OSGi

Tales from the Trenches

Bertrand Delacretaz
Senior R&D Developer, Day Software, www.day.com
Apache Software Foundation Member and Director
bdelacretaz@apache.org
blog: http://grep.codeconsult.ch
twitter: @bdelacretaz

ApacheCon Europe 2009, Amsterdam
slides revision: 2009-03-25

Special thanks to:
Filippo Diotalevi (osgilook.com)
and The Day R&D team
for additional input!

What?

Share our experience using Apache Felix at Day, for a major rewrite of our content management products.

More than two years working with OSGi, very high impact on developers, customers, service people, *mostly* in a positive way.

OSGi is no silver bullet either.

Ok but what?

the

GOOD 

the

BAD 

the

UGLY 

and the

FUTURE

symbols by ppdigital , o0o0xmods0o0oon and clarita, on morguefile.com

Introduction

First a warning

is **not** an OSGi guru!

I'm just a poor lonesome user...

OSGi ?

OSGi™

The **Dynamic Module System** for Java™

http://www.osgi.org (see also Wikipedia)

Consortium founded 1999, 100 companies.

Initially meant for **mobile devices**.

Now moving to **server-side**, fast.

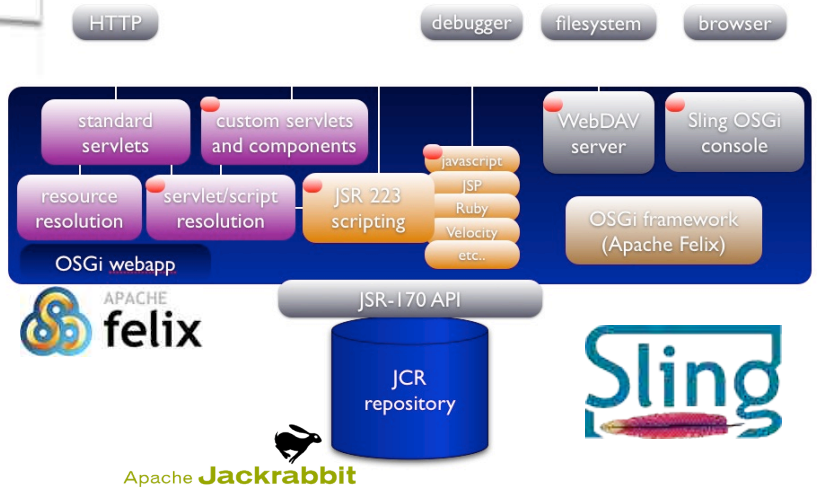
Eclipse, LinkedIn, GlassFish, WSO2, WebSphere and many others.

@apache: Felix, ServiceMix, Sling, CXF, Tuscany and more, growing.

Trenches?

Day

family of content management products, R&D team of about 30



built on Apache Sling, uses Apache Felix and Jackrabbit

http://www.day.com/cq5
http://jackrabbit.apache.org
http://felix.apache.org
http://incubator.apache.org/sling

What we use from OSGi

Bundles (using Maven plugins)

Lifecycle, Service Tracker

Configurations and Felix Web Console

Declarative Services (using Maven plugins)

Sling's **jcrinstall** module

(bundles and configs loaded from the JCR repository)

Log, HTTP, Event services

presented by Carsten at 10:30

Famous quotes

Famous Quotes, #1

“Effectively, OSGi brings many of the desirable aims of SOA into the JVM."”

Paul Fremantle
http://pzf.fremantle.org/2009/02/wso2-carbon-part-1.html

Famous Quotes, #2

"Each (OSGi) bundle can serve as a micro application, having it's own lifecycle, having it's own citizens and each bundle can carefully decide which objects to expose to the outside world"

Peter Rietzler
http://peterrietzler.blogspot.com/2008/12/is-osgi-going-to-become-next-ejb-bubble.html

Famous Quotes, #3

"OSGi is great, but the tooling is not quite there yet. Not every library is a bundle and many JARs don't have OSGi manifests"

Matt Raible in
http://blog.linkedin.com/2008/06/23/osgi-at-linkedin-integrating-spring-dm-part-1/

