1.- Concepto de entorno de desarrollo. Evolución histórica.
1.1.- Evolución Histórica.
2.- Funciones de un entorno de desarrollo.
3.- Entornos integrados libres y propietarios.
4.- Estructura de entornos de desarrollo.
5.- Configuración y personalización de entornos de desarrollo.
6.- Gestión de módulos y plugins.
6.1.- Añadir.
6.2.- Eliminar.
6.3.- Funcionalidades.
6.4.- Herramientas concretas.
7.- Uso básico de entornos de desarrollo.

1.- Concepto de entorno de desarrollo. Evolución histórica.

En la unidad anterior tratamos las fases en el proceso de desarrollo de software. Una de ellas era la fase de codificación, en la cual se hacía uso de algún lenguaje de programación para pasar todas las acciones que debía llevar a cabo la aplicación a algún lenguaje que la máquina fuera capaz de entender y ejecutar.

También se hizo alusión a herramientas de apoyo al proceso de programación.

En esta unidad vamos a analizar, instalar y ejecutar estas herramientas para entender su acción y efecto.

Muchas personas aprenden a programar utilizando un editor de texto simple, compilador y depurador. Pero la mayoría, finalmente, terminan haciendo uso de algún entorno de desarrollo integrado para crear aplicaciones.

Un entorno integrado de desarrollo (IDE), es un tipo de software compuesto por un conjunto de herramientas de programación.

En concreto, el IDE se compone de:

- Editor de código de programación.
- Compilador.
- Intérprete.
- Depurador.
- Constructor de interfaz gráfico.

Los primeros entornos de desarrollo integrados nacieron a principios de los años 70, y se popularizaron en la década de los 90. Tienen el objetivo de ganar fiabilidad y tiempo en los proyectos de software. Proporcionan al programador una serie de componentes con la misma interfaz gráfica, con la consiguiente comodidad, aumento de eficiencia y reducción de tiempo de codificación.

Normalmente, un IDE está dedicado a un determinado lenguaje de programación. No obstante, las últimas versiones de los IDE tienden a ser compatibles con varios lenguajes (por ejemplo, Eclipse, NetBeans, Microsoft Visual Studio) mediante la instalación de plugins adicionales.

En este tema, nuestro interés se centra en conocer los entornos de desarrollo, los tipos, en función de su licencia y del lenguaje de programación hacia el cual están enfocados.

Casi todas las personas que empiezan a programar utilizan un editor simple de textos y un compilador-depurador instalado en su equipo. Sin embargo, prácticamente todas acaban utilizando un entorno de desarrollo.

1.1.- Evolución Histórica.

En las décadas de utilización de la tarjeta perforada como sistema de almacenamiento el concepto de Entorno de Desarrollo Integrado IDE sencillamente no tenía sentido.

Los programas estaban escritos con diagramas de flujo y entraban al sistema a través de las tarjetas perforadas. Posteriormente, eran compilados.

El primer lenguaje de programación que utilizó un IDE fue el BASIC (que fue el primero en abandonar también las tarjetas perforadas o las cintas de papel).

Éste primer IDE estaba basado en consola de comandos exclusivamente (normal por otro lado, si tenemos en cuenta que hasta la década de los 90 no entran en el mercado los sistemas operativos con interfaz gráfica). Sin embargo, el uso que hace de la gestión de archivos, compilación y depuración; es perfectamente compatible con los IDE actuales.

A nivel popular, el primer IDE puede considerarse que fue el IDE llamado Maestro.

Nació a principios de los 70 y fue instalado por unos 22.000 programadores en todo el mundo. Lideró este campo durante los años 70 y 80.

El uso de los entornos integrados de desarrollo se ratifica y afianza en los 90 y hoy en día contamos con infinidad de IDE, tanto de licencia libre como no.

No hay unos entornos de desarrollo más importantes que otros. La elección del IDE más adecuado dependerá del lenguaje de programación que vayamos a utilizar para la codificación de las aplicaciones y el tipo de licencia con la que queramos trabajar.

2.- Funciones de un entorno de desarrollo.

Como sabemos, los entornos de desarrollo están compuestos por una serie de herramientas software de programación, necesarias para la consecución de sus objetivos.

