Unidad Didáctica II

CASO

__José__ está viendo cómo la empresa cada vez abarca más proyectos y a su vez más complejos. Se está dando cuenta que utilizan gran cantidad de tecnología Java y que muchas veces están perdiendo el tiempo en el paso de unas herramientas a otras. Una herramienta que abarcara todas esas tecnologías y cubriera todo el ciclo de vida del software les permitiría aumentar el rendimiento y a la vez reutilizar conocimiento y código.

Un amigo le ha hablado de unas Jornadas que se celebran dentro de poco sobre tecnologías Java y su integración y le ha dado una invitación para que asista. __José__ al principio era reacio a asistir a estas Jornadas, por falta de tiempo debido a su gran carga de trabajo, pero finalmente se ha decidido a acudir a las mismas.

__José__ ha venido entusiasmado de las Jornadas, ya que le han hablado muy bien de una herramienta que cubre todo el ciclo de vida de desarrollo de software y aúna casi todas las tecnologías Java existentes en el mercado. Está dispuesto a probar dicha herramienta. Habla con __Carmen__ y __Víctor__ para que empiecen a hacer unas primeras pruebas. __Carmen__ le comenta que ella ya ha utilizado dicha herramienta cuando estudiaba el Ciclo de DAI en el IES Aguadulce, aunque en una versión anterior a la última que hay en el mercado. Comenta que es una maravilla y que a __Víctor__ le costará poco familiarizarse con ella. Ambos van a ponerse manos a la obra.

Marcar como leído

Unidad Didáctica II

¿Qué es Oracle JDeveloper?

Seguramente, igual que Víctor, habrás oído hablar de muchas tecnologías relacionadas con Java. Muchas de ellas ya las has utilizado como: Java, Swing, UML, SQL, PL/SQL, pero otras muchas sólo te sonarán de oídas o ni siquiera, como pueden ser: XML, HTML, JavaScript, J2EE, EJB, JSP, JSF, Oracle ADF, BPEL, PHP, etc. No te pongas nervioso por tanta sigla, ya que a lo largo de esta unidad y todas las que componen este módulo pretendemos que la mayoría te queden claras y además sepas utilizarlas. Ahora bien, para que podamos llegar a buen puerto, necesitamos que repases tus conocimientos de Java, que aquí daremos por conocidos, ya que de lo contrario sería imposible avanzar. Recuerda que a la programación en Java le dedicamos el módulo de PLE (Programación en Lenguajes Estructurados), por lo que te pedimos que le des un repaso antes de seguir.

PARA SABER MÁS

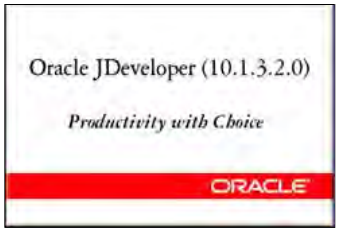
En los siguientes enlaces podrás encontrar dos tutoriales de Java en castellano, por si no has cursado el módulo de PLE y tus conocimientos de Java son escasos. Si ese es tu caso, te recomendamos encarecidamente que los mires y te los estudies , ya que de lo contrario te será muy difícil seguir este módulo. Te llevará un tiempo, pero es necesario. Si has cursado ya PLE, quizá un vistazo más rápido te sirva para refrescar y recordar lo que ya sabes.

El primero es un tutorial de Java básico, en el que podrás encontrar todos los fundamentos de la programación en Java.

Tutorial de Java básico

Este segundo enlace es otro tutorial en castellano, pero esta vez está dedicado a la programación en Swing y JFC (Java Foundation Class).

Tutorial de Swing y JFC



Oracle JDeveloper 10g es un Entorno de Desarrollo Integrado (IDE - Integrated Development Enviroment) que nos permite construir tanto aplicaciones Java, como servicios Web usando los últimos estándares para Java, y las tecnologías más utilizadas en el mercado (todas las siglas de las que hemos hablado antes y que seguramente te sonaban a chino).

Es una herramienta que podríamos considerar como herramienta CASE, ya que nos permite automatizar los aspectos clave de todo el proceso de desarrollo de un sistema, aumentando la productividad en el desarrollo reduciendo costes. También podríamos denominarla herramienta RAD, que aunque a veces se confunden no es lo mismo. Ambas hacen más rápido el desarrollo de aplicaciones, pero la diferencia es que las primeras suelen cubrir más etapas del ciclo de vida de desarrollo de aplicaciones.

Dicho entorno soporta el ciclo de vida completo de desarrollo del software (éste será orientado a objetos debido a las características de Java) incluyendo características de:

- modelado,
- codificación,
- depuración,
- testeo,
- documentación,
- optimización y
- despliegue de aplicaciones.