Famous Quotes, #4

"The lifecycle model of OSGi makes life complicated. Actually, tracking services and managing all the aspects of what to do when services come and go is nasty"

Peter Rietzler
http://peterrietzler.blogspot.com/2008/12/is-osgi-going-to-become-next-ejb-bubble.html

Famous Quotes, #5

"The challenge for the coming year is to make OSGi more in line with the expectations of the average J2EE programmer because we see that need"

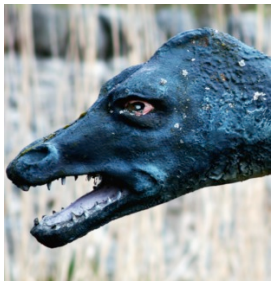
Peter Kriens in a comment at http://peterrietzler.blogspot.com/2008/12/is-osgi-going-to-become-next-ejb-bubble.html

Famous Quotes, #6

*"OSGi makes "**impossible**" things **easy**: hot deploy/upgrade, service discovery, ... and **trivial** things **hard**: hibernate, tag libraries, even deploying a simple war!" But, for the first time in my career, I see software reusability that works: **service reusability**.*

Filippo Diotalevi

Our opinion

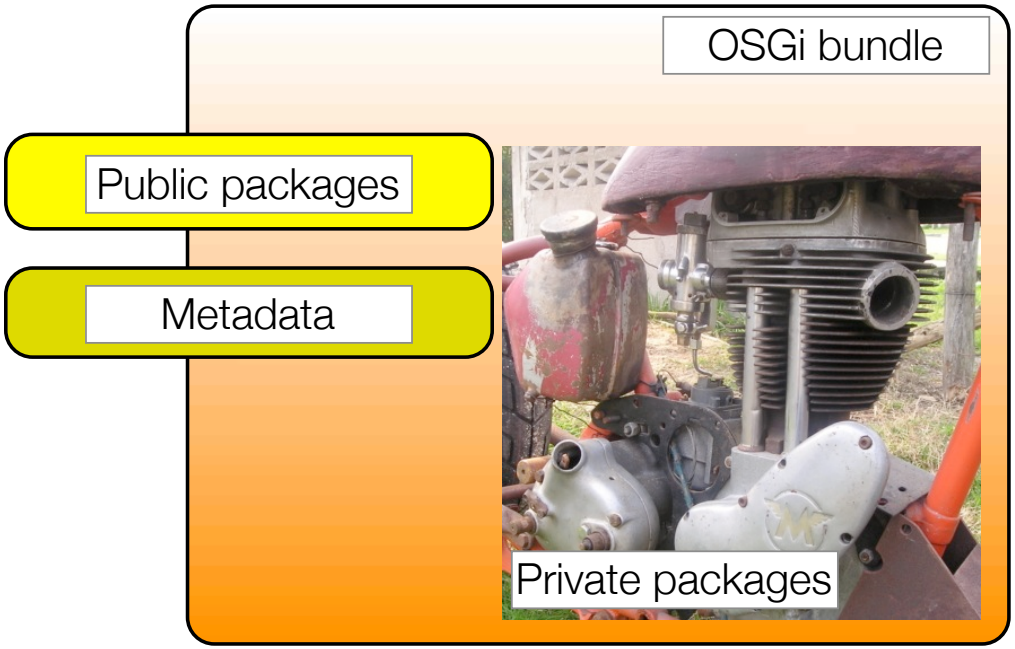


The Good

Modularity

Classloading distinct from class visibility.

Bundles as reusable components. At last!



Matchless picture: Alvimann on morguefile.com

Declarative Services

```
/**
* Excerpts from a Sling Servlet, processes *.query.json requests
* Uses Felix's maven-scr-plugin annotations
* @scr.component immediate="true"
* @scr.service interface="javax.servlet.Servlet"
* @scr.property name="sling.servlet.resourceTypes" value="sling/servlet/default"
* @scr.property name="sling.servlet.extensions" value="json"
* @scr.property name="sling.servlet.selectors" value="query"
*/
public class JsonQueryServlet extends SlingSafeMethodsServlet {
  /** @scr.reference (injected by framework) */
  private SlingRepository repo;

  // activate(ComponentContext) and deactivate(ComponentContext)
  // methods are called by framework if present
  ...
}
```

