JBoss Certified Technology Partner

Israel JBoss User Group

Session 08 / 16.4.2008

Open Your Eyes To OSGi

By : Yanai Franchi, Chief Architect , Tikal

Agenda

- What is OSGi ?
- The problems it helps us to solve
- OSGi programming model
- Spring Dynamic Modules

OSG - *what?*

OSGi Acronym

O = Open

S = Services

G = Gateway

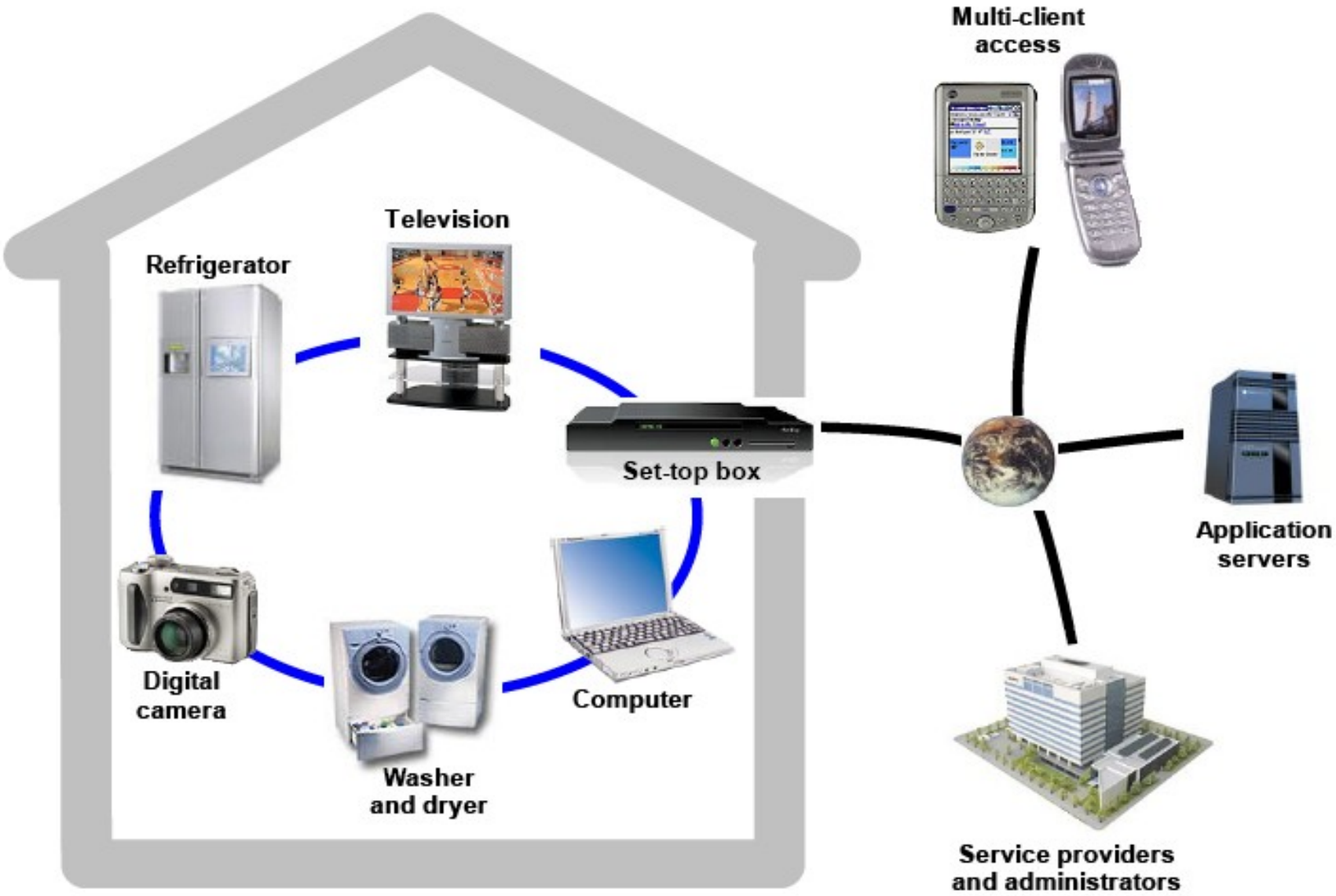
i = initiative

OSGi TM

OSGi ™

The Dynamic Module System For Java

Original OSGi Vision



Who's Doing It ?

- Open source implementation
  - Equinox
  - Felix
  - Knoplerfish
- Significant enterprise Java usage
  - Eclipse
  - IBM – WebSphere, Lotus
  - JonAS
  - BEA
  - Oracle
  - JBoss
  - SpringSource

So What ?

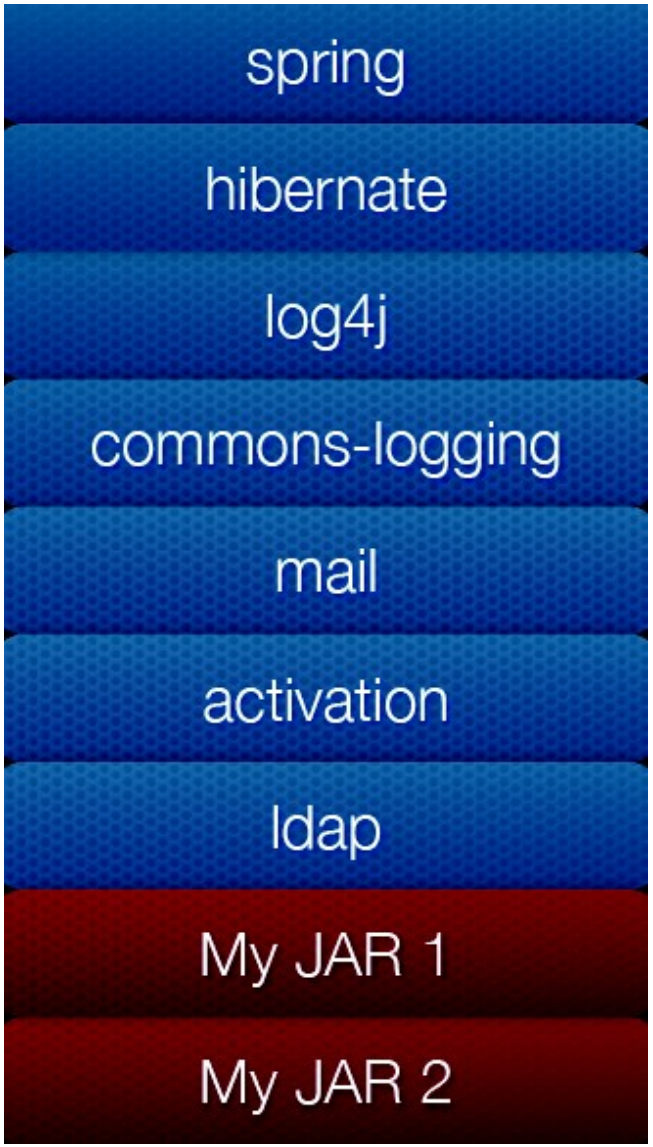
- Visibility
- Versioning
- Operational Control

The Humble JAR File

Typical Java Application

JAR HELL

Our Application Stack...



No Runtime Representation

```
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
org.bar.foo.Flibble
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
javax.naming.InitialContext
javax.naming.NamingException
com.sun.internal.DontUseThisClass
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
org...jdbc.core.RowMapper
org...jdbc.core.SqlParameter
org...jdbc.support.GeneratedKeyHolder
com.foo.bar.Wibble
java.security.Principal
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
java.util.List
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
com.sun.internal.DontUseThisClass
com.foo.bar.Wibble
org.bar.foo.Flibble
javax.naming.InitialContext
javax.naming.NamingException
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
```

Loading Classes...

```
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
org.bar.foo.Flibble
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
javax.naming.InitialContext
javax.naming.NamingException
com.sun.internal.DontUseThisClass
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
org...jdbc.core.RowMapper
org...jdbc.core.SqlParameter
org...jdbc.support.GeneratedKeyHolder
com.foo.bar.Wibble
java.security.Principal
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
java.util.List
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
com.sun.internal.DontUseThisClass
com.foo.bar.Wibble
org.bar.foo.Flibble
javax.naming.InitialContext
javax.naming.NamingException
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
```

com.foo.bar.Wibble?

...The ClassLoader Found One

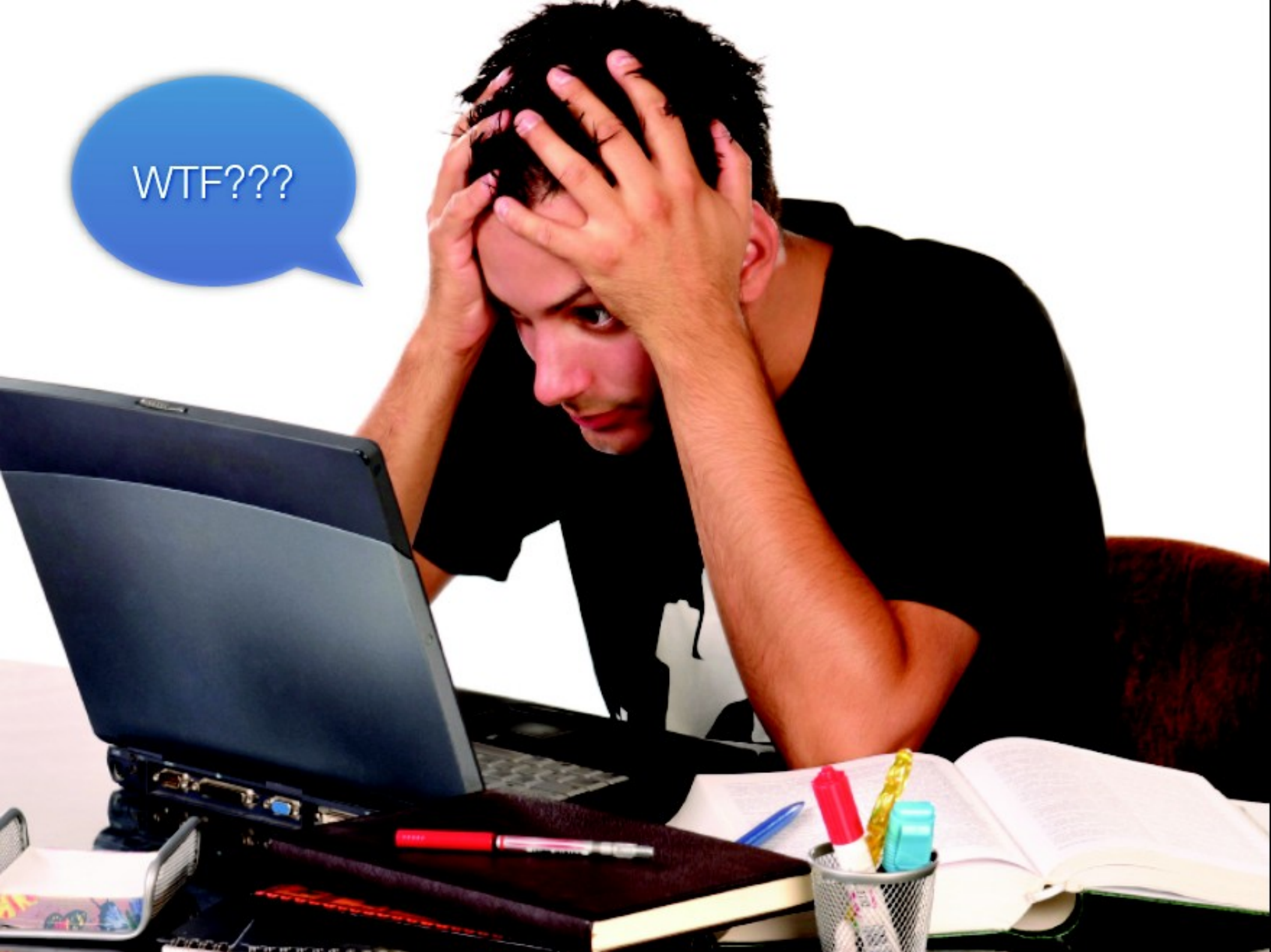
```
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
org.bar.foo.Flibble
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
javax.naming.InitialContext
javax.naming.NamingException
com.sun.internal.DontUseThisClass
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
org...jdbc.core.RowMapper
org...jdbc.core.SqlParameter
org...jdbc.support.GeneratedKeyHolder
com.foo.bar.Wibble
java.security.Principal
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
java.util.List
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
com.sun.internal.DontUseThisClass
com.foo.bar.Wibble
org.bar.foo.Flibble
javax.naming.InitialContext
javax.naming.NamingException
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
```

com.foo.bar.Wibble?

