The Ultimate Dependency Manager Shootout

Xander Uiterlinden @uiterlix

Sander Mak @sander_mak

Luminis Technologies

Who we are

Sander Mak

http://branchandbound.net

@sander_mak

Xander Uiterlinden

http://blog.uiterlinden.nl

@uiterlix

luminis
Conversing worlds

Agenda

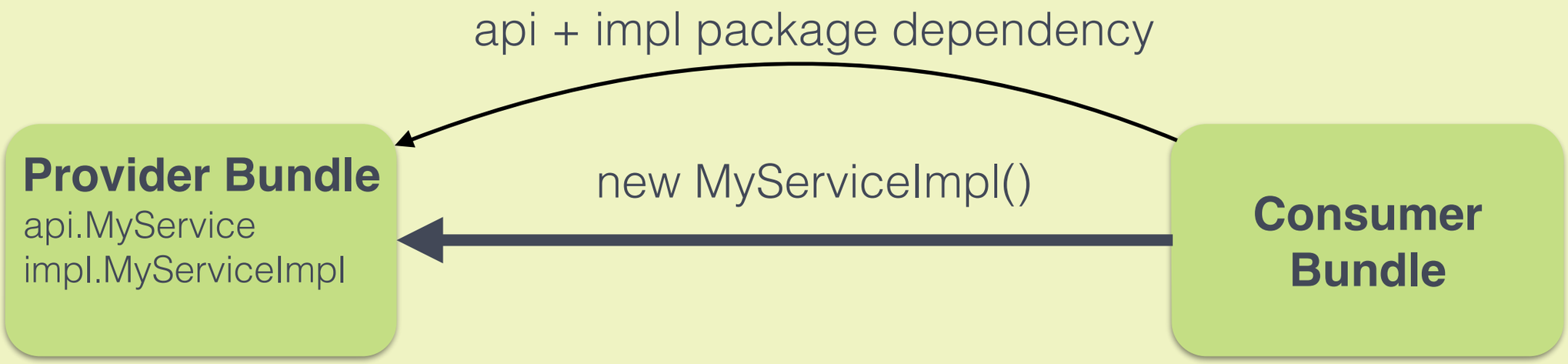
- OSGi Services
- Dependency Managers
- Comparison
  - Overview/community
  - Code
  - Performance
- Conclusion

@uiterlix
@sander_mak

luminis
Conversing worlds

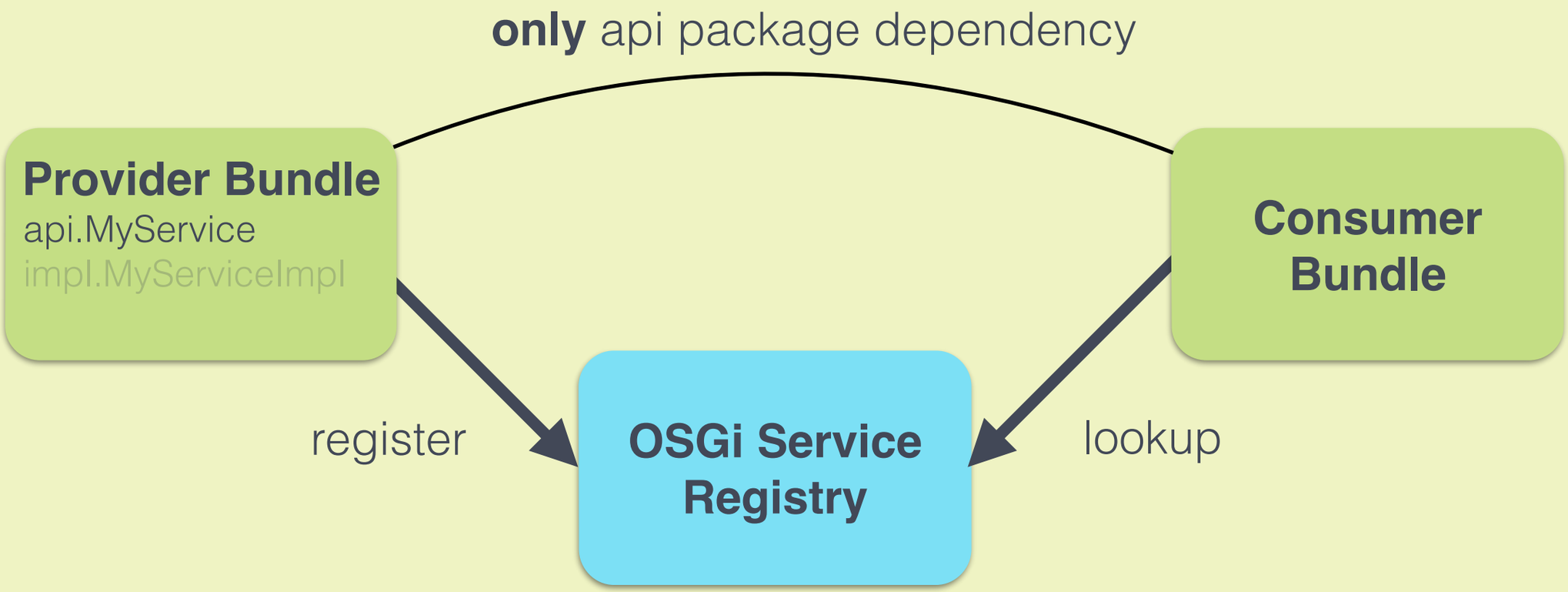
OSGi Services

OSGi Services



- Strong coupling to implementation
- What if provider bundle not available/stops?
- What about configuration?

OSGi Services



- Decoupling through interfaces
- Inversion of Control
- Service dynamics

@uiterlix
@sander_mak

luminis
Conversing worlds

Dependency Managers

Why Dependency Managers?

```
public class MathServiceTracker extends ServiceTracker {
    private MathServiceConsumer mathServiceConsumer;

    public MathServiceTracker(
            BundleContext bundleContext,
            MathServiceConsumer mathServiceCon
        super(bundleContext, IMathService.clas
        this.mathServiceConsumer = mathService
    }

    @Override
    public Object addingService(ServiceRefere
        System.out.println("hello");
        IMathService mathService = (IMathServ
        mathServiceConsumer.setMathService(ma
        return mathService;
    }

    @Override
    public void modif
        this.mathServi
        IMathService m
        this.mathServi
    }

    @Override
    public void remov
        IMathService m
        //for clean u
        this.mathServ
        context.unget
    }
}
```

```
public class Activator implements BundleActivator {

    private MathServiceTracker mathServiceTracker;

    public void start(BundleContext bundleContext) throws Exception {
        MathServiceConsumer consumer = new MathServiceConsumer();
        mathServiceTracker =
                new MathServiceTracker(bundleContext, consumer);
        mathServiceTracker.open();
    }
```

```
public class Activator implements BundleActivator {
    private ServiceRegistration serviceRegistration;

    public void start(BundleContext bundleContext) throws Exception {
        MathService mathService = new MathService();
        serviceRegistration = bundleContext.registerService(
                IMathService.class.getName(), mathService, null
            );
    }

    public void stop(BundleContext bundleContext) throws Exception {
        serviceRegistration.unregister();
    }
}
```

```
throws Exception {
```

@uiterlix
@sander_mak

luminis
Conversing worlds

Why Dependency Managers?

- Cut down the **boilerplate**
- **Decouple** from OSGi APIs
- Dependency Injection
  - Testability
- Component models
  - Aspects
  - Adapters
  - Etc.

Components…?

org.apache.felix.ipojo.annotations.**Component**
org.osgi.service.component.annotations.**Component**
aQute.bnd.annotation.component.**Component**
org.apache.felix.dm.annotation.api.**Component**
org.springframework.stereotype.**Component**
….

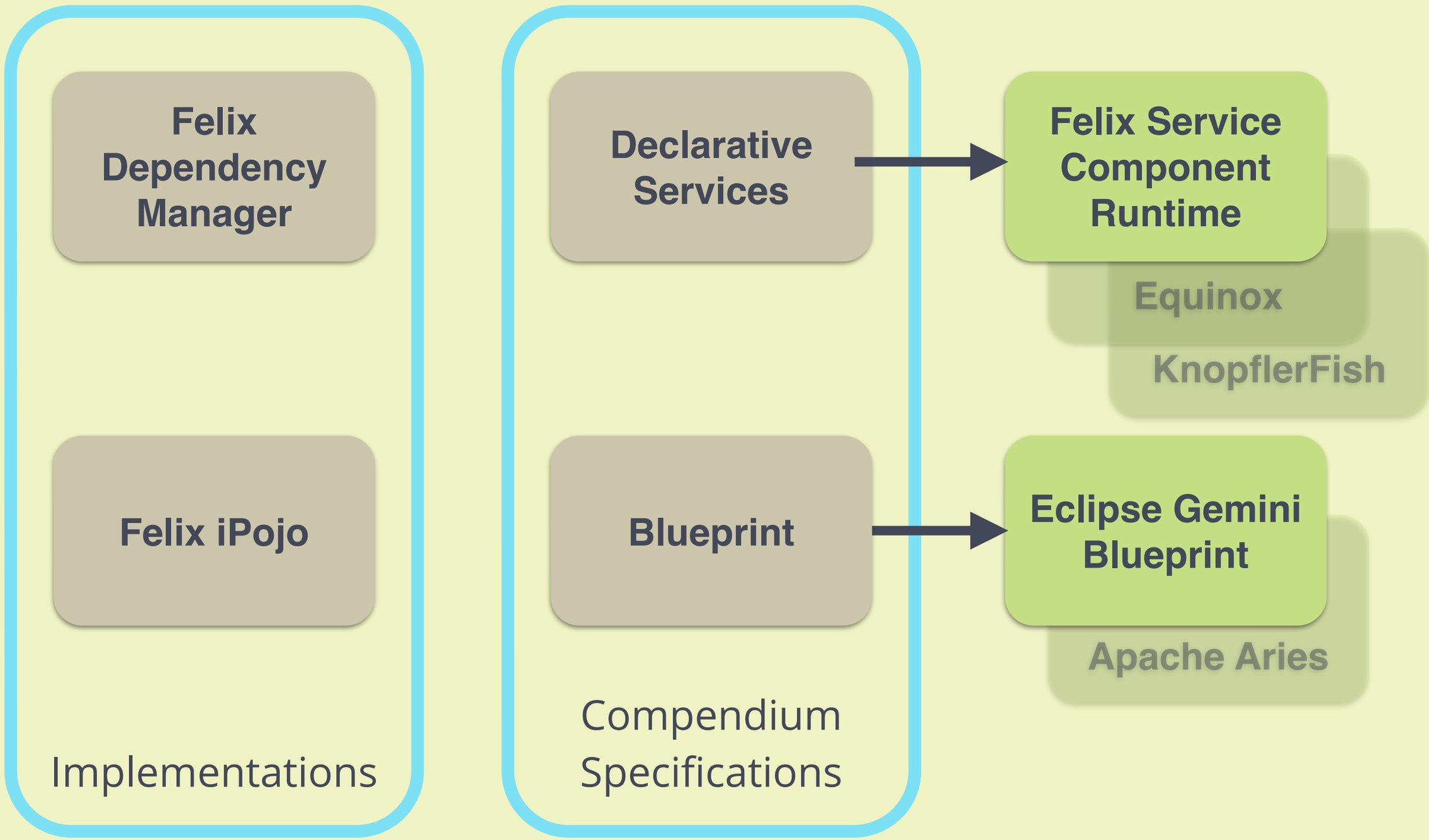
Notions: contract-based, reactive, explicit lifecycle