Annotated Java class +
Maven plugins == OSGi service

Clean API

installBundle

```
public Bundle installBundle(java.lang.String location,
                            java.io.InputStream input)
                     throws BundleException
```

getBundles

```
public Bundle[] getBundles()
```

Returns a list of all installed bundles.

addServiceListener

```
public void addServiceListener(ServiceListener listener,
                               java.lang.String filter)
                        throws InvalidSyntaxException
```

update

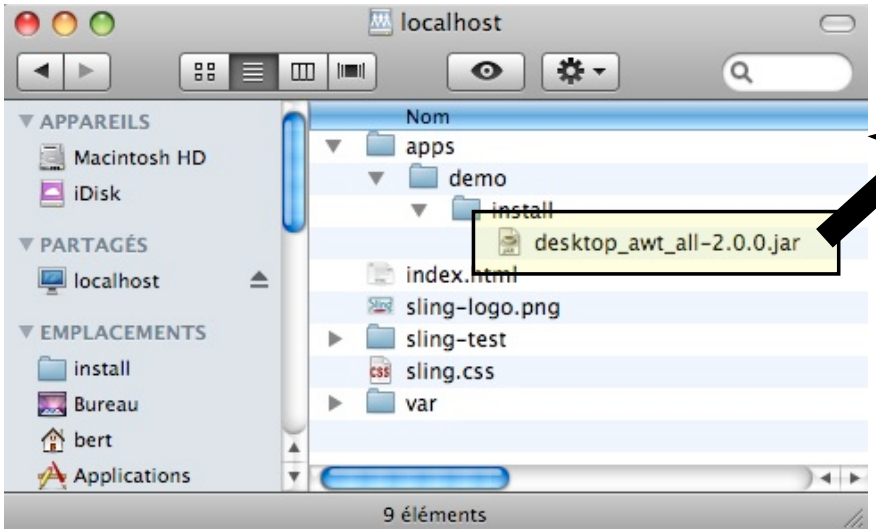
```
public void update(java.io.InputStream in)
                   throws BundleException
```

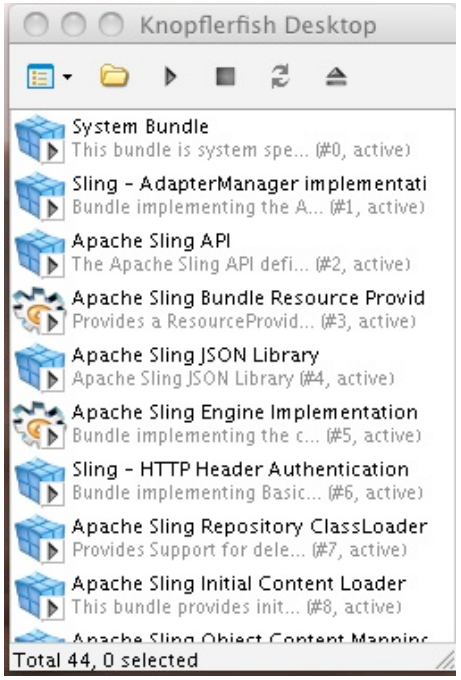
Updates this bundle from an `InputStream`.

Just a few
basic examples...

Dynamic loading / unloading

Just copy bundle jar to Sling's JCR repository (WebDAV)





Bundle activated and started.

(using Sling's jcrinstall module)