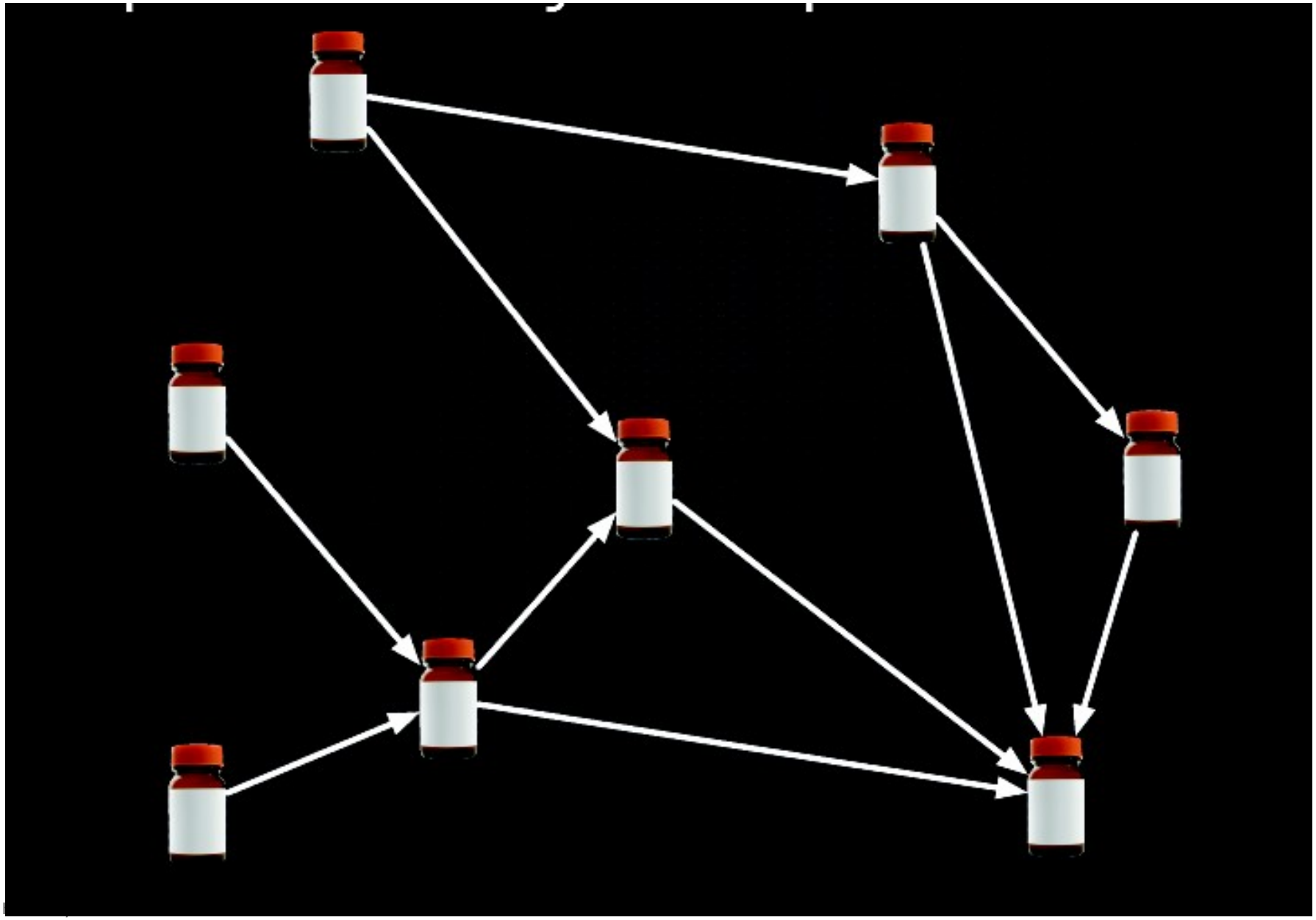
...But I Need The Other Class :(

```
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
org.bar.foo.Flibble
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
javax.naming.InitialContext
javax.naming.NamingException
com.sun.internal.DontUseThisClass
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
org...jdbc.core.RowMapper
org...jdbc.core.SqlParameter
org...jdbc.support.GeneratedKeyHolder
com.foo.bar.Wibble
java.security.Principal
javax.sql.DataSource
java.rmi.RemoteException
java.security.Principal
java.sql.ResultSet
java.sql.SQLException
java.sql.Types
java.util.List
javax.ejb.EJBException
javax.ejb.SessionBean
javax.ejb.SessionContext
javax.naming.Context
com.sun.internal.DontUseThisClass
com.foo.bar.Wibble
org.bar.foo.Flibble
javax.naming.InitialContext
javax.naming.NamingException
javax.sql.DataSource
org...jdbc.core.JdbcTemplate
org...jdbc.core.PreparedStatementCreator
```

com.foo.bar.Wibble?


WTF???

Dependency Graph

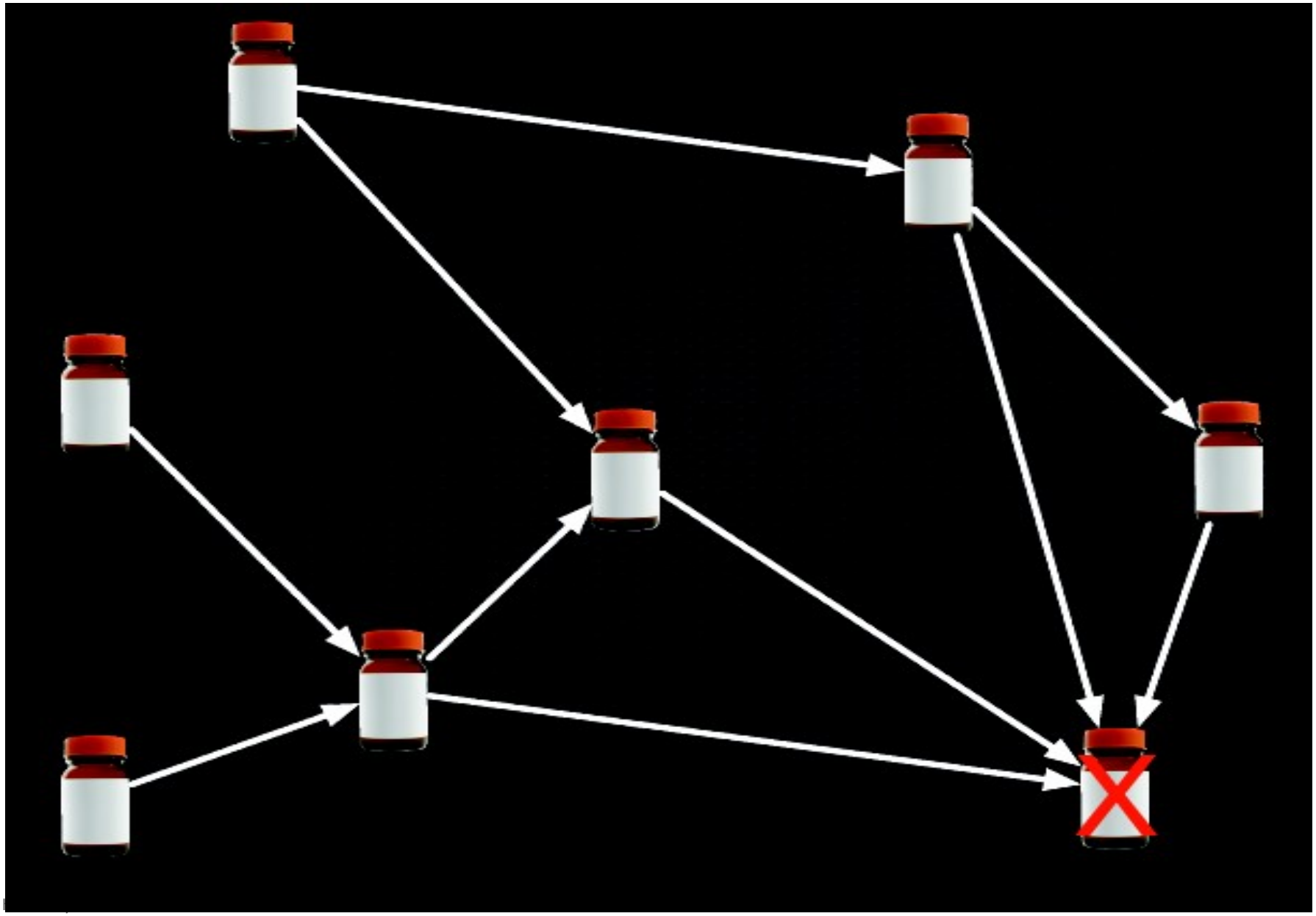


Still Something Wrong

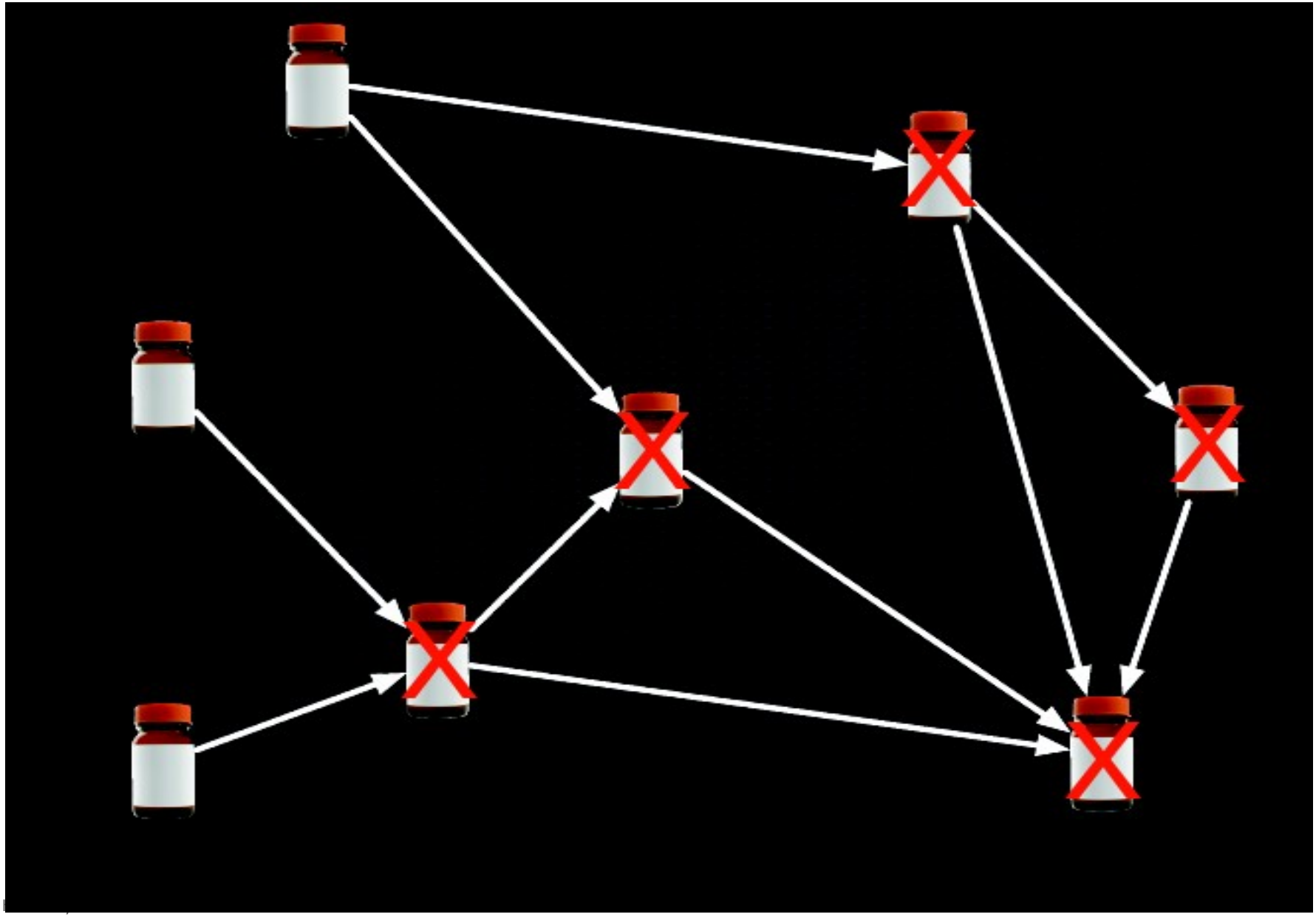
- Classes can be tightly coupled
- Even across module boundaries


