Introducción a Java. Gestores de dependencias. Aprender con Inteligencia Artificial. JSP.

Gestores de dependencias

En IntelliJ tienes varias opciones para gestionar las dependencias de tus proyectos Java, como el gestor de dependencias propio de IntelliJ, **Maven** o **Gradle**. Aquí te doy una visión rápida de cada uno, con recomendaciones sobre cuál podría ser la mejor opción según tu caso.

1. Gestor de dependencias de IntelliJ

El gestor de dependencias integrado en IntelliJ se encarga de gestionar bibliotecas manualmente a través de configuraciones locales en el proyecto. Aunque puede ser útil para proyectos pequeños o sencillos, tiene limitaciones importantes:

Ventajas del gestor de IntelliJ

- Fácil de usar para proyectos muy pequeños o para aquellos que no requieren muchas dependencias externas.
- Permite agregar librerías manualmente (en formato `.jar`) sin necesidad de un gestor externo.

Desventajas del gestor de IntelliJ

- **No es escalable**: Si el proyecto crece y tiene muchas dependencias, manejar esto manualmente puede ser muy complicado y propenso a errores.
- **No maneja automáticamente las dependencias transitorias**: Si tu biblioteca necesita otras bibliotecas, tendrás que gestionarlas tú manualmente.
- **No es compatible con automatización o integración continua**: No se integra fácilmente con herramientas de automatización de construcción como Jenkins o CI/CD.
- **Menor soporte para plugins o configuraciones avanzadas**.

¿Cuándo usaremos el gestor de IntelliJ?

Usar el gestor de dependencias propio de IntelliJ solo es recomendable para proyectos **muy pequeños** o **prototipos** que no requieran integración continua, automatización, o un sistema de construcción más avanzado.

> **Actividad**
>
> Crea un proyecto en Java usando el gestor de dependencias propio de IntelliJ y un programa que, al ejecutarlo, escriba por consola "Hola Mundo".

2. Maven

Maven es un gestor de proyectos y dependencias ampliamente utilizado en proyectos Java. Define la estructura del proyecto y gestiona las dependencias a través de un archivo XML llamado `pom.xml`.

Ventajas de Maven

- **Estandarización**: Tiene una estructura estándar de proyectos Java, lo que facilita la colaboración entre equipos.
- **Automatización de dependencias**: Puedes agregar dependencias fácilmente, y Maven descargará automáticamente las bibliotecas necesarias y sus dependencias transitorias desde repositorios centralizados.
- **Integración continua**: Está bien soportado en herramientas de integración continua (CI) como Jenkins, GitLab CI, GitHub Actions, etc.
- **Popularidad y soporte**: Hay una enorme comunidad y mucha documentación disponible.

Desventajas de Maven

- **XML**: Algunos desarrolladores consideran que editar archivos `pom.xml` es tedioso y difícil de leer cuando el archivo crece.
- **Rendimiento**: Comparado con Gradle, Maven puede ser un poco más lento, especialmente en proyectos grandes.

¿Cuándo usaremos Maven?

Maven es una excelente opción si trabajas en **proyectos medianos o grandes** o si tu equipo ya está familiarizado con él. También es ideal para proyectos con equipos distribuidos, ya que su estructura estándar y amplio uso en la industria lo hacen fácil de adoptar.