@uiterlix
@sander_mak

luminis
Conversing worlds

Comparison

The lineup



@uiterlix
@sander_mak

luminis
Conversing worlds

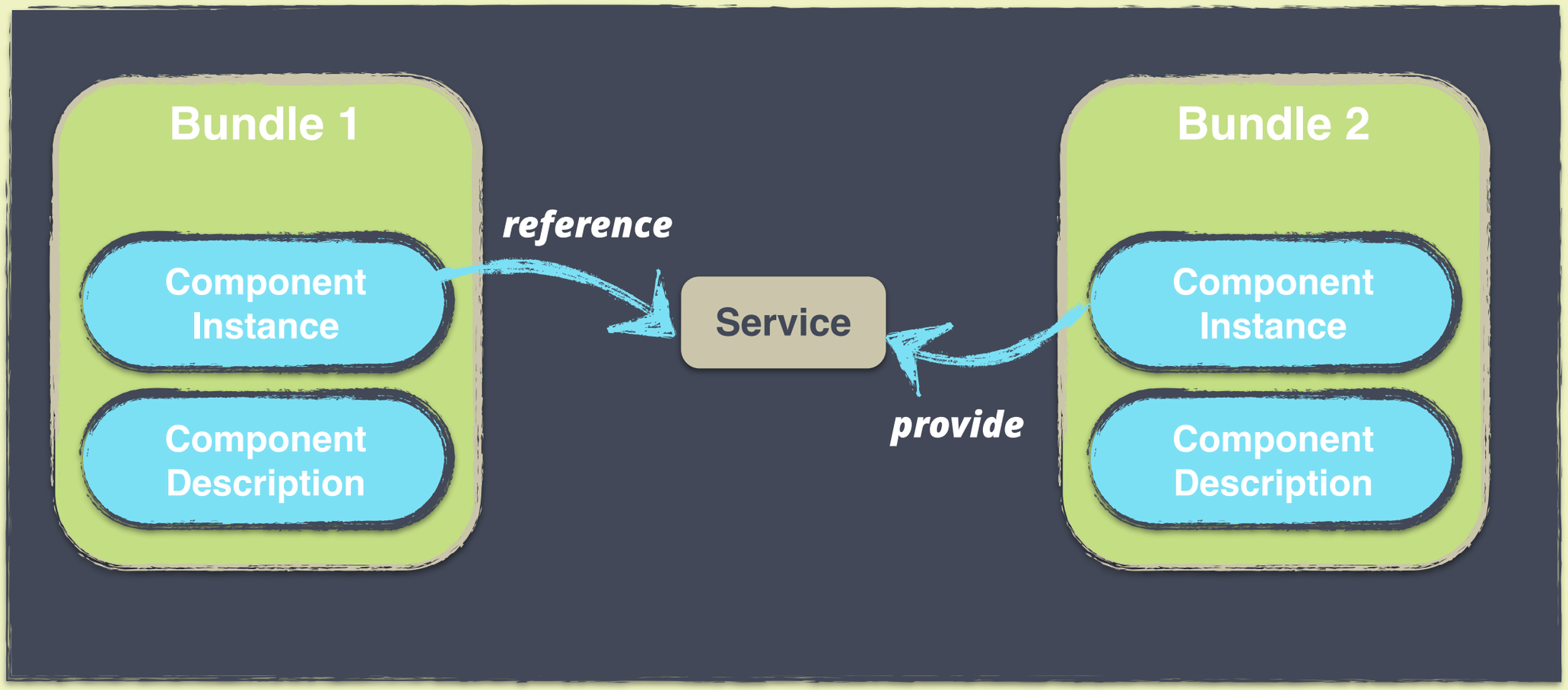
The lineup: disclaimer

We primarily work with (and on) Felix DM

@uiterlix
@sander_mak

luminis
Conversing worlds

Declarative Services



Additional abstractions:

- ComponentContext for each component
- ComponentFactory: client initiates instantiation

