An Oracle White Paper
July 2013

Using Maven with Oracle TopLink 12.1.2

Introduction

Maven is one of the most popular build management systems and with the 12.1.2 release TopLink includes POM files for all jars as well as a utility to install them into your Maven repository. This document describes how to install TopLink into your Maven repository, the Maven coordinates for all TopLink jars, and which dependencies are required for different usage scenarios including Java SE, Java EE in WebLogic, and using TopLink with Oracle Coherence.

Installation

Like other Oracle Fusion Middleware components, TopLink 12.1.2 provides support for developing with Maven by providing POMs for all distributed jars. To get started you'll need to populate your Maven repository with the jars from the TopLink install.

Note that TopLink is also included in the WebLogic and Coherence installs. Regardless of which installer you use, TopLink will be installed into the same location in the root ORACLE_HOME directory. The installation instructions are the same for all three install options.

The install process involves two steps. First the installation of the oracle-maven-sync utility plugin that will be used to populate the repository, and then the actual population of the repository with the TopLink jar and POM files. This process is documented in detail in the "Installing and Configuring Maven for Build Automation and Dependency Management"[1] chapter of the "Developing Applications Using Continuous Integration"[2] document that's a part of the Oracle Fusion Middleware 12.1.2 documentation set. The TopLink Install Utility[3] simplifies the two installation steps. It is available from the Java.net TopLink Samples Project[4].

Install oracle-maven-sync

Installing the oracle-maven-sync plugin and populating your repository is achieved by cd'ing into the directory containing the TopLink Install Utility files and, on Mac OS X and Linux, issuing the following command where {ORACLE_HOME} is your TopLink install directory:

```
./install.sh {ORACLE_HOME}
```

On Windows the command would be:

```
install.bat {ORACLE_HOME}
```

Once the install script completes you can verify success by looking in your Maven repository where you should find the com/oracle/toplink directory structure populated with the TopLink artifacts.

[1] http://docs.oracle.com/middleware/1212/core/MAVEN/config_maven.htm

[2] http://docs.oracle.com/middleware/1212/core/MAVEN/index.html

[3] http://www.oracle.com/technetwork/middleware/toplink/documentation/toplink-maven-install-1983778.zip

[4] https://java.net/projects/toplink-samples/

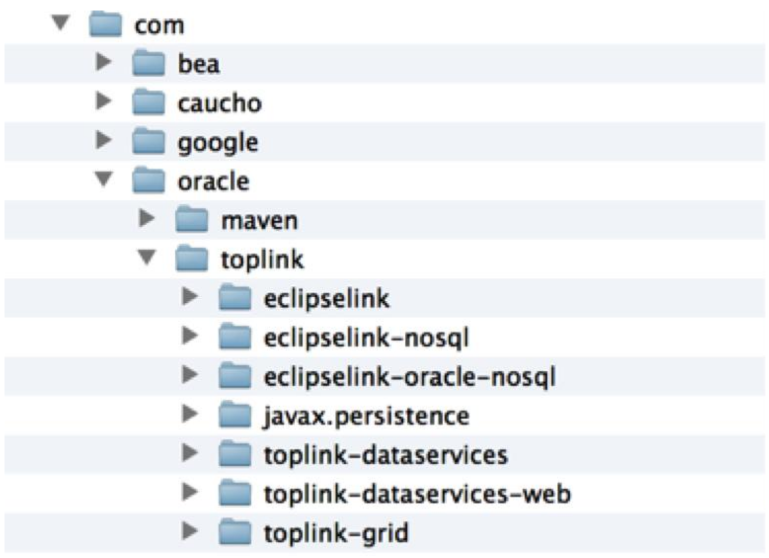

Figure 1: Maven repository contents after install

Note that if you are installing into a remote repository you cannot use the TopLink Install POM. Instead you should directly use the oracle-maven-sync plugin following the directions provided in the chapter the "Installing and Configuring Maven for Build Automation and Dependency Management"[5] chapter of the "Developing Applications Using Continuous Integration"[6] document.

Using Maven

After completing the installation, your repository will contain a number of jars with the associated group and artifact ids listed in the following table:

TOPLINK MAVEN COORDINATES

JAR	GROUP ID	ARTIFACT ID	VERSION
javax.persistence	com.oracle.toplink	javax.persistence	12.1.2-0-0
toplink-grid	com.oracle.toplink	toplink-grid	12.1.2-0-0
toplink-dataservices	com.oracle.toplink	toplink-dataservices	12.1.2-0-0
toplink-dataservices-web	com.oracle.toplink	toplink-dataservices-web	12.1.2-0-0

[5] http://docs.oracle.com/middleware/1212/core/MAVEN/config_maven.htm
[6] http://docs.oracle.com/middleware/1212/core/MAVEN/index.html

eclipselink	com.oracle.toplink	eclipselink	12.1.2-0-0
eclipselink-nosql	com.oracle.toplink	eclipselink-nosql	12.1.2-0-0
eclipselink-oracle-nosql	com.oracle.toplink	eclipselink-oracle-nosql	12.1.2-0-0

All of the jars in the TopLink install have the "com.oracle.toplink" group id and the version number of the TopLink release. TopLink includes jars from the open source EclipseLink project and these coordinates distinguish the bundled versions of those jars from those available from the EclipseLink project. This prevents problems arising from accidentally mixing and matching versions of TopLink jars with incompatible EclipseLink jars.

