Hacking Java EE

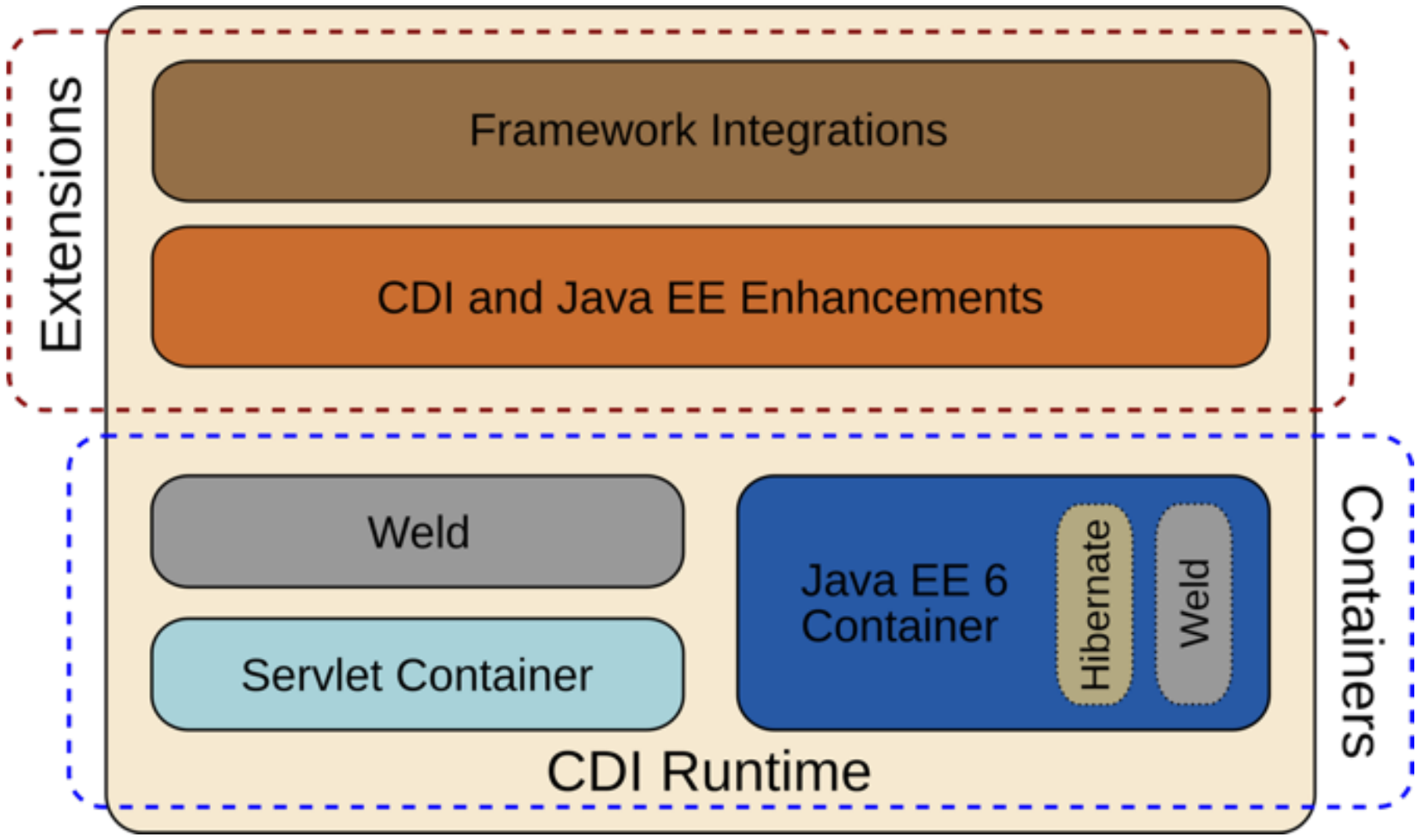
Pete Muir (@plmuir)
Principal Software Engineer
Red Hat, Inc.