Plugins for everything

Servlets

Content editors based on JCR node properties

Mime-type based handlers

Debugging/monitoring tools

Content renderers and decorators

Legacy integration gateways

Mail and messaging services

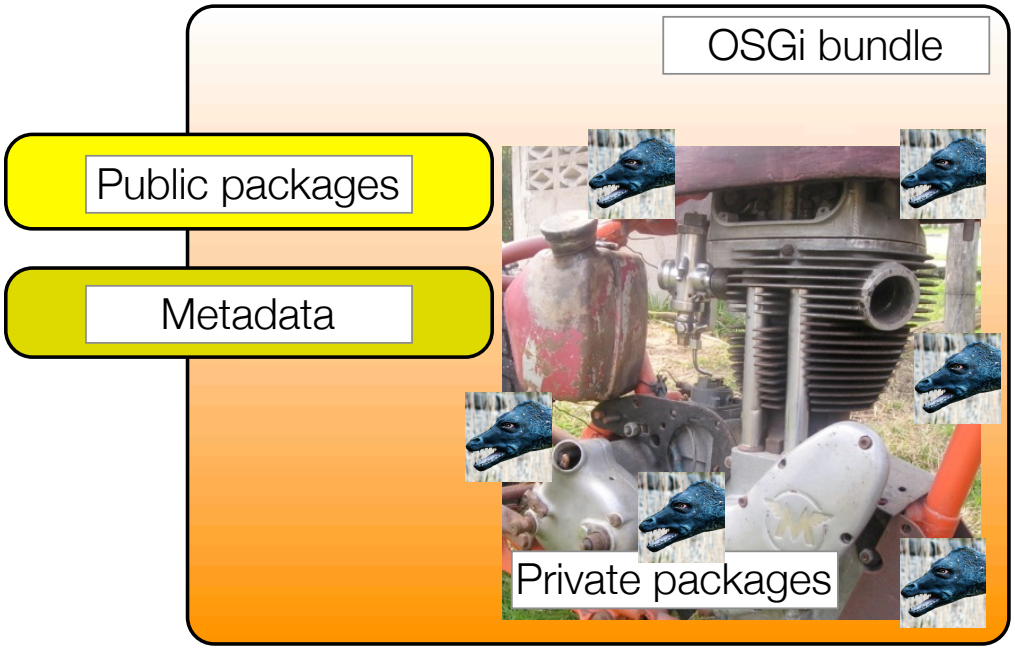
etc, etc...

Plugins: ServiceTracker

```
// Enumerate currently available AuthenticationHandler
// services, and select the one to use
ServiceTracker st = new
  ServiceTracker( bundleContext,
  AuthenticationHandler.class.getName());
ServiceReference[] sr = st.getServiceReferences();

for (int i = 0; i < sr.length; i++) {
  AuthenticationHandler h = (AuthenticationHandler)
    authHandlerTracker.getService(services[i]);
// ...
```

Legacy/customer code

Ugly or incompatible code segregated via private packages



Private / public packages

maven-bundle-plugin instructions:

```
<Export-Package>
  sling.jcr.jackrabbit.server.security
</Export-Package>

<Private-Package>
sling.jcr.jackrabbit.server.impl.*
</Private-Package>
```

The Bad

Granularity is a hard problem

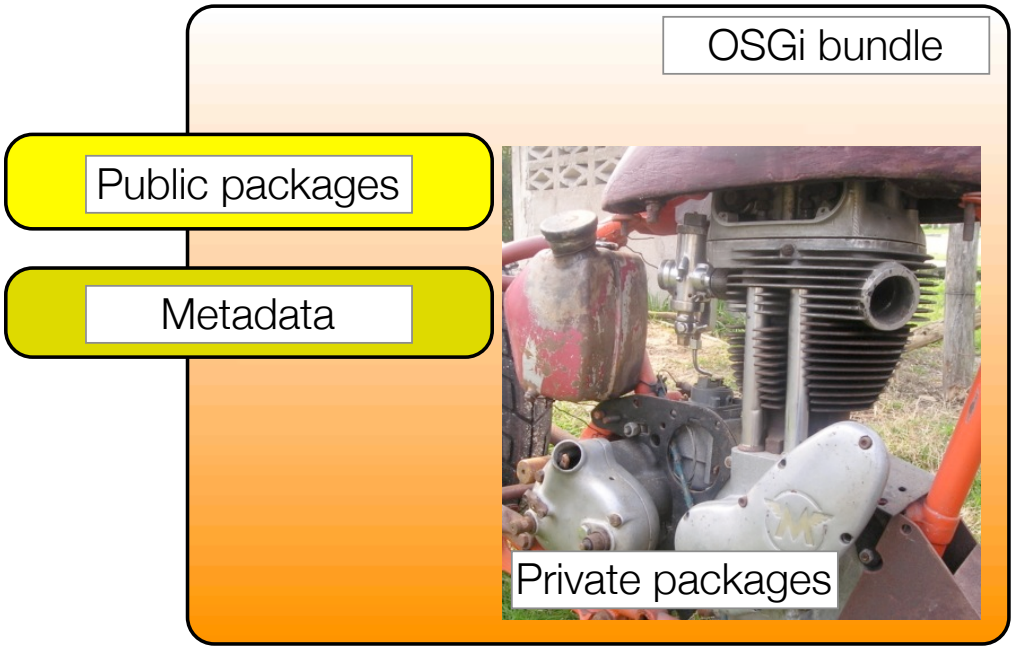
How many bundles?
> *100 currently in cq5*

How to handle “implementation details” libraries.
Extra bundles or private packages?