FRAGILE

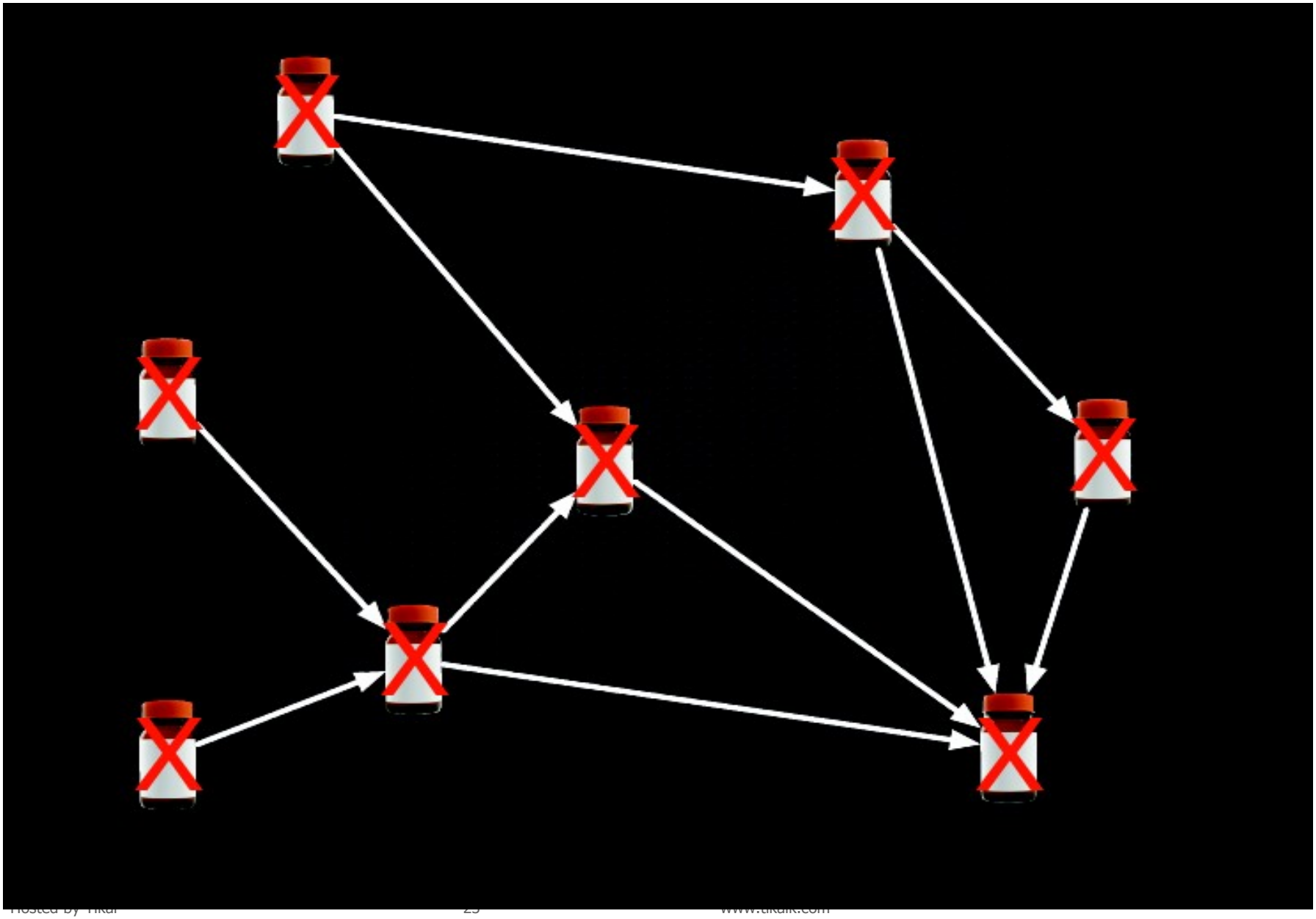
We Have a Problem…



...A Big One...



...It Crashed :(



What Is Missing ???

Dynamic!

Install bundles (plug-ins)