@uiterlix
@sander_mak

luminis
Conversing worlds

Felix SCR

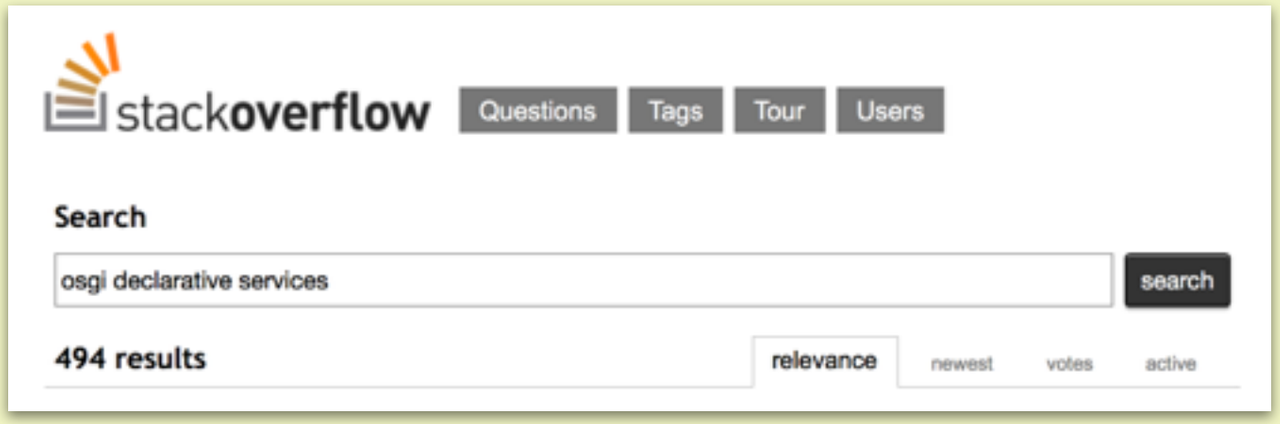
License Apache 2.0

Since 2007

Last release March 2014

User list ~50 msg

Open issues 25



@uiterlix
@sander_mak

luminis
Conversing worlds

Eclipse Gemini Blueprint

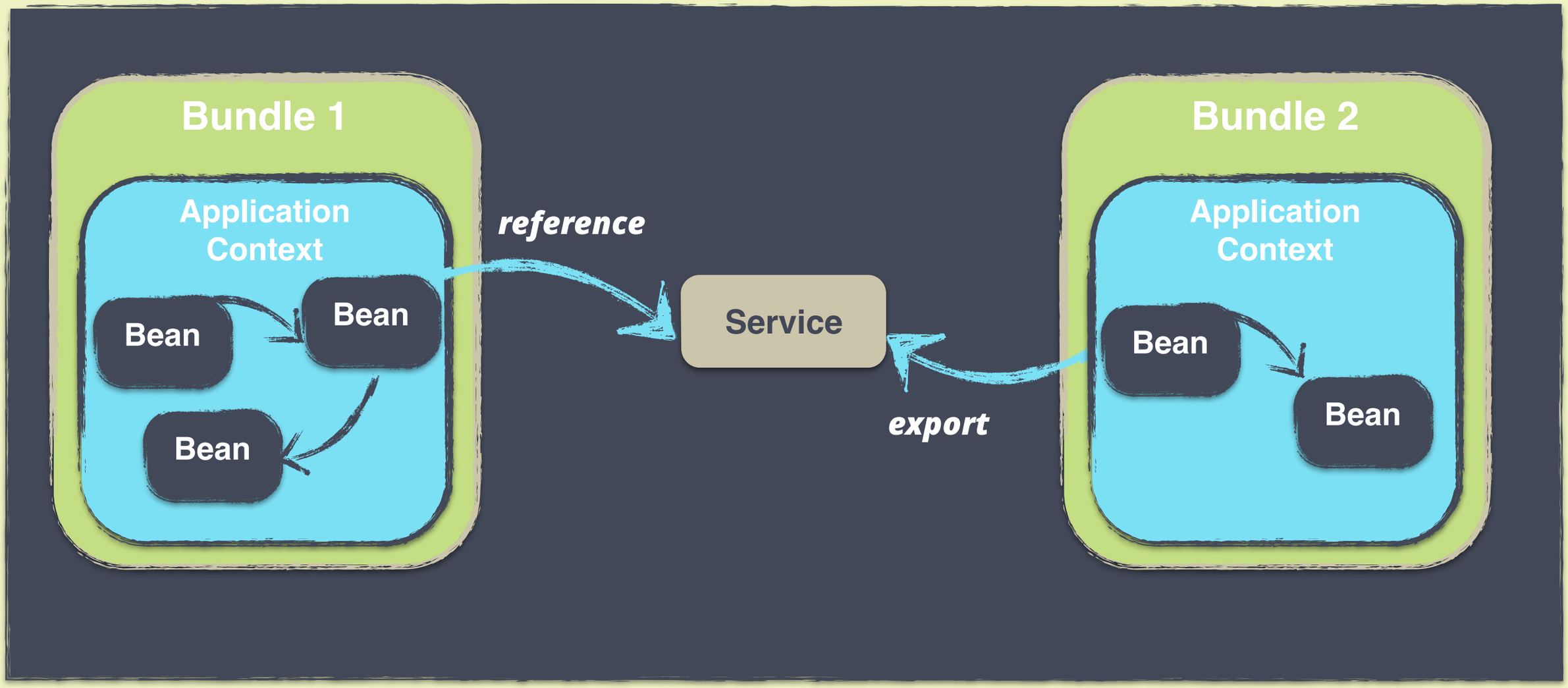
spring source

DM Server

2009

eclipse

Gemini Blueprint



@uiterlix
@sander_mak

luminis
Conversing worlds

Eclipse Gemini Blueprint

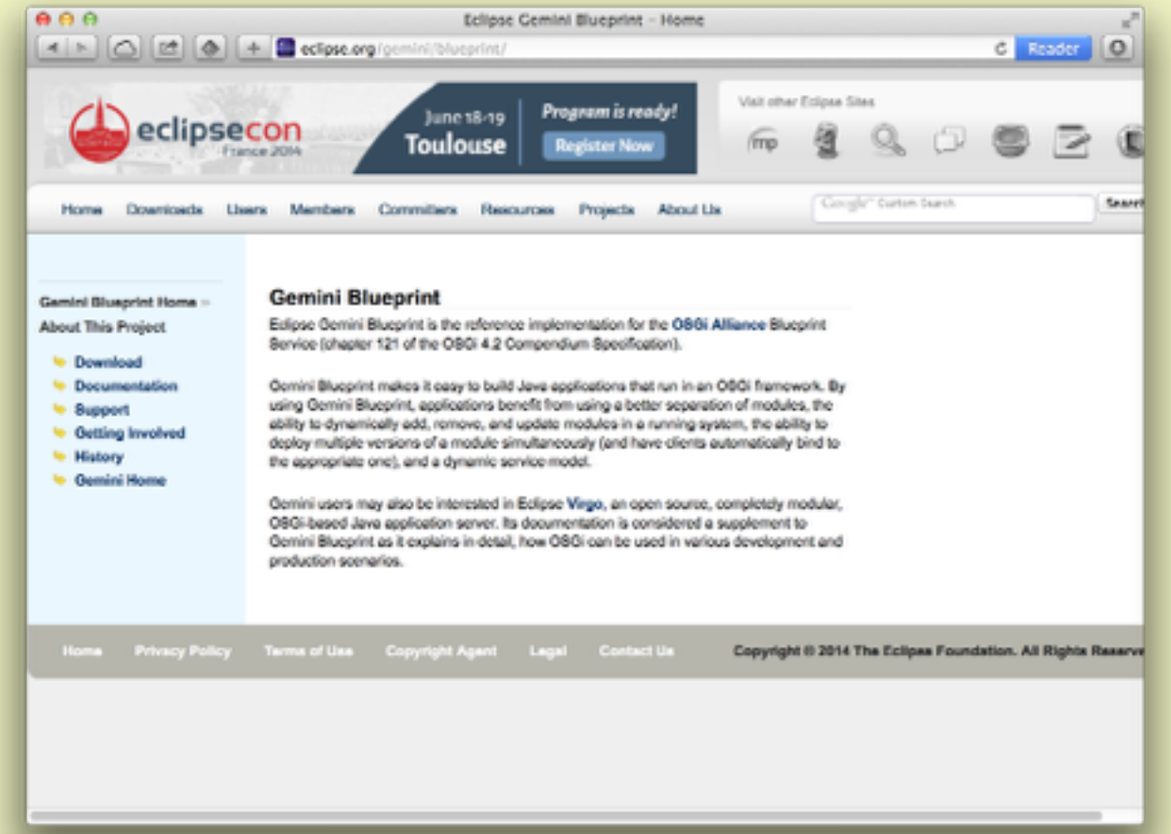
License EPL

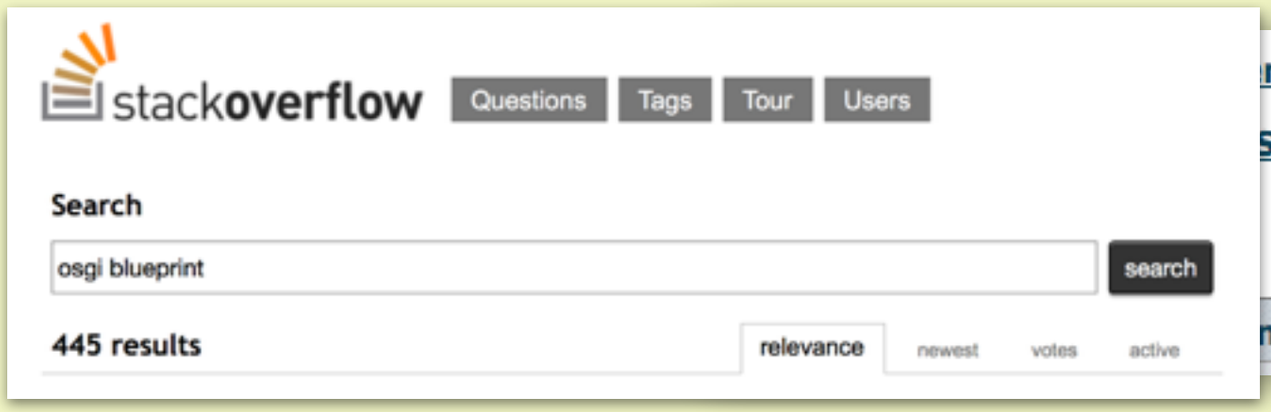
Since 2009

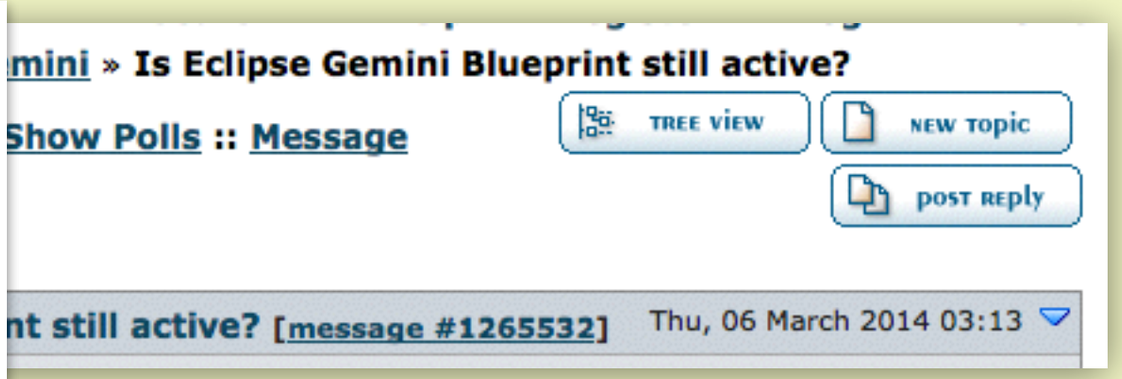
Last release August 2012

User list ~60 msg

Open issues 17



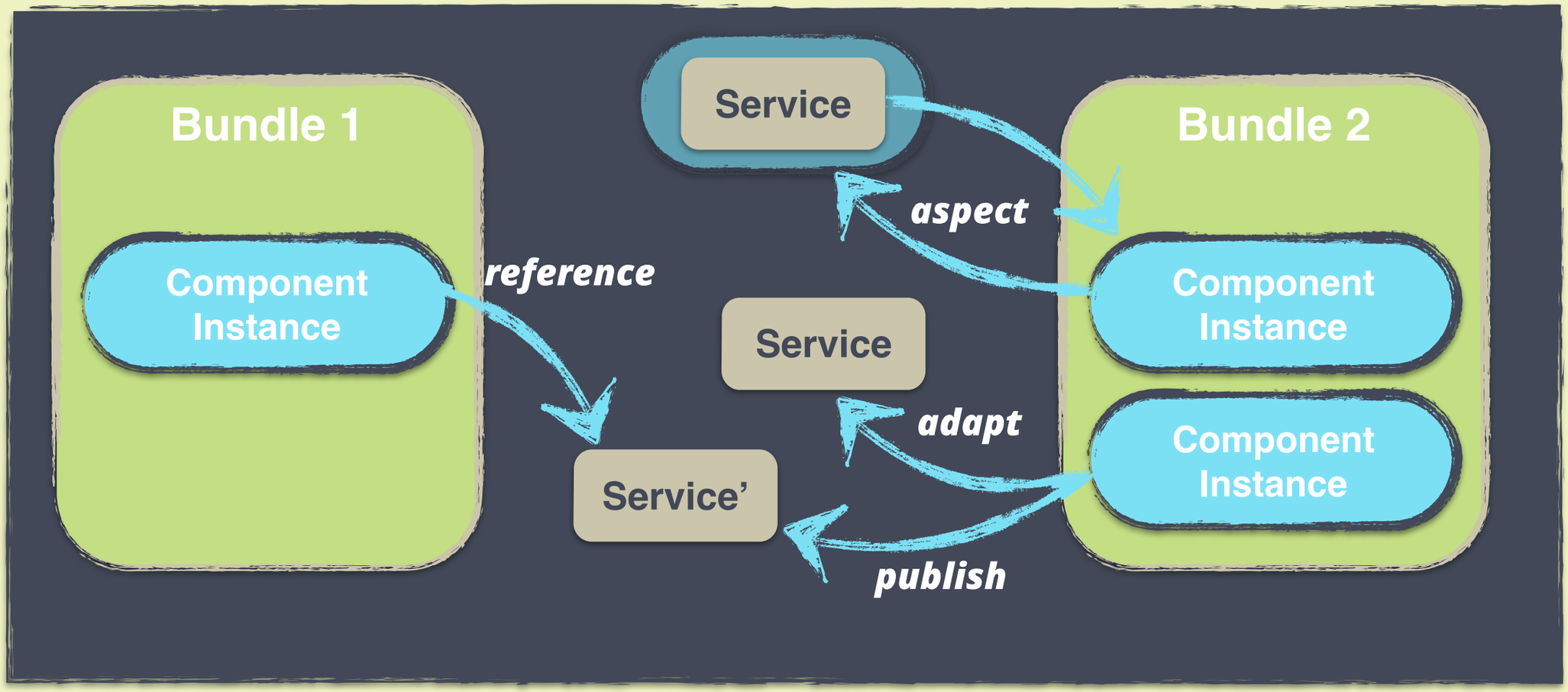




@uiterlix
@sander_mak

luminis
Conversing worlds

Felix Dependency Manager



Additional abstractions:

- Aspect: intercept and republish service
- Adapter: publish service with adapted interface