Strict version management required.
Are we there yet?

Clean up those packages!

Clean separation of interface and implementation packages required



but  when done!

Integration testing required

In-system testing?

but when done!

Is OSGi scary?

"OSGi is difficult to sell - it is adopted in some really visible products,like websphere, glassfish, eclipse, but people don't know that." (Filippo Diotalevi)

Why so many bundles?

Where's my J2EE? *Is the book ready?*

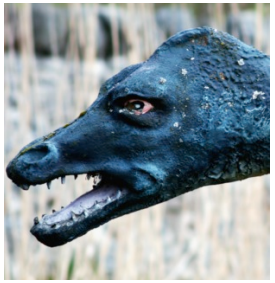
Whaddyamean "provisioning"?

Is it too early?

me: *Starting two years later might have helped* ***avoid initial pains*** *and incomplete implementations.* 

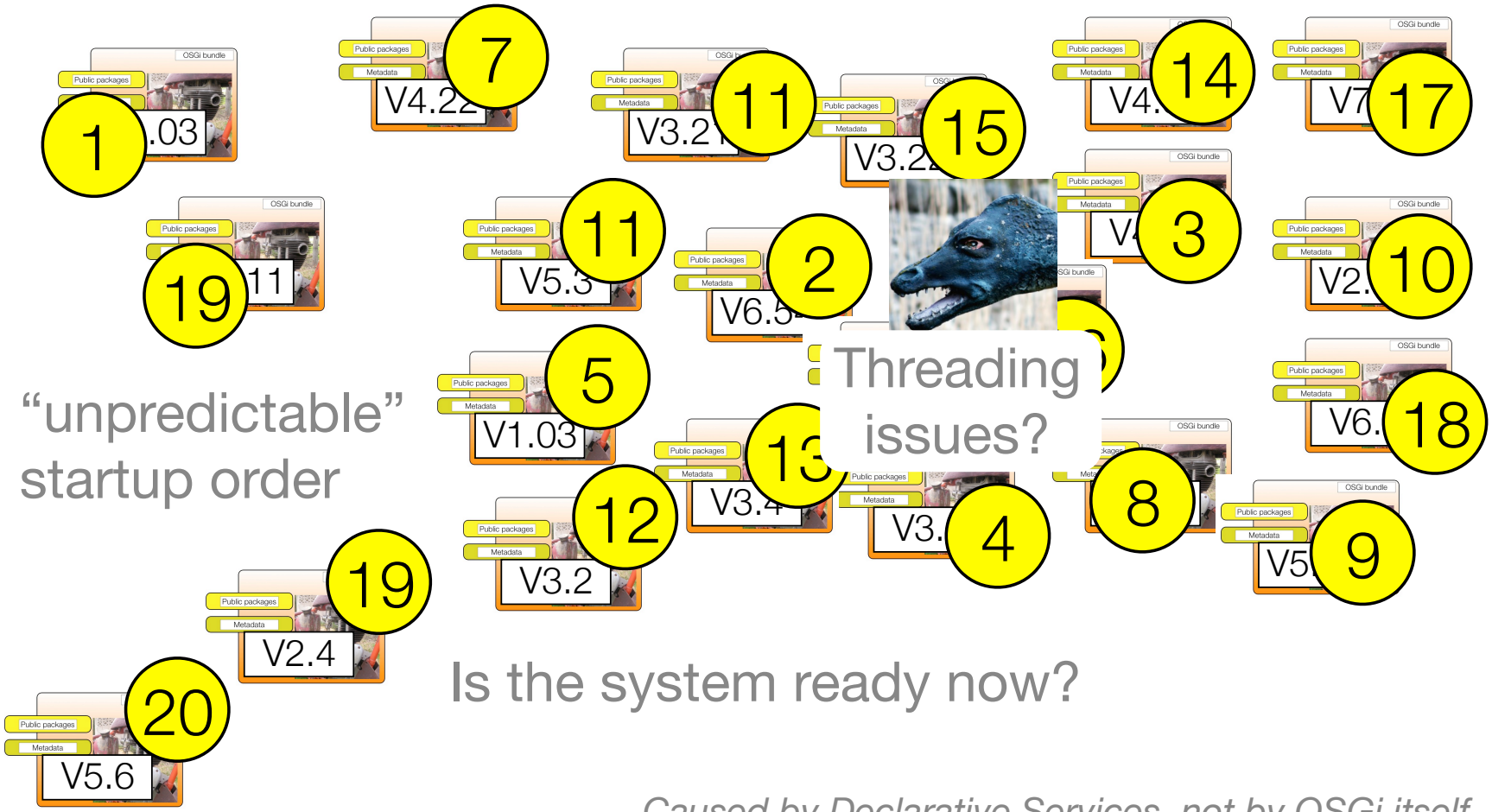
Felix Meschberger: *If we would have done that,* ***we would be nowhere near*** *where we are now!* 

