Ejercicio 1

D)

Se ejecutan las goals:

--- maven-clean-plugin:2.5:clean (default-clean) @ dbunitexample ---

--- maven-resources-plugin:2.6:resources (default-resources) @ dbunitexample ---

--- maven-compiler-plugin:3.1:compile (default-compile) @ dbunitexample ---

--- maven-resources-plugin:2.6:testResources (default-testResources) @ dbunitexample ---

--- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ dbunitexample ---

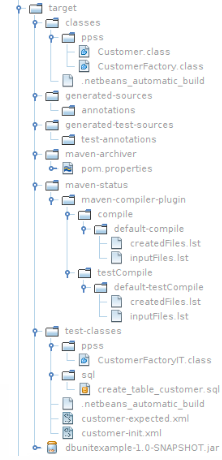
--- maven-surefire-plugin:2.12.4:test (default-test) @ dbunitexample ---

--- maven-jar-plugin:2.4:jar (default-jar) @ dbunitexample ---

--- sql-maven-plugin:1.5:execute (create-customer-table) @ dbunitexample ---

--- maven-install-plugin:2.4:install (default-install) @ dbunitexample ---

Y se generan los artefactos:



Ejercicio 2

1. Además de que todas las clases del proyecto terminan en TO son clases en las que se almacenan entidades de datos obtenibles de una bbdd, es decir Objetos de Transferencia de datos Transfer Objects.

Sobre este proyecto no es necesario realizar pruebas porque lo único que contiene son clases utilizadas para almacenar los datos obtenidos desde la base de datos.

1. –
2. Los ficheros que comienzan con una “I” serán las interfaces utilizadas para obtener los diferentes objetos sin mostrar el código y la implementación. Al ser una interfaz, en caso de cambiar la estructura de la bbdd sería fácil volver a reimplementar los métodos, de modo que no haría falta cambiar los nombres de dichos métodos, haciendo que el entorno de trabajo siga siendo aislado.

Dependencias: Junit: Se utiliza para realizar los test unitarios de los métodos.

Matriculacion-comun: Se utilizan los TO de matriculacion-comun para almacenar en memoria principal los datos obtenidos de la bbdd.

Mysql: Es el plugin necesario para poder realizar correctamente la conexión a la base de datos y poder obtener datos de ella.