~~When~~ Camel meets CDI

Łukasz Dywicki

Goals

- Determine state of DI support in Camel
- Possible extension points to use
- Transition from Spring to CDI
- Show running CDI + Camel application

Who am I?

- Łukasz Dywicki
  - Independent software contractor
- ServiceMix user since 2008
- Commiter of Apache Karaf
- Camel contributor

Camel + Red Hat

Brief introduction – CDI, JSR-299

Brief introduction – CDI

- Built on top of JSR-330
- Package javax.inject
  - @Inject
  - @Named
  - @Qualifier
  - @Scope
  - @Singleton
  - Provider<T>

Brief introduction – CDI cont.

```
@WebServlet("/greeting.do")
public class GreetingServlet extends HttpServlet {

    @Inject
    private Greeter greeter;

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
        throws ServletException, IOException {
        resp.getWriter().write(
            greeter.greet(req.getParameter("who"))
        );
    }

}
```

Brief introduction – CDI cont.

- CDI contains more elements specific to full featured DI container
  - Stereotypes
  - Application/Session/Request Scope
  - Bean Registry*
- Possibility to implement own scopes
- It's type safe by definition
- Handles dependent beans creation
- Lifecycle management with JSR-250

Brief introduction – CDI cont.

- Discovery of beans rely on presence of beans.xml file in META-INF or WEB-INF
- Container discovers beans automatically
- CDI is extensible
  - Extension interface
  - META-INF/services entry

Support for Dependency Injection in Camel

Support for DI in Camel

- XML based
  - Spring
  - Blueprint
- Annotation based
  - Guice
  - Spring JavaConfig

Support for DI in Camel

- XML based
  - Spring
  - Blueprint
- Annotation based
  - Guice
  - Spring JavaConfig
  - CDI

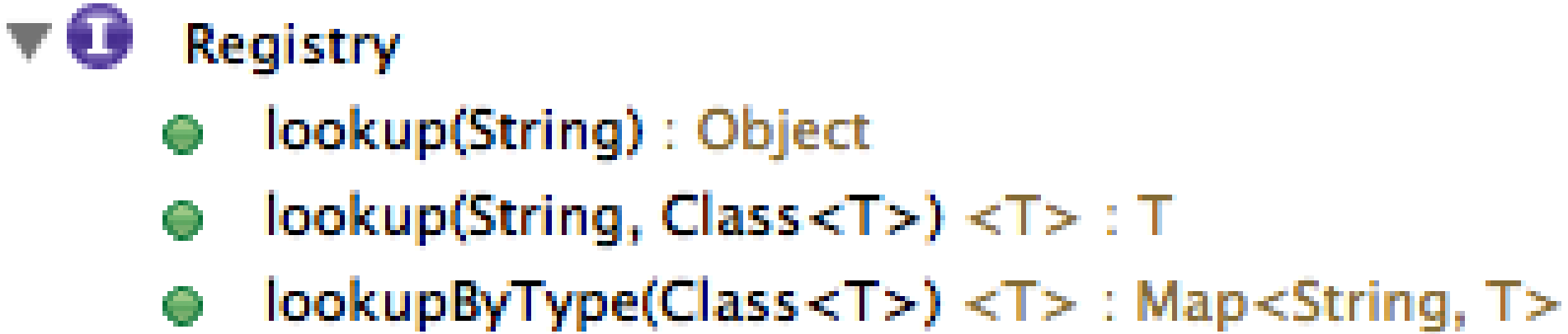
Some pictures for fun

Back to Camel

Support for DI in Camel cont.

