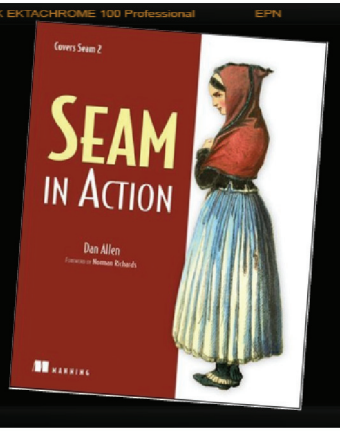

Seam 3

Portable, modular, type-safe extensions for CDI & Java EE 6



JBoss Community

Dan Allen

Principal Software Engineer

Seam Community Liaison

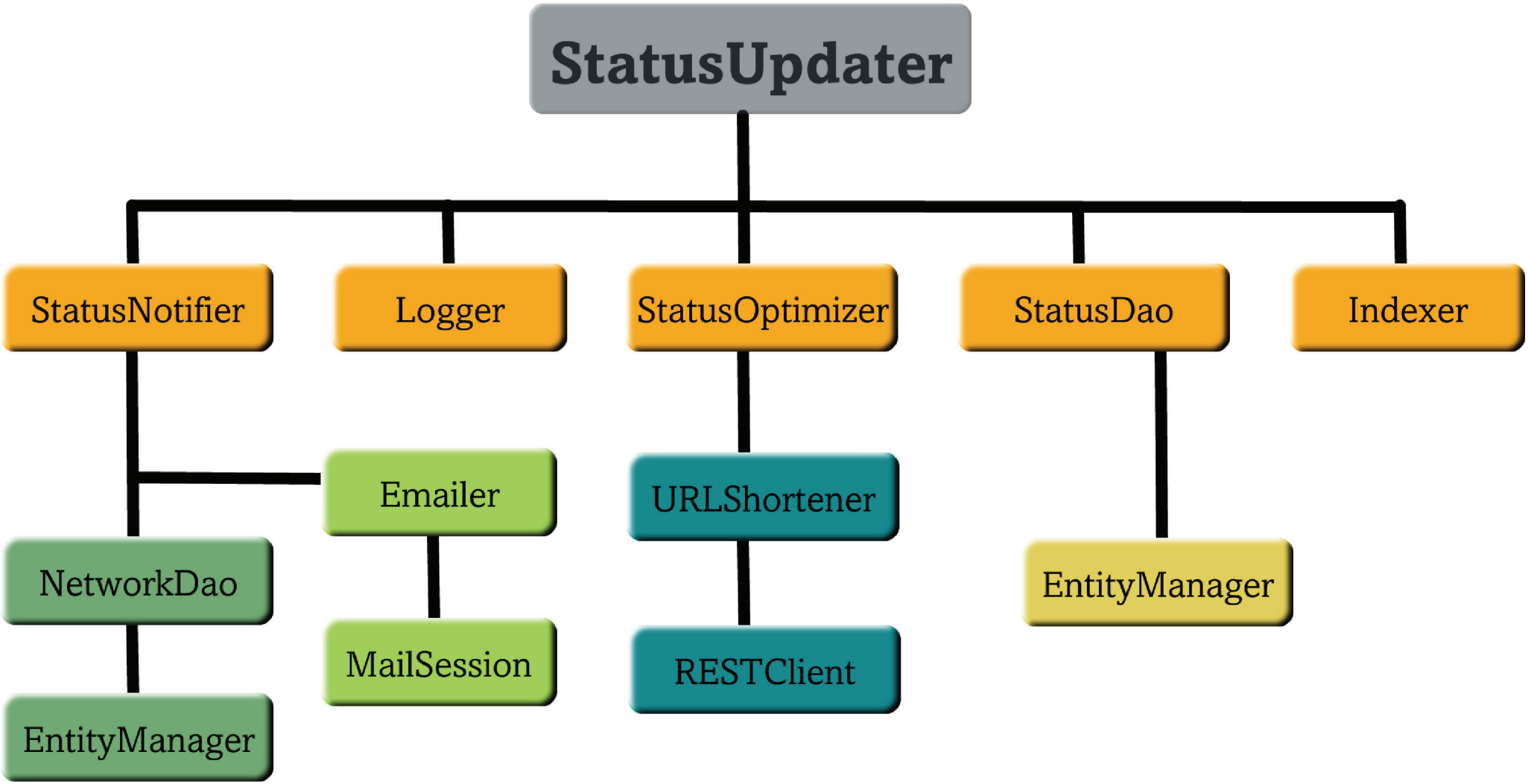
@mojavelinux

You're in the right place if you

Enterprise application development


HUGE!
It's gonna be big...


I need one of those cloud thing-a-ma-jigs.
And I need it in 10 minutes for a marketing meeting.


StatusUpdater
StatusNotifier
Logger
StatusOptimizer
StatusDao
Indexer
Emailer
NetworkDao
EntityManager
MailSession
URLShortener
RESTClient
EntityManager

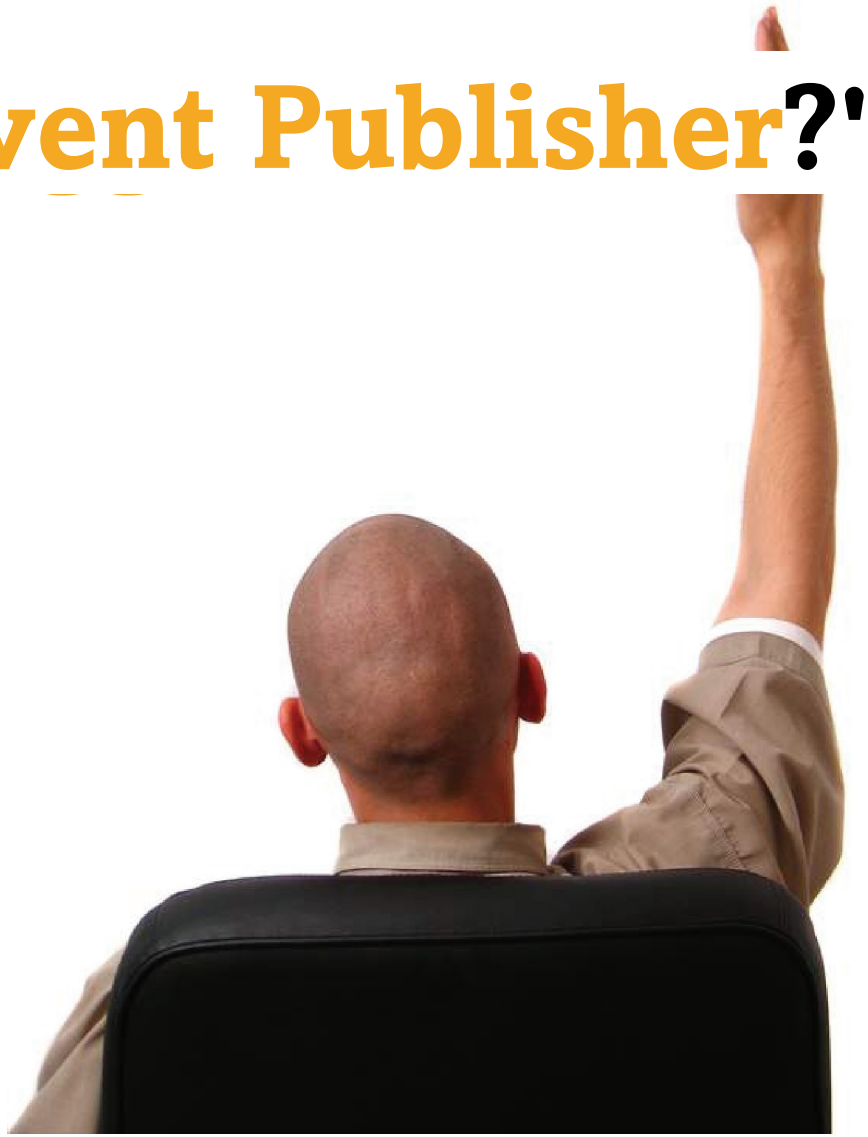
....in the beginning

there was...

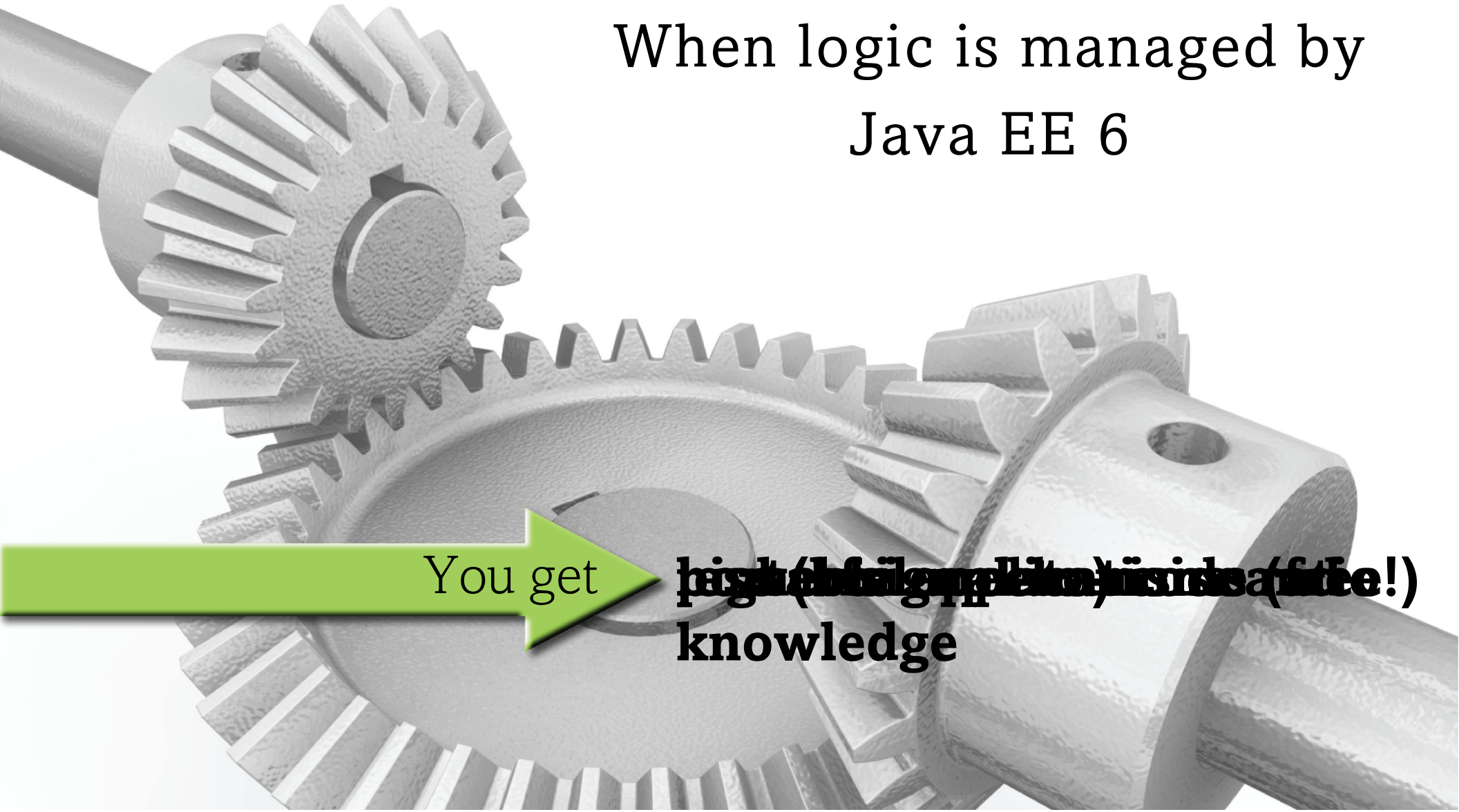
```
public class StatusUpdater {
    private UrlShortener shortener;
    private StatusPersister persister;
    public StatusUpdater() {
        this.shortener = new UrlShortener();
        this.persister = new StatusPersister();
    }

    public String post(String username, String status) {
        String sStatus = shortener.shortenUrls(status);
        persister.persist(username, status);
        return sStatus;
    }
}
```

```
StatusUpdater updater = new StatusUpdater();
```