TopLink JAXB and JSON

TopLink JAXB and JSON applications only depend on eclipselink.jar as the javax.xml.bind packages are included in JDK 1.6 and above. Add the following dependency to your POM for Java SE JAXB and JSON applications:

```
<dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>eclipselink</artifactId>
    <version>12.1.2-0-0</version>
</dependency>
```

Using TopLink JAXB and JSON in WebLogic

WebLogic includes eclipselink.jar on the server classpath so add `<scope>provided</scope>` to the dependency to prevent the dependent jar from being package in your deployment archive.

```
 <dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>eclipselink</artifactId>
    <version>12.1.2-0-0</version>
    <scope>provided</scope>
 </dependency>
```

TopLink JPA

Applications using TopLink JPA depend on both eclipselink.jar and the javax.persistence jar.

```
<dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>12.1.2-0-0</version>
</dependency>
<dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>eclipselink</artifactId>
```

```
        <version>12.1.2-0-0</version>
    </dependency>
```

Using TopLink JPA in WebLogic

WebLogic includes eclipselink.jar and the javax.persistence jar on the server classpath so add `<scope>provided</scope>` to these dependencies to prevent the dependent jars from being package in your deployment archive.

```
    <dependency>
        <groupId>com.oracle.toplink</groupId>
        <artifactId>javax.persistence</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>com.oracle.toplink</groupId>
        <artifactId>eclipselink</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
```

TopLink Data Services

In addition to depending on the two jars required for TopLink JPA, web applications using TopLink Data Services must also add a dependency the toplink-dataservices-web jars.

Using TopLink Data Services in WebLogic

The three dependencies are illustrated below. Note that the scope of javax.persistence and eclipselink.jar is set to provided because WebLogic includes these jars on the server classpath and need not be included in applications. The toplink-dataservices-web jar does need to be packaged in WEB-INF/lib so its scope is not set to provided.

```
    <dependency>
        <groupId>com.oracle.toplink</groupId>
        <artifactId>javax.persistence</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>com.oracle.toplink</groupId>
        <artifactId>eclipselink</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
```

```
<dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>toplink-dataservices-web</artifactId>
    <version>12.1.2-0-0</version>
</dependency>
```

TopLink and Coherence

Oracle TopLink and Oracle Coherence are integrated through TopLink's "TopLink Grid"[7] feature. TopLink Grid provides Coherence with optimized JPA CacheStore and CacheLoader implementations and enables TopLink JPA applications to leverage Coherence as both a scalable external entity cache and, in some configurations, as highly performant parallel query processor. See the Coherence Integration Guide for details.

Coherence provides Maven support similar to that provided by TopLink and also leverages the same oracle-maven-sync plugin to populate your Maven repository. Since the Coherence installer includes TopLink and installs both into the same ORACLE_HOME directory you can populate your repository with the jars and POMs of both products using the TopLink Install Utility. Follow the same installation instructions as for TopLink (above) but provide the Coherence install directory as the value of {ORACLE_HOME} when running the install script.

Regardless of whether you are developing a Coherence application using a JPA CacheStore or a JPA application with Coherence caching, your POM should include the following dependencies:

```
<dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>12.1.2-0-0</version>
</dependency>
<dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>eclipselink</artifactId>
    <version>12.1.2-0-0</version>
</dependency>
<dependency>
    <groupId>com.oracle.toplink</groupId>
    <artifactId>toplink-grid</artifactId>
    <version>12.1.2-0-0</version>
</dependency>
```

[7] http://www.oracle.com/technetwork/middleware/toplink/tl-grid-097210.html

```
    <dependency>
        <groupId>com.oracle.coherence</groupId>
        <artifactId>coherence</artifactId>
        <version>12.1.2-0-0</version>
    </dependency>
```

Using TopLink and Coherence in WebLogic

WebLogic includes all the jars required for using TopLink with Coherence on the server classpath so add `<scope>provided</scope>` to all these dependencies to prevent the dependent jars from being package in your deployment archive.

```
    <dependency>
        <groupId>com.oracle.toplink</groupId>
        <artifactId>javax.persistence</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>com.oracle.toplink</groupId>
        <artifactId>eclipselink</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>com.oracle.toplink</groupId>
        <artifactId>toplink-grid</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>com.oracle.coherence</groupId>
        <artifactId>coherence</artifactId>
        <version>12.1.2-0-0</version>
        <scope>provided</scope>
    </dependency>
```

ORACLE®

Oracle is committed to developing practices and products that help protect the environment

Using Maven with Oracle TopLink
May 2013
Author: Shaun Smith
Contributing Authors: Doug Clarke

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200

oracle.com

Hardware and Software, Engineered to Work Together