

EclipseLink/Hibernate with Karaf

- [Introduction](#)
- [JAVA Persistence API Artifact](#)
 - [Problem](#)
 - [Solution](#)
- [Link EclipseLink with the JPA/JTA-Container](#)
 - [Problem](#)
 - [Solution](#)
- [ClassNotFoundExceptions](#)
 - [Problem](#)
 - [Solution](#)

Introduction

This article describes how to get the most important Java Persistence API (JPA) implementations, EclipseLink and Hibernate, running together within a single Apache Karaf. The article describes why this is not possible with the current Karaf enterprise feature definitions and what has to be done to make it work.

JAVA Persistence API Artifact

Problem

The JAVA Persistence API in version 2.1 is described by Oracle in the document JSR-000338. If you like to take a look at the specification you can use the following link:

http://download.oracle.com/otndocs/jcp/persistence-2_1-fr-eval-spec/index.html

Normally Oracle provides an API artifact for their specifications. The most popular example of this is the Java Database Connectivity (JDBC). This API is implemented by different database providers like Oracle, PostgreSQL and Microsoft. All of these database providers implement the JDBC API artifact provided by Oracle. However Oracle doesn't provide an artifact for JPA. As a consequence a lot of different artifacts exist. These are provided by organisations like Hibernate or Eclipse. The most popular ones are the Hibernate JPA API artifact (org.hibernate.javax.persistence/hibernate-jpa-2.1-api) and the EclipseLink JPA API artifact (org.eclipse.persistence/javax.persistence/2.1.0).

It's not necessary to have an API artifact from Oracle as long as the API artifacts provided by the organisations export the 'javax.persistence.*' packages with the specific version number (e.g. 2.1.0) and comply with the specification. In this case it would be possible to compile the JPA implementations of Hibernate and EclipseLink against all JPA API artifacts. It would also be possible to wire the implementations against all API artifacts at runtime. However, this is not possible in the real world. To understand why, we have to analyse the import and export section of the MANIFEST files of the JPA API and the implementation artifacts of Hibernate and EclipseLink.

The Hibernate JPA API artifact exports all 'javax.persistence.*' packages with the specific version number (e.g. 2.1.0), but the EclipseLink JPA API artifact is slightly different. While all 'javax.persistence.*' packages are also

exported with the specific version number, an additional option `jpa="2.1"` is added between the package name and the version number as shown below.

JAVA Persistence API Artifact - Hibernate

```
Export-Package: javax.persistence;uses:="javax.persistence.metamodel,javax.persistence.criteria,javax.persistence.spi";version="2.1.0",javax.persistence.criteria;uses:="javax.persistence.metamodel,javax.persistence";version="2.1.0",javax.persistence.metamodel;version="2.1.0",javax.persistence.spi;uses:="javax.persistence,javax.sql";version="2.1.0"
Implementation-Title: Java Persistence API
```

JAVA Persistence API Artifact - EclipseLink

```
Manifest-Version: 1.0
Export-Package: javax.persistence;jpa="2.1";version="2.1.0",javax.persistence.criteria;jpa="2.1";version="2.1.0",javax.persistence.metamodel;jpa="2.1";version="2.1.0",javax.persistence.spi;jpa="2.1";version="2.1.0",org.osgi.service.jpa;version="1.1.0"
Implementation-Version: 2.1.0
Bundle-ClassPath:
```

The Hibernate JPA implementation imports all `'javax.persistence.*'` packages with the specific version number. The EclipseLink JPA implementation imports all `'javax.persistence.*'` packages with the specific version number and adds the option `jpa="2.1"` between the package name and the version number.