@uiterlix
@sander_mak

luminis
Conversing worlds

Felix Dependency Manager

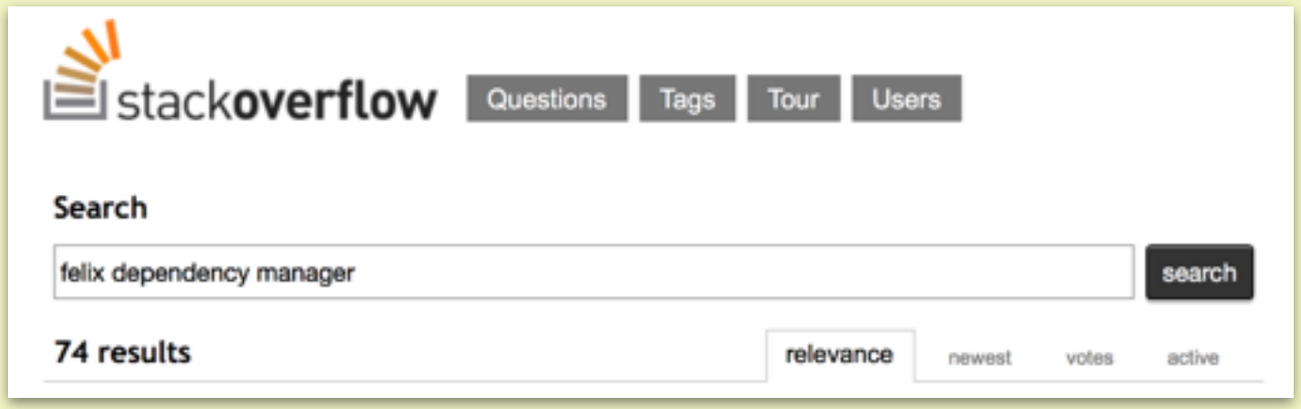
License Apache 2.0

Since 2004

Last release Januari 2013

User list ~1 msg

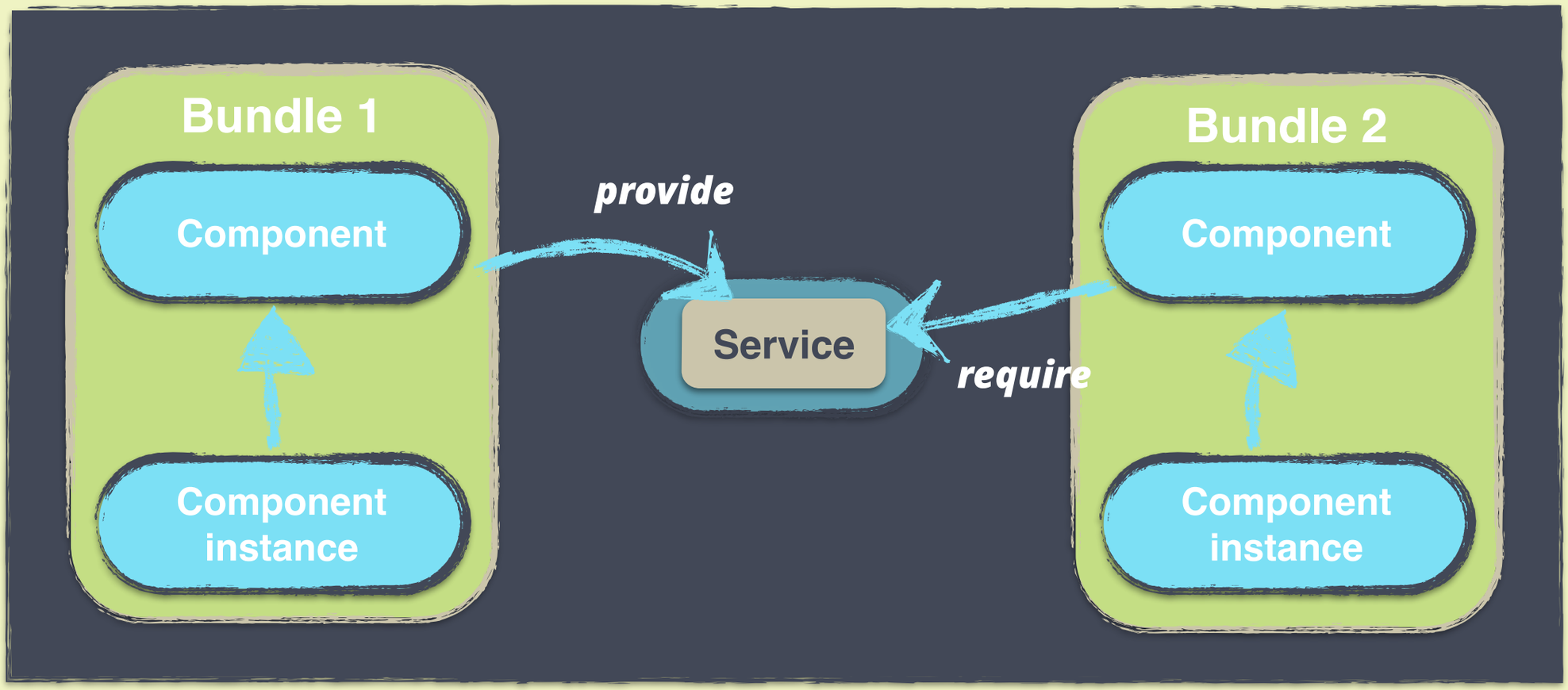
Open issues 22



@uiterlix
@sander_mak

luminis
Conversing worlds

Felix iPojo



Additional abstractions:

- Composites: intra-bundle mechanism
- Scoped service registries
- Pluggable handlers

@uiterlix
@sander_mak

luminis
Conversing worlds

Felix iPojo

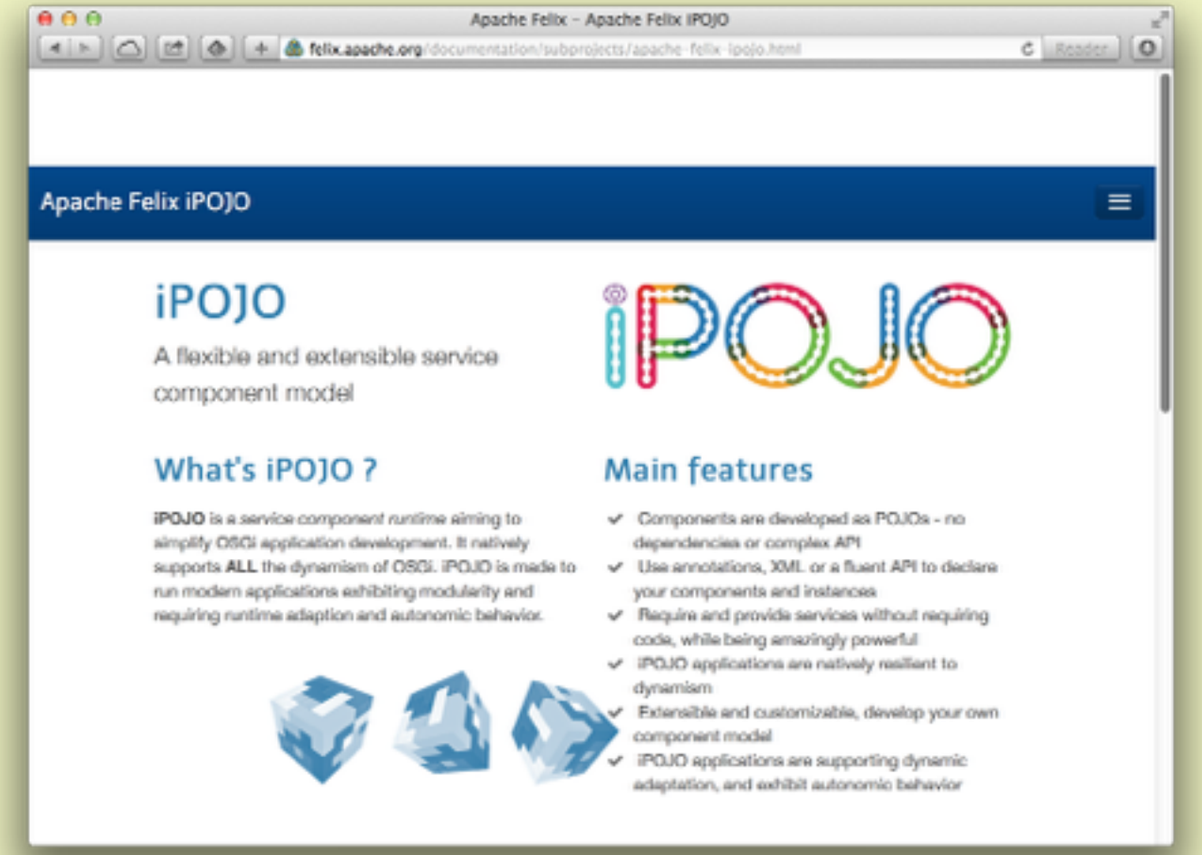
License Apache 2.0

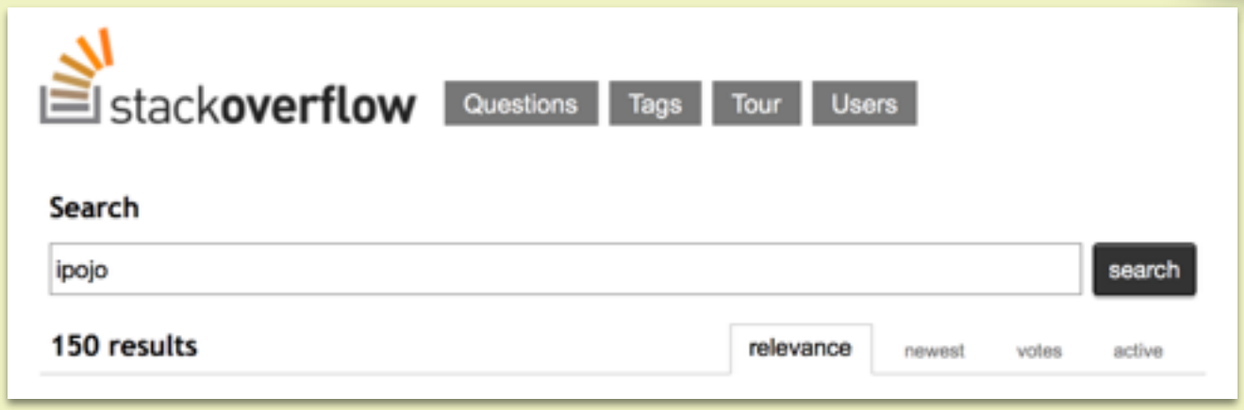
Since 2008

Last release March 2014

User list ~130 msg

Open issues 30





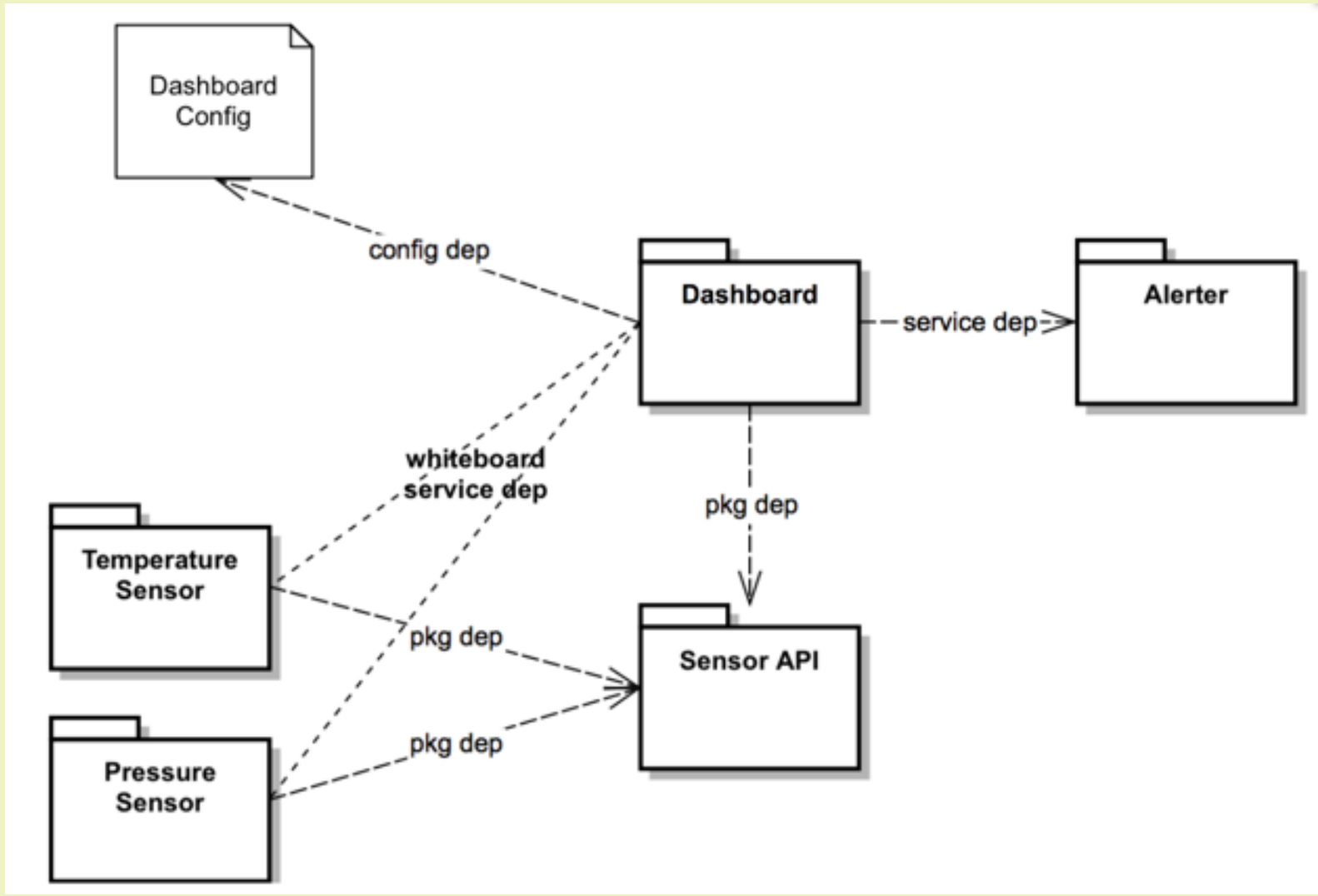
Best site among DMs!