Server-side OSGi is still fairly new...



The Ugly

Asynchronous startup



"unpredictable" startup order

Is the system ready now?

Caused by Declarative Services, not by OSGi itself.

Unpredictable assemblies

Spot the monsters!

Security and **lots** of discipline would help

Testing

Testing?

JUnit and mocks, no OSGi	JUnit with OSGI	Integration testing, OSGi, HTTP
Test the **code**	Test **bundles**	Test **assemblies**
Required. Sufficient?	in realistic conditions	in realistic conditions
Many examples in Sling	We don't use this at this time.	Sling launchpad and jcrinstall

JUnit testing w/mocks

JUnit
and mocks,
no OSGi

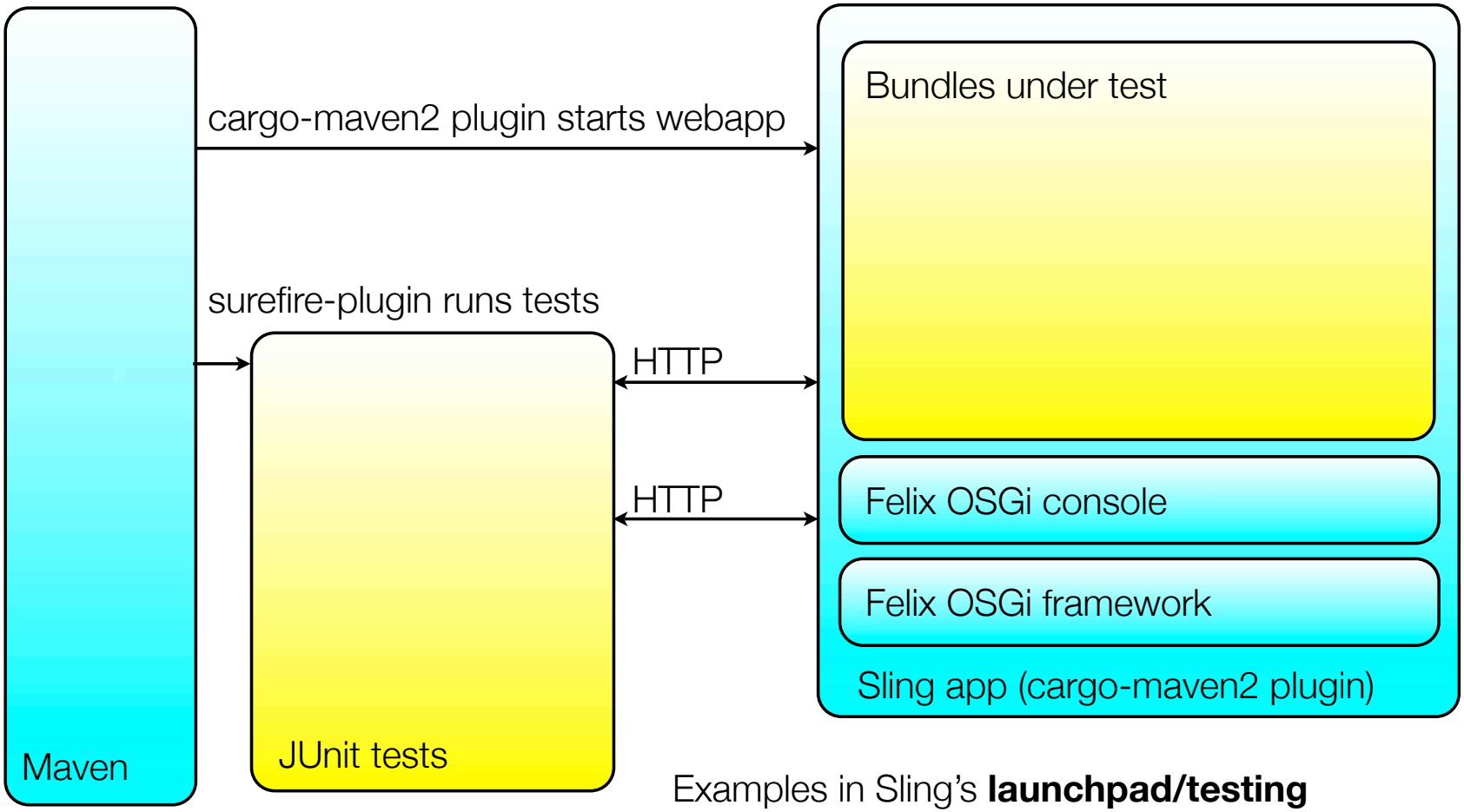
No OSGi needed, fast!
But **what** am I testing exactly?
Sometimes hard to follow or modify.