Estas herramientas son:

- Un editor de código fuente.
- Un compilador y/o un intérprete.
- Automatización de generación de herramientas.
- Un depurador.

Las funciones de los IDE son:

- Editor de código.
- Auto-completado de código, atributos y métodos de clases.
- Identificación automática de código.
- Herramientas de concepción visual para crear y manipular componentes visuales.
- Asistentes y utilidades de gestión y generación de código.
- Archivos fuente en unas carpetas y compilados a otras.
- Compilar proyectos complejos en un solo paso.
- Control de versiones: tener un único almacén de archivos compartido por todos los colaboradores de un proyecto. Ante un error, mecanismo de auto-recuperación a un estado anterior estable.
- Soportar cambios de varios usuarios de manera simultánea.
- Generar de documentación integrado.
- Detectar de errores de sintaxis en tiempo real.

Otras funciones importantes son:

- Ofrece refactorización de código: cambios menores en el código que facilitan su legibilidad sin alterar su funcionalidad (por ejemplo cambiar el nombre a una variable).
- Permite introducir automáticamente tabulaciones y espaciados para aumentar la legibilidad.
- Depuración: seguimiento de variables, puntos de ruptura y mensajes de error del intérprete.
- Aumentar las funcionalidades a través de la gestión de sus módulos y plugins.
- Administrar las interfaces de usuario (menús y barras de herramientas).
- Administrar configuraciones del usuario.

3.- Entornos integrados libres y propietarios.

El autor o autores, pueden conceder u otorgar distintos tipos de licencias, pueden sólo autorizar su uso, su modificación, instalación, etc...

Existen algunos tipos, como por ejemplo: GPL, LGPL, propietario, etc...

Entornos Integrados Libres

Son aquellos con licencia de uso público. El autor deja libertad a los usuarios, por tanto, el programa puede ser usado, copiado, estudiado, modificado y redistribuido libremente, pero eso no quiere decir que tenga que ser obligatoriamente gratis, podemos encontrarnos programas bajo esta licencia que son de pago, aunque suelen ser muy económico. El software libre es un asunto de libertad, NO DE PRECIO.

Los más conocidos y utilizados son Eclipse y NetBeans, hay bastantes más.

Tipos de entornos de desarrollo libres más relevantes en la actualidad. IDE NetBeans. Eclipse. Gambas. Anjuta. Geany. GNAT Studio.
C/C++, Java, JavaScript, PHP, Python. Ada,PHP. Basic. Fortran.

El aspecto de la licencia del IDE que se elija para el desarrollo de un proyecto es una cuestión de vital importancia. En su elección prevalecerá la decisión de los supervisores del proyecto y de la dirección de la empresa.

Entornos Integrados Propietarios

Son aquellos entornos integrados de desarrollo que necesitan licencia, para ser: usado, copiado, estudiado, modificado o redistribuido, es lo contrario a software libre.

El más conocido y utilizado es Microsoft Visual Studio, que usa el framework .NET y es desarrollado por Microsoft.

El software de propietario, igual que el software libre, es un asunto de libertad, NO DE PRECIO. También puede existir gratuito y de pago.

Tipos de entornos de desarrollo propietarios más relevantes en la actualidad. IDE Microsoft Visual Studio. FlashBuilder. C++ Builder. Turbo C++ profesional. JBuilder. JCreator. Xcode, C#. ActionScript.

4.- Estructura de entornos de desarrollo.

Los entornos de desarrollo, ya sean libres o propietarios, están formados por una serie de componentes software que determinan sus funciones.

Estos componentes son:

Editor de textos: Resalta y colorea la sintaxis, tiene la función de autocompletar código, ayuda y listado de parámetros de funciones y métodos de clase. Inserción automática de paréntesis, corchetes, tabulaciones y espaciados.

Compilador/intérprete: Detección de errores de sintaxis en tiempo real. Características de refactorización.

Depurador: Botón de ejecución y traza, puntos de ruptura y seguimiento de variables. Opción de depurar en servidores remotos.

Generador automático de herramientas: Para la visualización, creación y manipulación de componentes visuales y todo un arsenal de asistentes y utilidades de gestión y generación código.

Interfaz gráfica: Nos brinda la oportunidad de programar en varios lenguajes con un mismo IDE. Es una interfaz agradable que puede acceder a innumerables bibliotecas y plugins, aumentando las opciones de nuestros programas.