June 2011

Building on Common Ground

Pete Muir

Building on Common Ground



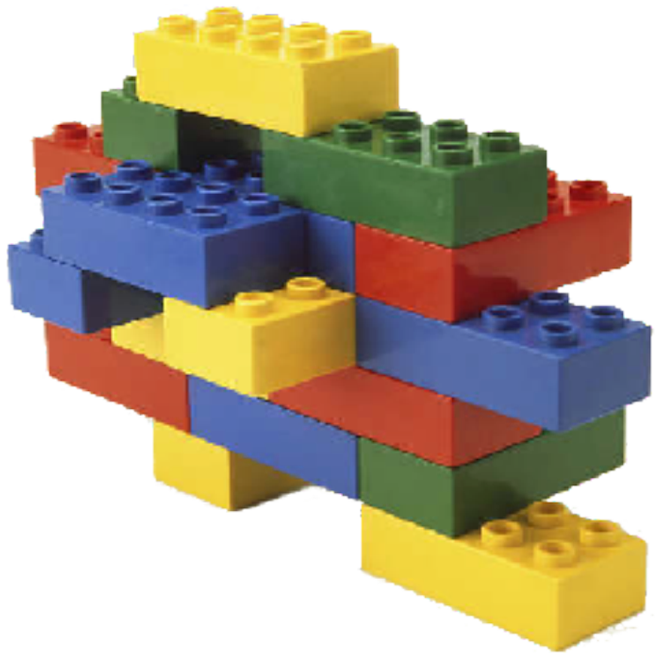
JBoss Community

JBoss Community

Pete Muir

SPI for hacking Java EE:

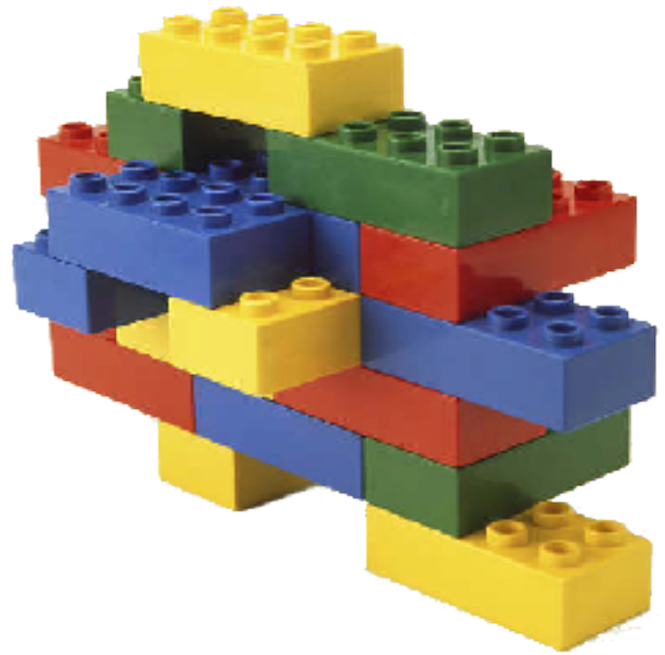
- register additional beans
- satisfy injection points



- introduce custom scopes
- augment or override bean metadata

SPI for hacking Java EE:

- ☑ register additional beans
- ☑ satisfy injection points



- ☑ introduce custom scopes
- ☑ augment or override bean metadata

How does it work?

Based on Service Provider Interface (SPI) pattern

How does it work?

Based on Service Provider Interface (SPI) pattern

Implement empty Extension interface

How does it work?

Based on Service Provider Interface (SPI) pattern

1. Implement empty Extension interface
2. Observe container events to alter deployment

How does it work?

Based on Service Provider Interface (SPI) pattern

1. Implement empty Extension interface
2. Observe container events to alter deployment
3. Register extension

JBoss Community

CDI deployment lifecycle with possible activities

JBoss Community

Pete Muir

CDI deployment lifecycle with possible activities

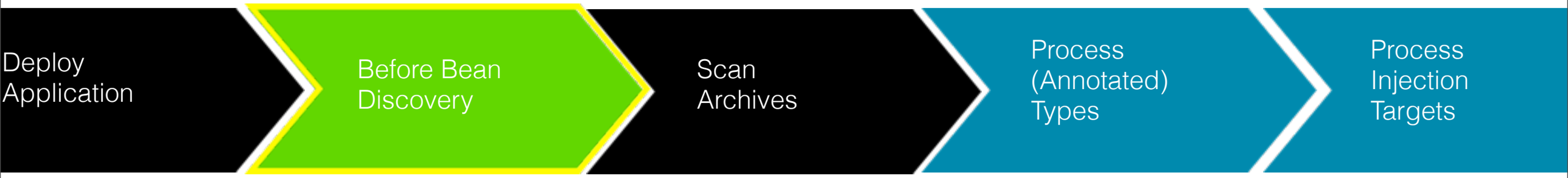
CDI deployment lifecycle with possible activities