Todas estas tareas se pueden realizar desde el mismo entorno, con la comodidad y aumento de productividad que ello conlleva.

JDeveloper incluye un editor visual del tipo WYSIWYG para HTML, JSP, JSF y Swing que permite a los desarrolladores modificar la disposición y propiedades de los componentes de una forma visual y los cambios son reflejados inmediatamente en el código. A su vez cualquier cambio hecho en el código es reflejado en la vista de diseño. Algo similar ocurre para los flujos de páginas para JSF y Struts. Además ofrece la novedosa tecnología Oracle ADF (Oracle Application Development Framework o Esqueleto de Desarrollo de Aplicaciones de Oracle) que simplifica el desarrollo de aplicaciones J2EE por medio de los patrones de diseño y reduce las tediosas tareas de codificación de dichas aplicaciones.

Es una herramienta 100% basada en Java, lo que la convierte en una herramienta **multi-plataforma** que puede ser ejecutada en Sistemas Operativos para los que haya una implementación de la máquina virtual de Java (Windows, Linux, Unix, Macintosh, ...). Las aplicaciones construidas con Oracle JDeveloper pueden correr en cualquier Sistema Operativo compatible con Java (en el caso de aplicaciones cliente) o pueden ser desplegadas en cualquier servidor compatible con J2EE (para el caso de aplicaciones basadas en esta tecnología), y cualquiera de las dos anteriores puede acceder a cualquier Base de Datos compatible con JDBC.

La primera versión de JDeveloper surge de un acuerdo entre Oracle y Borland que permitió a Oracle modificar el código de la herramienta de desarrollo en Java ideada por Borland cuyo nombre era JBuilder. Después Oracle decidió reescribir completamente su herramienta y así hacer que esta fuera 100% compatible con Java y a su vez escrita en este mismo lenguaje. Posteriormente añadió su tecnología ADF y poco a poco fue añadiendo tecnologías y utilidades punteras en el mercado hasta llegar a la herramienta de la que actualmente disponemos. Su última versión a día de hoy es la 10.1.3.2 que fue liberada en Enero de 2007, aunque también existe una versión preliminar liberada en Mayo de 2007 y bautizada como Oracle JDeveloper 11g.

El tipo de licencia a la que está sujeta Oracle JDeveloper nos permite su descarga y su uso con fines de aprendizaje con un coste cero, pero siempre tendremos que tener en cuenta que si queremos utilizarlo para su uso en un proyecto real, tendremos que pagar la licencia.

PARA SABER MÁS

En el siguiente enlace podrás encontrar el tipo de licencia que nos ofrece Oracle, para la utilización de su herramienta JDeveloper 10g. Es recomendable que la leas, como suele ocurrir con cualquier producto que utilices, para que conozcas los términos legales de su utilización. La página está en inglés, como muchos de los enlaces que te recomendamos, pero ya debes saber que este es el lenguaje más utilizado en el argot informático.

Licencia de uso de Oracle JDeveloper 10g **[Versión en caché]**

1 **Autoevaluación**

Respecto a la licencia de uso de Oracle JDeveloper podemos afirmar...

a) JDeveloper no posee licencia de uso.
b) La licencia de uso de JDeveloper es una licencia GPL.
c) JDeveloper permite que lo usemos con fines de aprendizaje con coste cero.
d) No podemos descargarnos JDeveloper hasta que no pagamos por medio de una tarjeta VISA.

Comprobar

Marcar como leído

Unidad Didáctica II

Descarga e Instalación de Oracle Jdeveloper 10g.

CASO. Carmen *tiene la tarea de enseñar a* ***Víctor*** *a utilizar la herramienta que* ***José*** *les ha propuesto. Seguramente será la que utilice todo el equipo de* ***SI Andalucía*** *a partir de ahora para aumentar el rendimiento en todos sus proyectos y aprovechar al máximo todo lo que saben, así como el código que vayan haciendo.*

Para ello ***Carmen*** *y* ***Víctor*** *van a descargar e instalar la herramienta en las dos plataformas que más utilizan (Windows y Linux) y posteriormente empezar a realizar sus primeros*

programas que es como se aprende verdaderamente.

Como hemos dicho anteriormente la descarga de Oracle JDeveloper 10g es gratuita y lo único que nos pedirá es que nos registremos. Para este menester, nosotros ya nos hemos registrado por ti (aunque si quieres lo puedes hacer por tu cuenta). El usuario que hemos dado de alta es "**fpd@iesaguadulce.org**" y la contraseña "**clavefpd**". La versión que utilizaremos es la 10.1.3.2 que a día de hoy es la última estable.