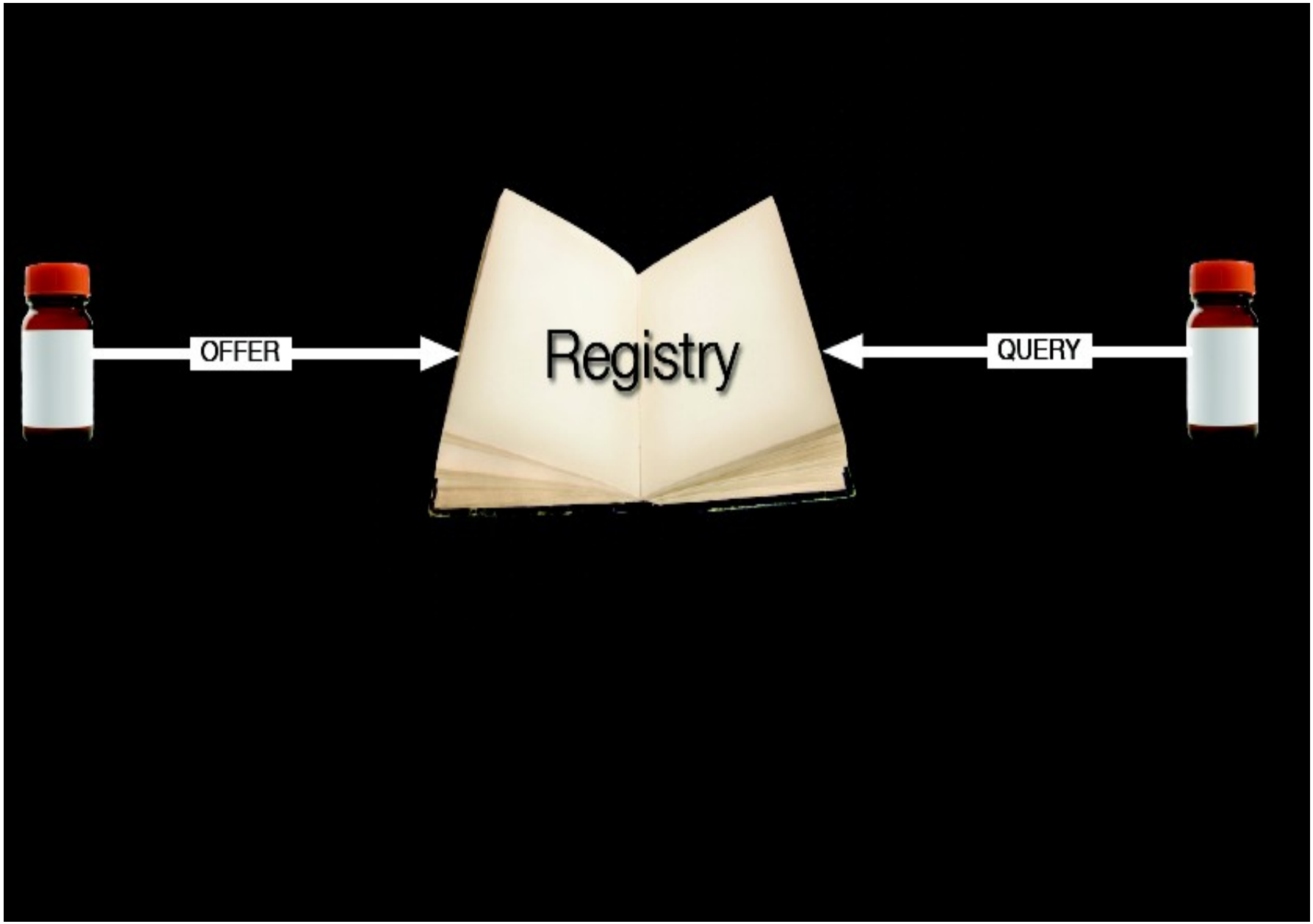
Update bundles

Remove bundles

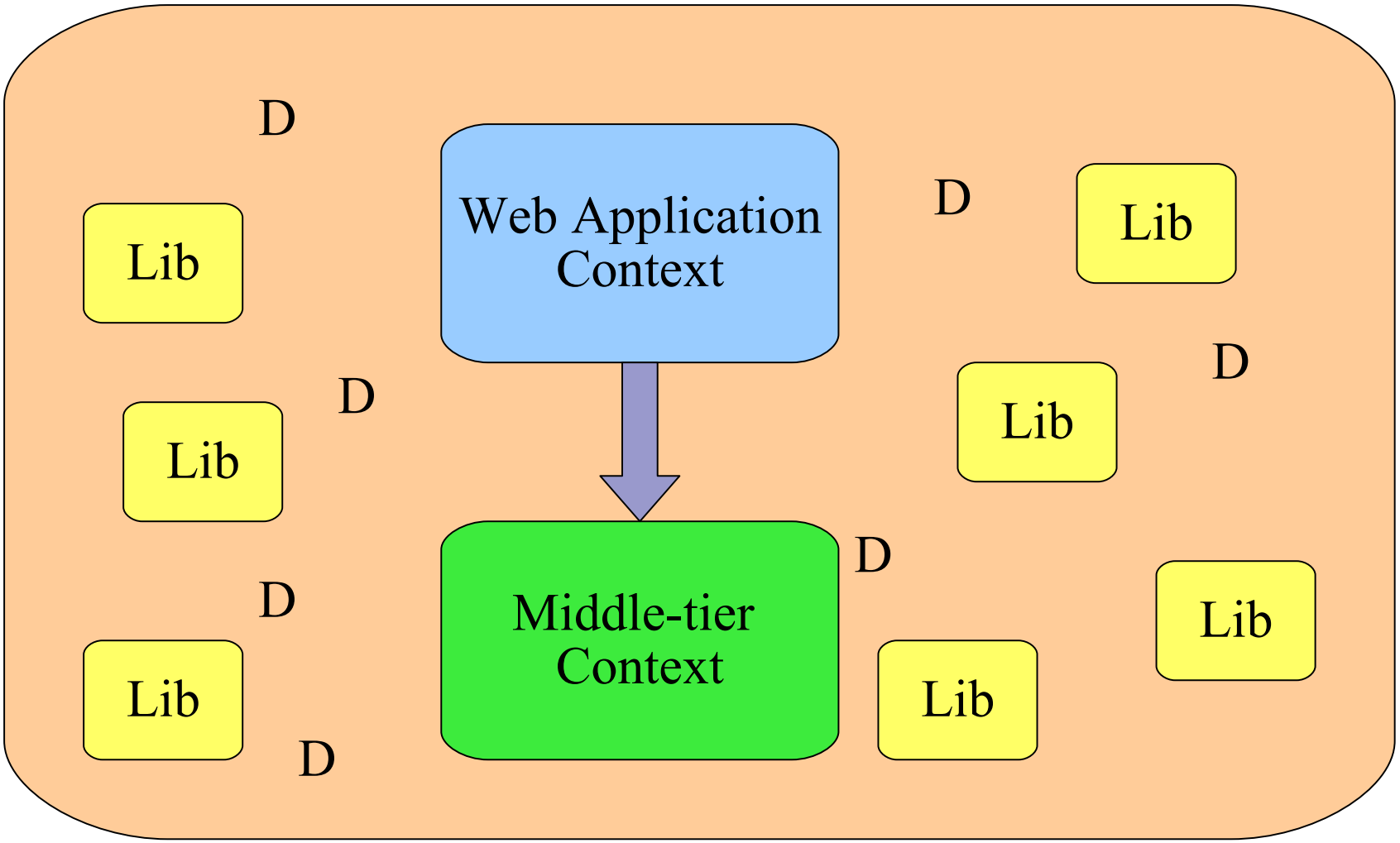
... all on-the-fly

Need to Decouple

Late Binding

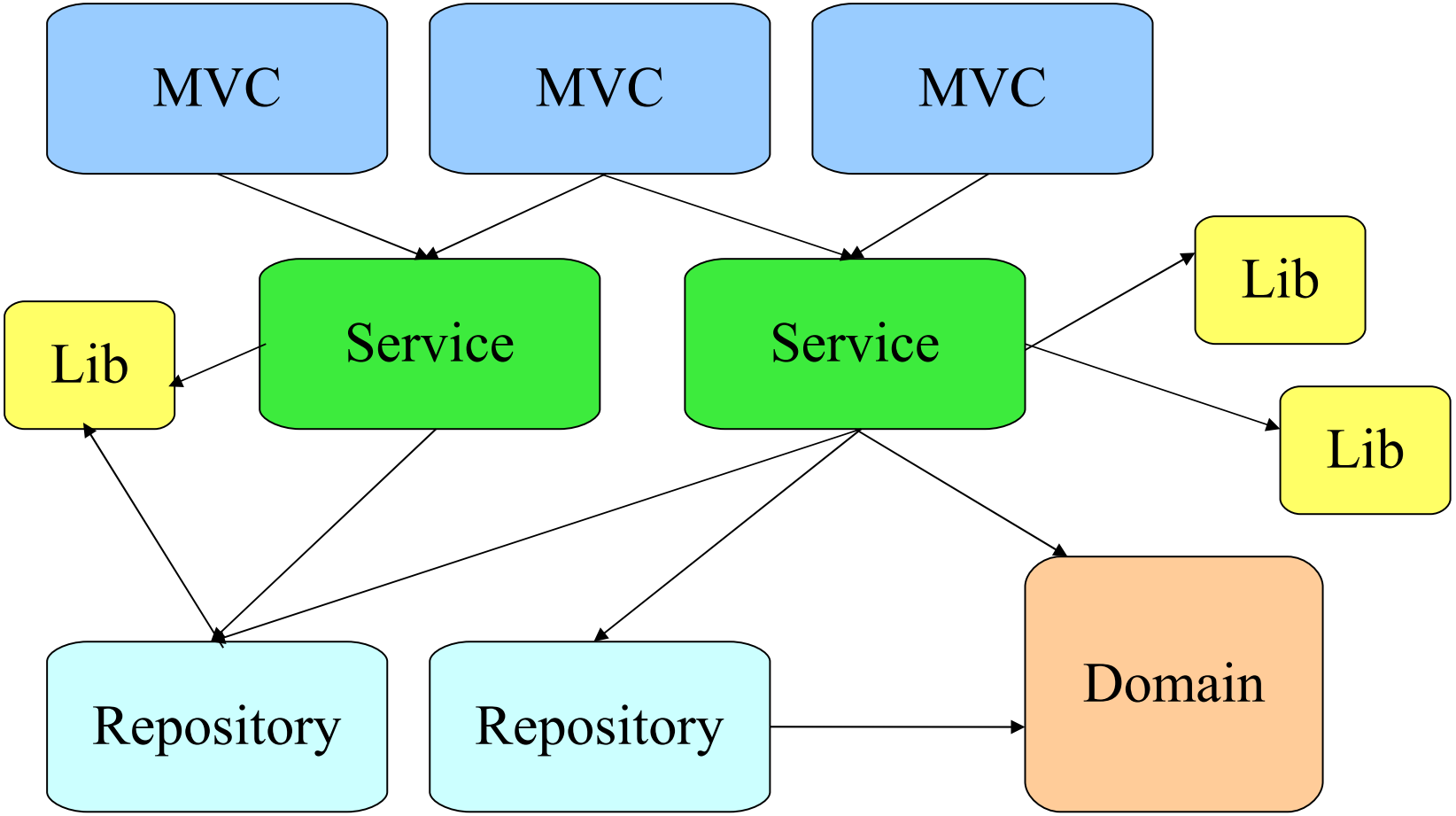


A Typical JEE Application



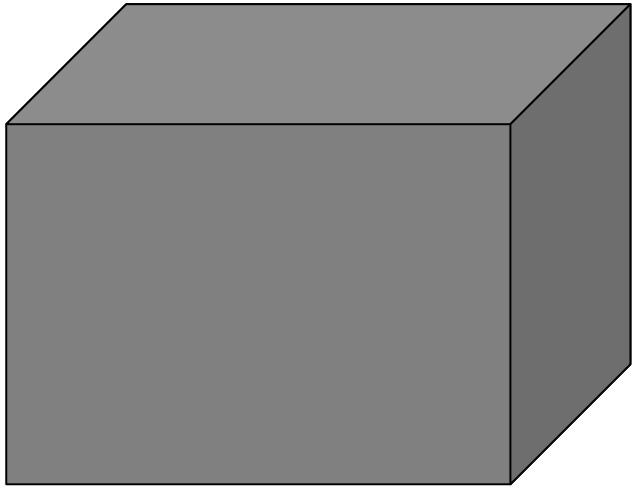
A Typical OSGi Application

Each bundle is a segregated class space

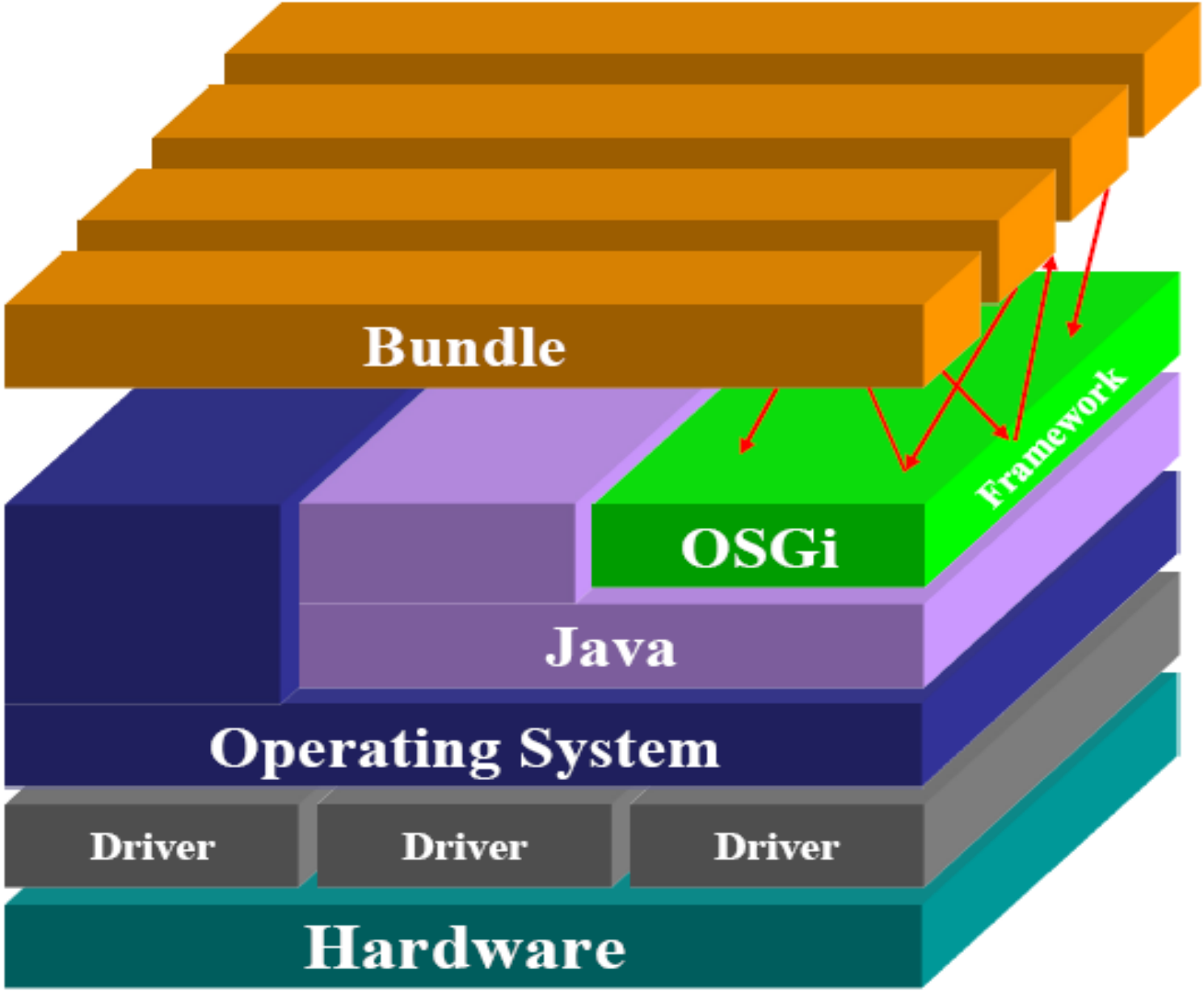


Bundle

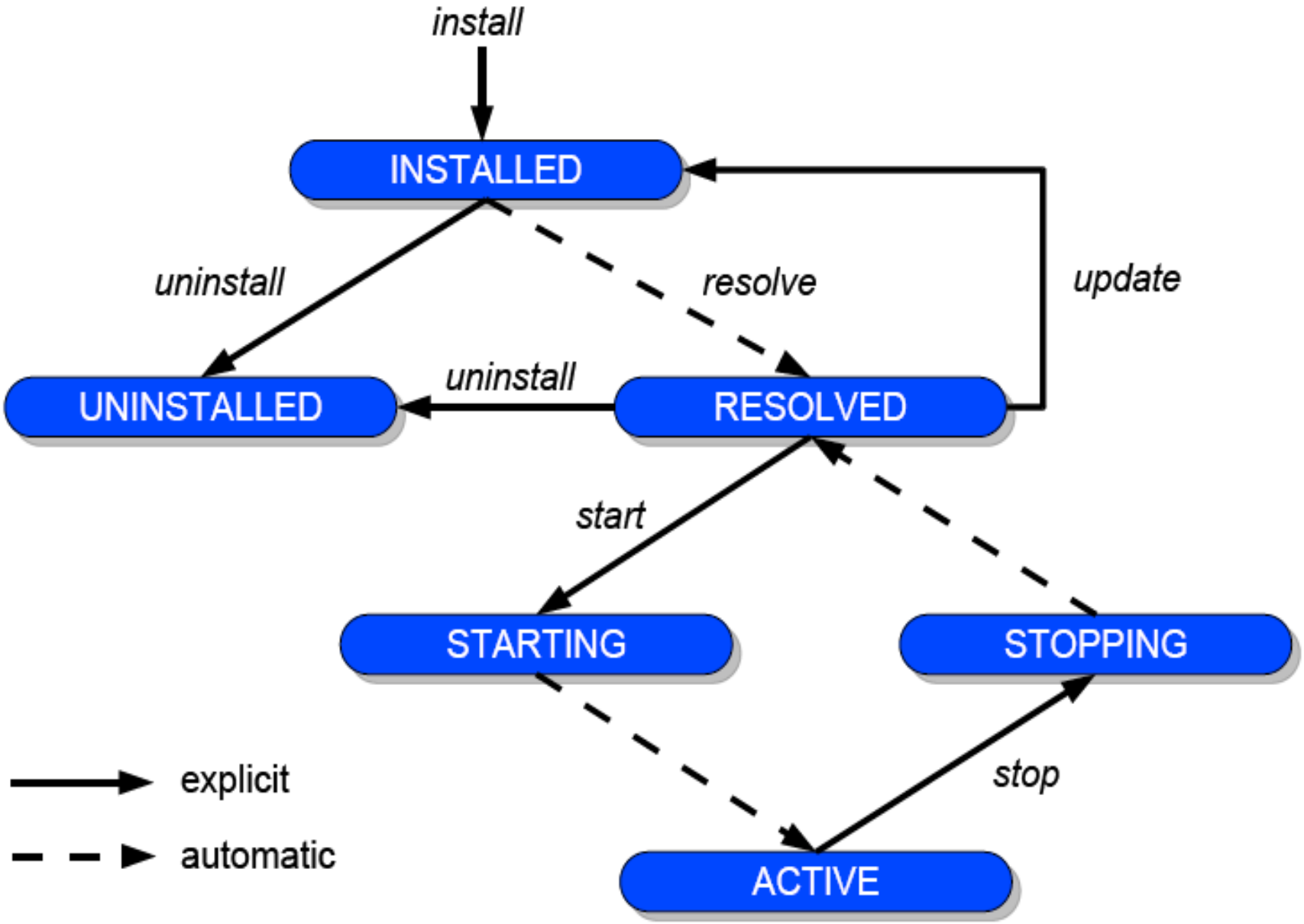
- By default a bundle is a black box
  - Completely protected
  - You can NOT see inside it
    - Not even by reflection
    - Or any other class loading trickery



OSGi Architectural



Bundle's Life Cycle- Dynamic !



Bundle Visibility

- Exposing Types can be done explicitly
  - A bundle can export one or more packages
  - Optionally with versioning information
- Only exported packages are visibly outside of the exporting bundle.
  - Stops unintended coupling between bundles
  - Enable independent development
  - Faster development cycles

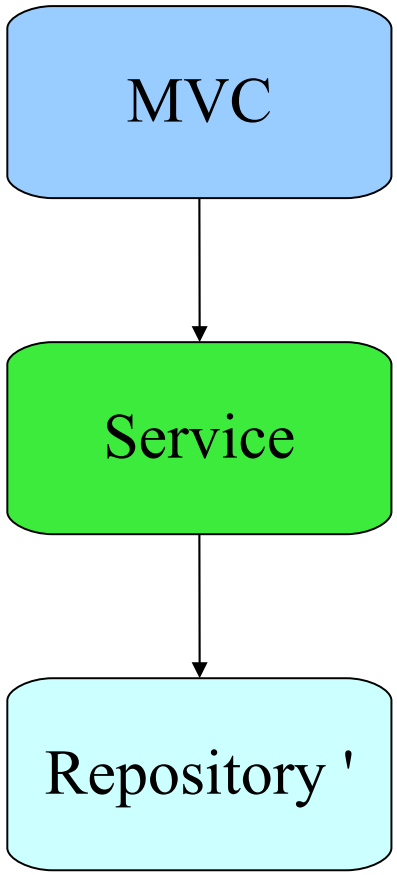
How Does It Look Like ?

- Its a jar file!
- No complicated tools or sophisticated packaging required.

META-INF/MANIFEST.MF

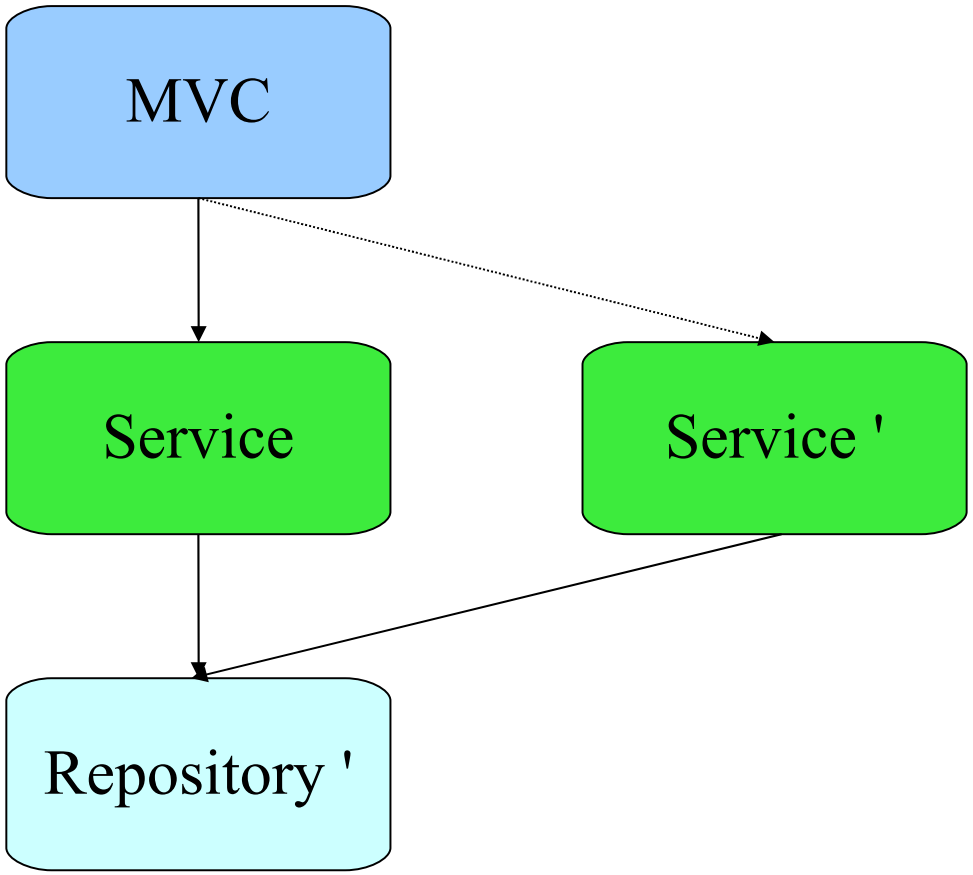
```
Bundle-Name: Hello World
Bundle-SymbolicName: com.tikal.osgi.sample.hellojbug


Bundle-Version: 2.1.0
Export-Package: com.tikal.osgi.sample
Import-Package: com.tikal.osgi.weather ,org.osgi.framework
Bundle-Activator: com.tikal.osgi.sample.HelloActivator
```