Deploy Application

Before Bean Discovery

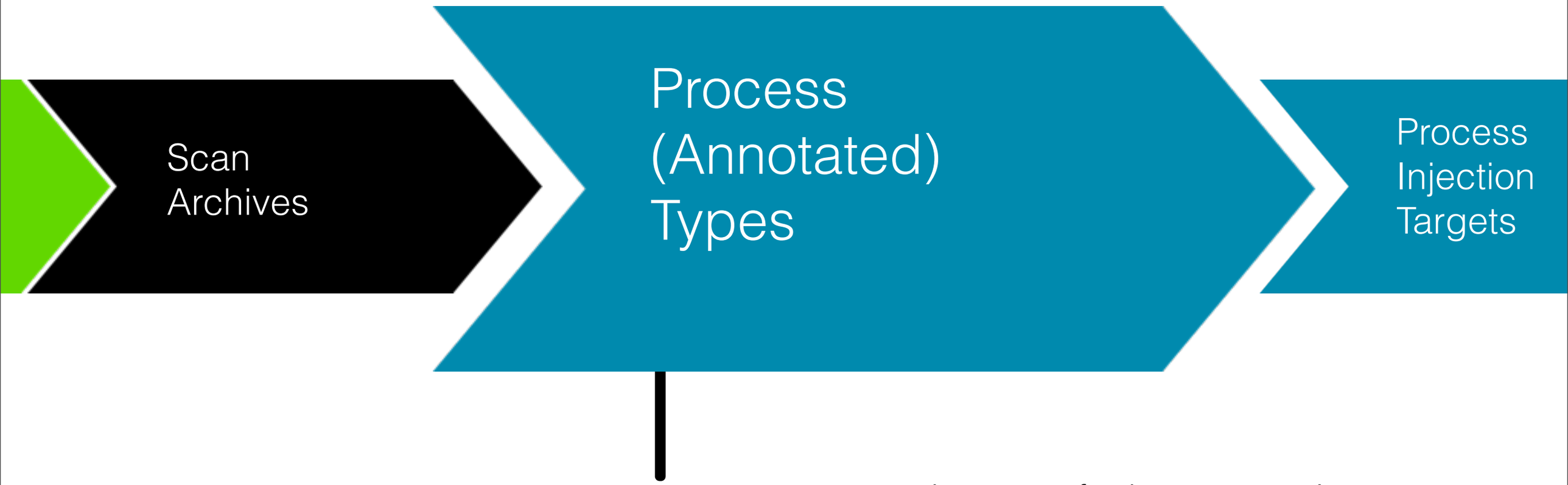
Scan Archives

+ add scope
+ add annotated type
+ add qualifier
+ add interceptor binding
+ add stereotype


Deploy Application
Before Bean Discovery
Scan Archives
Process (Annotated) Types
Process Injection Targets


Deploy Application
Before Bean Discovery
Scan Archives
Process (Annotated) Types
Process Injection Targets


Deploy Application
Before Bean Discovery
Scan Archives
Process (Annotated) Types
Process Injection Targets



+ veto types and prevent further processing

+ replace annotated type

Process (Annotated) Types

Process Injection Targets

Process Beans

Process Producers

Process Observer Methods

JBoss Community

Pete Muir

Process (Annotated) Types

Process Injection Targets

Process Beans

Process Producers

Process Observer Methods

JBoss Community

Pete Muir


Process
(Annotated)
Types
Process Injection Targets
Process
Beans
replace injection target