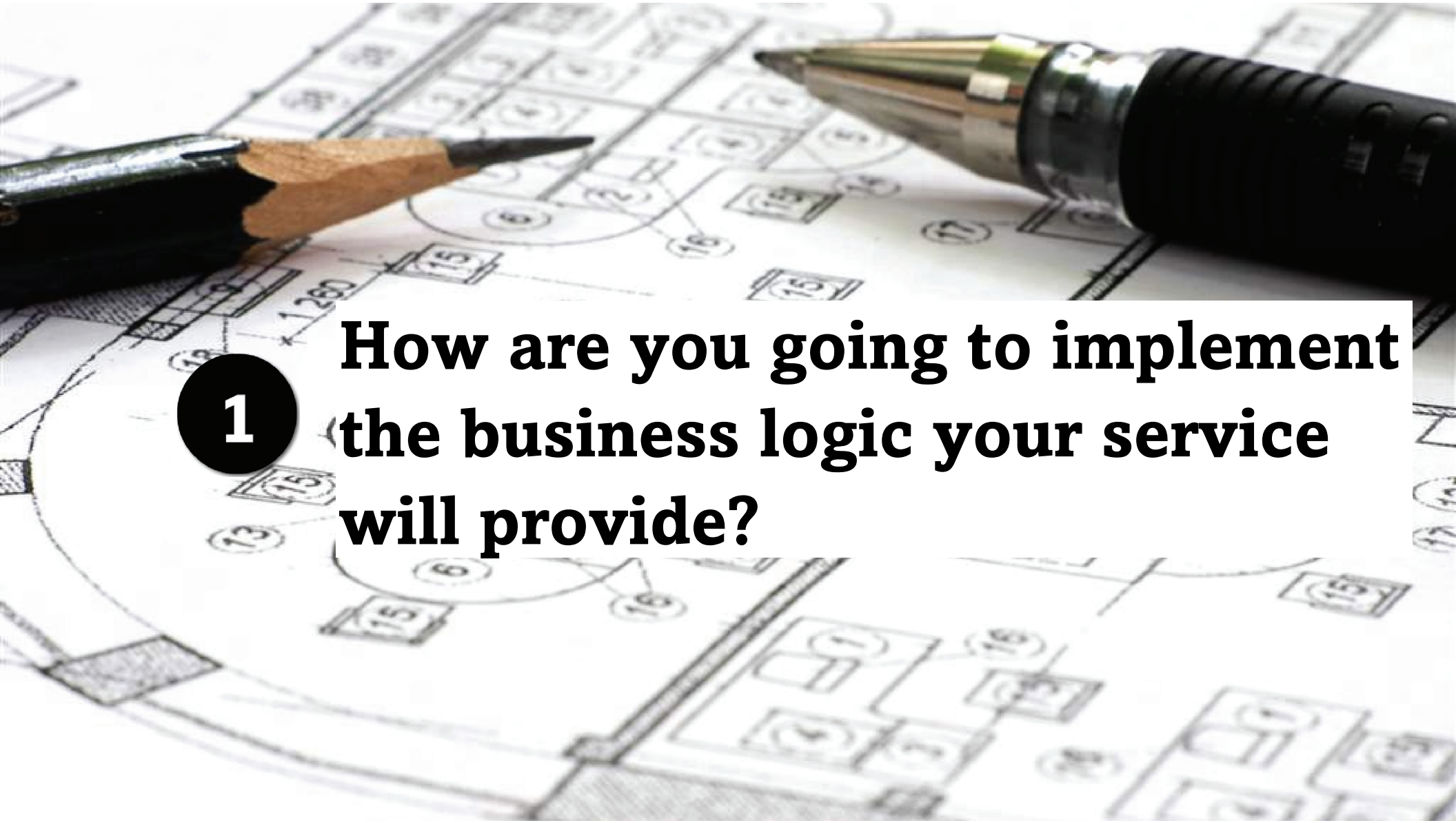
"But what about Event Publisher?"




DUDE! I THINK I FOUND MY COMMODOR 64 DOWN HERE...


When logic is managed by
Java EE 6
You get
knowledge

There are 3 important questions you need to answer



1 How are you going to implement the business logic your service will provide?

2 What is your service's lifespan?

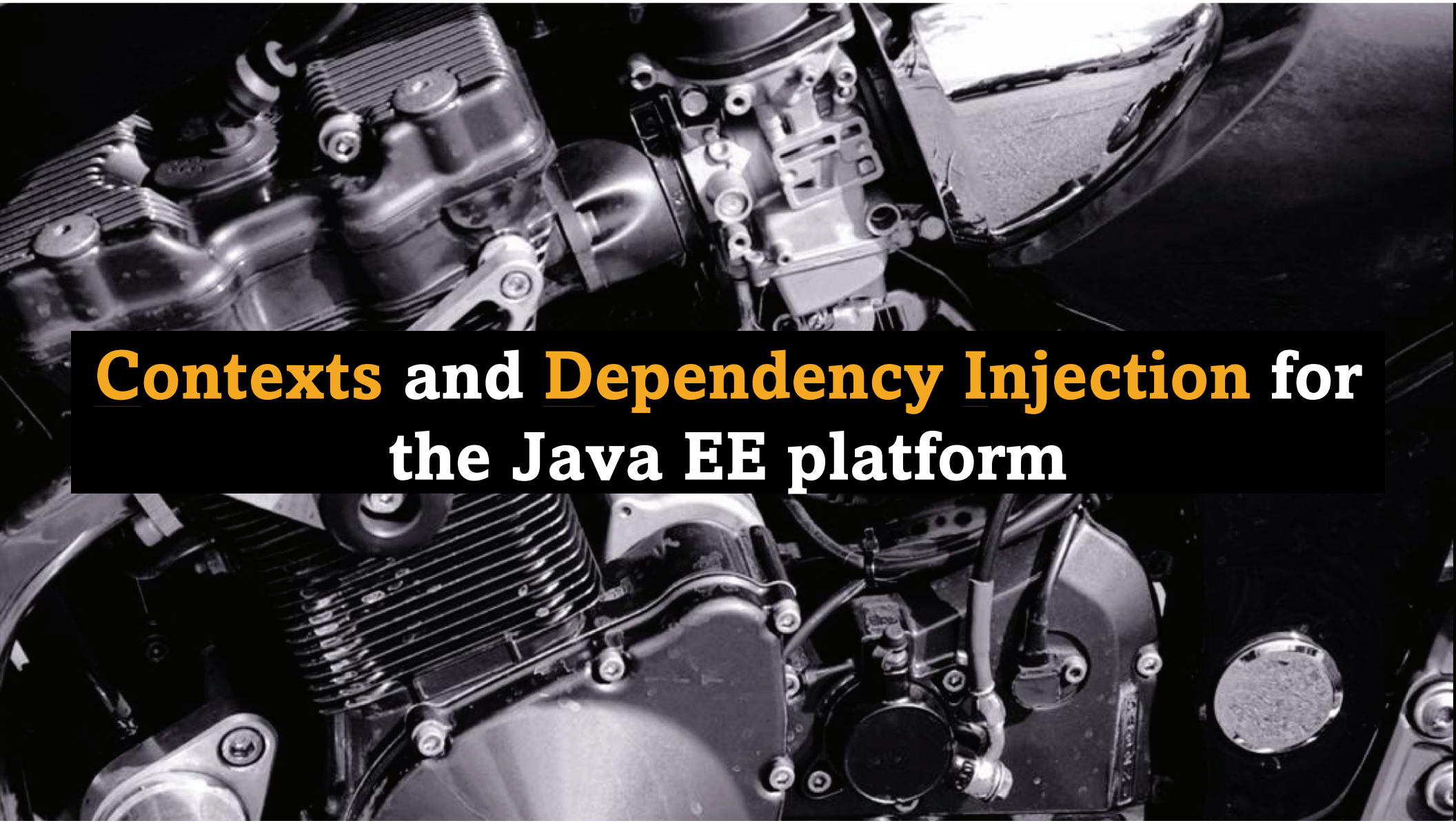
3

What service are you going to call?

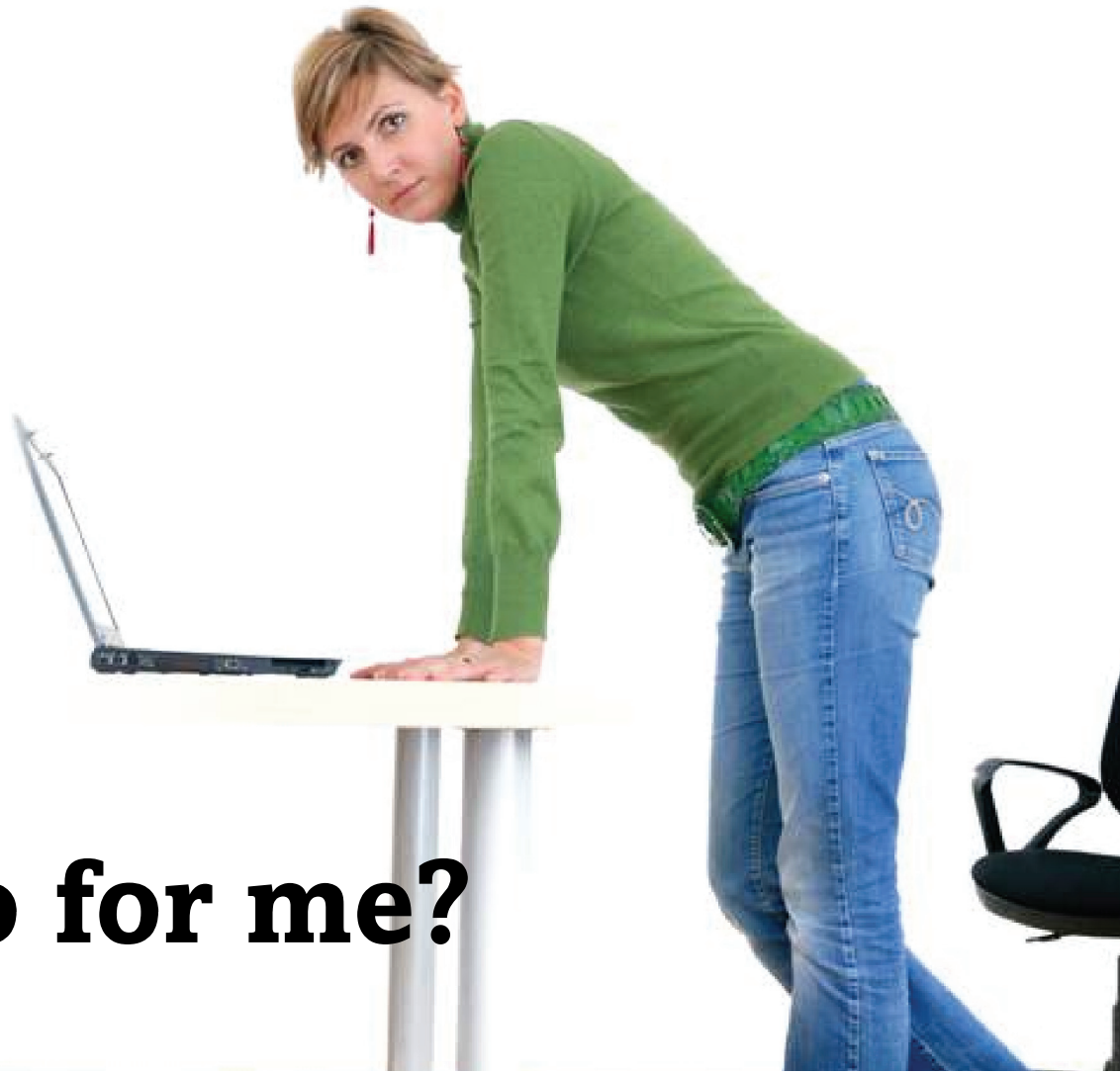
And did you get its number last night?