Example, using **jmock.org**:
JsonReaderTest in Sling's contentloader module.

```
@org.junit.Test public void testEmptyObject()
throws Exception {
  this.mockery.checking(new Expectations() {{
    allowing(creator).createNode(null, null, null);
    inSequence(mySequence);
    allowing(creator).finishNode();
    inSequence(mySequence);
}});
this.parse(""); }
```

https://svn.eu.apache.org/repos/asf/incubator/sling/trunk/bundles/jcr/contentloader/src/test/java/org/apache/sling/jcr/contentloader/internal/JsonReaderTest.java

Integration tests, OSGi + HTTP



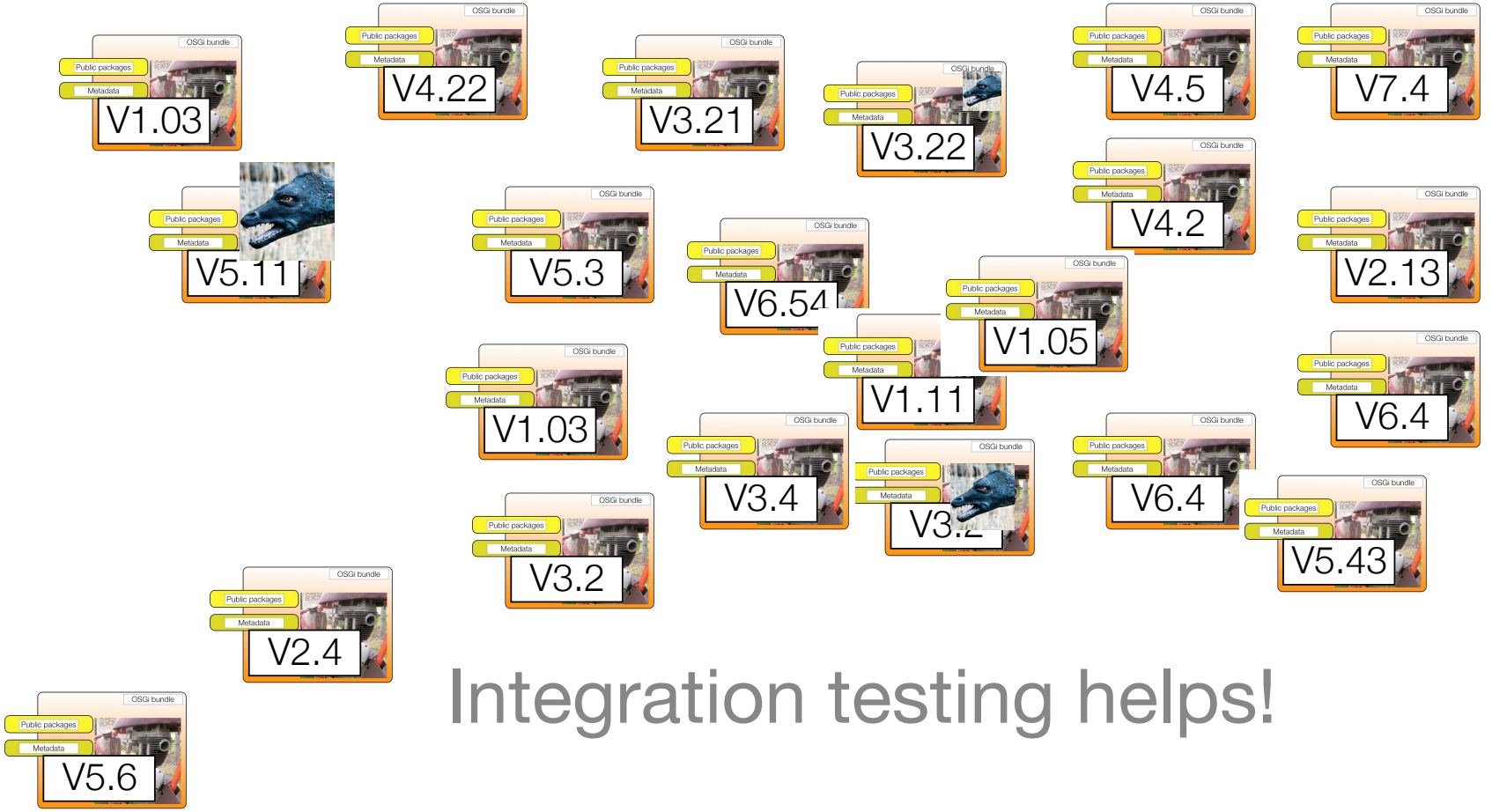
Examples in Sling's **launchpad/testing** and **jcrinstall** modules.

Integration test: JsonRenderingTest

```
public void testRecursiveInfinity() throws IOException {

    final Map<String, String> props = new HashMap<String, String>();
    props.put("text", testText);
    props.put("a/b/c/d/e/f/g/h/i/j/k/l/m/n/o/p/q/r/s/t/u/v/w/x/y", "yes");
    final String url = testClient.createNode(postUrl, props);

    final String json = getContent(url + ".infinity.json",
        CONTENT_TYPE_JSON);