Versioning

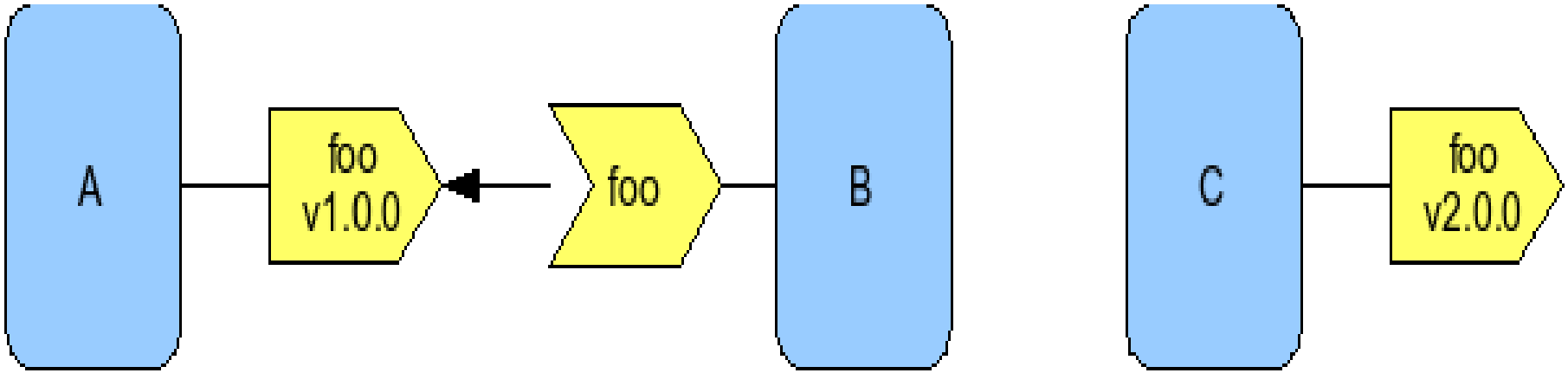


Versioning

Two versions of the same service type... at the same time!



Versioning




```
Export-Package: foo;
version="1.0.0"
```

```
Import-Package: foo;
version="[1.0,2.0)"
```

```
Export-Package: foo;
version="2.0.0"
```

Operational Control

- See all bundles and their status
  - OSGi console
  - JMX
- Get information on wiring
- Install new bundles
- Activate/Deactivate bundles (and publish services)
- Refresh bundles
- Stop bundles
- Uninstall bundles

All without stop or restart the application

OSGi Programming

- All access to OSGi is through BundleContext
- This is supplied to our BundleActivator

BundleActivator

```
public class HelloActivator implements BundleActivator {
  public void start(BundleContext context) throws Exception {
    System.out.println("Hello, world!");
    System.out.println(Arrays.asList(context.getBundles()));
  }
  public void stop(BundleContext context) throws Exception {
    System.out.println("Goodbye, world!");
  }
}
```

Hello-JBUG DEMO

Extender vs Service


“I made this specially for *you*”

OSGi Extender Model

Service interface types exported with version information

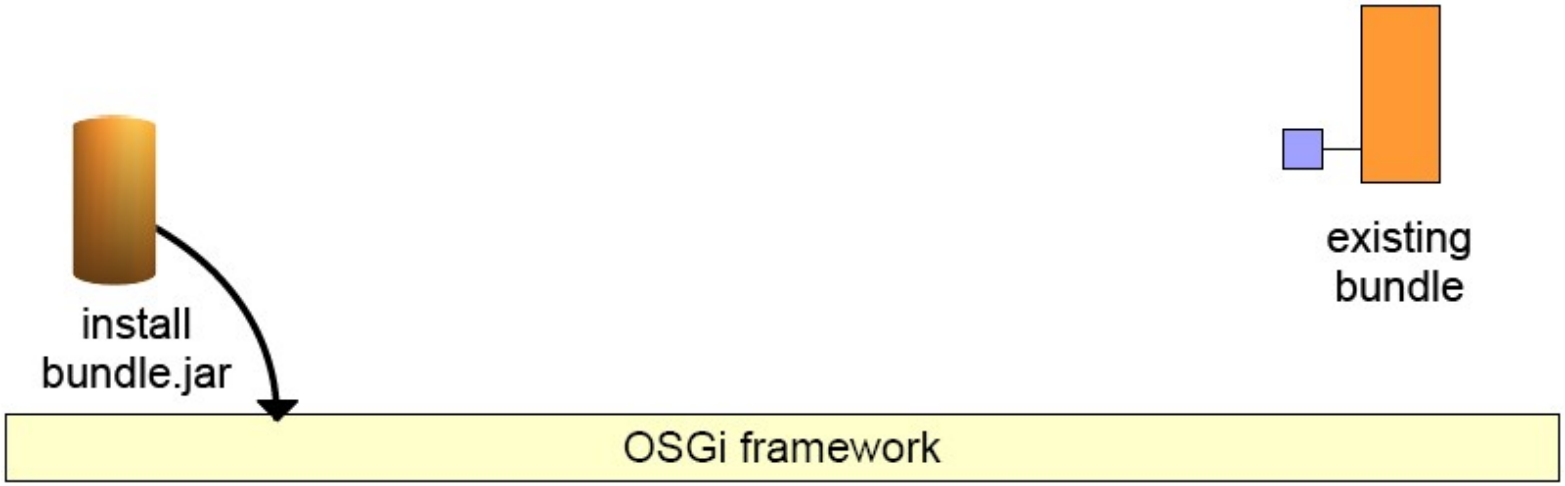
Export-Packages: a,b,c

private packages

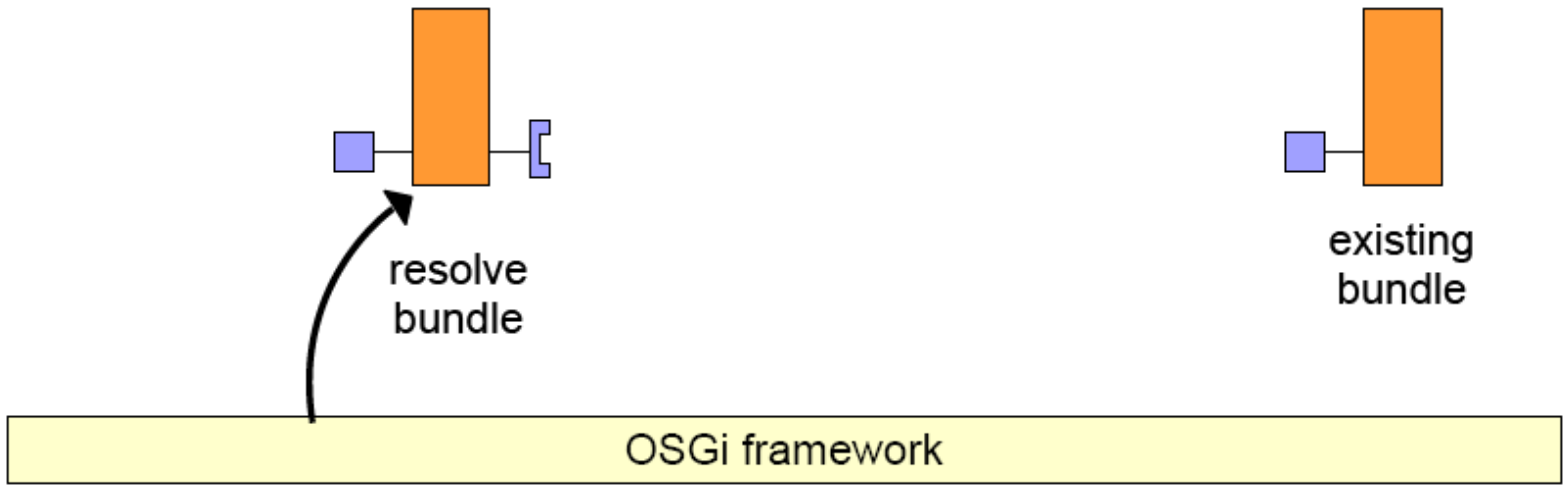
Service implementation locked away

"Passive" contribution
Types added to type space bundles see new version on resolution after install/refresh

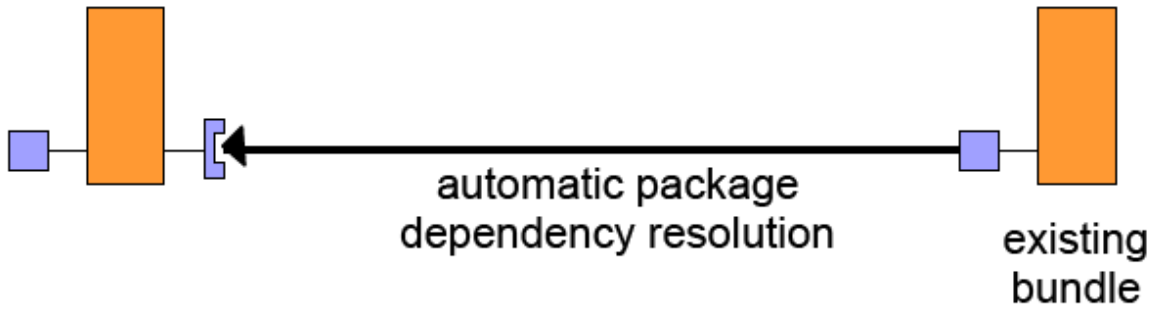
OSGi Extender Model



OSGi Extender Model



OSGi Extender Model

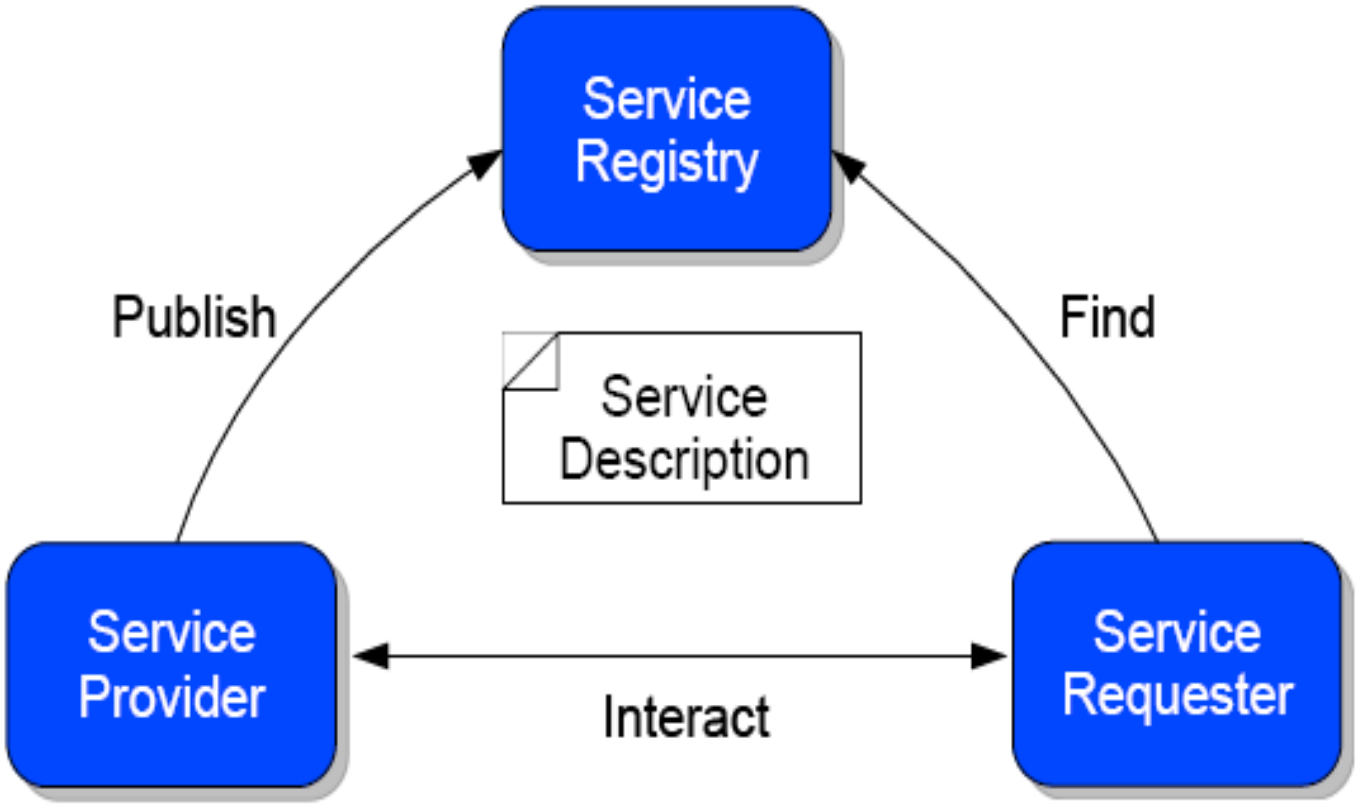


OSGi framework


“I made these. You
can use them.”