Focus with dependency injection

```
public class StatusUpdater {
    @Inject
    private UrlShortener shortener;

    @Inject
    private StatusPersister persister;

    public String post(String username, String status) {
        String sStatus = shortener.shortenUrls(status);
        persister.persist(username, status);
        return sStatus;
    }
}
```


Contexts and Dependency Injection for
the Java EE platform

What can CDI do for me?



CDI can inject dependencies into its context

But wait! There's more...

CDI also provides

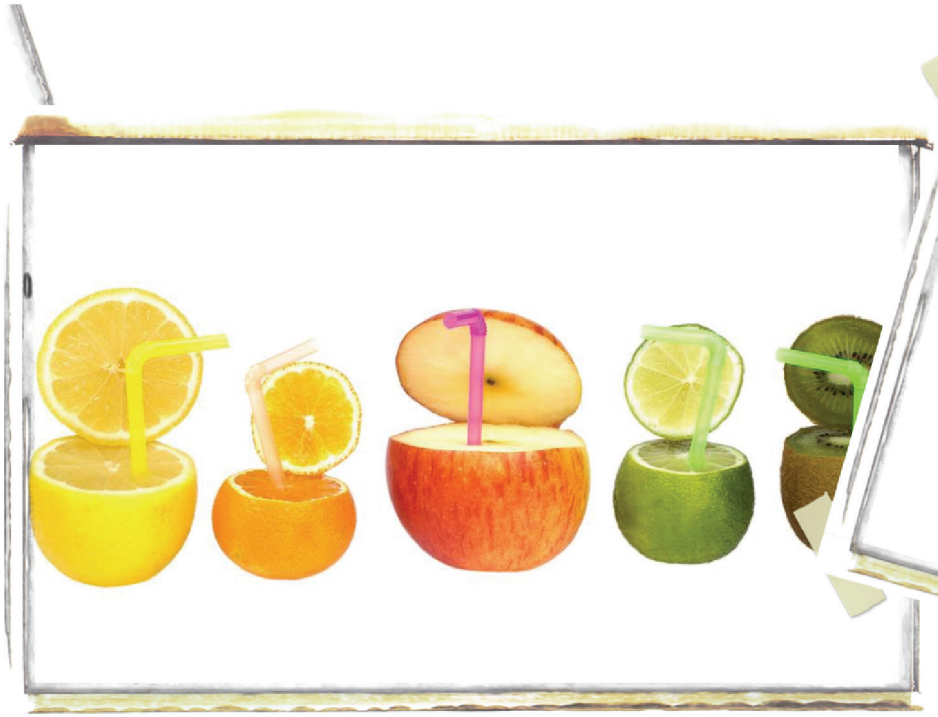
We're not done yet!


Portable extensions

SPI for hacking Java EE to:

- ☑ register additional beans
- ☑ satisfy injection points
- ☑ introduce custom scope and backing context
- ☑ augment or override bean metadata

Java EE is your oyster



portable + flexible + extensible

=


KODAK EKTACHROME 100 Professional
EPN
172241/21
KODAK EKTACHROME 100 Professional
EPN


CDI
services for components
extension points

"What makes CDI different?"

STANDARDIZED

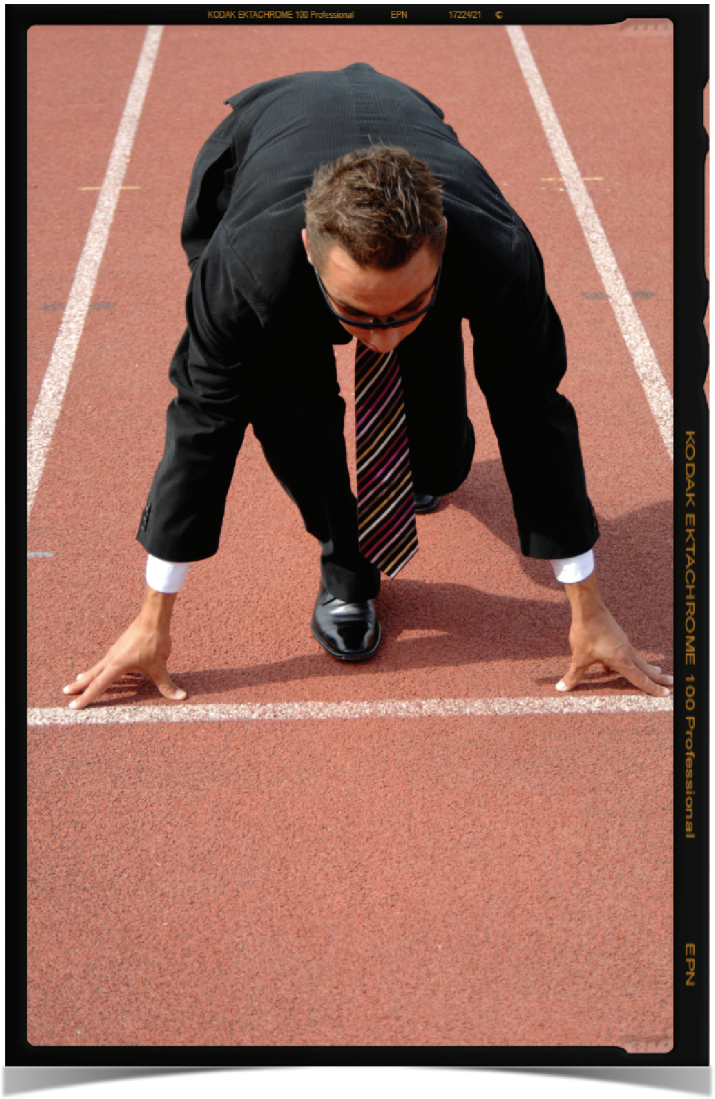
TYPE-SAFE

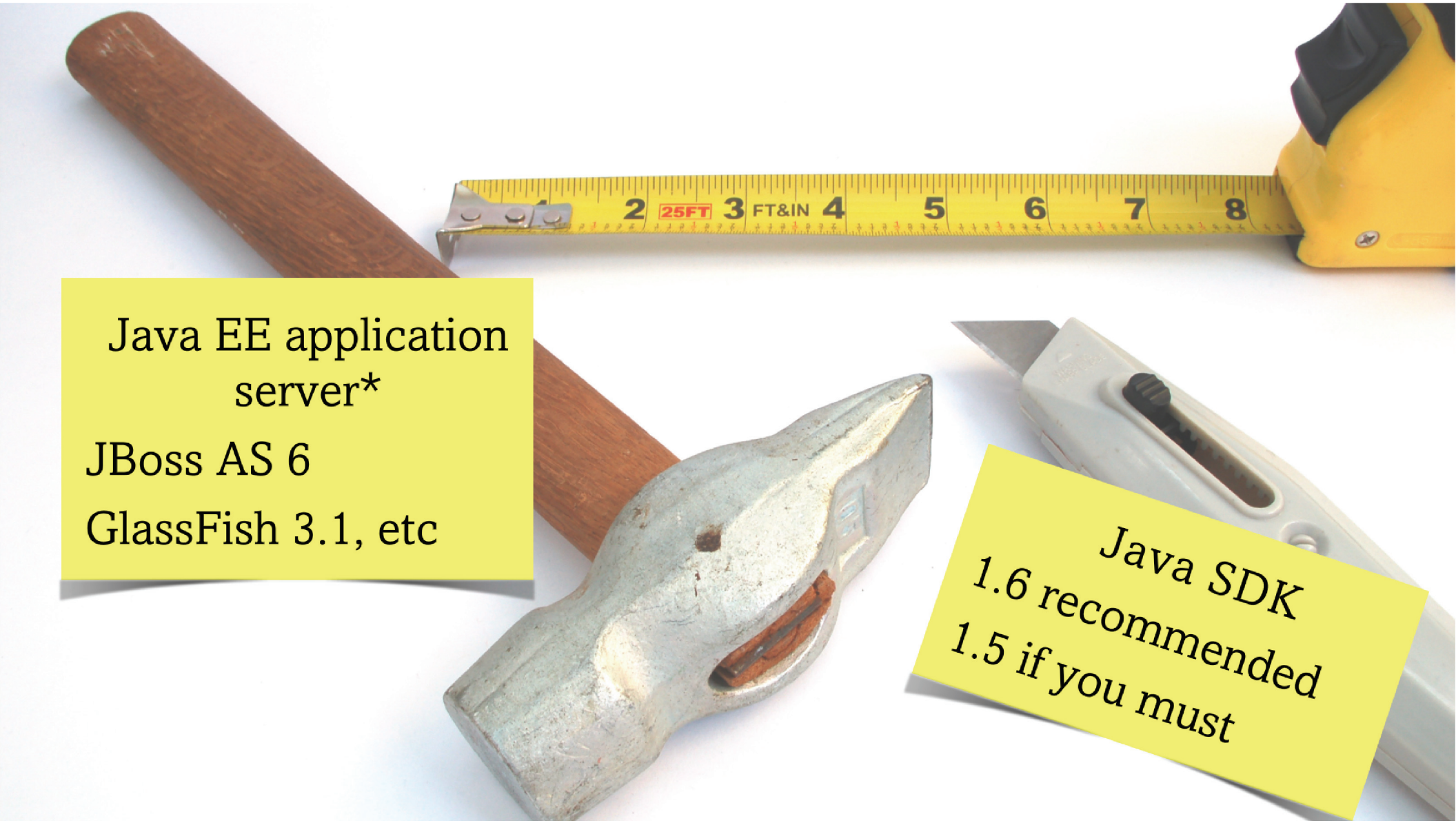
EXTENSIBLE

Loose coupling

Strong Typing

Let's get started




Java EE application server*
JBoss AS 6
GlassFish 3.1, etc
Java SDK
1.6 recommended
1.5 if you must