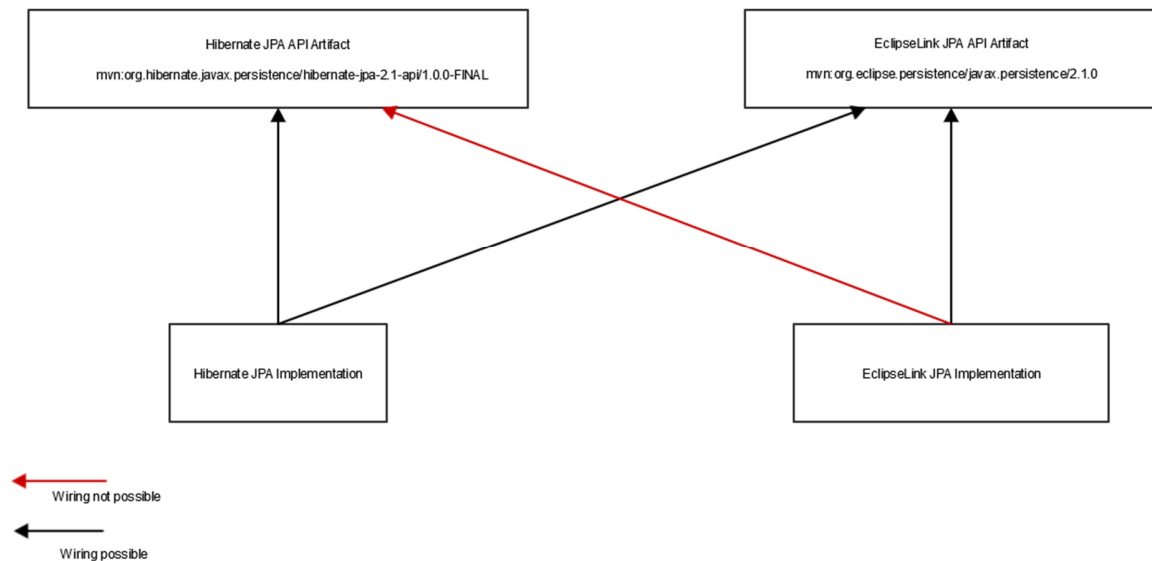
JPA Implementation Artifact - Hibernate

```
Import-Package: javax.persistence;version="2.1.0",javax.persistence.criteria;version="2.1.0",javax.persistence.metamodel;version="2.1.0",javax.persistence.spi;version="2.1.0",javax.transaction;version="(1.1,2)",javassist.bytecode;version="(3.18,4)",javassist.bytecode.annotation;version="(3.18,4)",javax.naming,javax.sql,javax.xml.parsers,javax.xml.transform,javax.xml.transform.dom,javax.xml.transform.stream,javax.xml.validation,org.hibernate;version="[4.3,5)",org.hibernate.annotations.common;version="[4.0,5)",org.hibernate.annotations.common.reflection
```

JPA Implementation Artifact - EclipseLink

```
Created-By: 1.6.0_21 (Sun Microsystems Inc.)
Import-Package: javax.naming;resolution:=optional,javax.persistence;jpa="2.1";version="[1.1.0,2.2)",javax.persistence.criteria;jpa="2.1";version="[1.1.0,2.2)",javax.persistence.metamodel;jpa="2.1";version="[1.1.0,2.2)",javax.persistence.spi;jpa="2.1";version="[1.1.0,2.2)",javax.sql;resolution:=optional,javax.transaction;version="1.1.0";resolution:=optional,javax.transaction.xa;version="1.1.0";resolution:=optional,javax.validation;resolution:=optional,javax.validation.groups;resolution:=optional,javax.xml.parsers;resolution:=optional,javax.xml.transform;resolution:=optional,javax.xml.transform.stream;resolution:=optional,javax.xml.validation;resolution:=optional,org.apache.tools.ant;resolution:=optional,org.apache.tools.ant.types;resolution:=optional,
```

The added option `jpa="2.1"` is the reason why the EclipseLink JPA implementation can only import the packages exported by the EclipseLink JPA API artifact at runtime. It's not possible for EclipseLink to import the packages exported by the Hibernate JPA API. But the Hibernate JPA implementation can import the packages exported by the JPA API artifact provided from Hibernate or EclipseLink, because it ignores the `jpa="2.1"` option. The relationship is shown in the following diagram.



Solution

Even though several JPA API artifacts exist, only one (per specification version) can be running within an OSGi-Container at the same time. Therefore we have to choose an artifact. As the EclipseLink JPA API artifact is the officially referenced implementation of the JPA specification, the author recommends that we use that artifact. More information regarding the official reference implementations of the JAVA EE specification can be found here:

<https://glassfish.java.net/downloads/ri/>

Additionally, because the EclipseLink JPA API artifact can be used by Hibernate's implementation, it is our prime choice.

Now let's take a look at what we have to change in the Karaf enterprise feature definition (XML file) to get Hibernate and EclipseLink working at the same time. The changes must be done in the following file:

```
<groupId>org.apache.karaf.features</groupId>
<artifactId>enterprise</artifactId>
<version>${karaf.version}</version>
<classifier>features</classifier>
<type>xml</type>
```

Inside the Karaf enterprise feature definition the features JPA and Hibernate both use the Hibernate JPA API artifact. This features must be changed to use the EclipseLink JPA API artifact. The changes are marked in the following pictures.