@uiterlix
@sander_mak

luminis
Conversing worlds

Comparison: **code**

The example

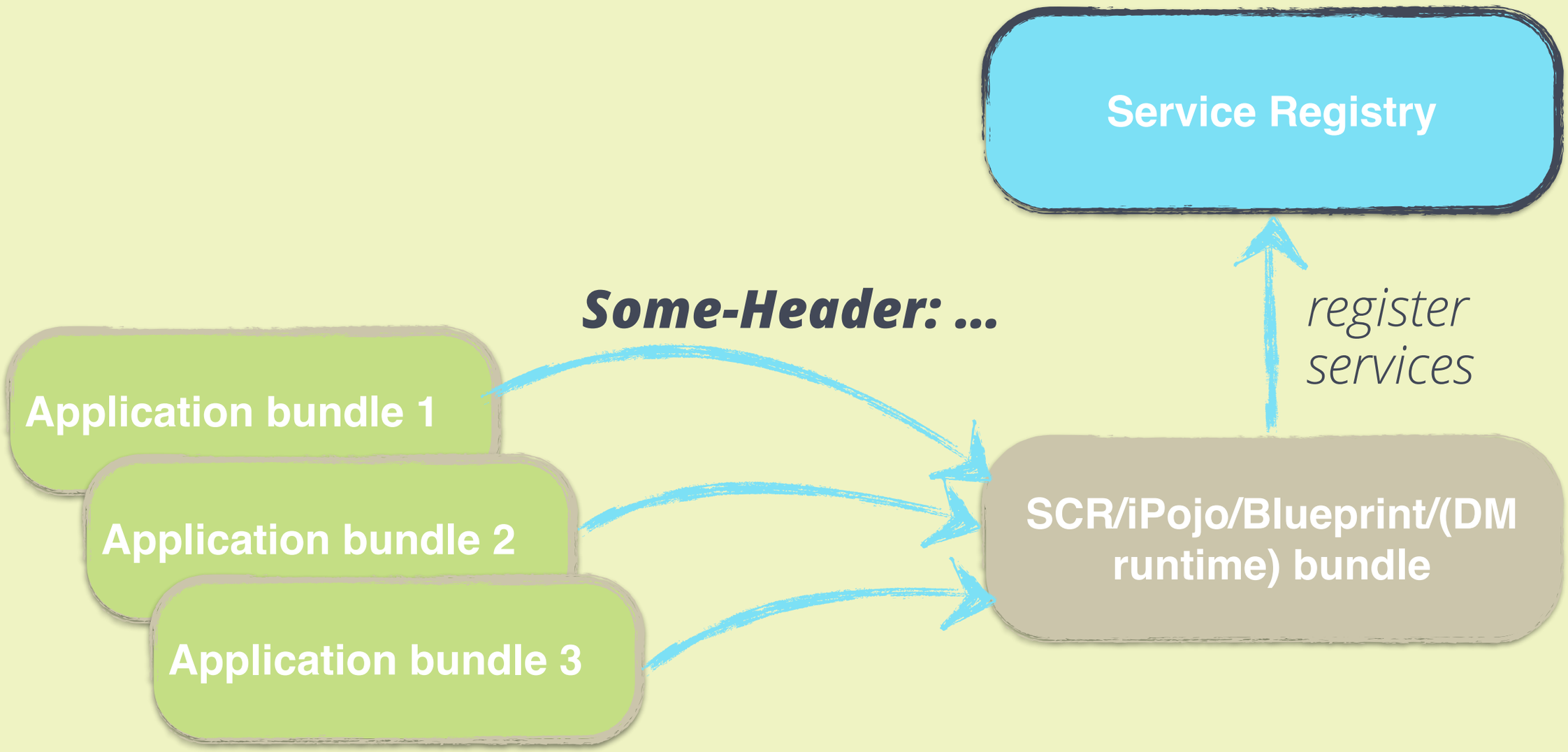


http://bit.ly/dmshootout

@uiterlix
@sander_mak

luminis
Conversing worlds

Extender pattern



@uiterlix
@sander_mak

luminis
Conversing worlds

Declarative Services

Temperate Sensor Bundle

META-INF/MANIFEST.MF
OSGI-INF/temp.xml

TemperatureSensor.class

```
Bundle-SymbolicName: sensors.ds.temp
Import-Package: shootout.ds.sensor.api;version="[1.0,2)"
Include-Resource: OSGI-INF/temp.xml=OSGI-INF/temp.xml
Private-Package: shootout.ds.sensor.temp
Service-Component: OSGI-INF/temp.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<component name="shootout.ds.sensors.temp" immediate="false">
  <implementation class="shootout.ds.sensor.temp.TemperatureSensor" />
  <property name="service.description" value="Temperature Sensor" />
  <service>
    <provide interface="shootout.ds.sensor.api.Sensor" />
  </service>
</component>
```

@uiterlix
@sander_mak

luminis
Conversing worlds

Declarativ

Dashboard Bundle

```
<?xml version="1.0" e
<component xmlns="htt
  immediate="true" co
  <implementation cla
  <property name="ser
  <service>
    <provide interfac
  </service>
  <reference name="se
    cardinality="1..n
    <reference name="
        bind="setAler
    cardinality="1..1
</component>
```

```
public class DashboardImpl implements Dashboard {

  private Alerter alerter;

  public synchronized void setAlerter(Alerter alerter) {
    this.alerter = alerter;
  }

  public synchronized void unsetAlerter(Alerter alerter) {
    if (this.alerter == alerter) {
      this.alerter = null;
    }
  }

  protected void activate(Map<String, Object> properties) {
    System.out.println("DashboardImpl activated");
    properties.get("refreshinterval"));
  }

  protected void deactivate() {
    System.out.println("DashboardImpl deactivated");
  }

  private Set<Sensor> sensors = Collections
      .newSetFromMap(new ConcurrentHashMap<Sensor, Boolean>());

  public void sensorRemoved(Sensor sensor) {
    System.out.println("Sensor removed " + sensor.toString());
    sensors.remove(sensor);
  }

  public void sensorAdded(Sensor sensor) {
    System.out.println("Sensor added " + sensor.toString());
    sensors.add(sensor);
  }
}
```

@uiterlix
@sander_mak

Declarative Services

- No direct field injection
- Defaults:
  - Delayed component (activate on use)
  - Static policy -> many reactivations

What if ‘better’ service arrives?

	Static	Dynamic
Reluctant	Do nothing	Rebind optionals
Greedy	Reactivate	Rebind

@uiterlix
@sander_mak

luminis
Conversing worlds

BluePrint

Temperate Sensor Bundle

```
Bundle-SymbolicName: sensors.blueprint.temp
Import-Package: shootout.blueprint.sensor.api;version="[1.0,2)"
Private-Package: shootout.blueprint.sensor.temp
```

Bundle-Blueprint header optional

META-INF/MANIFEST.MF
OSGI-INF/blueprint/temp.xml

TemperatureSensor.class

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <service id="tempService" interface="shootout.blueprint.sensor.api.Sensor"
    ref="temp" />

  <bean id="temp" class="shootout.blueprint.sensor.temp.TemperatureSensor">
  </bean>

</blueprint>
```

@uiterlix
@sander_mak

luminis
Conversing worlds

BlueP

Dashboard B

```
<?xml version
<blueprint  .

  <service id
    ref="dash

  <bean id="d
    class="sh

    <property
      <refere
    </propert
    <property
      <refere
    </propert

    <!-- The
      the giv
    <bp-comp:
  </bean>

</blueprint>
```

```
public class DashboardImpl implements Dashboard {

  private List<Sensor> sensors;
  private Alerter alerter;
  private Integer refreshinterval;

  // No special name, referenced with init-method property. Must be public
  public void myActivate() {
    System.out.println("DashboardImpl activated");
  }

  // No special name, referenced with destroy-method property. Must be public.
  public void myDeactivate() {
    System.out.println("DashboardImpl deactivated");
  }

  public synchronized void setSensors(List<Sensor> sensors) {
    this.sensors = sensors;
  }

  public synchronized void setAlerter(Alerter alerter) {
    this.alerter = alerter;
  }

  public synchronized void setRefreshinterval(Integer refreshinterval) {
    this.refreshinterval = refreshinterval;
  }

}
```

@uiterlix
@sander_mak

luminis
Conversing worlds

BluePrint

- Strong focus on intra-bundle composition
- Hides dynamicity (until TimeOutException…)
  - If dependency goes away, component is not stopped. If service is exposed, unregisters it.
  - No lazy activation like DS
  - Whole ‘applicationContext’ started or not
- (Custom) TypeConverters
- Constructor/setter injection, no field injection
- Namespace hell :(

@uiterlix
@sander_mak

luminis
Conversing worlds

Felix Dependency Manager

```
Bundle-SymbolicName: sensors.dm.temp
Import-Package: org.apache.felix.dm;version="[3.0,4)",org.osgi.framework
 ;version="[1.6,2)",shootout.dm.sensor.api;version="[1.0,2)"
Private-Package: shootout.dm.sensor.temp
Bundle-Activator: shootout.dm.sensor.temp.Activator
```

Temperate Sensor Bundle

META-INF/MANIFEST.MF

Activator.class
TemperatureSensor.class

```
public class Activator extends DependencyActivatorBase {

    @Override
    public void init(BundleContext context, DependencyManager manager)
        throws Exception {
    Dictionary<String, String> props = new Hashtable<String, String>();
    props.put("sensor.type", "temperature");

    manager.add(createComponent()
        .setInterface(Sensor.class.getName(), props)
        .setImplementation(TemperatureSensor.class)
    );

    }

    @Override
    public void destroy(BundleContext context, DependencyManager manager)
        throws Exception {
    // Nothing to do
    }

}
```

@uiterlix
@sander_mak

luminis
Conversing worlds

Felix De

Dashboard Bund

```
public class Ac

  public void i
      throws Ex

    manager.add
        .setInt
        .setImp
        .add(cr
            .se
        .add(cr
        .add(cr
        );
  }

}
```

```
public class DashboardImpl implements Dashboard, ManagedService {

  private volatile Alerter alerter;

  // Special method called by Felix DM
  public void start() {
    System.out.println("DashboardImpl activated");
    // ...
  }

  // Special method called by Felix DM
  public void stop() {
    System.out.println("DashboardImpl deactivated");
    // ...
  }

  private Set<Sensor> sensors = Collections
      .newSetFromMap(new ConcurrentHashMap<Sensor, Boolean>());

  public void sensorAdded(Sensor sensor) {
    System.out.println("Sensor added " + sensor.toString());
    sensors.add(sensor);
  }

  public void sensorRemoved(Sensor sensor) {
    System.out.println("Sensor removed " + sensor.toString());
    sensors.remove(sensor);
  }

  @Override
  public void updated(Dictionary<String, ?> properties)
      throws ConfigurationException {
    // ...