5.- Configuración y personalización de entornos de desarrollo.

Al abrir un proyecto existente, o bien crear un nuevo proyecto, seleccionaremos un desplegable con el nombre de “configuración” desde el que podremos personalizar distintas opciones del proyecto.

Podemos personalizar la configuración del entorno sólo para el proyecto actual, o bien para todos los proyectos, presentes y futuros.

Parámetros configurables del entorno:

Carpeta o carpetas donde se alojarán todos los archivos de los proyectos (es importante la determinación de este parámetro, para tener una estructura de archivos ordenada).

Carpetas de almacenamiento de paquetes fuente y paquetes prueba.

Administración de la plataforma del entorno de desarrollo.

Opciones de la compilación de los programas: compilar al grabar, generar información de depuración.

Opciones de empaquetado de la aplicación: nombre del archivo empaquetado (con extensión .jar, que es la extensión característica de este tipo de archivos empaquetados) y momento del empaquetado.

Opciones de generación de documentación asociada al proyecto.

Descripción de los proyectos, para una mejor localización de los mismos.

Opciones globales de formato del editor: número de espaciados en las sangrías, color de errores de sintaxis, color de etiquetas, opción de autocompletado de código, propuestas de insertar automáticamente código.

Opciones de combinación de teclas en teclado. Etc.

6.- Gestión de módulos y plugins.

Con la plataforma dada por un entorno de desarrollo como NetBeans podemos hacer uso de módulos y plugins para desarrollar aplicaciones.

En la página oficial de NetBeans encontramos una relación de módulos y plugins, divididos en categorías. Seleccionando la categoría Lenguajes de Programación, encontraremos aquellos módulos y plugins que nos permitan añadir nuevos lenguajes soportados por nuestro IDE.

Un módulo es un componente software que contiene clases de Java que pueden interactuar con las API del entorno de desarrollo y el manifest file, que es un archivo especial que lo identifica como módulo.

Los módulos se pueden construir y desarrollar de forma independiente.

Esto posibilita su reutilización y que las aplicaciones puedan ser construidas a través de la inserción de módulos con finalidades concretas. Por esta misma razón, una aplicación puede ser extendida mediante la adición de módulos nuevos que aumenten su funcionalidad.

Tanto los módulos como los plugins se encuentran en NetBeans en Tools, Plugins.

Muchos autores utilizan el mismo nombre, porque se considera que un plugin es normalmente un conjunto de módulos.

Existen en la actualidad multitud de módulos y plugins disponibles para todas las versiones de los entornos de desarrollo más utilizados.

6.1.- Añadir.

Para añadir un nuevo módulo tenemos varias opciones:

1. Añadir algún módulo de los que NetBeans instala por defecto.
2. Descargar un módulo desde algún sitio web permitido y añadirlo.
3. Instalarlo on-line en el entorno.

Por supuesto, una cuarta posibilidad es crear el módulo nosotros mismos (aunque eso no lo veremos aquí). Sin embargo, lo más usual es añadir los módulos o plugins que realmente nos interesan desde la web oficial de NetBeans. El plugin se descarga en formato .nbm que es el propio de los módulos en NetBeans. Posteriormente, desde nuestro IDE, cargaremos e instalaremos esos plugins. A esta manera de añadir módulos se le conoce como adición off-line. También es habitual instalarlos on-line, sin salir del IDE.

La adición on-line requiere tener instalado el plugin Portal Update Center en NetBeans 6.9.1 y consiste en instalar complementos desde nuestro mismo IDE, sin tener que descargarlos previamente.

6.2.- Eliminar.

Cuando consideramos que algún módulo o plugin de los instalados no nos aporta ninguna utilidad, o bien que el objetivo para el cual se añadió ya ha finalizado, el módulo deja de tener sentido en nuestro entorno. Es entonces cuando nos planteamos eliminarlo.

Eliminar un módulo es una tarea trivial que requiere seguir los siguientes pasos:

1. Encontrar el módulo o plugin dentro de la lista de complementos instalados en el entorno.
2. A la hora de eliminarlo, tenemos dos opciones:
    1. Desactivarlo: El módulo o plugin sigue instalado, pero en estado inactivo (no aparece en el entorno).
    2. Desinstalarlo: El módulo o plugin se elimina físicamente del entorno de forma permanente.