Build tool
Gradel, Maven 3 or
ANT
IDE
Eclipse + JBoss tools
Netbeans or IDEA

CDI Reference Implementation & TCK

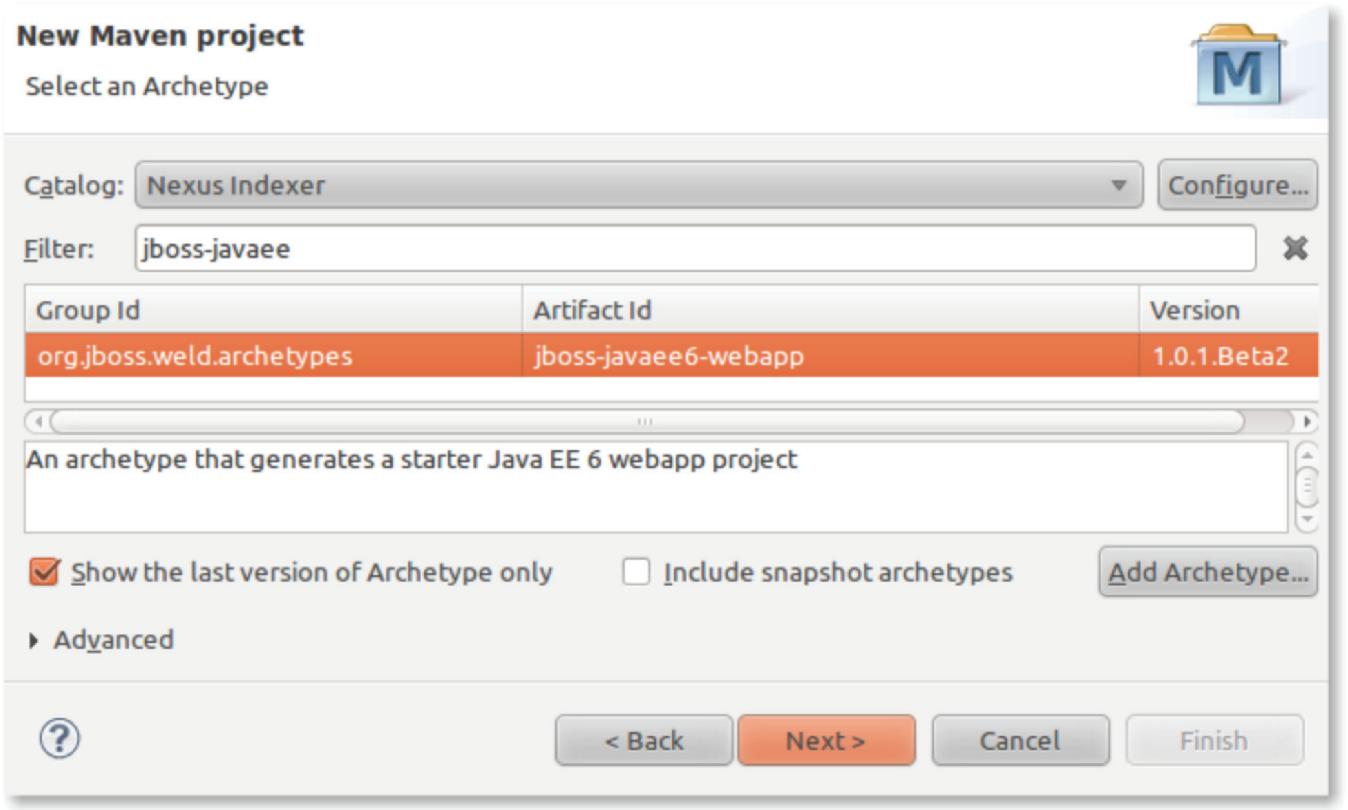
features Servlet container & Java SE support