  }
}
```

@uiterlix
@sander_mak

Felix Dependency Manager

- **It's code**: dynamically create and depend on services
- Annotations available (+ bnd plugin)
- Field injection, NullObject pattern
- Manual mgmt. of list of dependencies :(
- Services registered eagerly from Activator
- Features: aspects, adapters

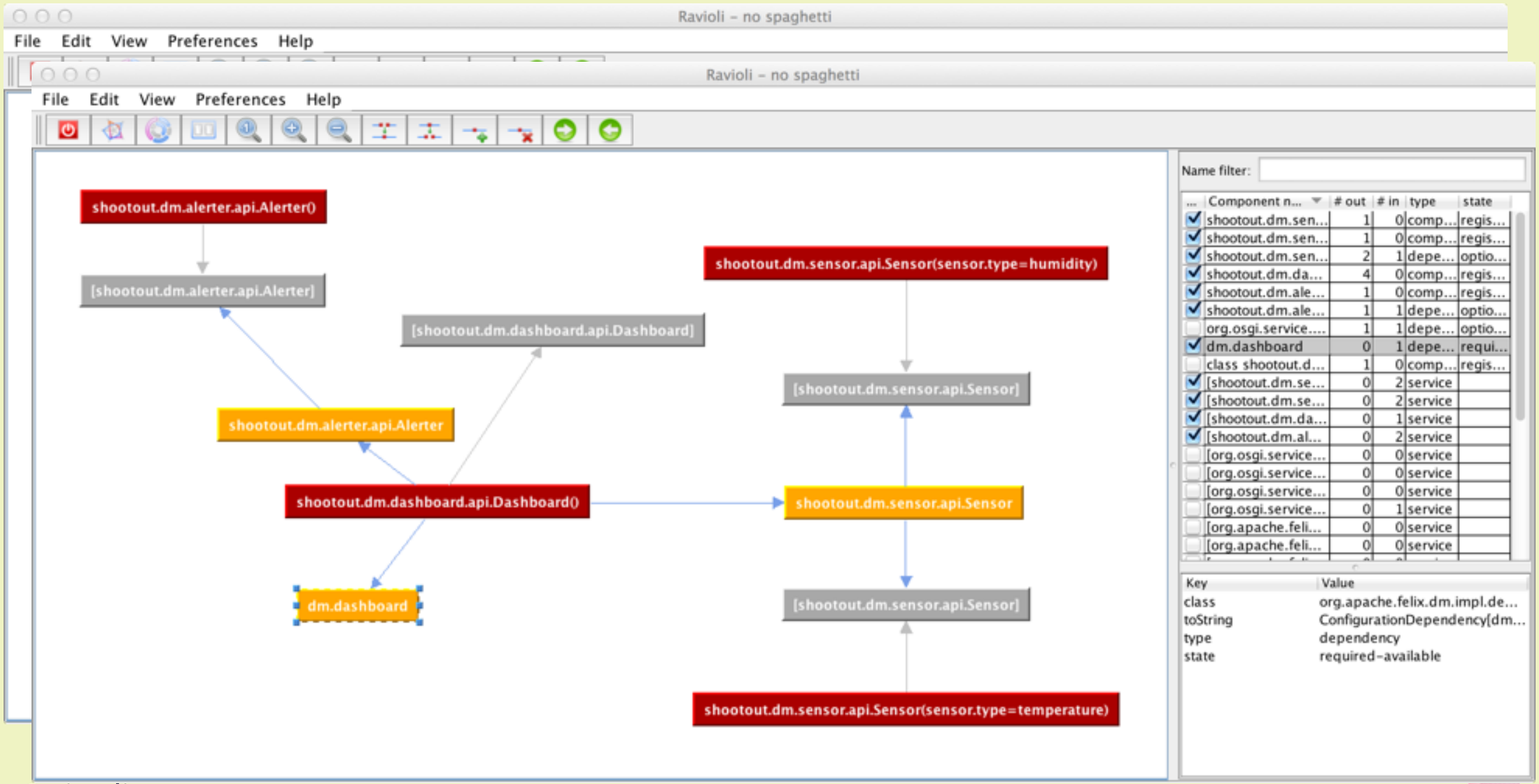
DM 4 coming up...

@uiterlix
@sander_mak

luminis
Conversing worlds

Felix Dependency Manager

- **https://bitbucket.org/uiterlix/ravioli**



@uiterlix
@sander_mak

luminis
Conversing worlds

Felix iPojo

Temperate Sensor Bundle

TemperatureSensor.class
metadata.xml

META-INF/MANIFEST.MF

```
Bundle-SymbolicName: dashboard
Bundle-Version: 0.0.0.201406012309
Export-Package: shootout.ipojo.dashboard.api;version="1.0"
Import-Package: org.apache.felix.ipojo;version="[1.11.2,2.0.0)",org.apac
 he.felix.ipojo.architecture;version="[1.11.2,2.0.0)",org.osgi.service.c
 m;version="[1.4,2)",org.osgi.service.log;version="1.3",shootout.ipojo.a
 lerter.api;version="[1.0,2)",shootout.ipojo.dashboard.api;version="[1.0
 ,2)",shootout.ipojo.sensor.api;version="[1.0,2)"
IPOJO-Components: instance { $component="shootout.ipojo.dashboard.Dashbo
 ardImpl" }instance { $component="shootout.ipojo.dashboard.ConfigCreator
 " }component { $name="shootout.ipojo.dashboard.DashboardImpl" $classnam
 e="shootout.ipojo.dashboard.DashboardImpl" properties { $pid="ipojo.das
 hboard" $updated="updated" }requires { $field="alerter" }requires { $ag
 gregate="true" $optional="false" $id="shootout.ipojo.sensor.api.Sensor"
  callback { $method="sensorAdded" $type="bind" }callback { $method="sen
 sorRemoved" $type="unbind" }}callback { $transition="validate" $method=
 "start" }callback { $transition="invalidate" $method="stop" }manipulati
 on { $classname="shootout.ipojo.dashboard.DashboardImpl" interface { $n
 ame="shootout.ipojo.dashboard.api.Dashboard" }interface { $name="org.os
 gi.service.cm.ManagedService" }field { $name="alerter" $type="shootout.
 ipojo.alerter.api.Alerter" }field { $name="refreshinterval" $type="java
 .lang.Integer" }field { $name="sensors" $type="java.util.Set" }field {
 $name="timer" $type="java.util.Timer" }method { $name="$init" }method {
```

Generated by IDE plugin by processing annotations and metadata.xml

@uiterlix
@sander_mak

luminis
Conversing worlds

Felix iPojo

```
@Component(managedservice="ipojo.dashboard")
@Provides
public class DashboardImpl implements Dashboard, ManagedService {

    @Requires
    private volatile Alerter alerter;

    // Special method called by iPojo
    @Validate
    public void start() {
        System.out.println("DashboardImpl activated");
        //..
    }

    @Bind(aggregate=true, optional=false)
    public void sensorAdded(Sensor sensor) {
        System.out.println("Sensor added " + sensor.toString());
        sensors.add(sensor);
    }

    @Unbind
    public void sensorRemoved(Sensor sensor) {
        System.out.println("Sensor removed " + sensor);
        sensors.remove(sensor);
    }

    @Override
    @Updated
    public void updated(Dictionary<String, ?> properties)
            throws ConfigurationException {
        //..
    }
}
```