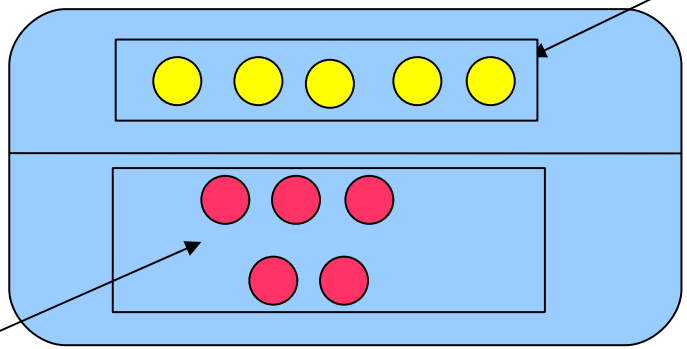
SOA Architecture

- The OSGi framework promotes a service-oriented interaction pattern among bundles



SOA For The JVM

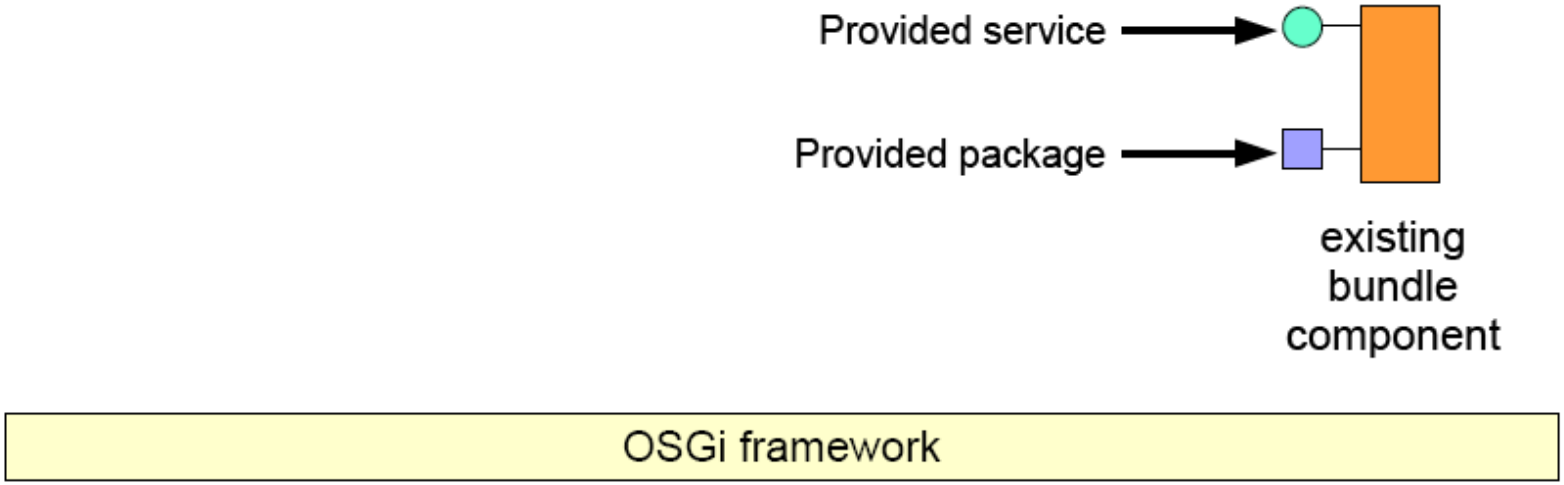
Published Services



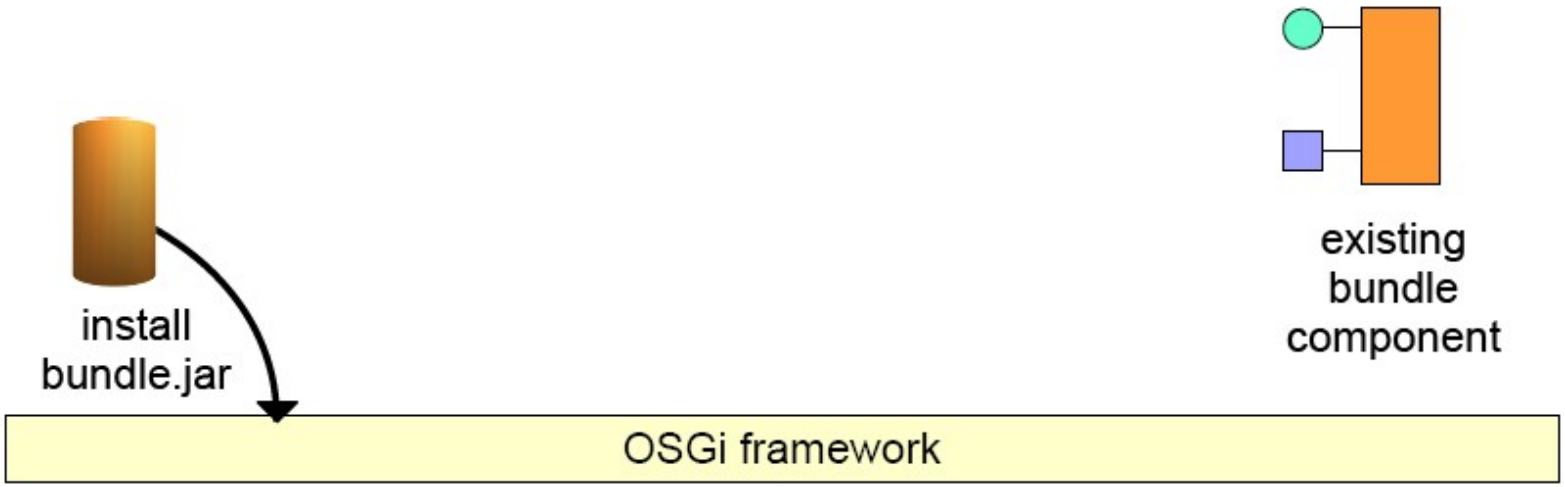
Private implementation objects

"Active" contribution
Services published in registry bundles see service changes immediately