Process (Annotated) Types

Process Injection Targets

Process Beans

Process Producers

Proces Observ Methoc

JBoss Community

Pete Muir


Process
(Annotated)
Types
Process
Injection
Targets
Process
Beans
Process
Producers
Proces
Observ
Methoc


Process
Injection
Targets
Process
Beans
Process
Producers
prepare additional beans


Process
Injection
Targets
Process
Beans
Process
Producers
Process
Observer
Methods
After
Disco

JBoss Community

Pete Muir

JBoss Community

Pete Muir

Process Producers

Process Observer Methods

After Bean Discovery

+ add bean

+ add observer method

+ add context

ean
ery

After
Deployment
Validation

Application
Running

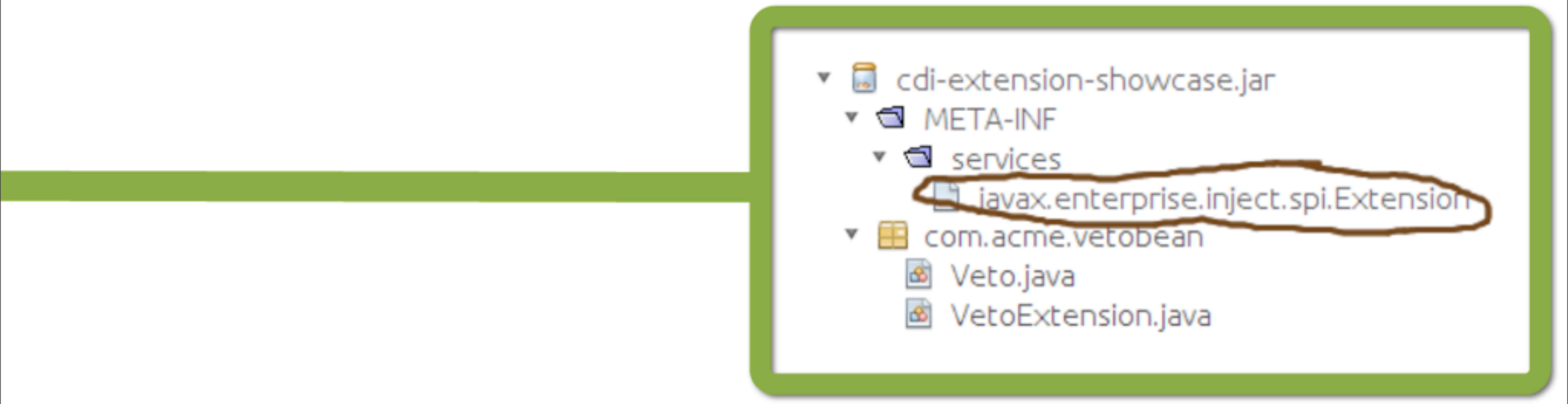
Before
Shutdow

+ assessment

+ cleanup


After Deployment Validation
Application Running
Before Shutdown
Undeploy Application


When is an extension recognized?



META-INF/services/javax.enterprise.inject.spi.Extension

```
com.acme.vetobean.VetoExtension
```

Extensions

"All we have to decide is what to do with the beans that are given to us."

Extending Java EE

- Portable extensions
- Built-in components: beans, producers interceptors, decorators, observers
- Entity listeners
- System event listeners
- Phase listeners
- Servlet filters
- Request listeners
- Servlet context listeners