Karaf Features with Hibernate JPA API Artifact

```
<feature name="jpa" version="2.1.0" description="OSGi Persistence Container" resolver="(obr)">
  <details>JPA implementation provided by Apache Aries JPA 1.0.2. NB: this feature doesn't provide the JPA engine, you have to
  <feature version="[1.1,2]">transaction</feature>
  <bundle start-level="30" dependency="true">mvn:org.hibernate.javax.persistence/hibernate-jpa-2.1-api/1.0.0.Final</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa-api/1.0.2</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa.blueprint.aries/1.0.4</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa.container/1.0.2</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa.container.context/1.0.4</bundle>
</feature>

<feature name="hibernate" version="4.3.6.Final" description="Hibernate 4.3.x JPA persistence engine support" resolver="(obr)">
  <details>Enable Hibernate 4.3.x as persistence engine.</details>
  <feature>http</feature>
  <bundle start-level="30" dependency="true">mvn:org.hibernate.javax.persistence/hibernate-jpa-2.1-api/1.0.0.Final</bundle>
  <bundle dependency="true">mvn:org.apache.servicemix.bundles/org.apache.servicemix.bundles.ant/1.8.2_2</bundle>
  <bundle dependency="true">mvn:org.apache.servicemix.bundles/org.apache.servicemix.bundles.dom4j/1.6.1_5</bundle>
  <bundle dependency="true">mvn:org.apache.servicemix.bundles/org.apache.servicemix.bundles.serp/1.14.1_1</bundle>
  <bundle dependency="true">mvn:com.fasterxml/classmate/1.1.0</bundle>
  <bundle dependency="true">mvn:org.javassist/javassist/3.18.1-GA</bundle>
  <bundle dependency="true">mvn:org.jboss.spec.javax.security.jacc/jboss-jacc-api_1.4_spec/1.0.2.Final</bundle>
  <bundle dependency="true">mvn:org.jboss.jandex/1.2.2.Final</bundle>
  <bundle dependency="true">mvn:org.jboss.logging/jboss-logging/3.1.4.GA</bundle>
  <bundle dependency="true">mvn:org.hibernate.common/hibernate-commons-annotations/4.0.4.Final</bundle>
  <bundle start-level="100">mvn:org.hibernate/hibernate-core/4.3.6.Final</bundle>
  <bundle start-level="100">mvn:org.hibernate/hibernate-entitymanager/4.3.6.Final</bundle>
  <bundle start-level="100">mvn:org.hibernate/hibernate-osgi/4.3.6.Final</bundle>
</feature>
```

Karaf Features with EclipseLink JPA API Artifact

```
<feature name="jpa" version="2.1.0" description="OSGi Persistence Container" resolver="(obr)">
  <details>JPA implementation provided by Apache Aries JPA 1.0.2. NB: this feature doesn't provide the JPA engine, you have to
  <feature version="[1.1,2]">transaction</feature>
  <bundle start-level="30" dependency="true">mvn:org.eclipse.persistence/javax.persistence/2.1.0</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa-api/1.0.2</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa.blueprint.aries/1.0.4</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa.container/1.0.2</bundle>
  <bundle start-level="30">mvn:org.apache.aries.jpa/org.apache.aries.jpa.container.context/1.0.4</bundle>
</feature>

<feature name="hibernate-test" version="4.3.6.Final" description="Hibernate 4.3.x JPA persistence engine support" resolver="(obr)">
  <details>Enable Hibernate 4.3.x as persistence engine.</details>
  <feature>http</feature>
  <bundle start-level="30" dependency="true">mvn:org.eclipse.persistence/javax.persistence/2.1.0</bundle>
  <bundle dependency="true">mvn:org.apache.servicemix.bundles/org.apache.servicemix.bundles.ant/1.8.2_2</bundle>
  <bundle dependency="true">mvn:org.apache.servicemix.bundles/org.apache.servicemix.bundles.dom4j/1.6.1_5</bundle>
  <bundle dependency="true">mvn:org.apache.servicemix.bundles/org.apache.servicemix.bundles.serp/1.14.1_1</bundle>
  <bundle dependency="true">mvn:com.fasterxml/classmate/1.1.0</bundle>
  <bundle dependency="true">mvn:org.javassist/javassist/3.18.1-GA</bundle>
  <bundle dependency="true">mvn:org.jboss.spec.javax.security.jacc/jboss-jacc-api_1.4_spec/1.0.2.Final</bundle>
  <bundle dependency="true">mvn:org.jboss.jandex/1.2.2.Final</bundle>
  <bundle dependency="true">mvn:org.jboss.logging/jboss-logging/3.1.4.GA</bundle>
  <bundle dependency="true">mvn:org.hibernate.common/hibernate-commons-annotations/4.0.4.Final</bundle>
  <bundle start-level="100">mvn:org.hibernate/hibernate-core/4.3.6.Final</bundle>
  <bundle start-level="100">mvn:org.hibernate/hibernate-entitymanager/4.3.6.Final</bundle>
  <bundle start-level="100">mvn:org.hibernate/hibernate-osgi/4.3.6.Final</bundle>
</feature>
```