Configuración de un proyecto con Maven

1. **Crear un proyecto con Maven**:
   - Al crear un nuevo proyecto en IntelliJ, selecciona **Maven** como opción de proyecto.
   - IntelliJ generará automáticamente un archivo `pom.xml` donde podrás agregar tus dependencias.
2. **Agregar dependencias en Maven**:
   - Abre el archivo `pom.xml` y añade las dependencias dentro de la sección `<dependencies>`:

```
<dependencies>
    <dependency>
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.13.1</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

3. **Ejecutar Maven**:
    - Puedes ejecutar comandos de Maven directamente en IntelliJ desde el panel de Maven (en la barra lateral derecha).
    - Comandos comunes incluyen `clean`, `install`, y `package`.

> **Actividad**
>
> Crea un proyecto en Java usando Maven y un programa que, al ejecutarlo, escriba por consola "Hola Mundo, vivo en Maven".

3. Gradle

Gradle es un gestor de construcción más moderno y flexible que Maven. En lugar de usar XML, Gradle utiliza archivos de configuración en **Groovy** o **Kotlin** para definir dependencias y tareas, lo que lo hace más legible y potente.

Ventajas de Gradle

- **Flexibilidad**: Gradle es más flexible que Maven, permitiendo configuraciones avanzadas sin tantas restricciones.
- **Rendimiento**: Gradle es generalmente más rápido que Maven debido a su sistema de ejecución incremental y su caché local.
- **Soporte para múltiples lenguajes**: Aunque es muy popular en proyectos Java, Gradle también es común en proyectos Android, Kotlin, y otros lenguajes como Scala o Groovy.
- **DSL legible**: Al usar un DSL en Groovy o Kotlin, los archivos de configuración (`build.gradle`) son más compactos y fáciles de leer que los archivos XML de Maven.

Desventajasde Gradle

- **Curva de aprendizaje**: Aunque más potente, puede ser más difícil de aprender al principio si vienes de Maven, sobre todo si necesitas personalizar tareas específicas.

- **Menos estándar**: Aunque cada vez es más común, no todos los proyectos Java adoptan Gradle. Por lo tanto, puede no ser la mejor opción si tu equipo ya está familiarizado con Maven.

¿Cuándo usamos Gradle?

Gradle es ideal para **proyectos grandes**, especialmente si necesitas flexibilidad o si trabajas en proyectos **multilenguaje** o **Android**. También es una excelente opción si te importa el rendimiento y la eficiencia en la construcción.

Configuración de un proyecto con Gradle

1. **Crear un proyecto con Gradle**:
   - Al crear un nuevo proyecto, selecciona **Gradle** como el sistema de construcción.
   - Se generará un archivo `build.gradle` que puedes usar para gestionar las dependencias.
2. **Agregar dependencias en Gradle**:
   - Abre el archivo `build.gradle` y agrega las dependencias en la sección `dependencies`:

```
dependencies {
    testImplementation 'junit:junit:4.13.1'
}
```

3. **Ejecutar Gradle**:
   - Al igual que con Maven, puedes ejecutar tareas de Gradle directamente desde IntelliJ usando el panel de Gradle. Tareas comunes incluyen `build`, `test`, y `clean`.

> **Actividad**
>
> Crea un proyecto en Java usando Gradle y un programa que, al ejecutarlo, escriba por consola "Hola Mundo, vivo en Gradle".

Durante el curso emplearemos Maven, por su mayor implantación en la actualidad.

Uso de funciones Hash

Programación en Java. Aprendizaje a través de la Inteligencia Artificial

Java es un lenguaje muy popular en el desarrollo de aplicaciones web, se emplea sobre todo en el campo del backend aunque puede ser empleado para el frontend usando extensiones como JavaServer Pages (JSP). Java emplea la misma lógica que otros lenguajes como C# o C++.

Al crear nuestro proyecto en Maven, en la carpeta src introducimos nuestros archivos de código fuente, en los que insertamos nuestras clases. En java, los archivos deben tener el mismo nombre que las clases públicas definidas y solo puede haber una clase pública por archivo. Dentro de la clase pública, podemos crear métodos estáticos, que nos permiten tratar la clase como si fuera un contenedor de funciones, o definirla como una clase normal, como las que trabajamos en la Unidad de Programación 3 sobre diseño orientado a objetos.

- *archivo Main.java*

```
import java.util.*;
import java.lang.*;
import java.io.*;

// The main method must be in a class named "Main".
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Este es el código de un `Hola Mundo` extraído de MyCompiler.io. Si usamos la autocompleción de IntelliJ, los `import` se añadirán solos. Es importante, por lo tanto, emplear esas herramientas que nos facilita el IDE, conocidas como herramientas de refactorización.