En los siguientes subapartados veremos cómo podemos llevar a cabo la descarga e instalación de la herramienta para el Sistema Operativo Windows (en particular para Windows XP Service Pack 2) y para el sistema Operativo Linux (en particular para Guadalinex v4).

Oracle JDeveloper 10g está certificado para funcionar con el jdk 5.0 actualización 6 o superior.

Esto es bastante importante ya que lo hemos probado con el jdk 6.0 y hay algunas cosas que no funcionan bien, por lo que te aconsejamos que utilices el jdk 5.0 con su última actualización.

Otro de los requisitos que aconseja Oracle para la utilización de su herramienta es disponer de una buena máquina y con al menos 1GB de RAM. Nosotros lo hemos probado en un portátil Intel Centrino a 1.6 Ghz, con 512 MB de RAM y funciona perfectamente, aunque es un poco lento al arrancar y cuando empezamos a trabajar con proyectos muy complejos se empieza a notar la lentitud.

ZONA DE DESCARGA

Si visitas este enlace podrás acceder a la página de descarga de Oracle JDeveloper 10g para todas las plataformas.

Zona de descarga de Oracle JDeveloper 10g (10.1.3.2)

Desde el siguiente enlace accederás a la zona de descarga del jdk 5.0 para todas las plataformas, en el cual deberás escoger la última actualización que a día de hoy es la 12.

Zona de descarga del JDK 5.0

1 **Autoevaluación**

A la hora de instalar Oracle JDeveloper ...

a) Debemos tener en cuenta la versión del jdk que tenemos instalada en nuestra máquina.
b) Deberemos disponer de una máquina medianamente potente.
c) Podremos hacerlo para casi cualquier Sistema Operativo.
d) Todas las respuestas anteriores son correctas.

Comprobar

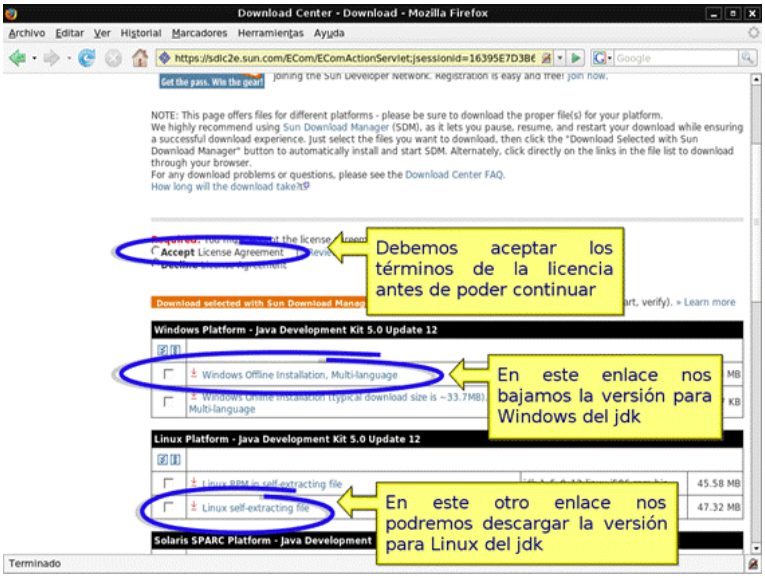
Marcar como leído

Unidad Didáctica II

Descarga e instalación de Oracle JDeveloper 10g en Windows XP SP2.

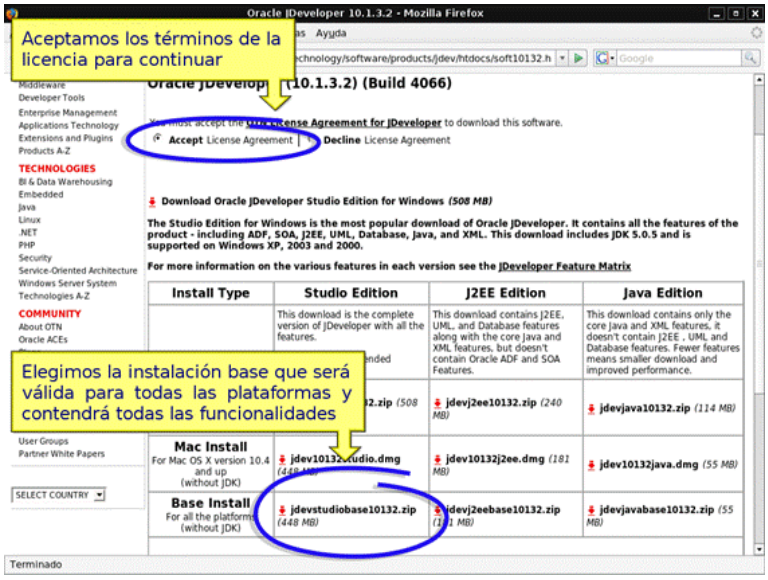
Para la descarga de nuestra herramienta para Windows, existen dos opciones, una que ya incluye el jdk y otra que no. Elegiremos la segunda, ya que es el método que se utiliza en todas las plataformas. Por tanto deberemos descargar e instalar primero el jdk 5.0 en su última actualización, para lo cuál accederemos a la zona de descarga del jdk y nos aparecerá una página como la siguiente, en la que optaremos por la descarga del jdk 5.0 actualización 12.

