Java9 Modules

BestSolution

Tom Schindl <tom.schindl@bestsolution.at>

Twitter: @tomsontom
Blog: http://tomsondev.bestsolution.at
Website: http://www.bestsolution.at

About Tom

BestSolution

- CTO BestSolution.at Systemhaus GmbH
- Eclipse Committer
  - e4
  - Platform
  - EMF
- Project lead
  - e(fx)clipse
- Twitter: @tomsontom
- Blog: tomsondev.bestsolution.at
- Corporate: http://bestsolution.at

Targets of Java9 Modulesystem

- Split up rt.jar in smaller junks
  - Avoids further introduction of cross-references
  - Allows smaller Java-Runtimes (eg for IoT-Devices, …)
- Restrict Access to internal APIs (eg com.sun.misc.Unsafe)
- Allow developers to adopt it for their code as well

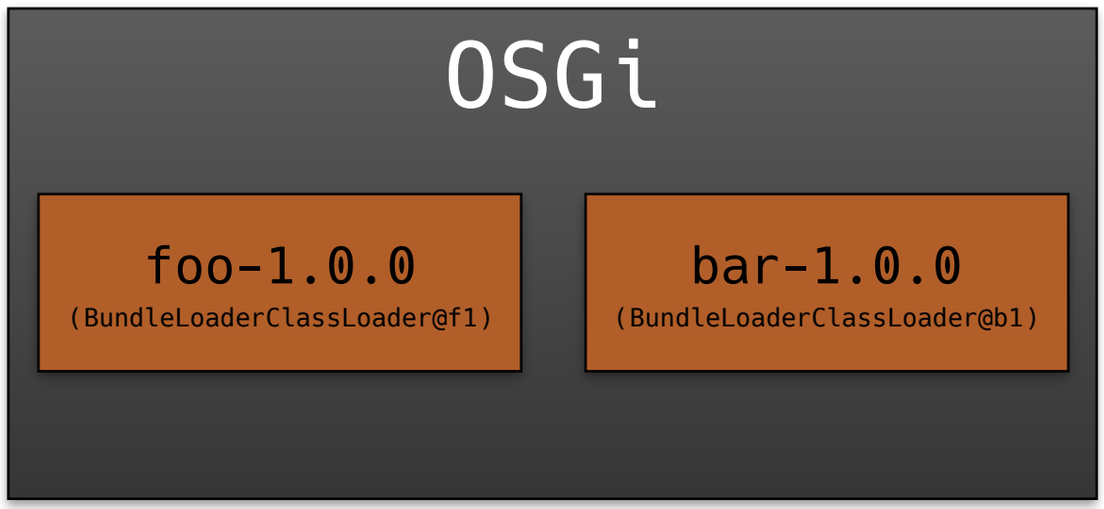
Java9 Modules vs OSGi

- The Implementation
  - OSGi Module System is built upon classloaders
  - Java9 Module System is implemented at the VM Level

Java9 Modules vs OSGi

BestSolution

- The Implementation
  - OSGi Module System is built upon classloaders
  - Java9 Module System is implemented at the VM Level



Java9 Modules vs OSGi

BestSolution

- The Implementation
  - OSGi Module System is built upon classloaders
  - Java9 Module System is implemented at the VM Level

Java9 Modules vs OSGi

- Dependency definition
  - OSGi uses custom MANIFEST.MF entries **Require-Bundle** and **Import-Package**
  - Java9 uses module-info.java and directive **requires**

Java9 Modules vs OSGi

BestSolution

- Dependency definition
  - OSGi uses custom MANIFEST.MF entries **Require-Bundle** and **Import-Package**
  - Java9 uses module-info.java and directive **requires**

OSGi

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: bar
Require-Bundle: foo
```

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: bar
Import-Package: foo
```

Java9 Modules vs OSGi

BestSolution

- Dependency definition
  - OSGi uses custom MANIFEST.MF entries **Require-Bundle** and **Import-Package**
- Java9 uses module-info.java and directive **requires**

OSGi

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: bar
Require-Bundle: foo
```

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: bar
Import-Package: foo
```

Java9

```
module bar {
  requires foo;
}
```

Java9 Modules vs OSGi

- Access Restrictions
  - OSGi uses custom **MANIFEST.MF** entry **Export-Package**
  - Java9 use module-info.java and directive **exports**

Java9 Modules vs OSGi

- Access Restrictions
  - OSGi uses custom **MANIFEST.MF** entry **Export-Package**
  - Java9 use module-info.java and directive **exports**

OSGi

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: foo
Export-Package: foo
```

Java9 Modules vs OSGi

- Access Restrictions
  - OSGi uses custom **MANIFEST.MF** entry **Export-Package**
  - Java9 use module-info.java and directive **exports**

OSGi

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: foo
Export-Package: foo
```

Java9

```
module foo {
  exports foo;
}
```

Java9 Modules vs OSGi

- Services
  - OSGi has eg Declarative Service Components and the BundleContext to lookup

Java9 Modules vs OSGi

BestSolution

- Services
  - OSGi has eg Declarative Service Components and the BundleContext to lookup

Provider

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: bar
Require-Bundle: foo
Service-Component: OSGI-INF/mycomponent.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0">
   <implementation class="bar.BarService"/>
   <service>
      <provide interface="foo.Service"/>
   </service>
</scr:component>
```

Java9 Modules vs OSGi

BestSolution

- Services
  - OSGi has eg Declarative Service Components and the BundleContext to lookup

Provider

```
Manifest-Version: 1.0
Bundle-ManifestVersion: 2
Bundle-SymbolicName: bar
Require-Bundle: foo
Service-Component: OSGI-INF/mycomponent.xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<scr:component xmlns:scr="http://www.osgi.org/xmlns/scr/v1.1.0">
   <implementation class="bar.BarService"/>
   <service>
      <provide interface="foo.Service"/>
   </service>
</scr:component>
```

Consumer

```
BundleContext ctx = getContext(cl);
ServiceReference<Service> ref =
  ctx.getServiceReference(Service.class);
Service s = ctx.getService(ref);
```

Java9 Modules vs OSGi

BestSolution

- Service
  - Java9 uses module-info.java and **ServiceLoader** for the lookup

Provider

```
module bar {
     requires foo;
     provides foo.Service with bar.BarService;
}
```

Consumer

```
module foo {
     requires foo;
     uses foo.Service;
}
```

```
ServiceLoader<GreetService> l =
   ServiceLoader.load(Service.class);

Service s = l.iterator().next();
```