Una vez que conocemos la estructura básica de un programa en Java, es hora de transferir nuestros conocimientos en cualquier otro lenguaje de programación a Java. Para ello, una herramienta fundamental es la **Inteligencia Artificial**. Podemos usar ChatGPT, Gemini o cualquier otra. Incluso podemos ir alternando para obtener diferentes perspectivas. La versión de Java que trabajaremos será la 21 LTS (Long Term Support), por temas de compatibilidad.

> **Actividad**
> Coge tus apuntes de Programación sobre `Entrada y salida de datos por consola` y pídele a un modelo de lenguaje que te los traduzca a Java.

Más adelante aprenderemos a traducir lo aprendido en la unidad de programación 3 a lenguaje Java y, aunque no es necesario para este módulo, sí que sería recomendable hacer un esfuerzo en aprender los siguientes puntos sobre Java:

- Uso de if, if-else, switch y enhaced switch en Java.
- Uso de while, do-while, for y enhanced for en Java.

> **Actividad**
> Pídele a un modelo de lenguaje que te haga un tutorial para aprender a manejar cada uno de los siguientes puntos en Java. Pídele un tutorial distinto por punto, para evitar respuestas demasiado largas o complejas. Pídele que te proponga ejercicios para que tú los resuelvas y luego él te los corrija.
>
> - Uso de if y de if-else
> - Uso de switch-case
> - Uso de while y do-while
> - Uso de for
> - Creación y consulta de arrays
> - Uso de enhanced for para recorrer arrays

Puedes emplear también la información obtenida en el apartado de Diagramas de Actividad de la Unidad de Programación 2, pero ten en cuenta que los diagramas que tratan la concurrencia son de carácter avanzado y pueden hacerse complicados de trabajar y depurar en Java.

Conforme vayas avanzando en el módulo de programación, procura tratar de convertir esos conocimientos a Java. Te será de utilidad en un futuro.

Uso de JSP y Jakarta EE en IntelliJ Ultimate

Jakarta EE es una plataforma de programación para desarrollar y ejecutar aplicaciones en lenguaje Java. Entre sus estándares incluye el uso de servlets y de JavaServer Pages (JSP), herramientas para trabajar en la programación de aplicaciones web usando Java.

JSP es similar a PHP, en el sentido que combina el lenguaje HTML con, en este caso, fragmentos de código en Java. En un archivo .jsp, introduce tu código html como harías de forma normal y, cuando requieras algo de java, introduce el código entre `<% %>`. Si deseas imprimir directamente un valor por pantalla, puedes usar `<%= %>`. Al código introducido se lo conoce como `scriptlet`.

A nivel interno, el archivo .jsp se traduce a un archivo .java que emplea un objeto derivado de la clase `HttpServlet` y el escribe todo dentro del método `doGet` a través de un objeto PrintWriter.out. Necesita un servidor web, como Tomcat, para poder funcionar. Vamos a aprender cómo configurarlo fácilmente en IntelliJ IDEA Ultimate Edition. Recuerda que con la licencia educativa puedes acceder de forma gratuita a ese IDE.

Necesitarás:

1. **IntelliJ IDEA Ultimate Edition:** Asegúrate de que tienes la versión Ultimate, ya que la Community Edition no tiene soporte para Java EE/Web.
2. **Java Development Kit (JDK):** Necesitas un JDK instalado en tu sistema. Normalmente, IntelliJ lo instala de forma guiada y fácil.
3. **Apache Tomcat** Descargado y descomprimido en tu sistema. Puedes bajarlo de aquí. Si tienes un Mac o un Linux, baja la versión tar.gz. Guárdalo en una carpeta que te sea accesible.

