
        [...]

    }
}
```

Demo 4: OSGi Services



Services können kommen und gehen



- » Die Suche nach einem Service kann scheitern, weil ...
 - » ... das implementierende Bundle noch nicht gestartet ist
 - » ... der Service noch nicht registriert ist
 - » ... das implementierende Bundle beendet wurde

ServiceListener / ServiceTracker

- » ServiceListener / ServiceTracker
 - » ServiceListener: Callbacks, wenn sich etwas ändert
 - » ServiceTracker: Verfolgen von Service Listener Events
(weniger Code als bei der direkten Verwendung von Service Listenern)
- » Empfehlung: Benutzen Sie ServiceTracker!
- » (Besser: Verwenden Sie deklarative Ansätze!)

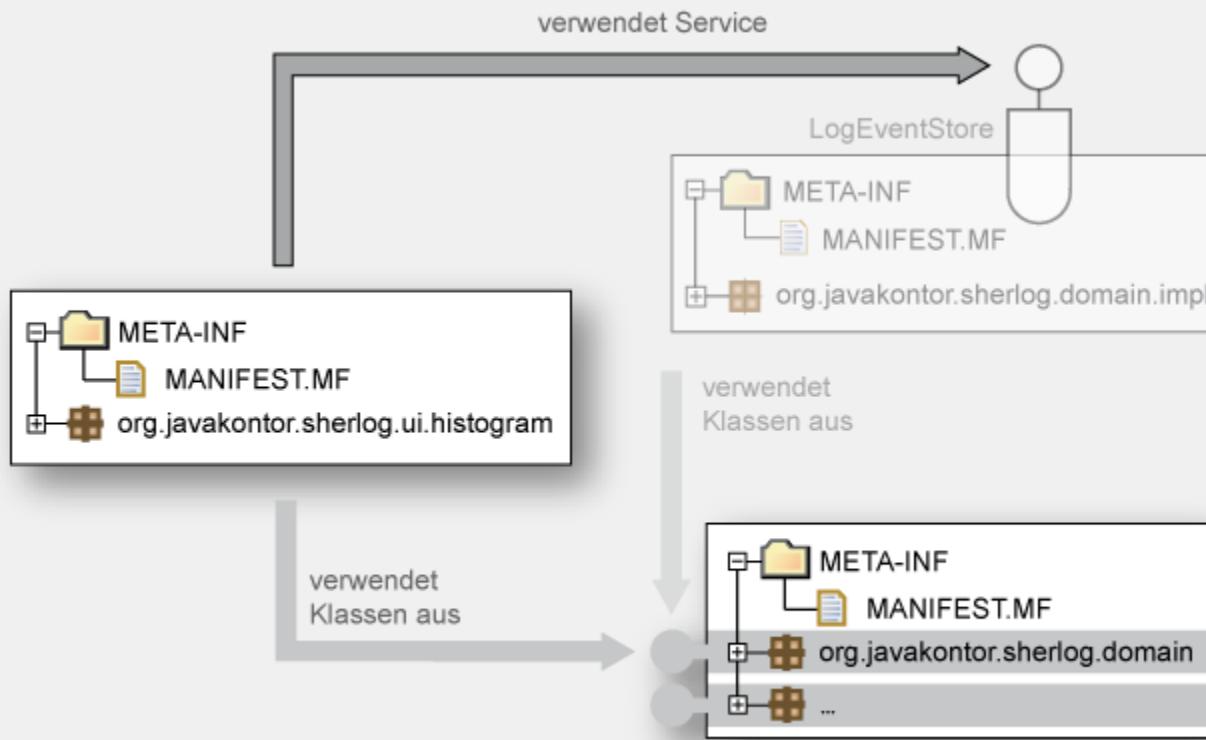
Service Tracker I

```
public class LogEventStoreServiceTracker extends ServiceTracker {  
  
    public LogEventStoreServiceTracker(BundleContext context) {  
        super(context, LogEventStore.class.getName(), null);  
    }  
  
    public Object addingService(ServiceReference reference) {  
        LogEventStore logEventStore =  
            (LogEventStore) super.addingService(reference);  
        // do something here...  
        return logEventStore;  
    }  
  
    public void removedService(ServiceReference reference,  
        Object service) {  
        // do something here...  
        super.removedService(reference, service);  
    }  
}
```

Service Tracker II

```
public class Activator implements BundleActivator {  
  
    private ServiceTracker _serviceTracker;  
  
    public void start(BundleContext context) throws Exception {  
        _serviceTracker =  
            new LogEventStoreServiceTracker(context);  
        _serviceTracker.open();  
    }  
  
    public void stop(BundleContext context) throws Exception {  
        _serviceTracker.close();  
    }  
}
```

Demo 5: Dynamic services



Deklarative Ansätze

» **Declarative Services**

- » Teil der OSGi-Spezifikation
- » Deklarative Beschreibung von Services mit XML
- » Kapitel 12 in “Die OSGi Service Platform”

» **Spring Dynamic Modules**

- » Spring wird mittels OSGi dynamisch
- » <http://www.springframework.org/osgi>
- » Blueprint Service in OSGi Specification R 4.2

» **iPojo**

- » “Original” DI framework for OSGi
- » <http://ipojo.org>

» **Guice - Peaberry**

- » Guice: Performantes, leichtgewichtiges DI Framework
- » Peaberry: Erweiterung zu Guice für OSGi
- » <http://code.google.com/p/peaberry/>
- » <http://code.google.com/p/google-guice/>

Vielen Dank! Fragen?