AnnotatedType

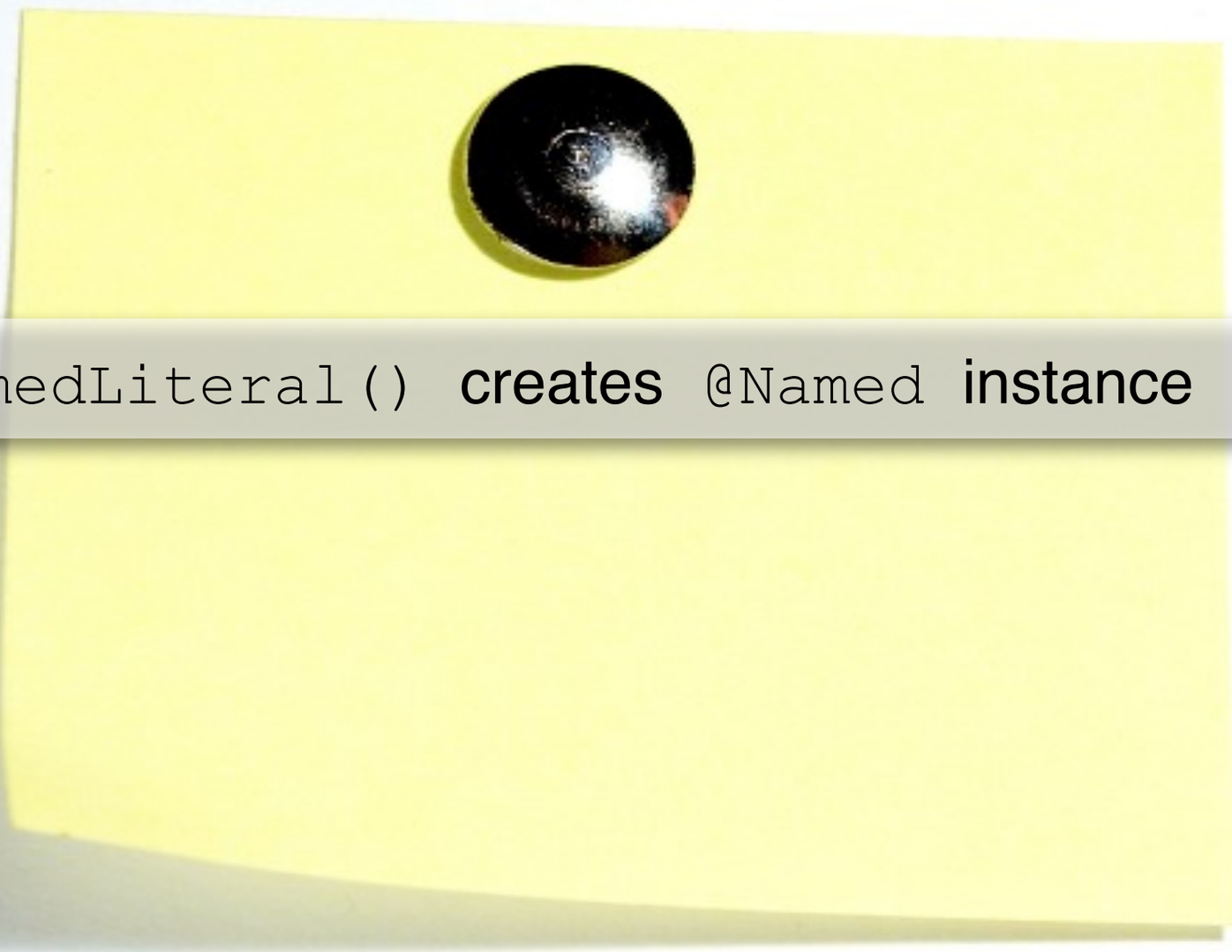
Bean

Substituting bean metadata


I think I need to create an annotation...

