Entity Enhancement

What is Enhancement Anyway?

The JPA spec requires some type of monitoring of Entity objects, but the spec does not define how to implement this monitoring. Some JPA providers auto-generate new subclasses or proxy objects that front the user's Entity objects at runtime, while others use byte-code weaving technologies to enhance the actual Entity class objects. OpenJPA supports both mechanisms, but strongly suggests only using the **byte-code weaving** enhancement. The subclassing support (as provided by OpenJPA) is not recommended (and is disabled by default in OpenJPA 2.0 and beyond).

For complete details on entity enhancement, please checkout the section in the latest User's Guide on Enhancement. The following sections will only cover the high-level concepts and the ways to use the enhancer in different build and runtime setups.

Byte-code Weaving Enhancement

Byte-code weaving is the preferred and recommended method of enhancement to use with OpenJPA. This byte-code enhancement can be performed at build-time or dynamically at run-time. Whichever approach is selected, the same enhancement processing is performed with the same level of functionality weaved into each Entity class.

Build Time Enhancement

Build time enhancement is probably the most common enhancement method to use with OpenJPA, especially in an automated JUnit test environment. The whole OpenJPA JUnit test bucket relies on build-time enhancement. Please follow the links below based on your development environment:

- Enhancement with ANT
- Enhancement with Maven
- Enhancement with Eclipse

Dynamic Enhancement

Dynamic run-time enhancement with OpenJPA comes in several different flavors, depending on your environment. The preferred and most reliable method of dynamic enhancement is via the defined container hook in a Java EE and OSGi environments. In a JSE environment, there are a couple of choices for configuring or using dynamic enhancement. The choice will depend on your usage patterns.

Java EE and OSGi environments

The Java EE specifications outline a mechanism for plugging in a JPA transformer (byte code enhancer) into the container's classloading processing. Most Java EE application servers (for example, IBM's WebSphere Application Server) support this mechanism. In addition, several of the OSGi container providers (for example, WebSphere's OSGi container) have followed a similar path and provide this classloading hook for dynamic enhancement. If your container environment supports this mechanism with OpenJPA, this would be the preferred and easiest method of performing the byte-code enhancement.

JSE Environment

Explicit javaagent support

The recommended way get runtime enhancement for the JSE environment is to provide a javaagent when launching the JVM that OpenJPA is running in. This is a common method to use when executing individual JUnits in a development environment because it is very painless and easy. All that is required to get runtime enhancement is to specify the -javaagent:openjpa-all-2.2.0-SNAPSHOT.jar (as an example) on the JVM configuration.

More information can be found on the Runtime Enhancement page.

Implicit javaagent support

OPENJPA-952 added the capability to have OpenJPA attempt to dynamically load the javaagent enhancer. If you see the following message, OpenJPA loaded the enhancer dynamically.

> *[java] 1453 jpa_app INFO [main] openjpa.Runtime - OpenJPA dynamically loaded the class enhancer. Any classes that were not enhanced at build time will be enhanced when the are loaded by the JVM.*

This method of enhancement is intended for first time users or developers as it has a number of caveats.

- It works with both the Sun/Oracle 1.6SDK and IBM 1.6JDK. The JRE is not sufficient.
- If any unenhanced Entities are loaded by the JVM before an EntityManagerFactory is created, this method of enhancement will not work. If this condition is encountered, you will see the following warning:

> *[java] 1047 jpa_app WARN [main] openjpa.Enhance - Unenhanced classes were detected even though the enhancer has ran. Ensure that the EntityManagerFactory is created prior to creating any Entities.*

If your application uses some other method of enhancement, this support can be explicitly disabled by setting the following property in your persistence.xml.

```
<property name="openjpa.DynamicEnhancementAgent" value="false"/>
```

Subclassing Enhancement

> ⚠ **Not Recommended**
>
> The use of OpenJPA's subclassing support is not recommended, and is disabled by default in OpenJPA 2.0 and beyond.

When running in a Java SE environment or in a non-Java EE 5 compliant container, OpenJPA can utilize runtime subclassing enhancement. The subclassing enhancement support was added originally as a convenience to new developers to reduce the amount of work to get a 'HelloWorld-ish' OpenJPA application working out of the box. It was never meant to run in production. So you're probably thinking that this sounds great! OpenJPA handles enhancement automatically for me and I can stop reading this post. Wrong! Subclassing has two major drawbacks. First off, it isn't nearly as fast as byte-code enhancement and the second drawback is that there are some documented functional problems when using the subclassing support. The moral of the story is, don't use this method of enhancement.

For reference, the property that enables/disables the subclassing support is openjpa.RuntimeUnenhancedClasses. The value "unsupported" is the recommended and default setting for this property.

Additional information regarding the subclassing enhancement can be found in the OpenJPA docs.

Author Attribution

The content for this page and sub-pages was adapted from content created by OpenJPA contributor Rick Curtis from the following WebSphere and Java Persistence blog entries:
http://websphere persistence.blogspot.com/2009/02/openjpa-enhancement.html
http://websphere persistence.blogspot.com/2009/04/openjpa-enhancement-eclipse-builder.html