Siempre nos pedirá elegir entre dos opciones: desactivar o desinstalar.

6.3.- Funcionalidades.

Los módulos y plugins disponibles para los entornos de desarrollo, en sus distintas versiones, tienen muchas y muy variadas funciones.

Podemos clasificar las distintas categorías de funcionalidades de módulos y plugins en los siguientes grupos:

Construcción de código: facilitan la labor de programación.

Bases de datos: ofrecen nuevas funcionalidades para el mantenimiento de las aplicaciones.

Depuradores: hacen más eficiente la depuración de programas.

Aplicaciones: añaden nuevas aplicaciones que nos pueden ser útiles.

Edición: hacen que los editores sean más precisos y más cómodos para el programador.

Documentación de aplicaciones: para generar documentación de los proyectos en la manera deseada.

Interfaz gráfica de usuario: para mejorar la forma de presentación de diversos

aspectos del entorno al usuario.

Lenguajes de programación y bibliotecas: para poder programar bajo un Lenguaje de Programación que, en principio, no soporte la plataforma.

Refactorización: hacer pequeños cambios en el código para aumentar su legibilidad, sin alterar su función.

Aplicaciones web: para introducir aplicaciones web integradas en el entorno.

Prueba: para incorporar utilidades de pruebas al software.

6.4.- Herramientas concretas.

Importador de Proyectos de NetBeans: permite trabajar en lenguajes como JBuilder.

Servidor de aplicaciones GlassFish: Proporciona una plataforma completa para aplicaciones de tipo empresarial.

Soporte para Java Enterprise Edition: Cumplimiento de estándares, facilidad de uso y la mejora de rendimiento hacen de NetBeans la mejor herramienta para crear aplicaciones de tipo empresarial de forma ágil y rápida.

Facilidad de uso a lo largo de todas las etapas del ciclo de vida del software. NetBeans Swing GUI builder: simplifica mucho la creación de interfaces gráficos de usuarios en aplicaciones cliente y permite al usuario manejar diferentes aplicaciones sin salir del IDE.

NetBeans Profiler: Permite ver de forma inmediata ver cómo de eficiente trabajará un trozo de software para los usuarios finales.

El editor WSDL facilita a los programadores trabajar en servicios Web basados en XML.

El editor XML Schema Editor permite refinar aspectos de los documentos XML de la misma manera que el editor WSDL revisa los servicios Web.

Aseguramiento de la seguridad de los datos mediante el Sun Java System Acces Manager.

Soporte beta de UML que cubre actividades como las clases, el comportamiento, la interacción y las secuencias.

Soporte bidireccional, que permite sincronizar con rapidez los modelos de desarrollo con los cambios en el código conforme avanzamos por las etapas del ciclo de vida de la aplicación. Etc.

7.- Uso básico de entornos de desarrollo.

Aquí irá apareciendo la relación de proyectos, archivos, módulos o clases que vayamos abriendo durante la sesión.
Cada proyecto comprende una serie de archivos y bibliotecas que lo componen.

El principal archivo del proyecto Java es el llamado Main.java.

Ventana derecha: espacio de escritura de los códigos de los proyectos.

Aquí aparece el esqueleto propio de un programa escrito en lenguaje Java.

Se ha añadido el código:
System.out.println("Hola Mundo");

Y veremos su significado en las siguientes páginas. De momento, saber que para escribir cualquier código, hay que hacerlo en esta ventana.
BARRA DE HERRAMIENTAS: Desde aquí podremos acceder a todas las opciones del IDE.

No hay que decir que la programación en Java no es objeto del presente módulo, pero puedes probar con algunos ejemplos en Java que tengas de otros módulos.

Mientras escribimos en el editor de textos nos percatamos de varias características de NetBeans que ya hemos señalado en páginas anteriores:
Autocompletado de código.
Coloración de comandos.

Subrayado en rojo cuando hay algún error y posibilidad de depuración y corrección de forma visual, mediante un pequeño icono que aparece a la izquierda de la línea defectuosa.

Una vez tenemos el código plasmado en la ventana de comandos y libre de errores de sintaxis, los siguientes pasos son: compilación, depuración, ejecución.