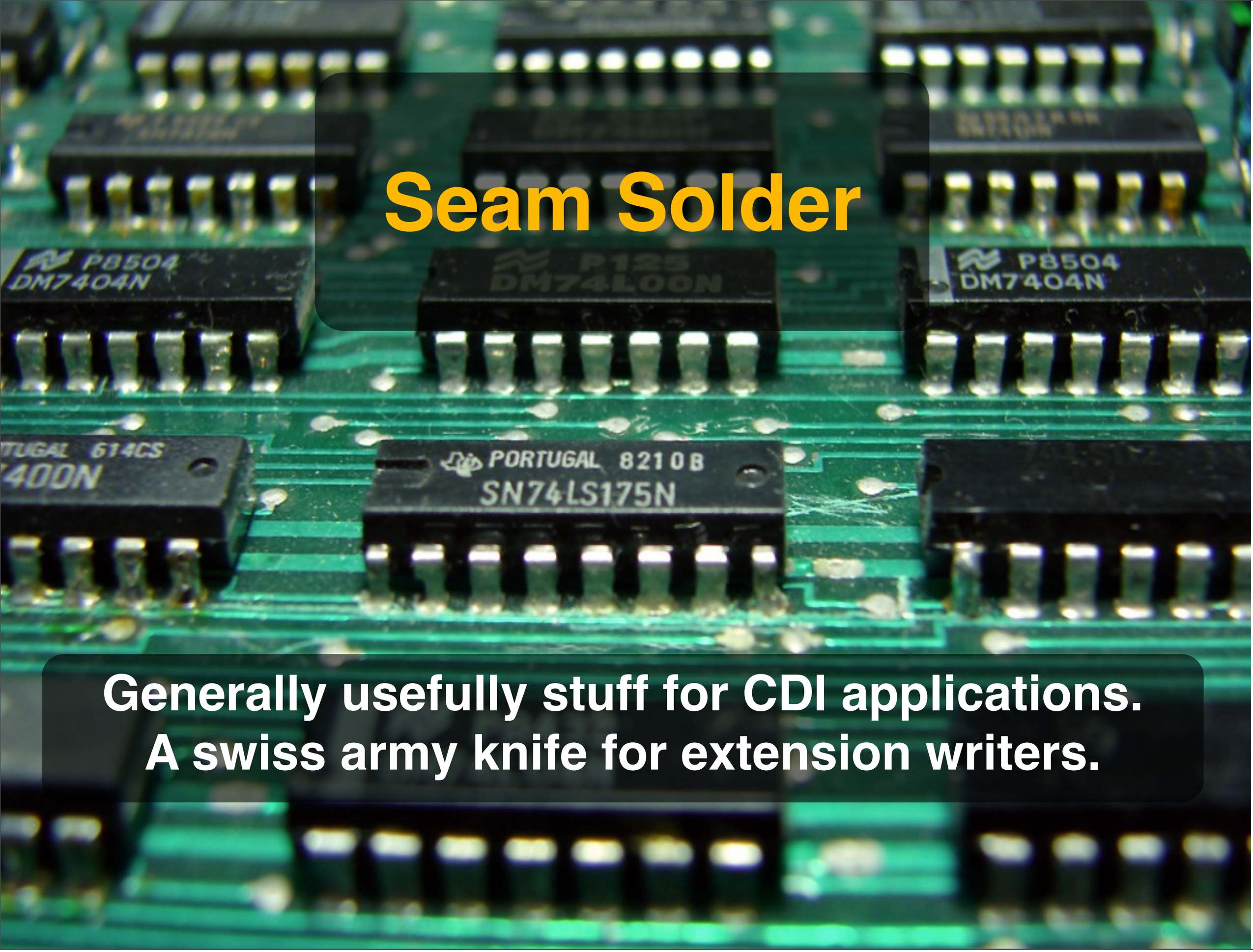
`new AnnotationLiteral<X>() {}` creates `annotation X` instance

`new AnnotationLiteral<Named>() {}` creates `@Named` instance



`new NamedLiteral()` creates `@Named` instance

Seam Solder



Generally usefully stuff for CDI applications.
A swiss army knife for extension writers.

AnnotatedType builder

Bean builders

Annotation inspectors

Reflection utilities

te Muir

BeanManager locator

Annotation literals
(for standard annotations)

Resource loading

Method injector

Pete Muir


Ah, but can we test it?

Pete Muir

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

   @Inject Greeter greeter;

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

Friday, 17 June 2011

Extension Showcase

- Alias annotations
- Veto beans
- Register beans from third-party library
- Introduce behaviors to built-in annotations
- Override injection point types
- Bridge Servlet events to CDI
- Expose standard component as beans
- Emulate EJB services for managed beans
- Set conversation boundaries declaratively
- Initialize beans on application startup

Who's writing extensions?

Seam 3 project - http://seamframework.org/Seam3

MyFaces CODI - http://www.irian.at/myfaces_codi

SoftwareMill - https://github.com/softwaremill/softwaremill-common

List your extension @
http://tinyurl.com/cdi-extensions

JBoss Community

Pete Muir

CDI combines loose coupling with strong typing.

CDI makes Java EE flexible, portable and extensible.

CDI extensions make Java EE competitive and awesome!

Pete Muir

redhat.

Q & A

email: pete.muir@jboss.org

twitter: @plmuir

blog: http://in.relation.to/Bloggers/Pete