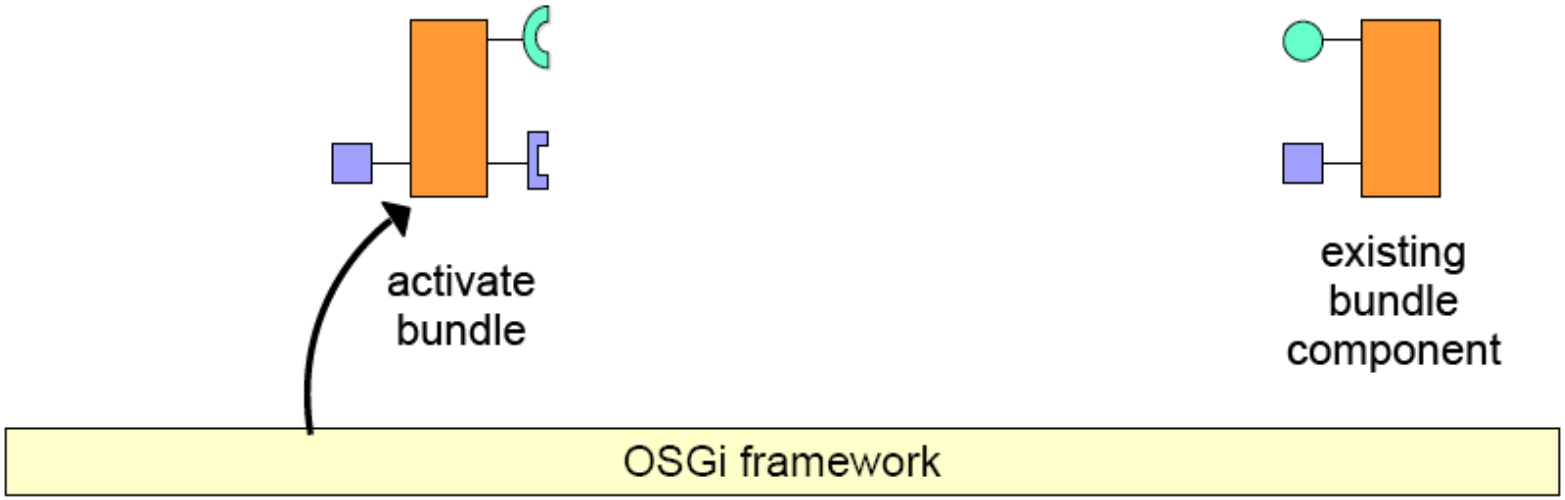
SOA in OSGi


Provided service
Provided package
existing bundle component
OSGi framework

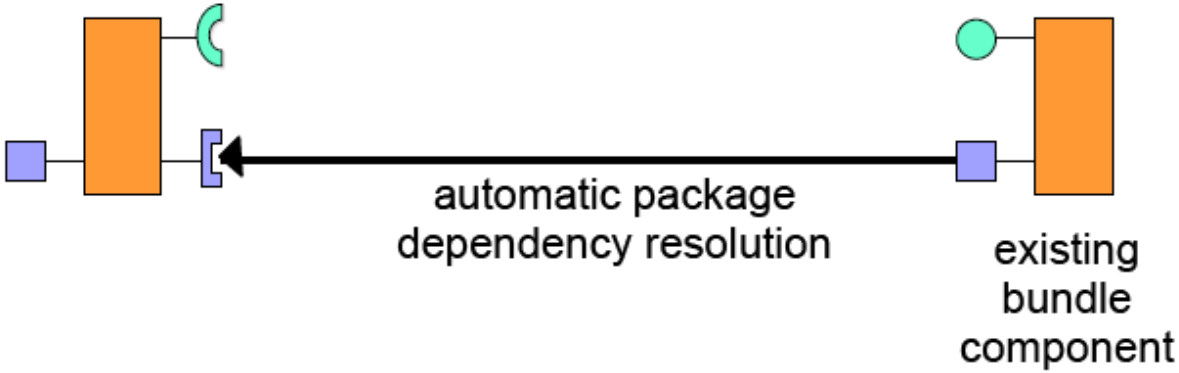
SOA in OSGi


install
bundle.jar
existing
bundle
component
OSGi framework

SOA in OSGi

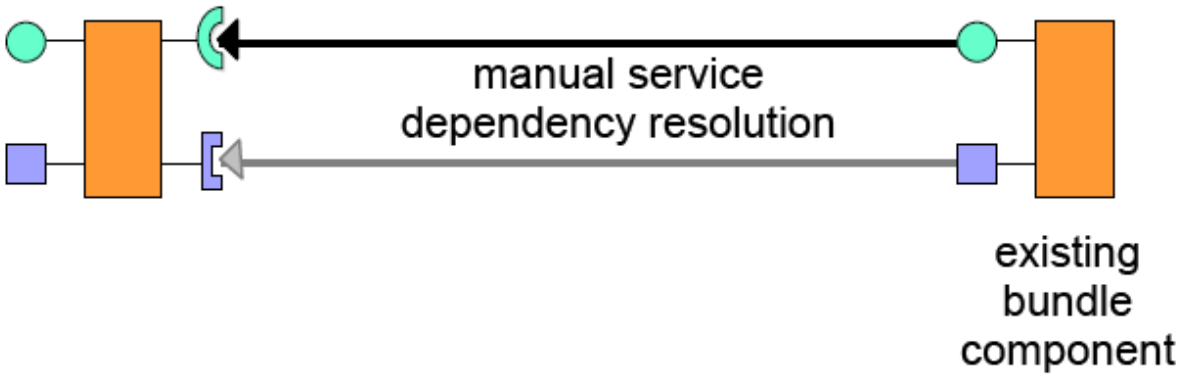

activate
bundle
existing
bundle
component
OSGi framework

SOA in OSGi



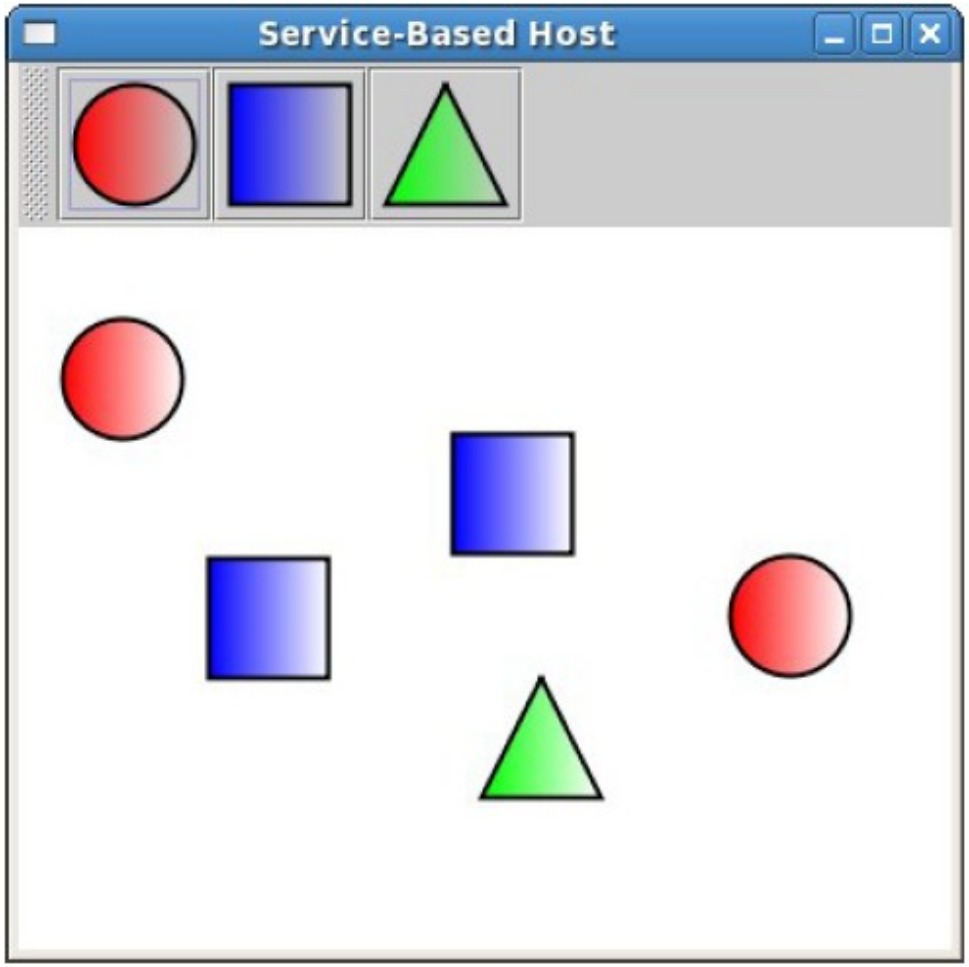
OSGi framework

SOA in OSGi

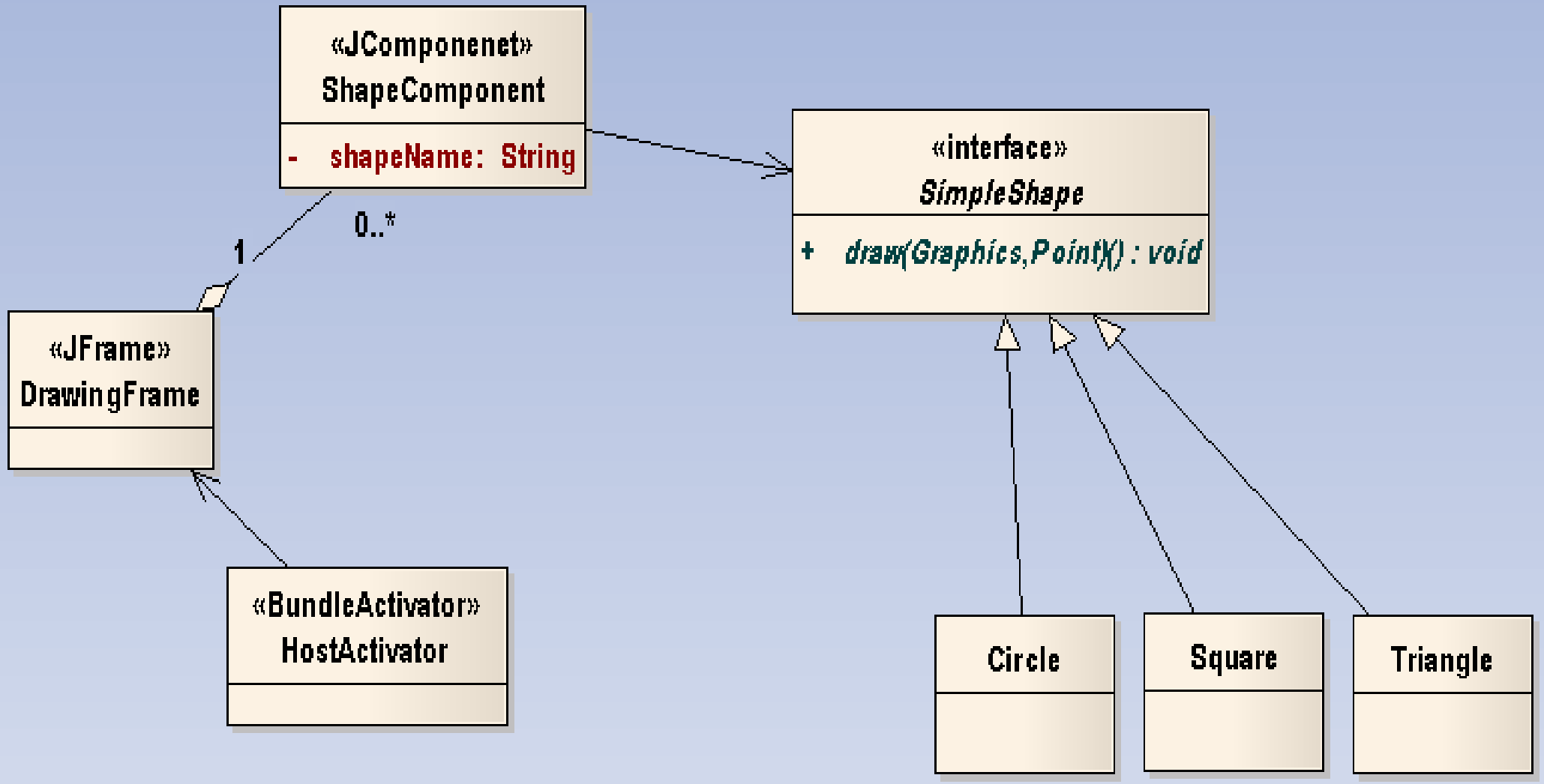


OSGi framework

Example: Paint Program



Naive Class Diagram



Register And Consume

- Register our Shape service is easy:

```
Hashtable dict = new Hashtable();
        dict.put(SimpleShape.NAME_PROPERTY, "Square");
        dict.put(SimpleShape.ICON_PROPERTY, new
ImageIcon(getClass().getResource("square.png")));
context.registerService(SimpleShape.class.getName(),
                                                new Square(),dict);
```

- Consuming the service can be...


Slippery

First Attempt

```
ServiceReference ref = context.getServiceReference(
                                    SimpleShape.class.getName());
SimpleShape shape = (SimpleShape) context.getService(ref)
shape.draw();
```

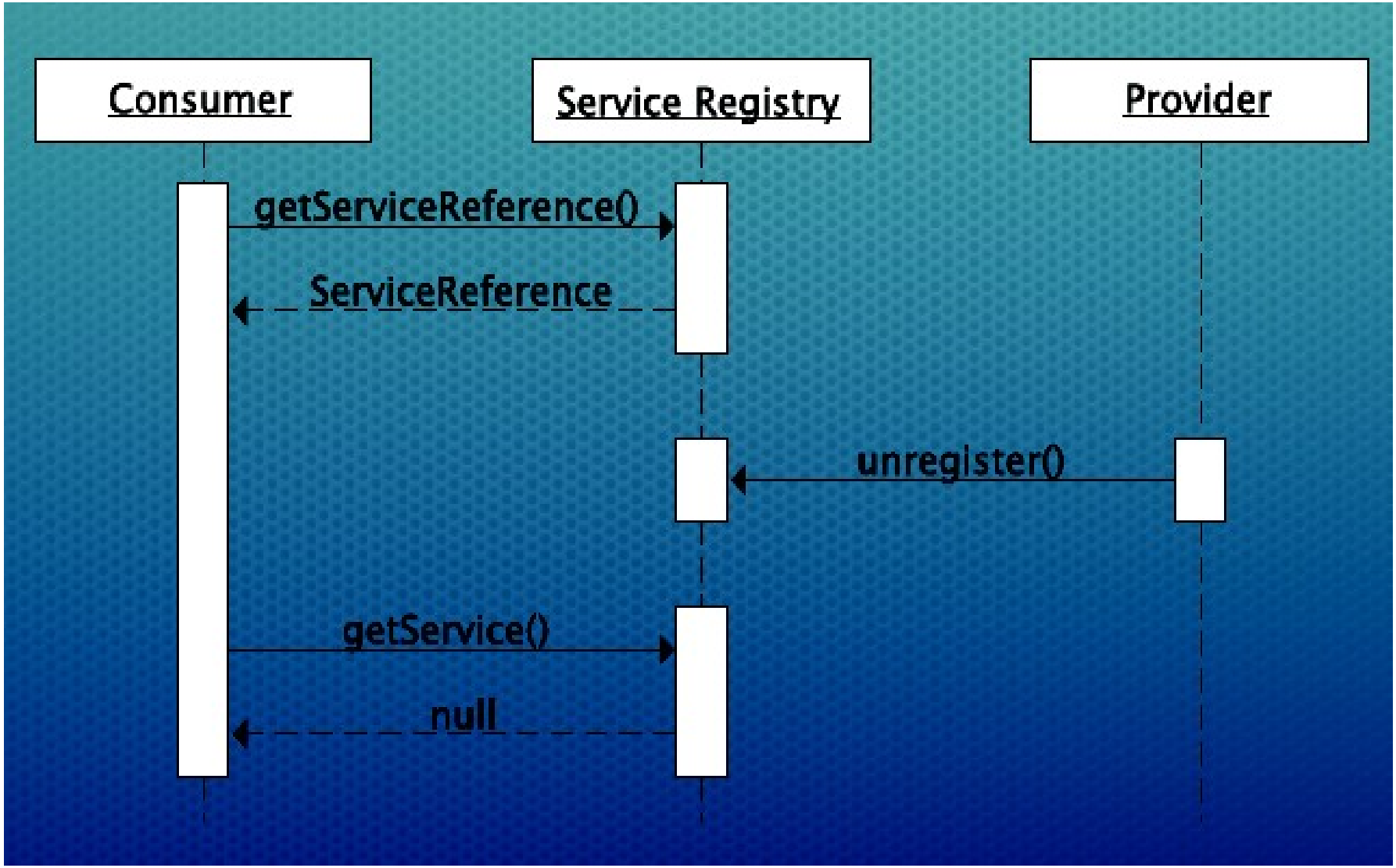
Oops, “ref” can be Null

```
ServiceReference ref = context.getServiceReference(
                                    SimpleShape.class.getName());

if(ref != null) {
  SimpleShape shape = (SimpleShape) context.getService(ref)
  shape.draw();
}
```

Oops, “shape” can be Null

```
ServiceReference ref = context.getServiceReference(
                                    SimpleShape.class.getName());

if(ref != null) {
  SimpleShape shape = (SimpleShape) context.getService(ref);
  if(shape != null)
    shape.draw();
}
```

How ???


Consumer
Service Registry
Provider
getServiceReference()
ServiceReference
unregister()
getService()
null

Oops, Need to Clean Up

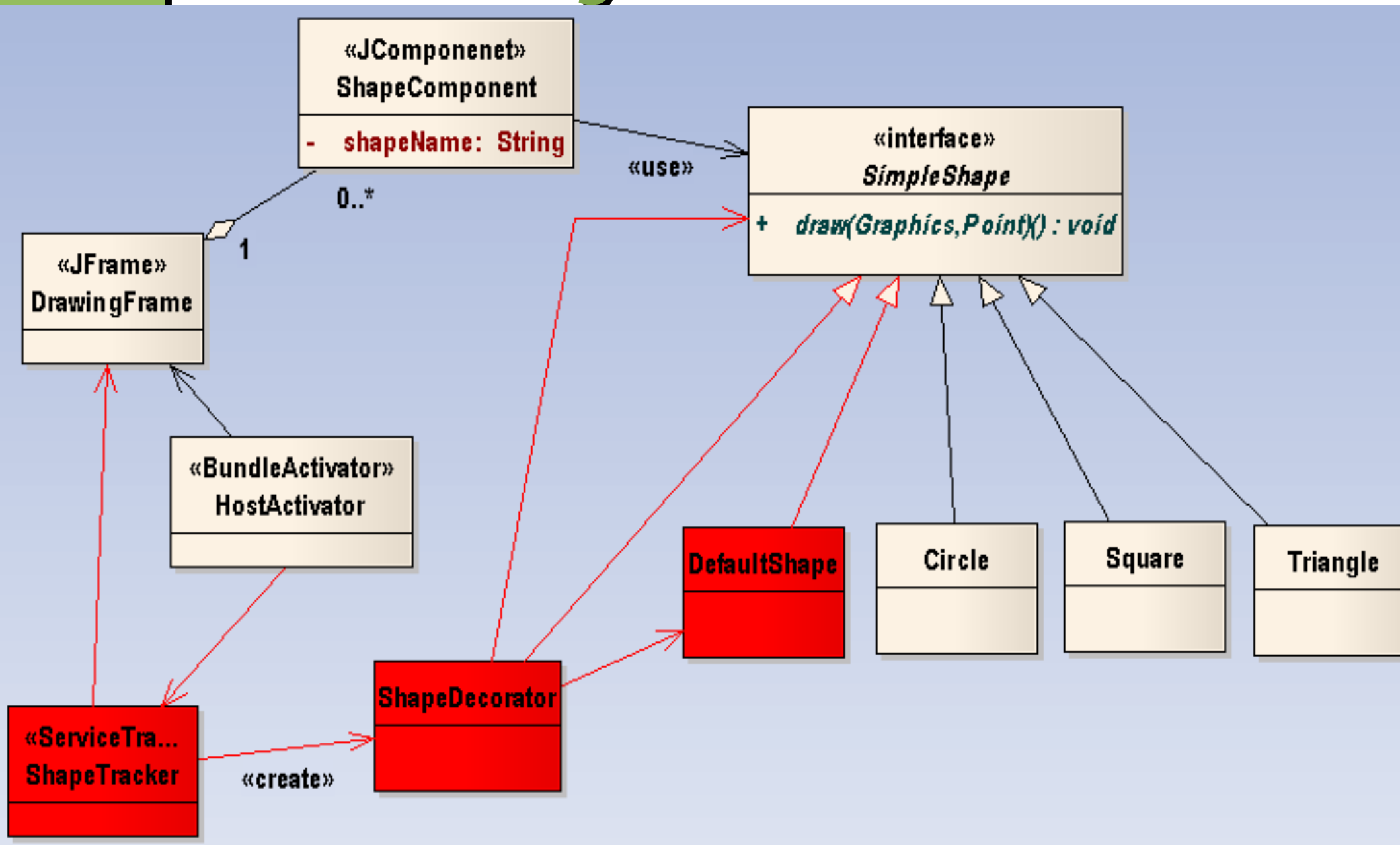
```
ServiceReference ref = context.getServiceReference(
                                    SimpleShape.class.getName());

if(ref != null) {
  SimpleShape shape = (SimpleShape) context.getService(ref);
  if(shape != null){
    try{
      shape.draw();
    }finally{
      context.ungetService(ref);
    }
}
```