Cranking out a project

Maven archetype from IDE or commandline

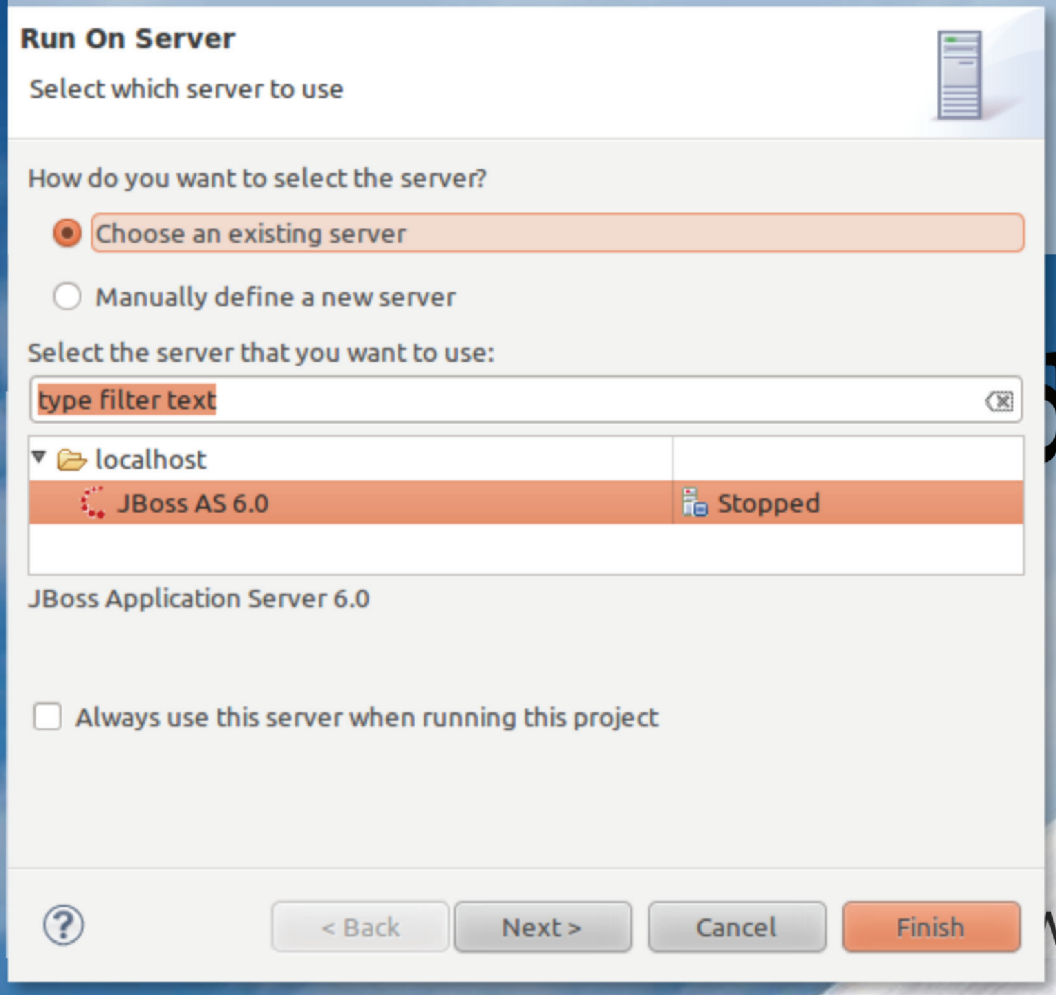
or

$ mvn archetype:generate



```
$ mvn archetype:generate
Choose an archetype:
...
289: remote -> jboss-javaee6-webapp (-)
290: remote -> jboss-jsf-weld-servlet-webapp (-)
...
Choose a number: 99: 289
Choose version:
1. 1.0.1.Beta1
2. 1.0.1.Beta2
Choose a number: 2:
Define value for property 'groupId': : com.socialize
Define value for property 'artifactId': : socialize
Define value for property 'version':  1.0-SNAPSHOT: : 1.0.0-SNAPSHOT
Define value for property 'package':  com.socialize: :
[INFO] Using property: name = Java EE 6 webapp project
Confirm properties configuration:
groupId: com.socialize
artifactId: socialize
version: 1.0.0-SNAPSHOT
package: com.socialize
name: Java EE 6 webapp project
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Deploying to JBoss AS from the IDE



ndline

d running

var

Hello, managed bean!

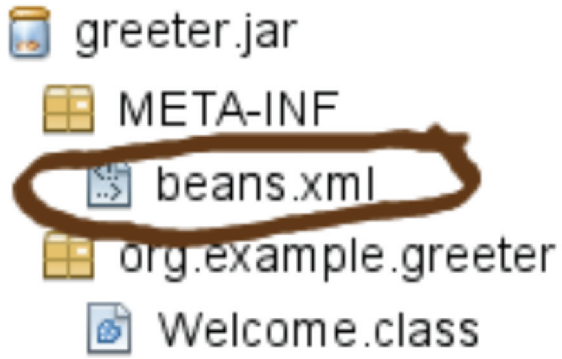
```
public class StatusUpdater {
    public String post(String username, String status) {
        System.out.println(username + " said " + status);
        return status;
    }
}
```

Hello, EJB 3!

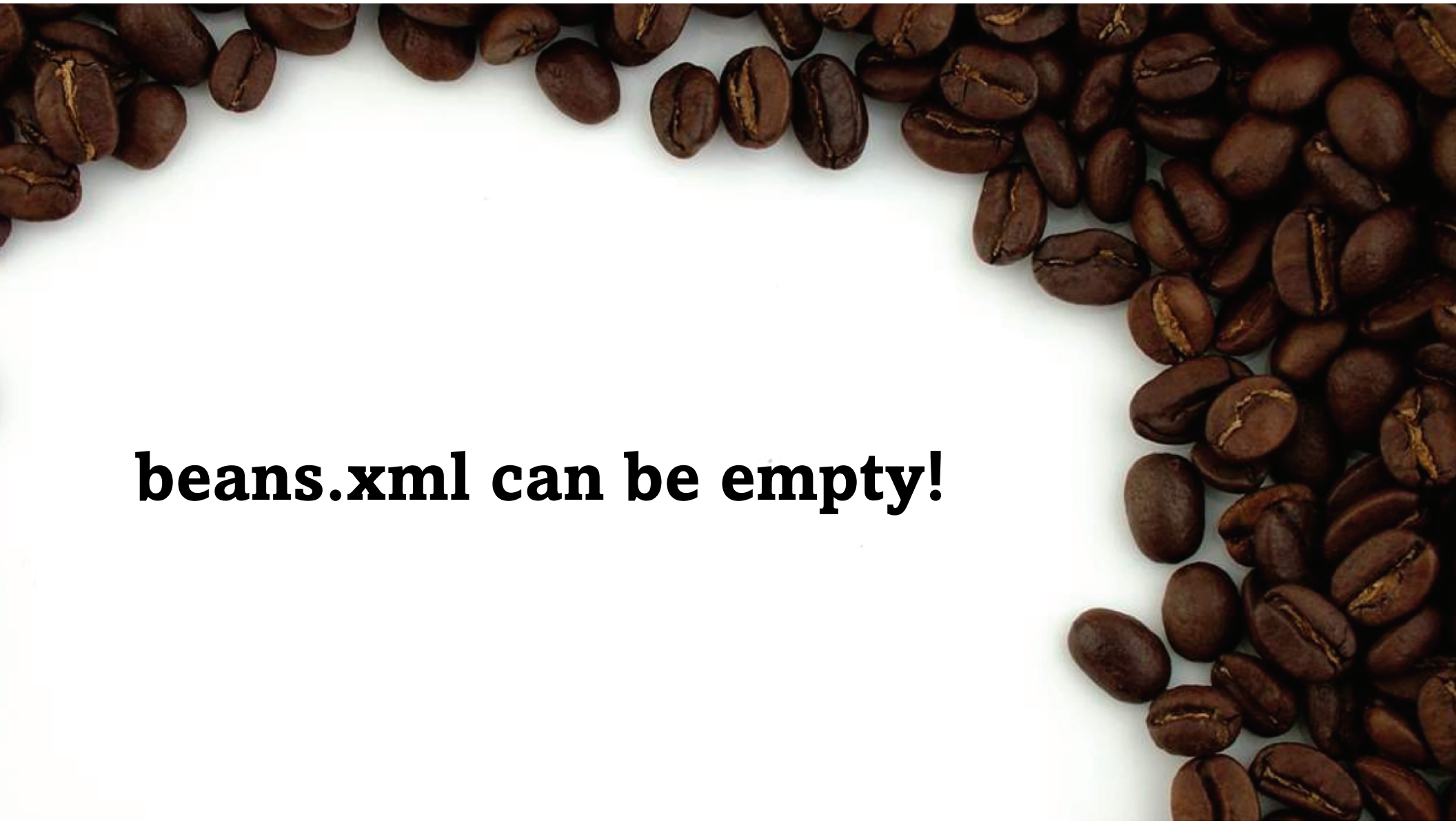
```
@Stateless
public class StatusUpdater {
    public String post(String username, String status) {
        System.out.println(username + " said " + status);
        return status;
    }
}
```


"When is a bean recognized?"

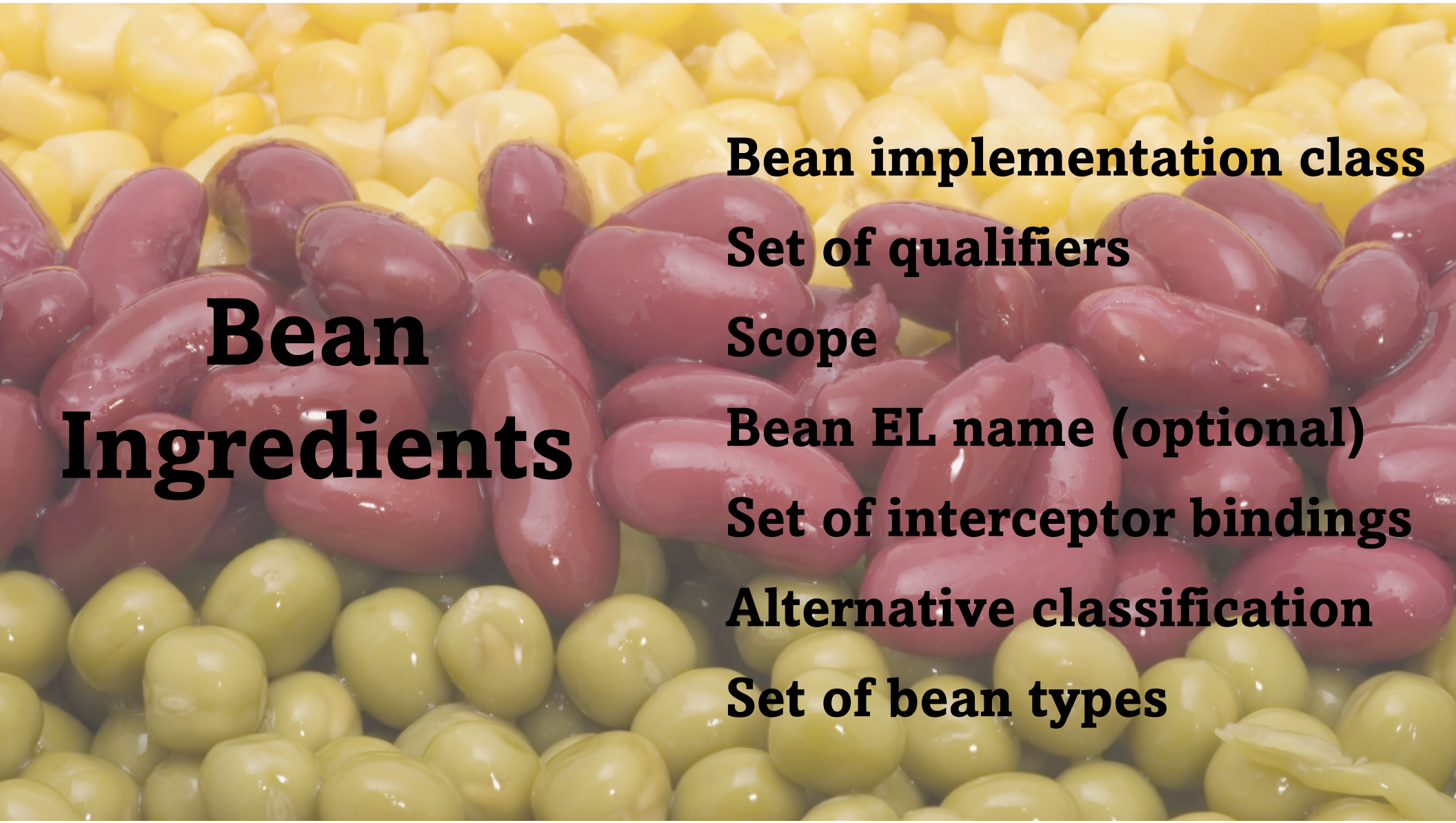
Bean archive (jar) - META-INF/beans.xml



Bean archive (war) - WEB-INF/beans.xml

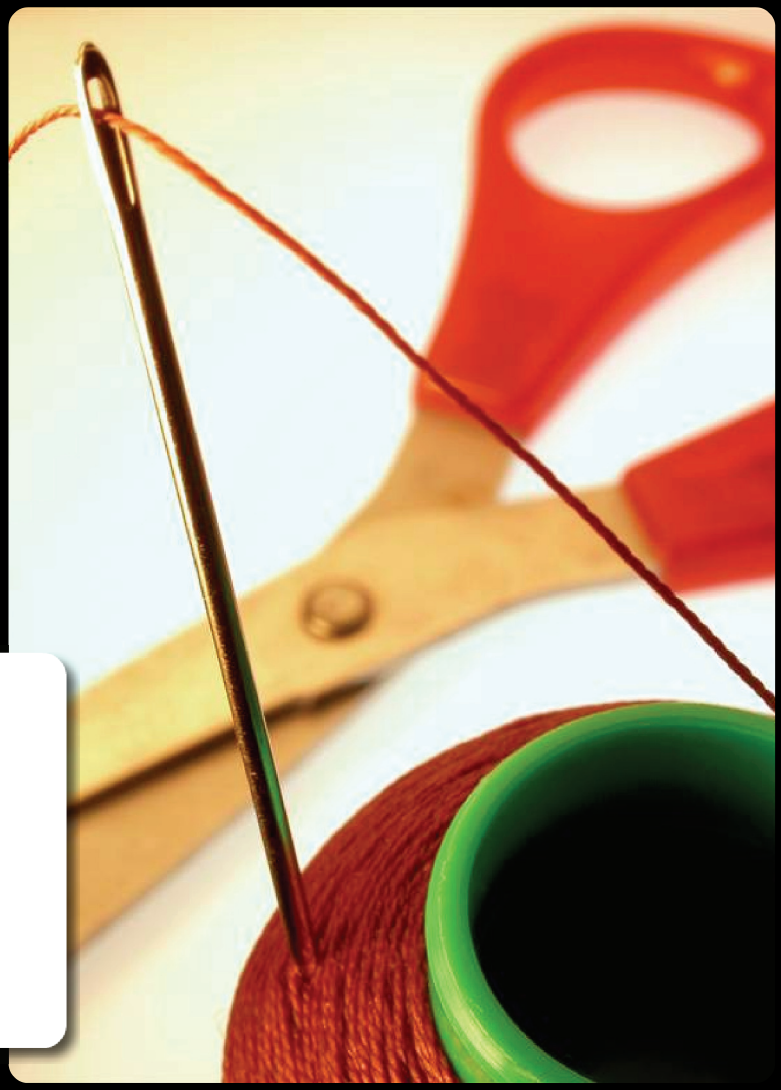


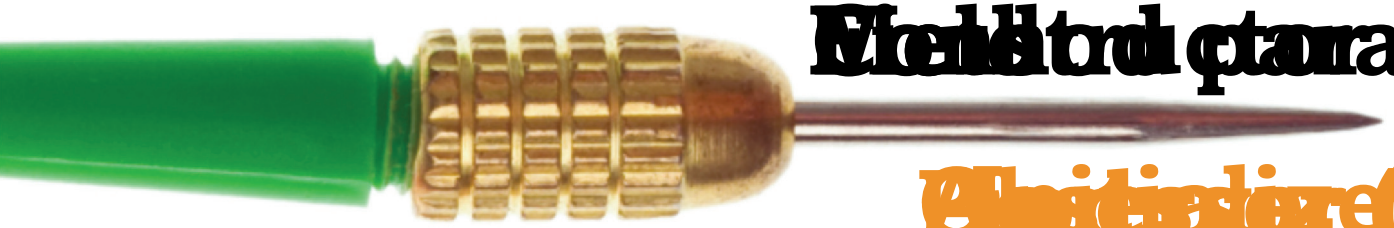
beans.xml can be empty!


Bean
Ingredients
Bean implementation class
Set of qualifiers
Scope
Bean EL name (optional)
Set of interceptor bindings
Alternative classification
Set of bean types

Our first collaborator

```
public class UrlShortener {
    public String shortenUrls(String text) {
        return text.replace("http://www.", "http://");
    }
}
```





Model parameter

Hyperparameter (automatically injected)

"Use the source!"

Types

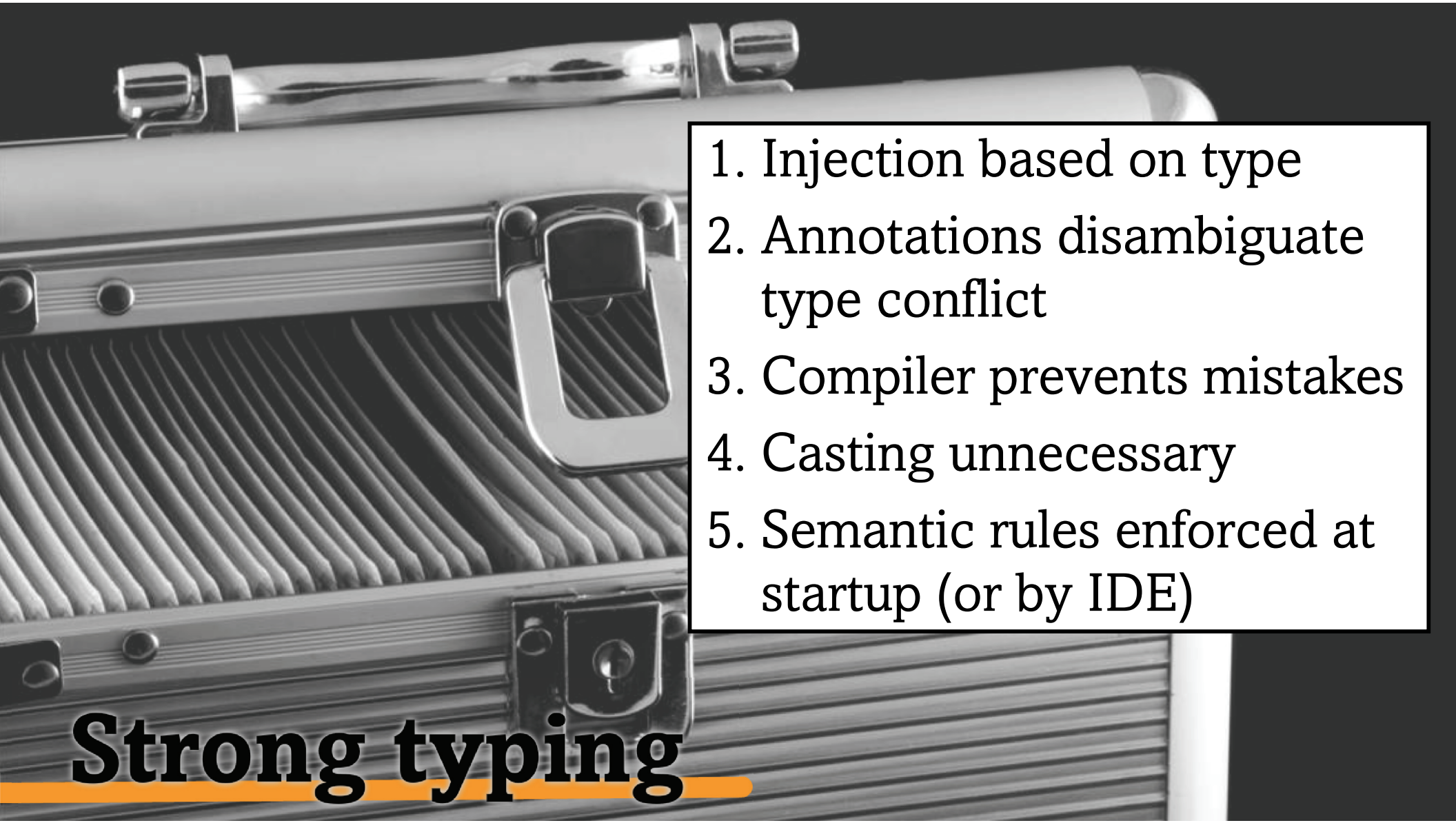
Type parameters

Annotations

Method signatures

Injection 101