Note: These changes must be made by the Apache Karaf team or we have to create a fork for our company. The preferred way is that the Karaf team accepts our change request.

With these changes we make sure that only one JAVA Persistence API artifact is running within Karaf and we can use the EclipseLink feature definition available at the following maven coordinates.

```
<groupId>de.kisters.karaf</groupId>
<artifactId>karaf-features.eclipselink</artifactId>
<version>${eclipselink.version}</version>
<classifier>features</classifier>
<type>xml</type>
```

The features

- jpa (JPA-Container)
- hibernate
- eclipselink-jpa

can now run together within Karaf because all JPA related artifacts use one JPA API artifact and no ClassCastExceptions or wiring problems occur. Which of these implementation is used must be defined by the Persistence Unit.

Link EclipseLink with the JPA/JTA-Container

Problem

Karaf provides the features JPA and JTA with the enterprise feature definition, that installs the JPA- and JTA-Container so that you can use container managed entity managers and transactions. An example how to use this can be found by looking at the energy-material slice (for more information ask [Andy Schmidt](#) or [Tim Rehfish](#)). If you install the Hibernate feature in Karaf, an adapter will automatically be installed, which links the Hibernate-Persistence-Provider with the JPA- and JTA-Container. There is no adapter for EclipseLink. However, the JPA- and JTA-Container looks for services which implement the interface "javax.persistence.spi.PersistenceProvider" and have a property set with the key "javax.persistence.provider" and the value "org.eclipse.persistence.jpa.PersistenceProvider".

Solution

With the knowledge that the JPA- and JTA-Container binds all services which implement the "PersistenceProvider" interface and have a specific service property set, we can create the EclipseLink-Persistence-Provider as a bean and register it as a service via blueprint. This can be done by creating a file named "eclipselink-persistence-provider.xml" with the following content.

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0">
  <bean id="EclipseLink.PersistenceProvider" class="org.eclipse.persistence.jpa.PersistenceProvider" />
  <service interface="javax.persistence.spi.PersistenceProvider" ref="EclipseLink.PersistenceProvider">
    <service-properties>
      <entry key="javax.persistence.provider">
        <value type="java.lang.String">org.eclipse.persistence.jpa.PersistenceProvider</value>
      </entry>
    </service-properties>
  </service>
</blueprint>
```

This file must be put into the deploy folder of Karaf.

Note: We should consider creating an adapter for EclipseLink like Hibernate has, to provide the EclipseLink-Persistence-Provider as a service and install the adapter with the "eclipselink-jpa" feature.

ClassNotFoundExceptions

Problem

When Hibernate or EclipseLink is used at runtime a `ClassNotFoundExceptions` occurs for the artifact which uses JPA to load or persist their entities.

Solution

In this case the packages which include the classes that cannot be found must be imported by the artifact. The following listing shows you which packages must be imported for Hibernate and EclipseLink so that it works at runtime.

Hibernate

- `org.hibernate.proxy`
- `javassist.util.proxy`

EclipseLink

- `org.eclipse.persistence.internal.weaving`
- `org.eclipse.persistence.internal.descriptors`
- `org.eclipse.persistence.internal.identitymaps;`
- `org.eclipse.persistence.queries`
- `org.eclipse.persistence.descriptors.changetracking`
- `org.eclipse.persistence.sessions`
- `org.eclipse.persistence.internal.jpa.rs.metadata.model`

We recommend that you import these packages with the resolution "optional" (`<package>;resolution:=optional`) to get your artifact working with Hibernate and EclipseLink.