```
<ipojo>
    <instance component="shootout.ipojo.dashboard.DashboardImpl"/>
</ipojo>
```

@uiterlix
@sander_mak

Felix iPojo

- Both ‚static' configuration and configuration using code
- Dependency manager with the most advanced features:
  - External handlers
  - Service binding interceptors
  - Stereotypes
- Annotations limited to declaring components and ‚simple' instances. XML recommended for declaring instances.
- Shell commands for component inspection

@uiterlix
@sander_mak

luminis
Conversing worlds

Comparison: **performance**

Benchmarks

- Don’t be afraid of too many services ?
- Promises…

- *„Each bundle may register zero or more services. Each bundle may also use zero or more services. There exists* ***no limit*** *on the number of services, more than the ones given by memory limits or java security permissions.”*

(Knoplerfish tutorial)

Benchmark scenario

Temperature sensors for each postal code in the Nederlands

463860 Sensor services

Service properties:

- Province
- Municipality
- City
- PostalCode

@uiterlix
@sander_mak

luminis
Conversing worlds

Benchmarks

- Publishing Sensor services
  - Felix, Equinox, Knoplerfish
  - Plain OSGi, DM, iPojo

- Consuming Sensor services
  - Whiteboard; all sensors in Amsterdam

    (&(province=Noord-Holland)(municipality=Amsterdam))
  - Felix, Equinox, Knoplerfish
  - DS, DM, iPojo, Blueprint

@uiterlix
@sander_mak

luminis
Conversing worlds

Disclaimer

- Results are as-is, no further investigation done
- No optimizations done

- You can repeat these tests yourself.

  Projects are available on github

http://bit.ly/dmshootout

@uiterlix
@sander_mak

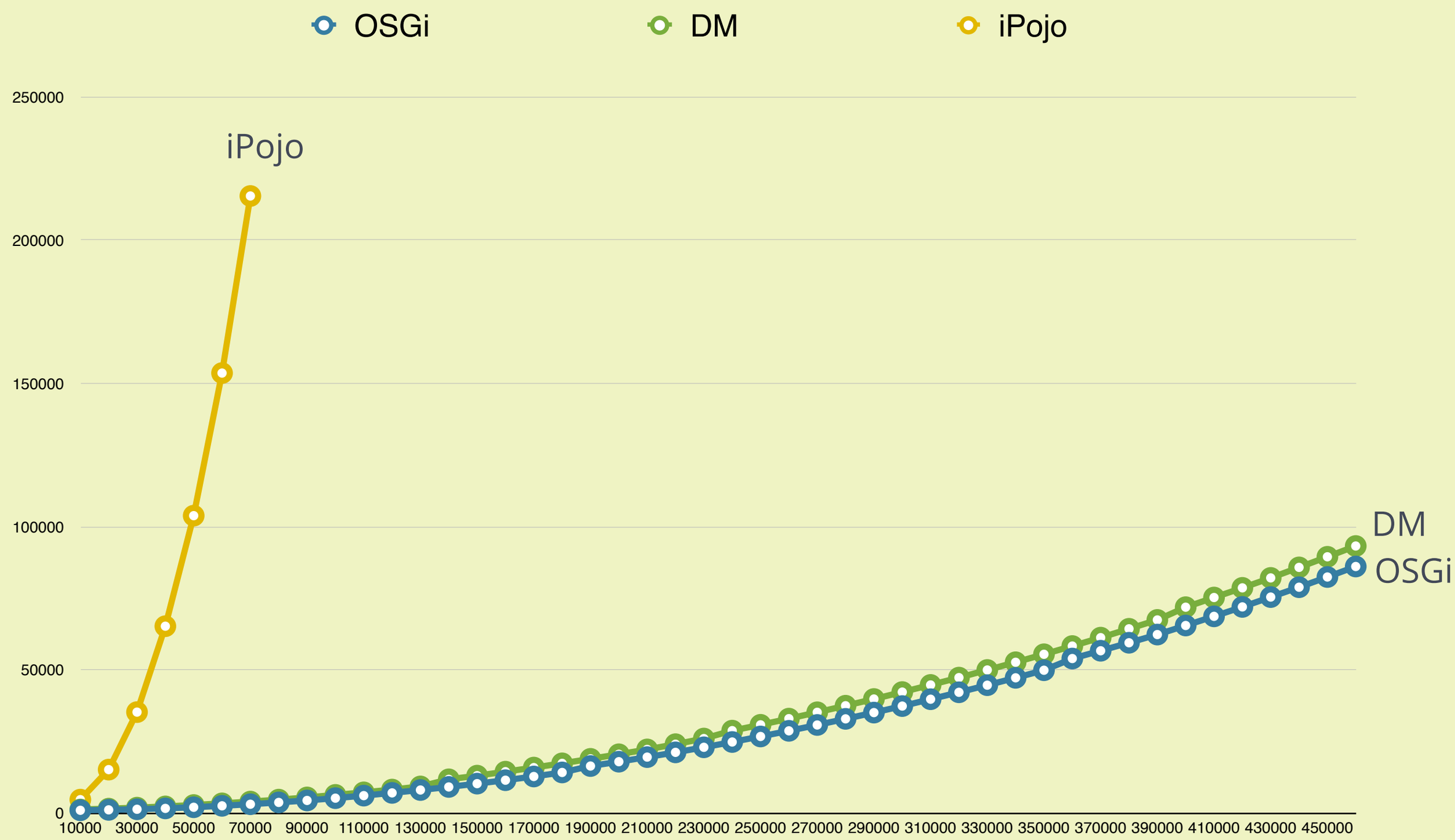
luminis
Conversing worlds

Register services (Equinox)

@uiterlix
@sander_mak

luminis
Conversing worlds

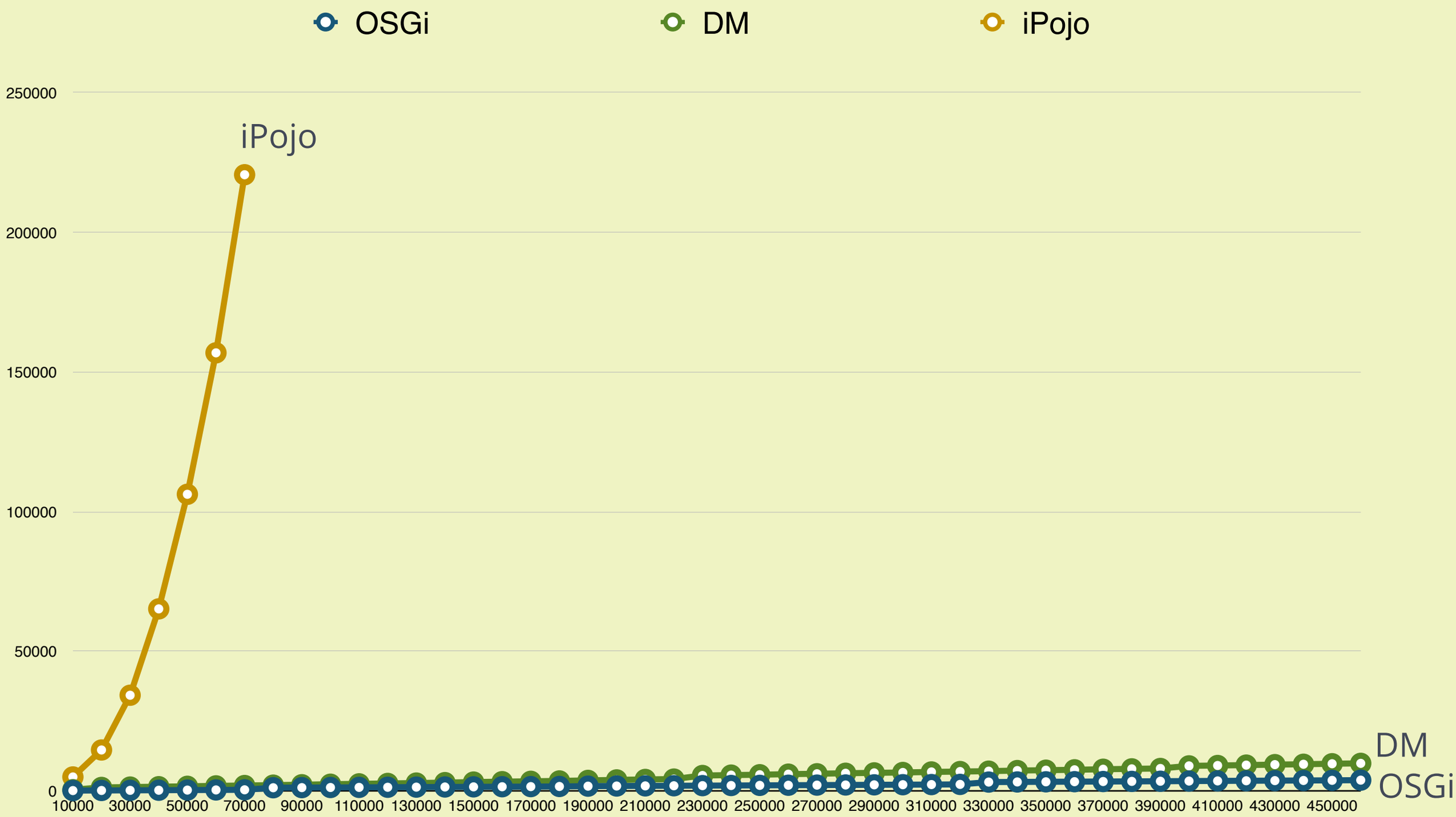
Register services (Felix)



@uiterlix
@sander_mak

luminis
Conversing worlds