```
public class StatusUpdater {
    @Inject
    private UrlShortener shortener;

    public void post(String username, String status) {
        String sStatus = shortener.shortenUrls(status);
        System.out.println(username + " said " + sStatus);
        return sStatus;
    }
}
```

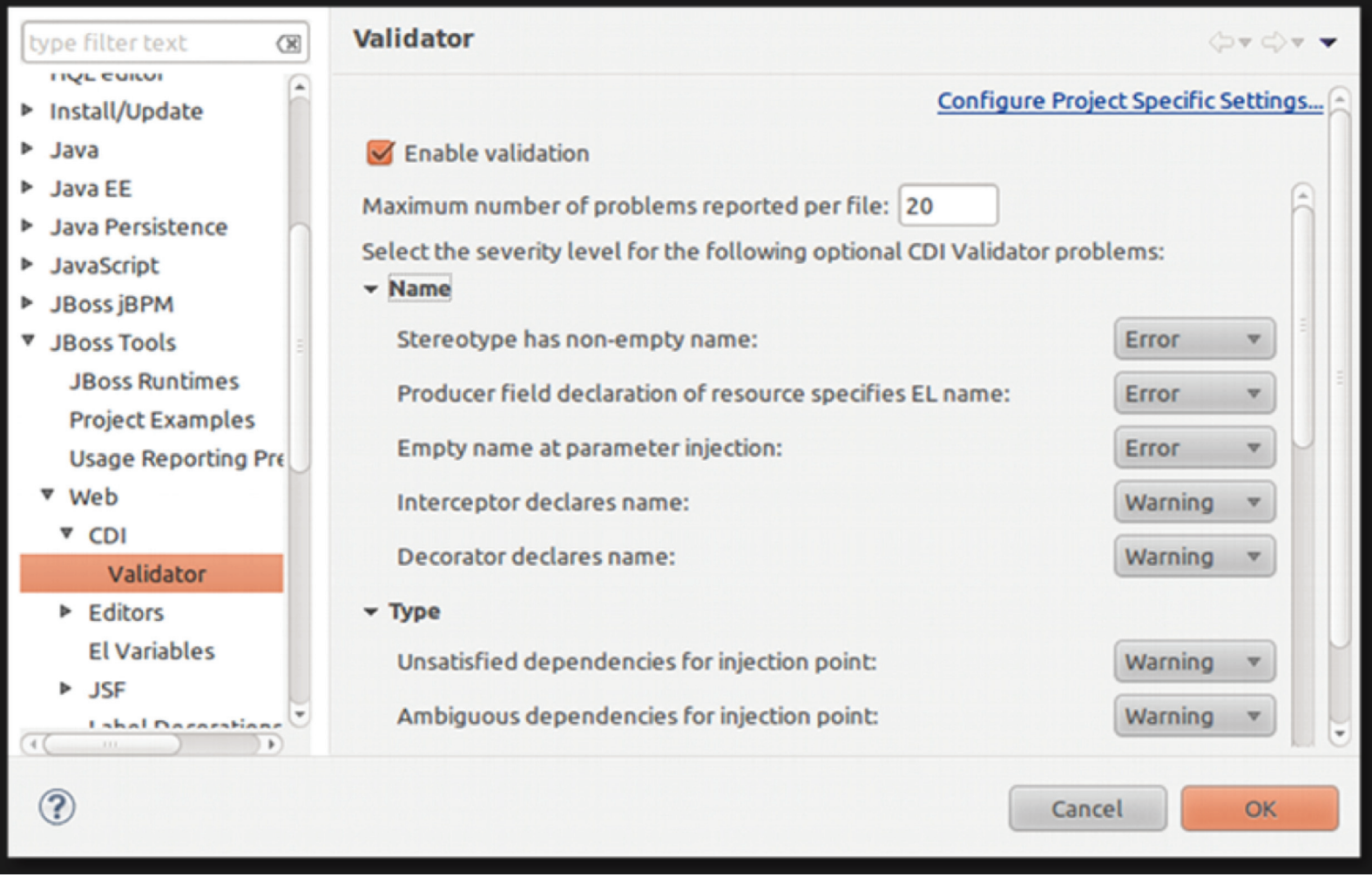


Strong typing

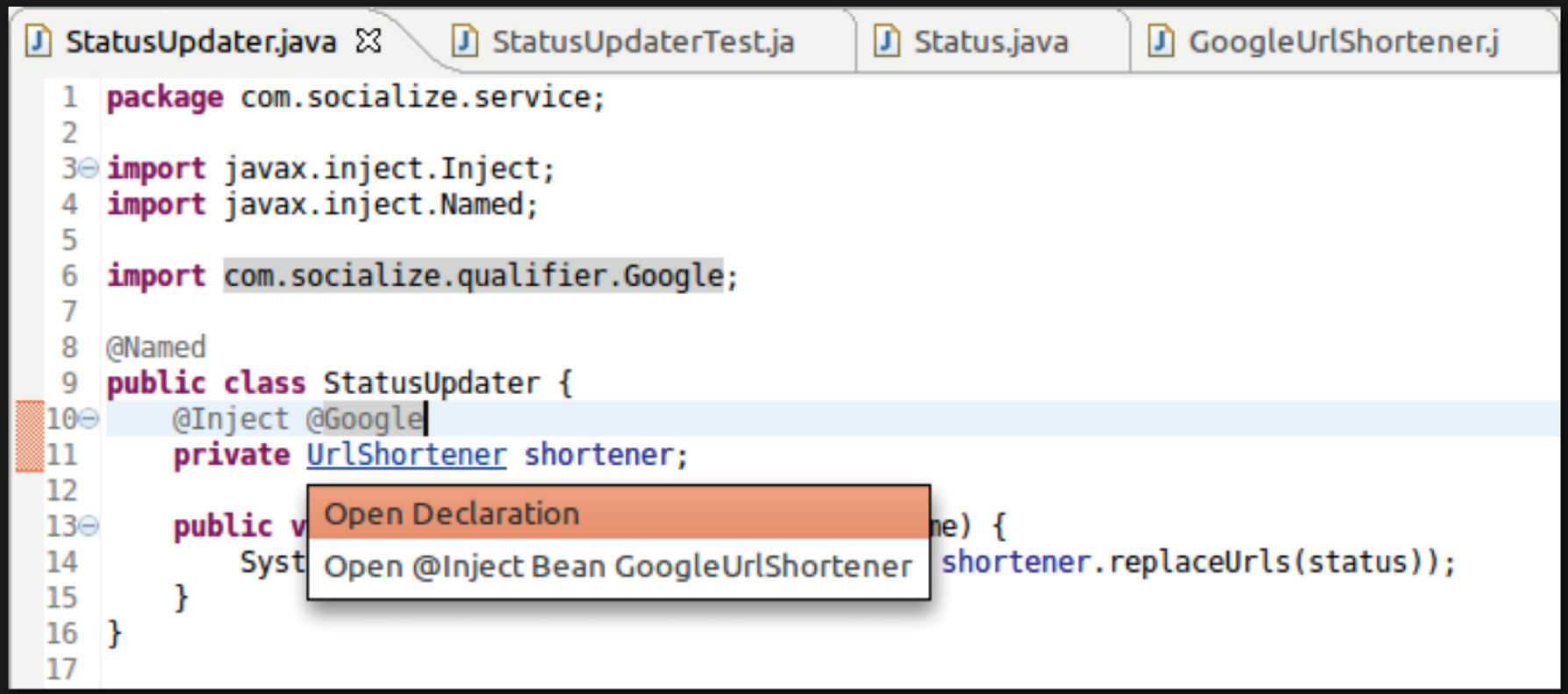
1. Injection based on type
2. Annotations disambiguate type conflict
3. Compiler prevents mistakes
4. Casting unnecessary
5. Semantic rules enforced at startup (or by IDE)

Strong Typing == Strong Tooling

CDI validations in JBoss Tools



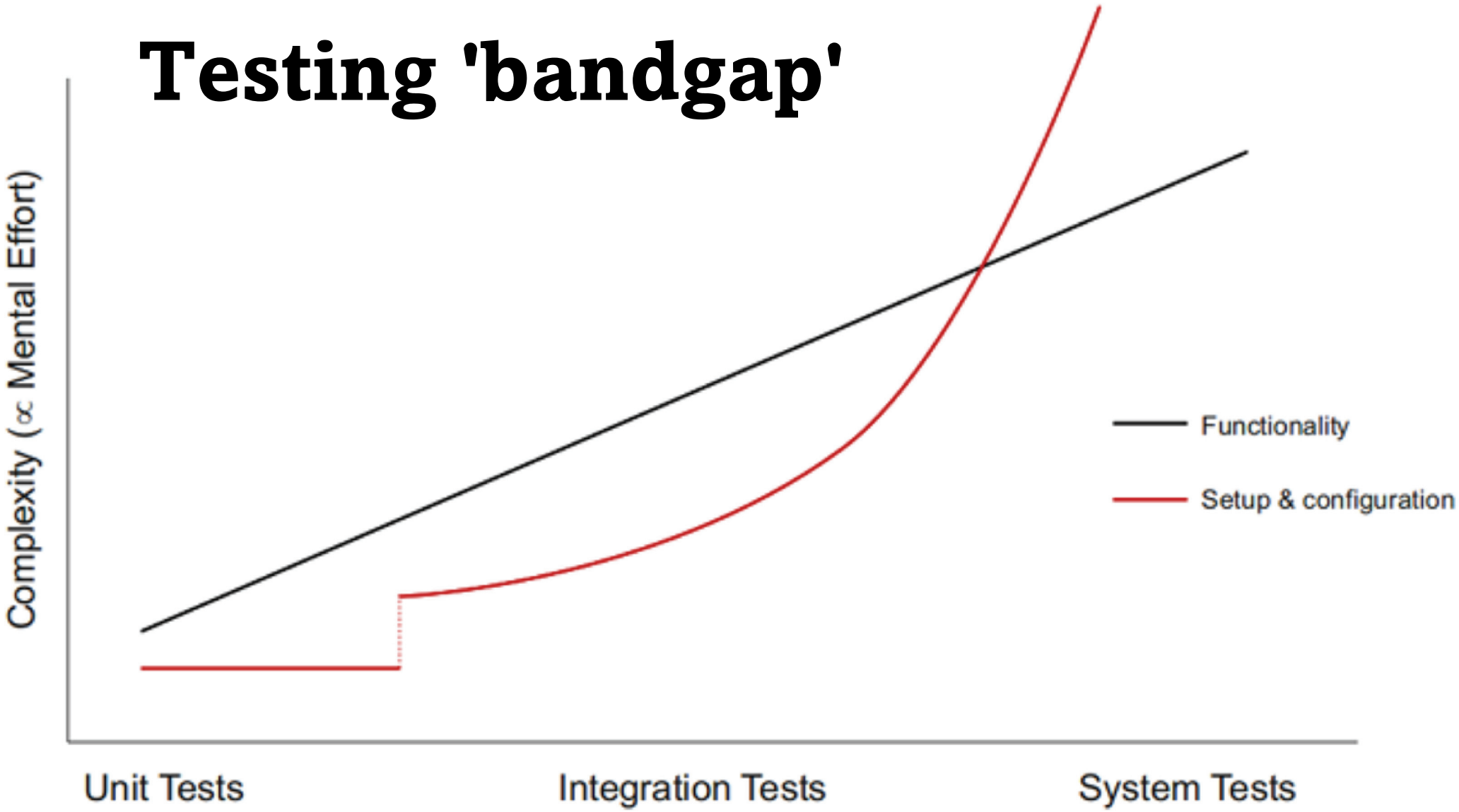
OpenOn injection point




Ah, but can we test it?

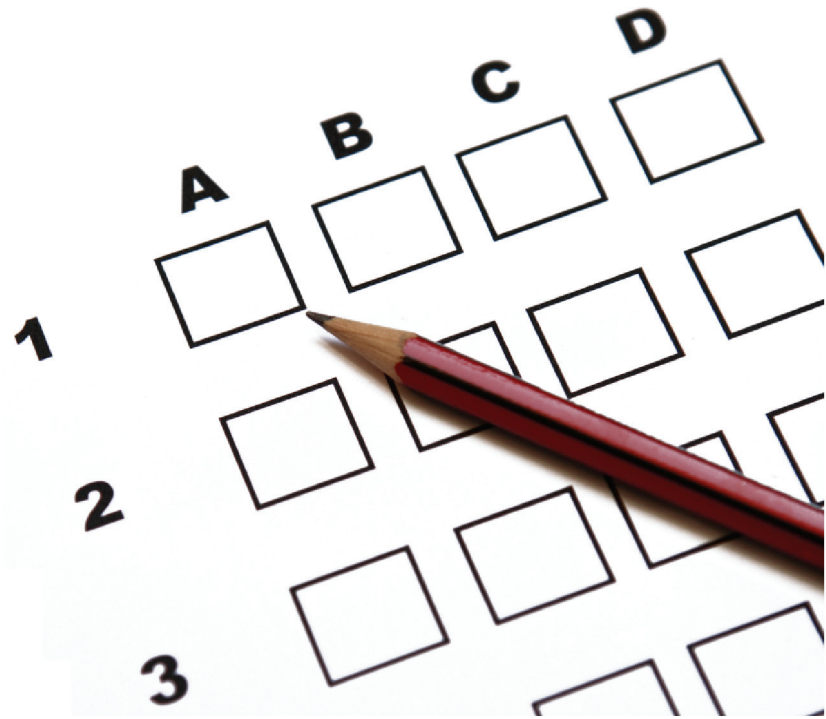

Testing 'bandgap'

Testing 'bandgap'



Unit testing - manual assembly, mocks

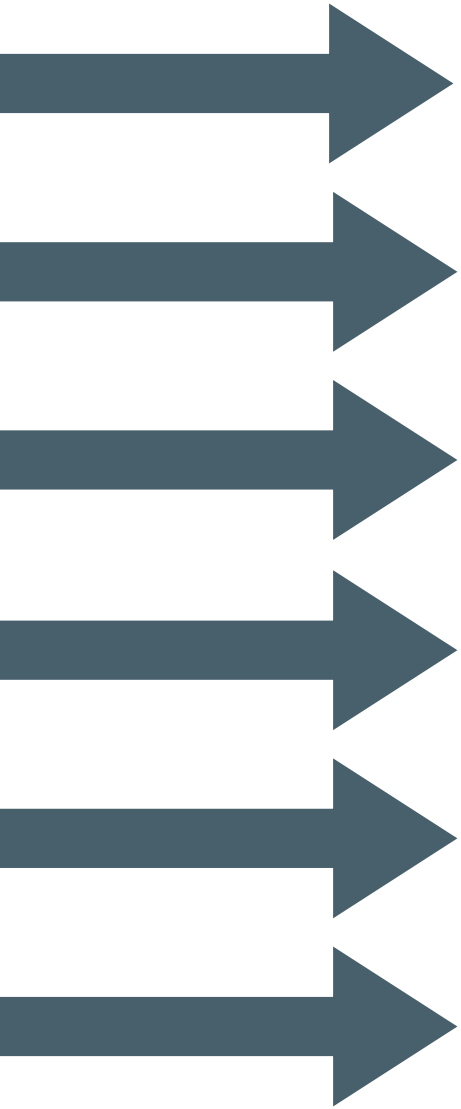
Integration testing - embedded or full container



Experience Ike

A *component model* for your tests

In-container integration testing steps



1. start or connect to container
2. package archive in Java
3. deploy test to container
4. execute test in container
5. capture and report results
6. undeploy test archive

Bring your test to the runtime…

…rather than managing the runtime from your test.