Pasos para crear y ejecutar un proyecto JSP en IntelliJ IDEA Ultimate:

1. Crear un Nuevo Proyecto Web

1. Abre IntelliJ IDEA.
2. Desde la pantalla de bienvenida, haz clic en **"New Project"**. Si ya tienes un proyecto abierto, ve a `File > New > Project...` .
3. En el panel izquierdo del asistente "New Project", selecciona **"Jakarta EE"** (o "Java Enterprise" en versiones más antiguas).
4. En el panel derecho:
   - **Project name:** Dale un nombre a tu proyecto (ej., `MiPrimerJSP`).
   - **Location:** Elige la ubicación donde se guardará tu proyecto.
   - **Build system:** Selecciona **"Maven"** (recomendado para proyectos Java EE modernos) o "Gradle". Para este tutorial, usaremos Maven.
   - **JDK:** Asegúrate de que se selecciona el JDK correcto. Si no tienes uno configurado, haz clic en "Add JDK..." y navega a la ubicación de tu JDK.
   - **Template:** Selecciona **"Web Application"**. Esto creará la estructura básica para una aplicación web, incluyendo un `index.jsp` y `web.xml` .
   - **Dependencies (solo si usas Maven/Gradle):** Por ahora, puedes dejarlo tal cual.
5. Haz clic en **"Next"**.

2. Configurar el Servidor de Aplicaciones (Tomcat)

1. En la siguiente pantalla, bajo **"Application Server"**:
   - Haz clic en el botón **"New..."**.
   - Selecciona **"Tomcat Server"**.
   - En la ventana emergente, haz clic en el botón `...` (Browse) y navega a la **carpeta raíz de tu instalación de Apache Tomcat** (donde se encuentran `bin` , `conf` , `lib` , etc.).
   - IntelliJ IDEA detectará la versión de Tomcat. Haz clic en "OK".

- Asegúrate de que el Tomcat recién configurado esté seleccionado en la lista "Application Server".

2. En la misma ventana:
   - **Jakarta EE version:** Elige la versión de Jakarta EE que deseas usar. Para la mayoría de los casos, la última versión estable (ej., `Jakarta EE 9` o `10`) estará bien. Si necesitas compatibilidad con versiones antiguas, elige `Jakarta EE 8` (que es el nombre anterior de Java EE 8).
   - **Language:** Deja "Java".
   - **Servlet version:** Deja la versión por defecto o la más reciente (ej., `6.0`).
   - **Generate web.xml: Asegúrate de que esta casilla esté marcada.** Es crucial para la configuración de tu aplicación web.
3. Haz clic en **"Create"**.

3. Explorar la Estructura del Proyecto

IntelliJ IDEA creará el proyecto con una estructura similar a esta (si usas Maven):

```
MiPrimerJSP/
├── .idea/                (Configuración de IntelliJ)
├── src/
│   └── main/
│       ├── java/         (Para tus clases Java, servlets, etc.)
│       └── webapp/       (La raíz de tu aplicación web)
│           ├── WEB-INF/
│           │   └── web.xml   (Descriptor de despliegue)
│           └── index.jsp  (Tu primera página JSP)
└── pom.xml               (Archivo de configuración de Maven)
```

- **`src/main/webapp/`**: Aquí es donde colocarás todos tus archivos JSP, HTML, CSS, JavaScript e imágenes. IntelliJ ya habrá creado un `index.jsp` por defecto.
- **`src/main/webapp/WEB-INF/web.xml`**: Este archivo es el descriptor de despliegue de tu aplicación web. Contiene configuraciones importantes, como la bienvenida, mapeos de servlets, etc.
- **`src/main/java/`**: Si vas a escribir servlets o clases Java que tu JSP va a usar, irán aquí.