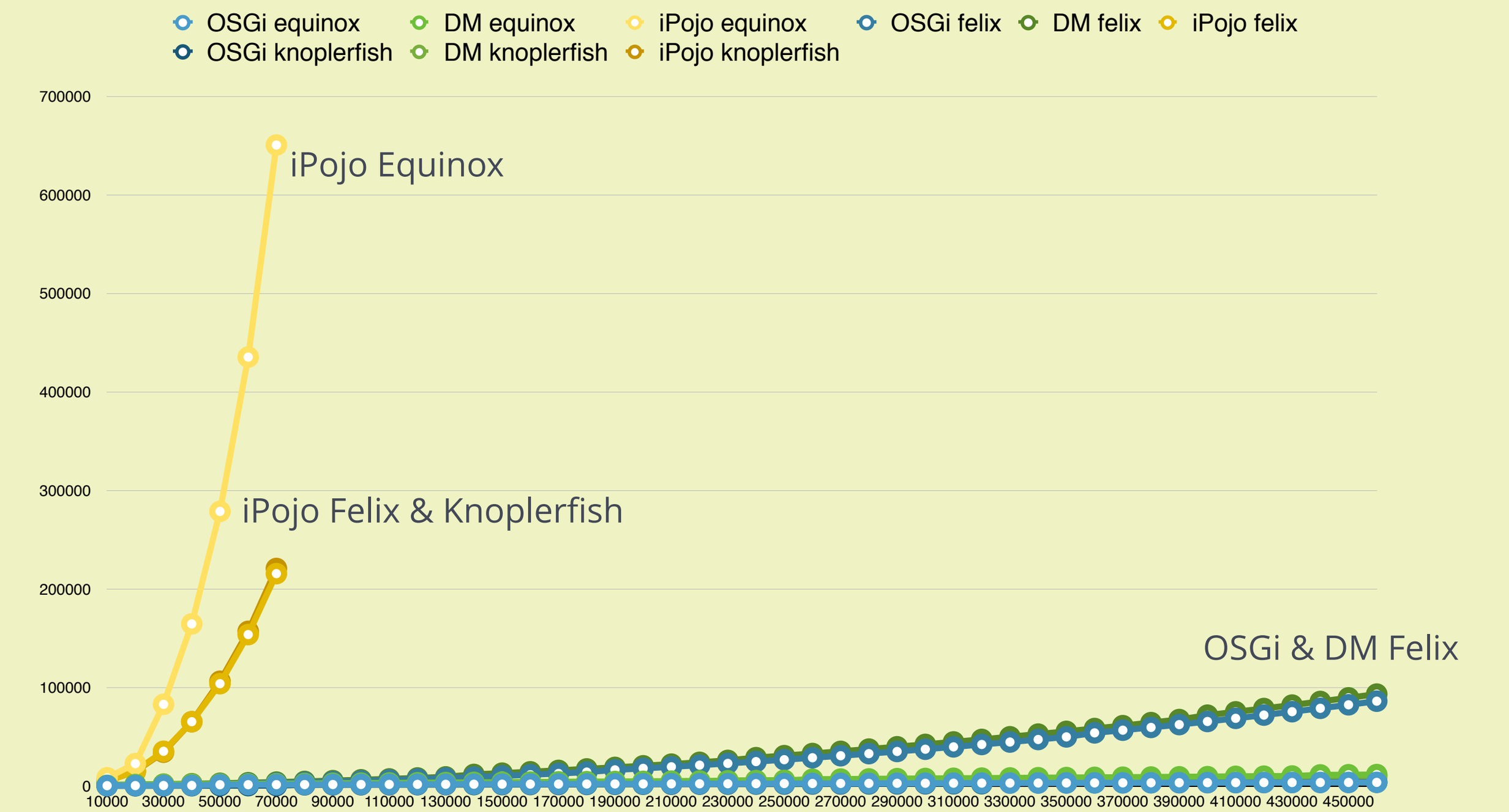
Register services (Knoplerfish)



@uiterlix
@sander_mak

luminis
Conversing worlds

Register services (combined)



@uiterlix
@sander_mak

luminis
Conversing worlds

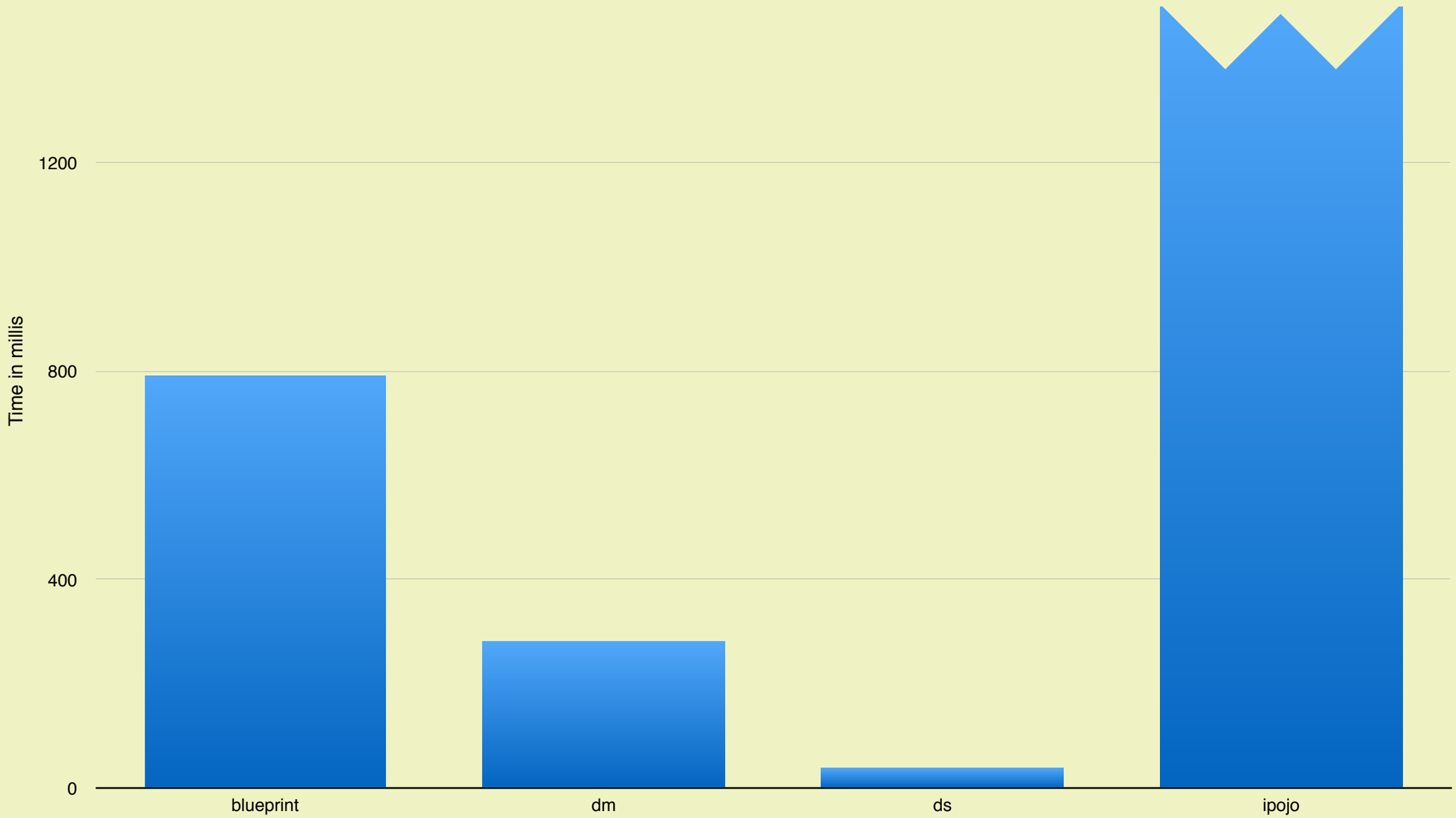
Register services

- Observations
  - Non linear results for registering services with Felix
  - Registering services using iPojo takes considerably longer compared to plain OSGi and DM

@uiterlix
@sander_mak

Consuming services

Injecting 19021 services into a single consumer on Equinox

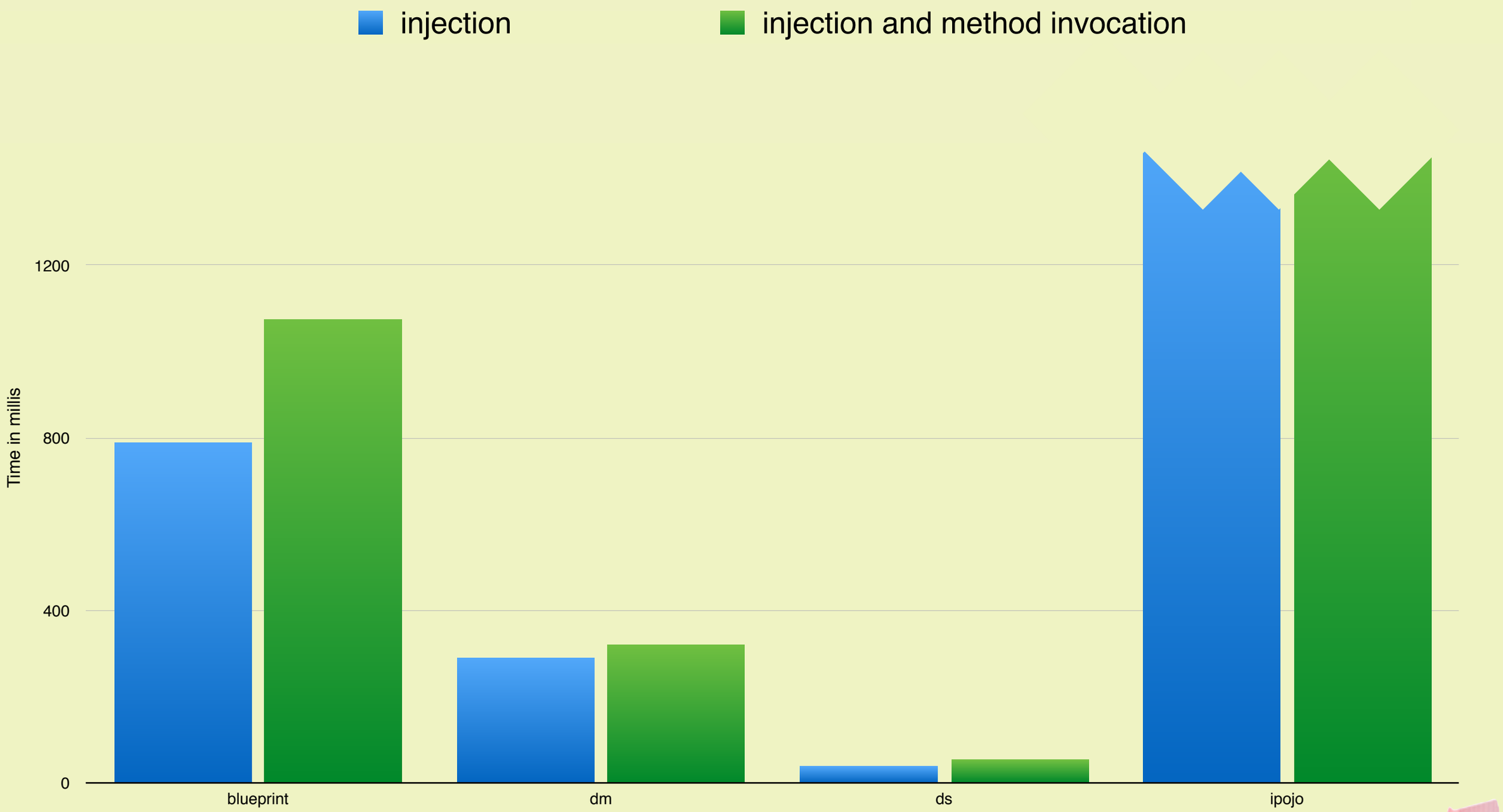


@uiterlix
@sander_mak

luminis
Conversing worlds

Method invocation overhead

Injecting 19021 services into a single consumer on Equinox,
and invoking a method on the injected services

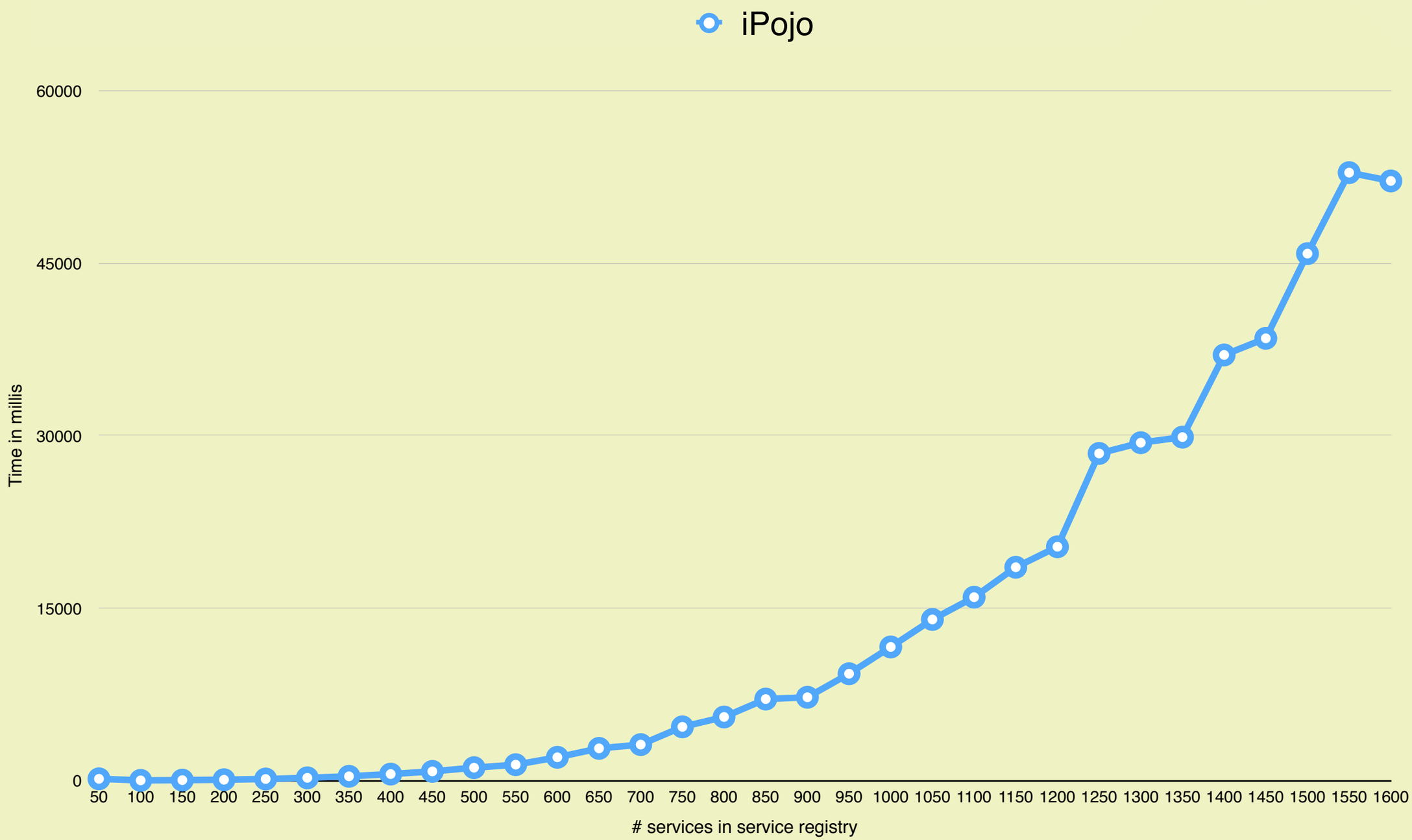


@uiterlix
@sander_mak

luminis
Conversing worlds

iPojo and # of services

Injecting a single service into a single consumer on Equinox

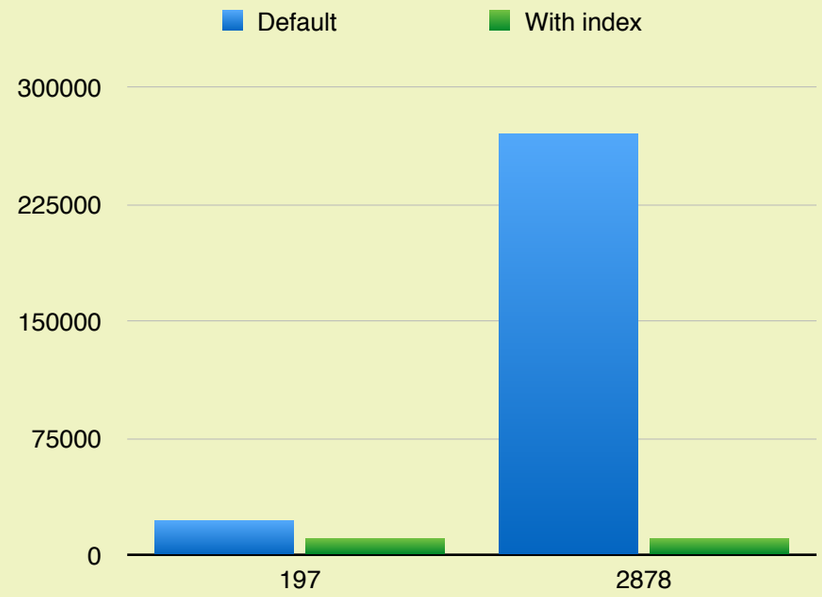


@uiterlix
@sander_mak

luminis
Conversing worlds

Performance considerations

- Use of LDAP filters
  - Filter complexity, service registry scans
  - Felix DM provides optimization though indexing service properties



- Bundle starting order
  - Service tracker obtain initial versus service listener events

@uiterlix
@sander_mak

luminis
Conversing worlds

Conclusion

Conclusion

- Features ranging from simple to advanced (DS, Blueprint < DM < iPojo)
- Annotation/XML-only frameworks are easy to use but provide less dynamics
- Consider the expected size of your application (services, dependencies) and run some test before choosing one!
- You can mix & match
  - … but don’t do that

@uiterlix
@sander_mak

Questions?

Code: **http://bit.ly/dmshootout**

Sander Mak

http://branchandbound.net

@sander_mak

Xander Uiterlinden

http://blog.uiterlinden.nl

@uiterlix

@uiterlix
@sander_mak

luminis
Conversing worlds