- Registry
- Injector

Registry


Registry
lookup(String) : Object
lookup(String, Class<T>) <T> : T
lookupByType(Class<T>) <T> : Map<String, T>

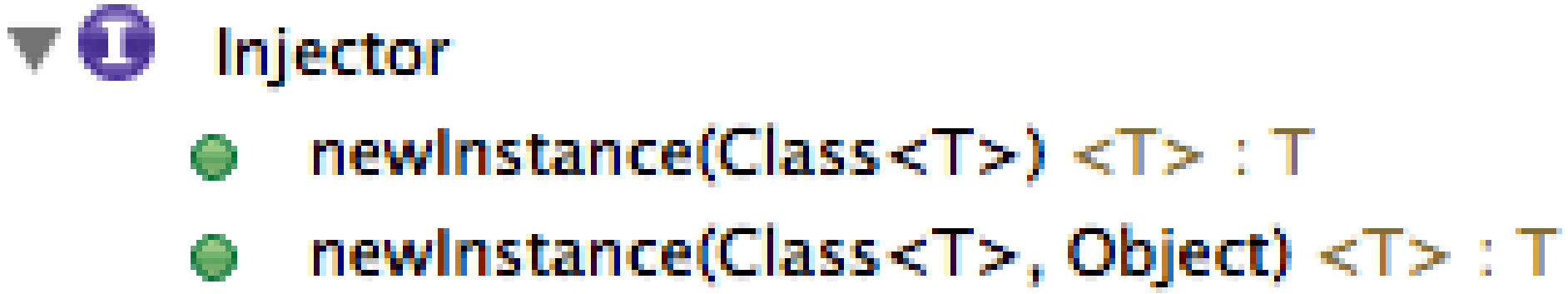
Core Registries

- CompositeRegistry
- SimpleRegistry
- JndiRegistry
- PropertyPlaceholderDelegateRegistry

3rd Party Registries

- ApplicationContextRegistry
- BlueprintContainerRegistry
- OsgiServiceRegistry
- CdiBeanRegistry

Injector


Injector
newInstance(Class<T>) <T> : T
newInstance(Class<T>, Object) <T> : T

Core Injectors

- ReflectionInjector
- DefaultInjector

3rd party injectors

- SpringInjector
- GuiceInjector
- CdiInjector

Transition from Spring to CDI

Transition from Spring to CDI

- No ApplicationContext instance
  - How do I get bean from it?
- No pre/post processors
  - CamelContextAware
- Multiple camel contexts
  - Linking beans with camel contexts
- No domain specific namespaces

How do I get bean

- Use Apache DeltaSpike BeanProvider class:
  - getContextualReference(String) = Object
  - getContextualReference(Class<T>) = T
  - getBeanDefinition(Class<T>) = Set<Bean<T>>
  - getContextualReferences(Class<T>) = List<T>

CamelContextAware

```
protected void contextAwareness(@Observes ProcessAnnotatedType<CamelContextAware> process)
    throws Exception {
    AnnotatedType<CamelContextAware> annotatedType = process.getAnnotatedType();
    Class<CamelContextAware> javaClass = annotatedType.getJavaClass();
    if (CamelContextAware.class.isAssignableFrom(javaClass)) {
        Method method = javaClass.getMethod("setCamelContext", CamelContext.class);
        AnnotatedTypeBuilder<CamelContextAware> builder = new AnnotatedTypeBuilder<CamelContextAware>()
                .readFromType(javaClass)
                .addToMethod(method, new InjectLiteral());
        process.setAnnotatedType(builder.create());
    }
}
```

Events

- BeforeBeanDiscovery
- AfterBeanDisovery
- ProcessBean
- ProcessInjectionTarget
- ProcessObserverMethod
- AfterDeploymentValidation
- BeforeShutdown

Linking beans

```
@ContextName("contextD")¶
public class RoutesContextD extends RouteBuilder {¶
¶
    @Inject @Uri("seda:D.a")¶
    Endpoint a;¶
¶
    @EndpointInject(uri = "mock:D.b")¶
    MockEndpoint b;¶
¶
    @Inject @Uri("seda:D.a")¶
    ProducerTemplate producer;¶
¶
    @Override¶
    public void configure() throws Exception {¶
        from(a).to(b);¶
    }¶
¶
}¶
```

Demo application

Common problems

Problems ...

- Domination of Spring style DI
- Multiple contexts vs simplicity of extension
- Magic done inside Camel
- Configuring Camel Context bean

TODO

- Support for discovering
  - Intercept Strategies
  - Event Notifiers
  - Executors
  - Resolvers
- Transactional services without spring-tx

Any questions?

Thank you for coming!