Desarrollo web en entorno servidor

UT1.2 – Programación web

1 – Dependencias – Maven – Gradle

IES Clara del Rey – Madrid

¿Qué son las dependencias?

En el desarrollo de software, una dependencia es cualquier elemento EXTERNO que se necesita para poder realizar un trabajo. Lo más habitual es la dependencia de otra clase o interfaz. Por ejemplo:

- Si un método de una clase "X" llama a Math.random(), la clase X depende para hacer su trabajo de la clase Math. Dicho de otra forma, la clase "Math" es una dependencia de la clase "X".
- Si una clase "X" necesita un objeto de la clase "Y" para realizar su trabajo, la clase X depende de la clase Y. La clase "Y" es una dependencia de la clase "X". la clase

En UML se representan con una línea terminada en flecha "abierta":

Dependencias internas vs externas

En un programa Java (y en el resto de los lenguajes) podemos encontrar dos tipos de dependencias:

- Dependencias internas: Cuando una clase necesita otra clase, que forma parte del mismo programa para poder realizar su trabajo.
- Dependencias externas: Cuando una clase necesita otra clase, pero de una biblioteca de clases externas. Por ejemplo:
  - Dependencia del driver de MySQL para acceder a una BD.
  - Dependencia de Log4J para poder hacer logging de la aplicación.
  - Dependencia de Spring para desarrollo de una aplicación web.

¿Por qué son necesarias las dependencias?

Usar dependencias en los proyectos aporta muchos beneficios, pero los más importantes son dos:

- Reutilización de código. Podemos:
  - Reutilizar, en distintos programas, código escrito por nosotros mismos.
  - Utilizar bibliotecas de clases escritas por terceros, para facilitar el trabajo de desarrollo, y ahorrar mucho código.
- Pruebas unitarias. Las dependencias, cuando se definen correctamente, permiten la realización de pruebas unitarias más cómodamente.

Dependencias a nivel de proyecto

Un proyecto Java puede depender de ciertos módulos o bibliotecas de clases externas, no incluidas por defecto en la plataforma.

Por ejemplo, para acceder a una BD dependerá del driver de la BD, o para generar valores aleatorios puede usar alguna clase especializada.

Para gestionar las dependencias en el proyecto hay dos opciones:

- Añadir las dependencias manualmente. Implica descargar uno o varios JAR, copiarlos en el proyecto y configurar el classpath. Tedioso, consume tiempo, y puede implicar añadir manualmente más dependencias transitivas (dependencias de las dependencias)
- Usar un gestor de dependencias como Maven o Gradle. Se encargan de la descarga de todas las dependencias, incluidas las transitivas.

Maven

Maven es un conjunto de utilidades que aporta, entre otras cosas:

- Gestión de dependencias – Descarga y configuración automática de dependencias del proyecto. Se configuran el un fichero "pom.xml"
- Repositorio centralizado – Todas las dependencias se encuentran en un repositorio https://mvnrepository.com, de donde Maven las descarga.
- Integración con los IDE – Todos los IDE incluyen integración con Maven de una forma u otra.
- Estructura estándar de proyecto – Maven define una estructura de proyecto estándar.
- Procesos y ciclo de vida – Incluye utilidades para comilar, ejecutar, empaquetar en JAR o WAR, ejecutar test unitarios, CI/CD, etc.

Gradle

Alternativa a Maven, que aporta:

- Usa el mismo repositorio que Maven para descargar dependencias, pero permite añadir más fácilmente otras fuentes.
- Uso de lenguaje de programación (Groovy o Kotlin) para configuración, en lugar de XML. Puede ser más expresivo y flexible.
- Permite añadir más tareas en el proceso de creación de la aplicación. Maven es más rígido y tiene fases concretas (validate, compile, test, package, etc.)
- Tiene mejor rendimiento que Maven. Útil cuando estamos trabajando en grandes proyectos.
- Es el utilizado por defecto en proyectos Android con Android Studio.