```java
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static Archive<?> createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class)
            .addManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }

    @Inject Greeter greeter;

    @Test
    public void shouldBeAbleToInvokeBean() throws Exception {
        Assert.assertEquals("Hello, Earthlings",
            greeter.greet("Earthlings"));
    }
}
```

```java
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static Archive<?> createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class);
    }

    @EJB Greeter greeter;

    @Test
    public void shouldBeAbleToInvokeBean() throws Exception {
        Assert.assertEquals("Hello, Earthlings",
            greeter.greet("Earthlings"));
    }
}
```

Less (boilerpate) test code

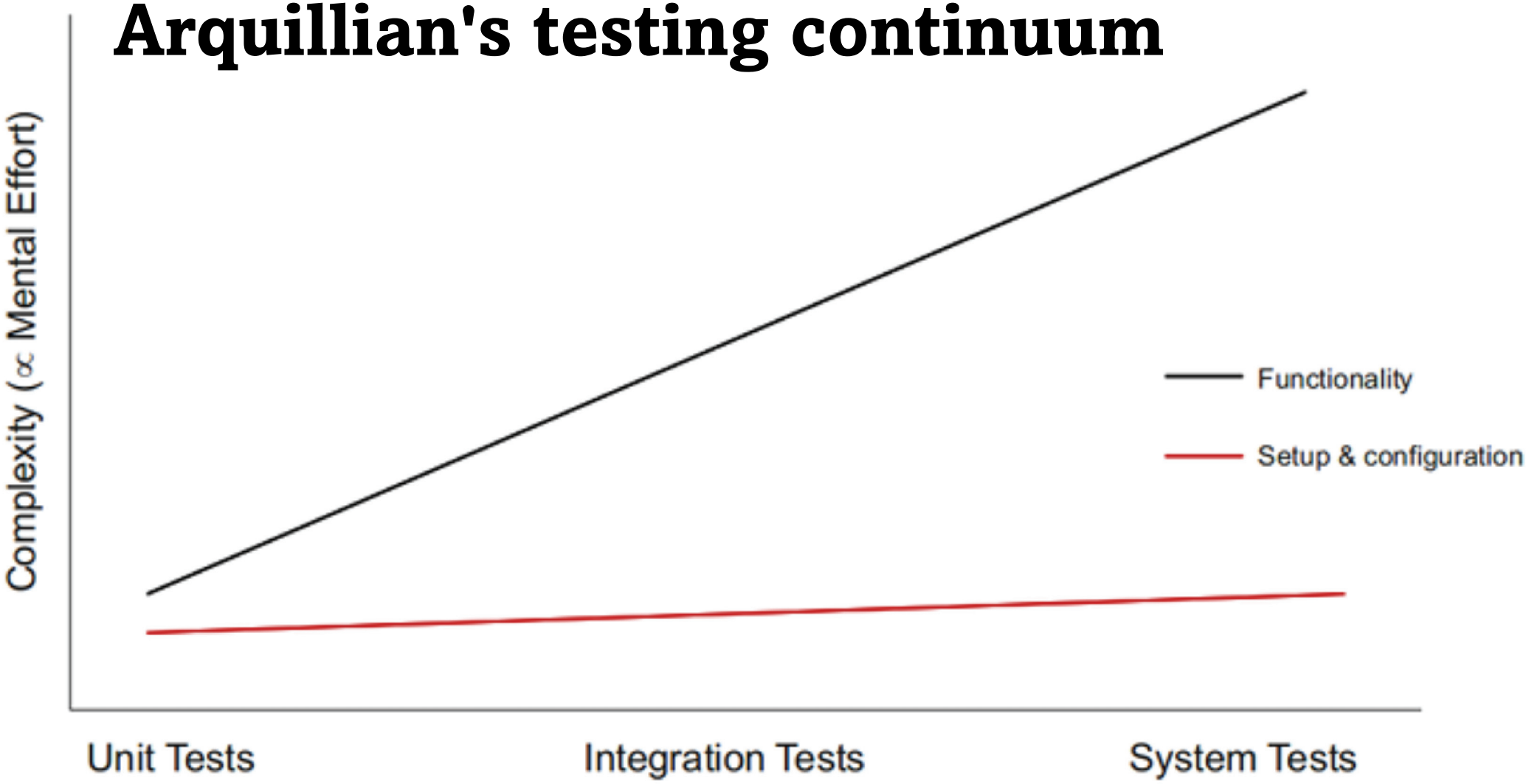
As much or as little integration as you need

Looks like a unit test, but executes in a true environment

Same test can be run in multiple environments

It's a *learning* environment

Arquillian's testing continuum




```
public class GoogleUrlShortener extends UrlShortener {
 }
```

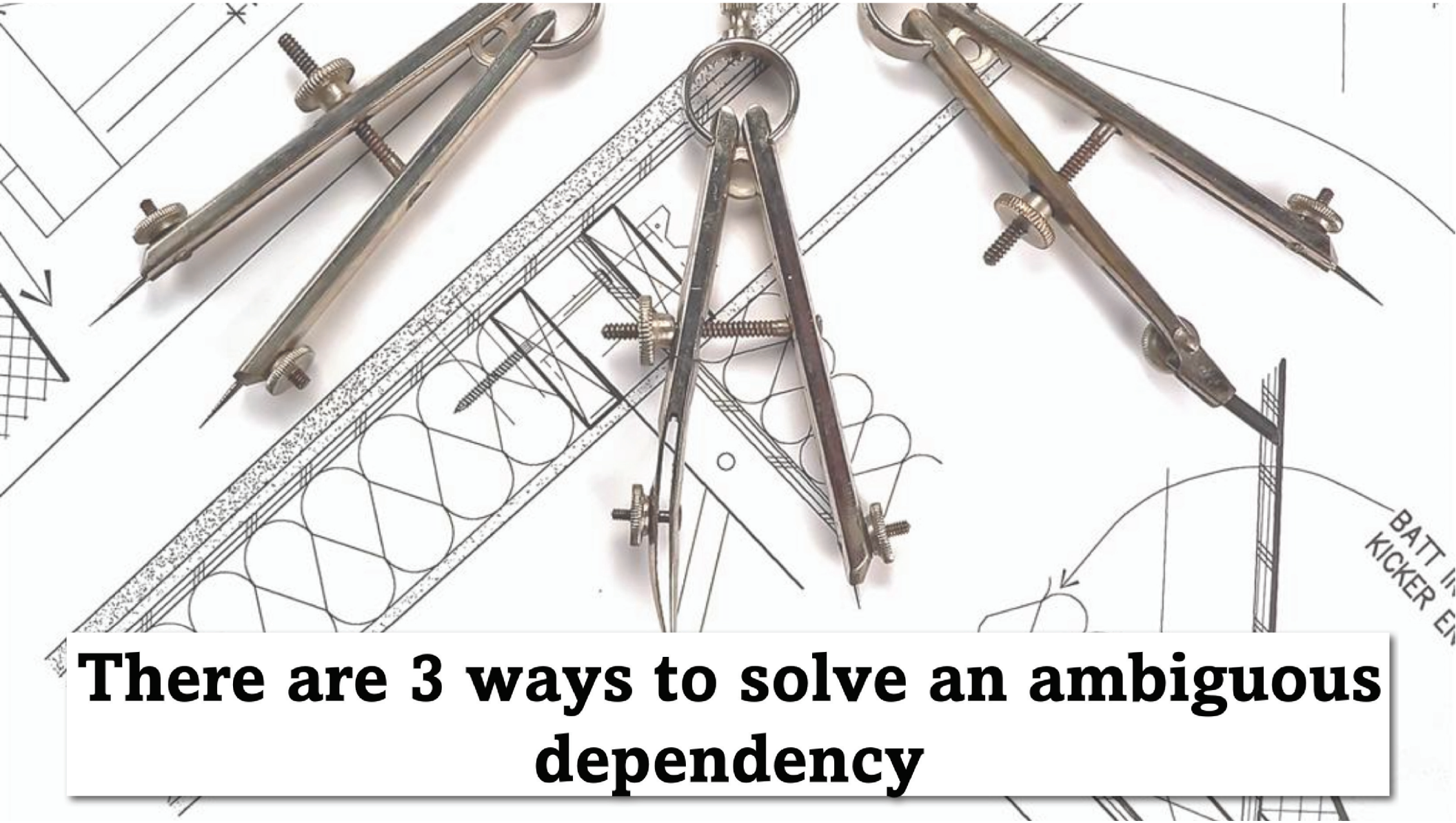
"What happens now?"