    assertJavascript(testText, json, "out.print(data.text)");
    assertJavascript("yes", json,
        "out.print(data.a.b.c.d.e.f.g.h.i.j.k.l.m.n.o.p.q.r.s.t.u.v.w.x.y)");
}
```

Simulate actual usage!

Unpredictable assemblies

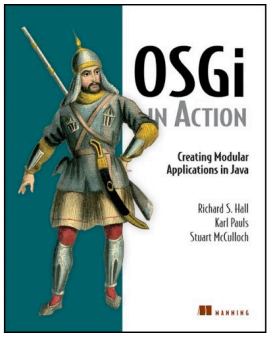


Integration testing helps!

The Future

OSGi @day, 2 years from now

Developers got used to it (and read the book).



Frameworks and tools improved.

Distributed OSGi? Maybe.

Customers understand OSGi and like it..

Apache Sling paved the way.

Do we need more features?

Today we use:

Bundles (using Maven plugins)
Lifecycle, Service Tracker
Configurations and Felix Web Console
Declarative Services (using Maven plugins)
Sling's **jcrinstall** module
Log, HTTP, Event services

Later:
Deployment packages. Security maybe.
More? Not really - tame what we use!

Conclusions

Good? Bad? Ugly?

Conclusions

- OSGi is great for **modularity**
- OSGi fosters **better structured** code
- Dynamic **services** and **plugins**
- **Tooling** needs to improve, but usable
- OSGi **skills** need to improve!
- **Asynchronous startup** can be problematic if using declarative services