Yuck!

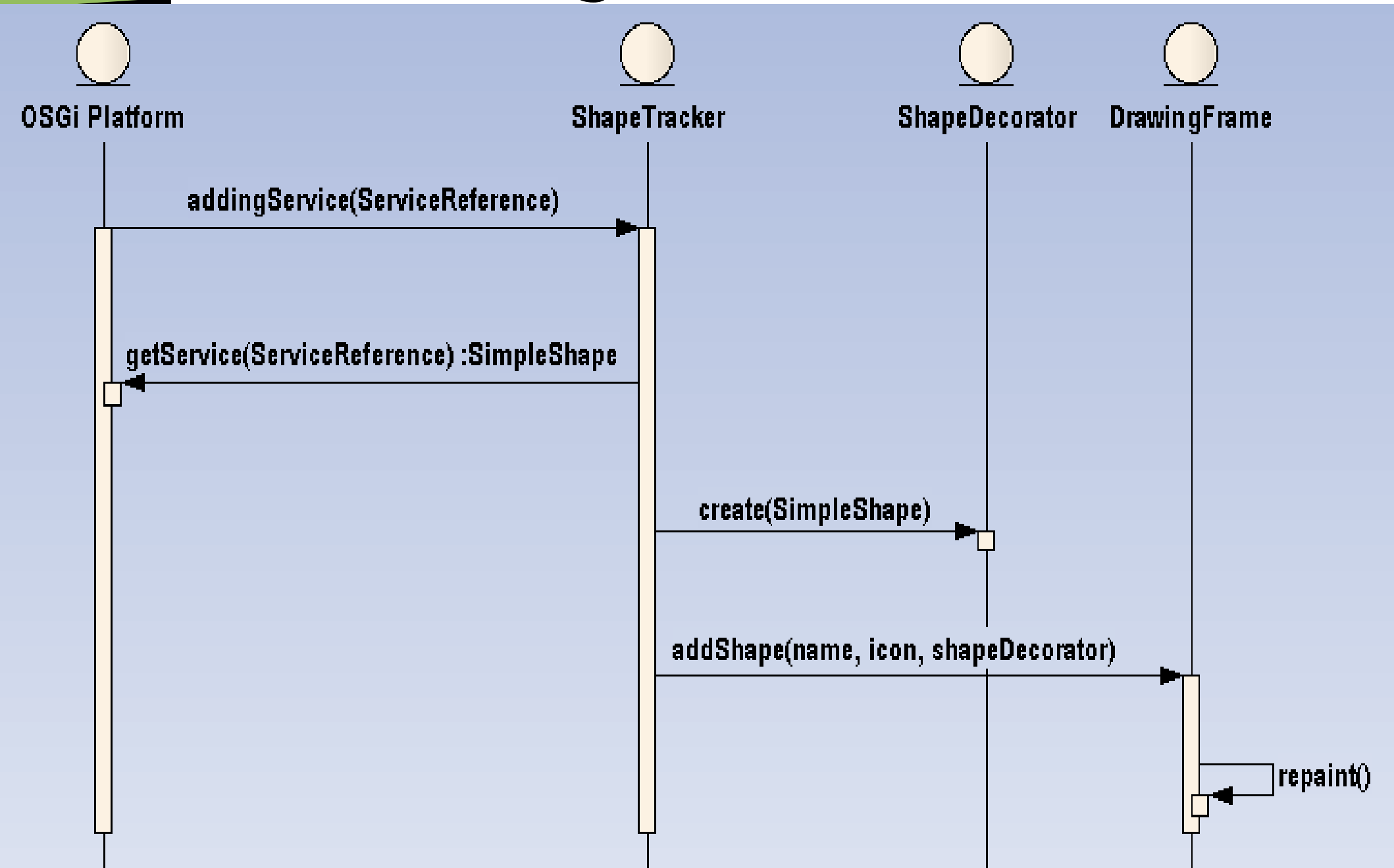
ServiceTracker

- Shape Tracker goal – Use Inversion of Control principles to inject shapes into application
  - Created by the BundleActivator
  - Puts tracking logic in one place
  - Isolates application from OSGi API
- User “Observer” design pattern to track services for clients.

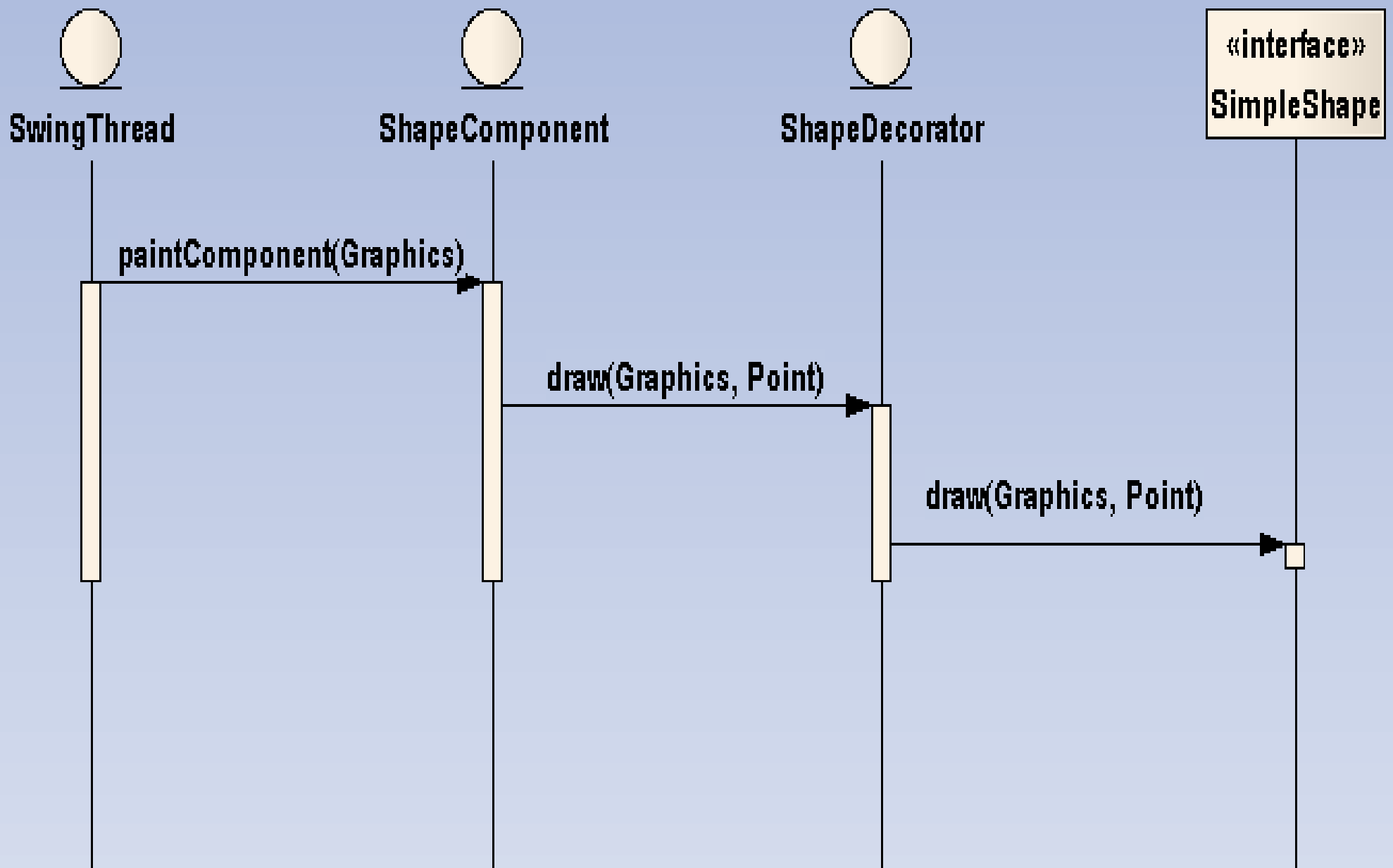
Connecting The Dots



Connecting The Dots



Connecting The Dots



Shapes DEMO

SOA Application Advantages

- Direct method invocation
- Structured code
  - Promotes separation of interface from implementation
  - Enables reuse, loose coupling, and late binding
- Dynamics
  - Support run-time management of modules
- Configuration is defined by the deployed bundles
  - Just deploy the bundles that you need

SOA Application Issues

- Complicated
  - Requires a different way of thinking
    - Services might not be there or go away at any moment
  - Must manually resolve service dependencies
  - Must track and manage service dynamics
  - Service Tracker can help but sill somewhat of a manual approach
- There is a declarative alternative…

JEE Application on OSGi ?

- How can we exploit OSGi
  - Without adding complexity ?
  - Retaining our programming model?
- We want to treat bundles as beans
  - Instantiated, Configure, Assemble, Decorate
  - We want to easy way to publish/consume services
    - Dynamic management
  - Should we code this ourselves?
  - Preserve ability to test

Spring DM

Spring-DM Key Features

- Integrates the simplicity and power of Spring...
- ...with the dynamic module system of OSGi
- Use spring container to configure bundles
- Make it easy to publish/subscribe services
- Application can be coded with minimal OSGi code
  - Also easy test/integration-test

POJO development on OSGi

Spring-DM Building Blocks

- NO OSGi code & NO Spring code
  - No BundleActivator
  - No ServiceTracker
  - No Spring invasive code
- A Spring application context based on OSGi bundle.
  - OsgiBundleApplicationContext
- Enables UnitTest and Integration Test

Bring POJO to OSGi Services

```
<bean name="bookDao" class="com....BookDaoImpl" />
    <property name="datasource" ref="datasource"/>
</bean>
```

```
<osgi:service id="bookDaoOsgi" ref="bookDao"
                                interface="com.tikal...BookDao" />
```

```
<bean name="bookService" class="com.tikal....BookServiceImpl">
  <property name="bookDao">
    <osgi:reference interface="com.tikal...BookDao" />
  </property>
</bean>
```

Spring-DM DEMO

Cardinality

- What happens if...?
  - » There isn't a matching service
  - » There are several matching services?
  - » A matched service goes away at runtime?
  - » New matching services become available at runtime?
- The osgi:reference element has cardinality attribute
  - » 0..1, 0..n , 1..1, 1..n
  - » default = 1..1

Looking Forward

- How do we do AOP in OSGi
- Integration with existing frameworks
  - SWF
  - Spring MVC
- Support for web application
  - Problem : In Traditional web apps there is no notion of bundle space or imported packages in a web application.
  - Solution: Bridging the web container and the OSGi space by Spring-DM.

JBoss Meet OSGi

- Have an OSGi based ClassLoader on JBoss
  - First will be done for runtime (services)
  - Later, will be Introduced for application developers.
- Create a full OSGi core spec v4.1 implementation.
  - Missing features in current implementations: AOP, legacy JMX, VFS etc.
- Benefits for users:
  - Classic JEE application developers currently won't see any benefits for now.
  - For service developers OSGi support will be provided out-of-the-box.

Can I Use OSGi ?

Still waiting on the sidelines

Summary

- OSGi is a dynamic module system for Java
  - Proven
  - Scalable (up and down)
- OSGi Offer benefits in terms of
  - Modularity (visibility)
  - Versioning
  - Operational Control
- Spring OSGi combines the simplicity and power of Spring with sophistication of OSGi platform.

Q & A

Thank You

yanai@tikalk.com