```
@Inject UrlShortener shortener;
```

Ambiguous dependency error

org.jboss.weld.exceptions.DeploymentException:

WELD-001409 Ambiguous dependencies for type [UrlShortener] with qualifiers [@Default] at injection point [[field] @Inject private com.socialize.service.StatusUpdater.shortener]. Possible dependencies [[Managed Bean [class com.socialize.service.UrlShortener] with qualifiers [@Any @Default], Managed Bean [class com.socialize.service.GoogleUrlShortener] with qualifiers [@Any @Default]]]



There are 3 ways to solve an ambiguous dependency


3

"What is an @Qualifier?"

Used to select an implementation variant at an injection point when type alone isn't enough

@Qualifier

Qualifiers as binding types

```
@Inject @GoogleUrlShortener;
```

```
@Google
public class Google UrlShortener implements UrlShortener {
}
```

provide context implementation

define scope annotations (e.g., @ViewScoped)

spans multiple requests

request, session, application

default; bound to lifecycle of bean/method defining injection

State management gets 'em talking

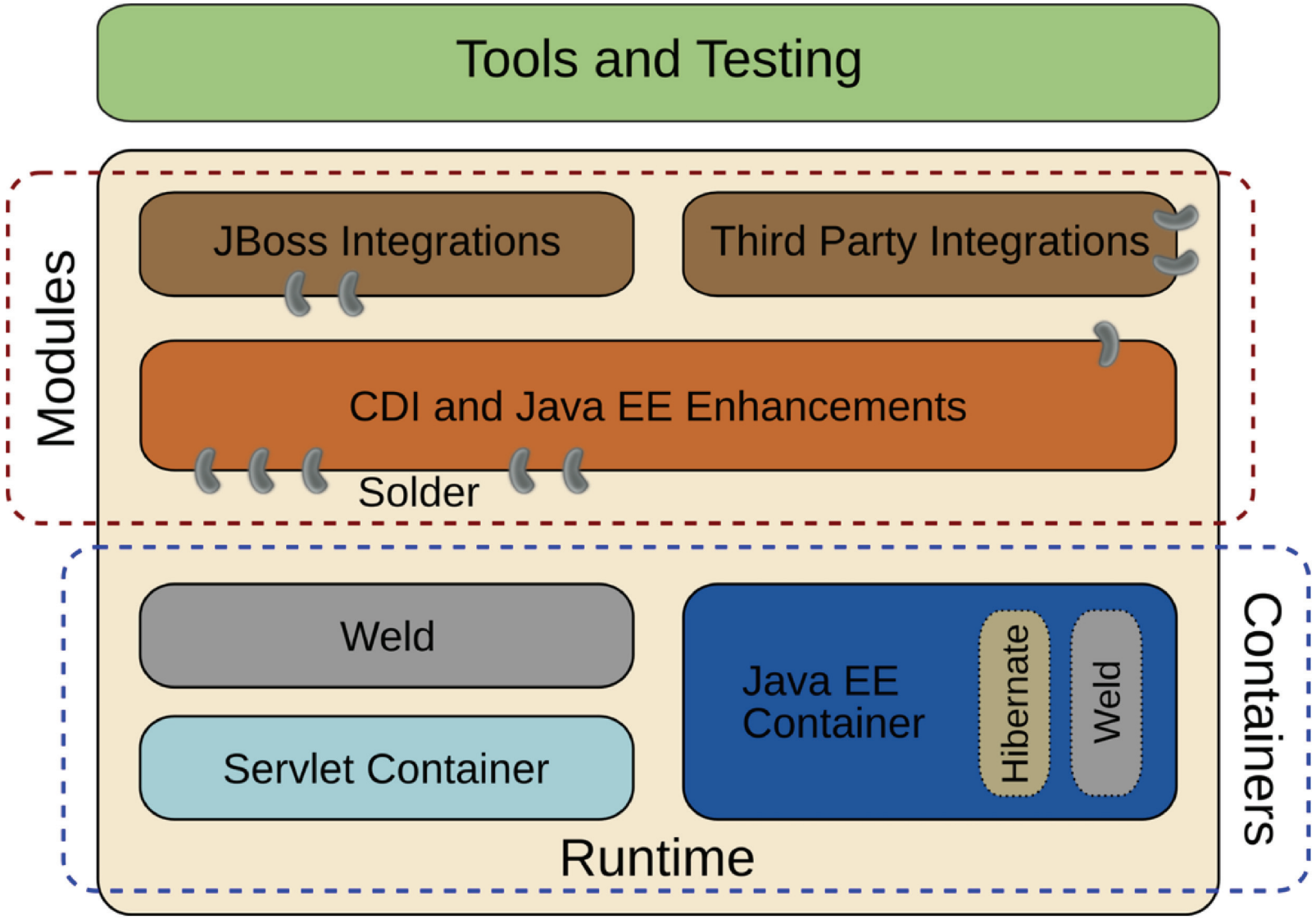

"Like a duck."


@Inteceptors

Seam's Mission

To provide a *fully-integrated* development platform for building rich, **standards-based** enterprise applications tailored for traditional and cloud deployments.

Seam 3 Stack


Tools and Testing
Modules
JBoss Integrations
Third Party Integrations
CDI and Java EE Enhancements
Solder
Weld
Servlet Container
Java EE Container
Hibernate
Weld
Runtime
Containers

github

SOCIAL CODING

seam (Seam)

ORGANIZATION

Name	Seam
Website/Blog	http://seamframework.org/Seam3
Location	Global
Member Since	Jul 13, 2010

26 public repos

22 Members

Public Repositories (26)

module ×

config

Java 8 5

Config Module: A portable extension for Java EE that provides alternate bean metadata providers, ...

Last updated about 4 hours ago

international

Java 5 4

International Module: A portable extension for Java EE that provides a unified approach to langua...

Last updated about 21 hours ago

Organization Members (22)

aslakknutsen (Aslak Knutsen)
16 Public Repositories, 20 followers

bleathem
3 Public Repositories, 1 followers

codylerum
2 Public Repositories, 1 followers

cpopetz (Clinton Popetz)
2 Public Repositories, 1 followers

gunnarmorling (Gunnar Morling)
8 Public Repositories, 3 followers

jganoff (Jordan Ganoff)
2 Public Repositories, 3 followers

Seam a la carte

Individual module artifacts:

Module distribution:

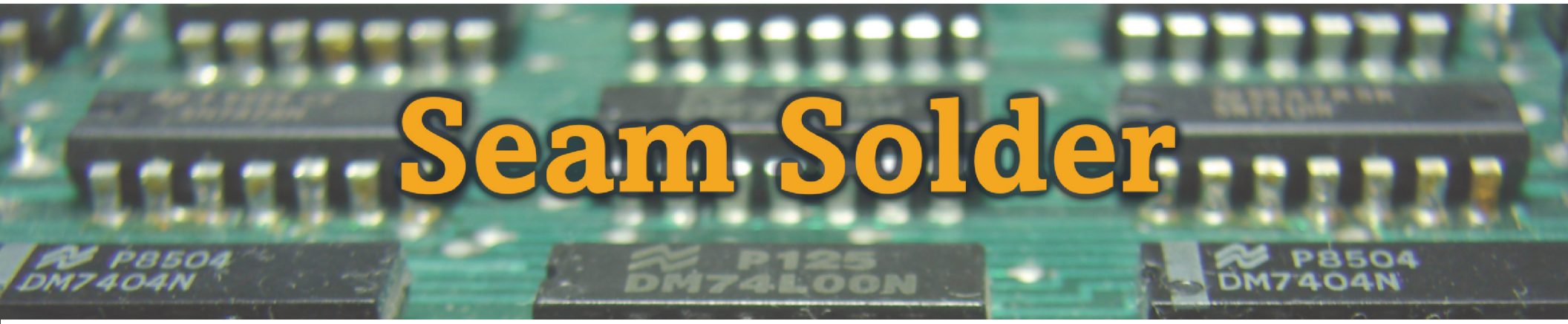
Highlight reel

01 02 03 04 05

Seam Catch

Seam Faces

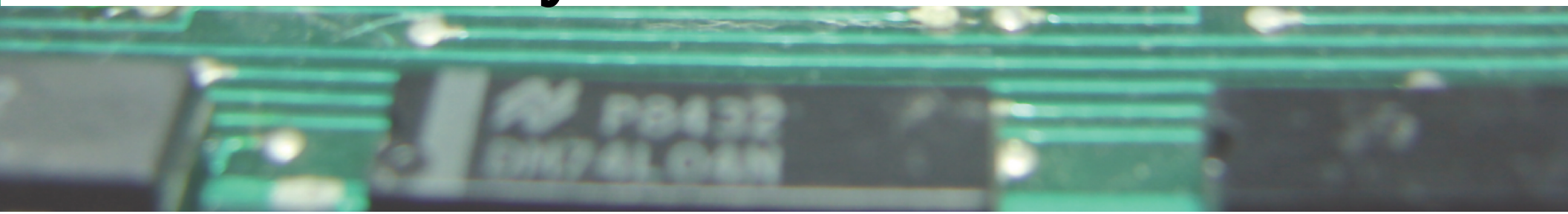
PHOTOGRAPHY PHOTOS SHOOTS IMAGES PHOTOGRAPHY PHOTOS SHOOTS IMAGES PHOTOGRAPHY PHOTOS SHOOTS IMAGES PHOTOGRAPHY PHOTOS SHOOTS IMAGES PHOTOGRAPHY PHOTOS SHOOTS IMAGES



Foundation Seam module

Generally usefully stuff for CDI applications

A swiss army knife for extension writers



A new breed of logger


Category as annotation
Exception codes
Type-safe loggers
Language support
An abstraction layer with fresh ideas


Addresses real-world scenarios
Developers code in Java
Translators edit message bundles

Loggers are serializable


@Inject

@Inject ~~Entity~~Manager

Seam Config: XML-based bean metadata

define

customize

wire

"Type-safe"

- Reference fully qualified types
- Import packages via XML namespaces

Keys to Seam 3 extensions and CDI

Focus on your business

CDI combines loose coupling with strong typing

CDI makes Java EE flexible, portable and extensible

Seam 3 rounds out Java EE

- **enhancements**
- **integrations**
- **tooling**

Download a Java EE 6 container:

JBoss AS 6 - http://jboss.org/jbossas

GlassFish V3 - http://glassfish.org

Create a Java EE 6 project:

Maven archetypes - http://tinyurl.com/goweld (TODO gojavaee)

JBoss Tools - http://jboss.org/tools

Add on a Seam 3 module:

http://seamframework.org/Seam3

Contact Information

email: dan.j.allen@gmail.com

twitter: @mojavelinux

website: mojavelinux.com

JBoss blog: community.jboss.org/people/dan.j.allen/blog