4. Modificar el `index.jsp` (Opcional)

Abre el archivo `src/main/webapp/index.jsp`. Verás un código HTML básico. Puedes modificarlo para incluir un poco de código JSP para probar:

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Mi Primer JSP</title>
</head>
<body>
<h1>¡Hola desde JSP en IntelliJ!</h1>
<%-- Código Java dentro de un JSP --%>
<%
    String mensaje = "La fecha actual es: " + new java.util.Date();
    out.println("<p>" + mensaje + "</p>");
%>
</body>
</html>
```

5. Configurar la Ejecución (Run Configuration)

IntelliJ IDEA suele crear automáticamente una configuración de ejecución para Tomcat cuando creas un proyecto web.

1. Mira en la barra de herramientas superior, cerca de los botones de "Play" (Run) y "Debug". Deberías ver un desplegable con el nombre de tu proyecto (ej., `Tomcat [nombre de tu proyecto]`).
2. Si no está o quieres verificar la configuración:
    - Haz clic en el desplegable y selecciona **"Edit Configurations..."**.
    - En el panel izquierdo, bajo "Tomcat Server", selecciona **"Local"** (o el nombre de tu configuración).
    - Asegúrate de que en la pestaña **"Server"**:
        - `Application server` apunta a tu instalación de Tomcat.
        - `URL` muestra algo como `http://localhost:8080/[nombre_de_tu_proyecto]_war_exploded` (esto es el "context path" por defecto que usa IntelliJ para el despliegue de desarrollo).
    - En la pestaña **"Deployment"**:
        - Deberías ver una entrada con el nombre de tu proyecto seguido de `_war_exploded`. Esto significa que IntelliJ está desplegando tu proyecto de forma "descomprimida" para un desarrollo rápido.
        - El "Application context" (la URL por la que se accederá) debería ser `/MiPrimerJSP` (o el nombre que le diste a tu proyecto). Si quieres cambiarlo, haz clic en el lápiz al final de la línea.

    - Asegúrate de que está seleccionada la opción `Deploy at startup`.
3. Haz clic en **"OK"**.

6. Ejecutar el Proyecto

1. En la barra de herramientas superior, haz clic en el botón verde de **"Play"** (Run) junto al desplegable de la configuración de Tomcat.
2. IntelliJ IDEA:
    - Construirá tu proyecto (Maven lo compilará y empaquetará).
    - Iniciará el servidor Tomcat.
    - Desplegará tu aplicación web en Tomcat.
    - Automáticamente abrirá tu navegador predeterminado en la URL de tu aplicación (ej., `http://localhost:8080/MiPrimerJSP/`).

Deberías ver tu `index.jsp` renderizado en el navegador.

Consejos adicionales:

- **Cambios en JSP:** Si modificas tu archivo `.jsp`, simplemente **refresca tu navegador**. IntelliJ IDEA y Tomcat están configurados para detectar cambios en JSPs sin necesidad de reiniciar el servidor.
- **Cambios en clases Java/Servlets:** Si modificas clases Java (archivos `.java`), necesitarás **recompilar el proyecto** (Build > Rebuild Project) y luego **reiniciar el servidor Tomcat** (puedes detenerlo y volver a iniciarlo con los botones de la barra de herramientas, o usar el botón de "Restart" que aparece a veces).
- **Ventana "Run":** La ventana "Run" (en la parte inferior de IntelliJ) mostrará los logs de Tomcat, lo cual es muy útil para depurar si algo sale mal.
- **Depuración:** Puedes poner puntos de interrupción en tu código Java (incluso en los scriptlets JSP) y ejecutar en modo "Debug" para depurar tu aplicación.

Uso de JSP en la actualidad

JSP no es una tecnología muy popular en la actualidad, ya que en general las páginas dinámicas (las que se generan en el servidor) han caído en popularidad en favor de las páginas estáticas generadas con React, Angular y otras tecnologías web. Sin embargo, JSP tiene algunas ventajas y además se utiliza todavía en páginas antiguas.

La ventaja es que podemos trabajar con los conocimientos que adquiramos en Java de forma directa y es una forma muy sencilla de poder conectar nuestros proyectos de Java a una interfaz gráfica

diseñada en HTML5, CSS y Javascript, tal y como se trabaja en el módulo profesional de Lenguaje de Marcas y Gestión de Sistemas de la información.

Si quieres profundizar más sobre el uso de JSP, puedes realizar la actividad de ampliación relacionada.

> **Actividad**
> Crea una página en JSP que calcule los primeros 7 números primos y los imprima en la página web. Usa scriptlets para el cálculo.