Aceptaremos los términos de la licencia y elegiremos la versión off-line para windows, para así poder realizar la instalación en otros equipos si fuese necesario sin necesidad de estar conectados a Internet.

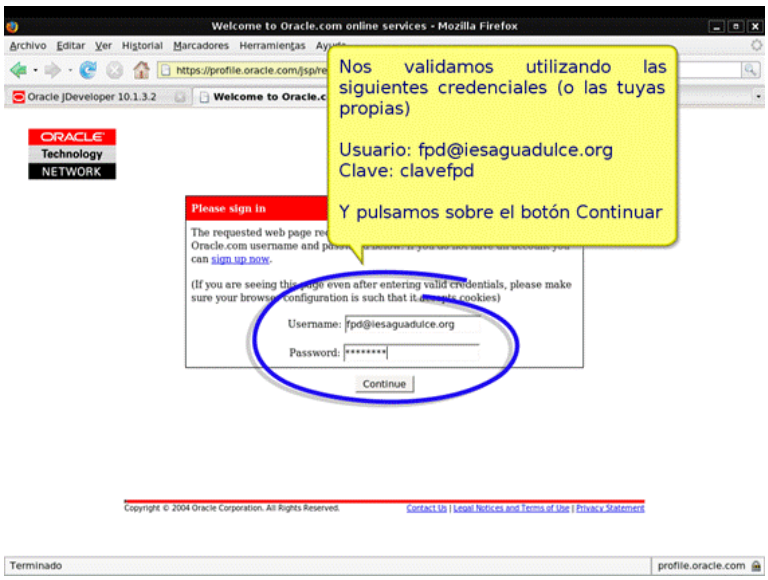


Una vez descargado el fichero simplemente tendremos que ejecutarlo para que comience su instalación y seguir los pasos por los que nos va guiando el asistente. Es muy sencilla y lo único que nos pregunta es el directorio en el que queremos que se instale el jdk.

Instalado correctamente el jdk, nos queda instalar nuestra herramienta para poder empezar a utilizarla. Lo primero que haremos será ir a la zona de descarga, aceptaremos los términos de la licencia y elegiremos la instalación base para la versión "Studio Edition" que es la versión completa.



Para que comience la descarga se nos pedirá que nos identifiquemos con los datos usados en nuestro registro, que como te dijimos hemos hecho por ti y son los que te comentamos al principio del apartado 2, o puedes usar los tuyos propios si es que te has registrado.



Una vez descargado el fichero, para lo que debemos ser pacientes debido a su tamaño, procedemos a instalarlo. JDeveloper **no tiene ningún instalador**, sino que **para instalarlo necesitarás simplemente descomprimir el fichero que te has bajado en un directorio cuyo nombre no contenga espacios en blanco** (por ejemplo, c:\jdeveloper, que a partir de ahora nos referiremos a él como jdev_home).

Una vez descomprimido, utilizamos un editor de texto que reconozca los caracteres fin de línea del tipo UNIX, como puede ser el WordPad de Windows. Editamos el fichero <jdev_home>\jdev\bin\jdev.conf para indicar la localización de nuestro jdk, por medio de la variable SetJavaHome. Por ejemplo, si hemos instalado el jdk en el directorio c:\jdk1.5_0_12, debemos editar el fichero y buscar la línea que haga referencia a la variable SetJavaHome y poner lo siguiente: SetJavaHome c:\jdk1.5.0_12. Ya tenemos instalada nuestra herramienta, ahora sólo ejecutamos el programa <jdev_home>\jdeveloper.exe o <jdev_home>\jdev\bin\jdevw.exe, o creamos un acceso directo en el escritorio por comodidad.

Marcar como leído

Unidad Didáctica II

Descarga e instalación de Oracle JDeveloper 10g en Guadalinex v4.

¿Eres usuario habitual de software libre?¿Tienes instalada alguna distribución Linux, como Guadalinex v4, y quieres seguir olvidándote de Windows? ¡No hay problema!

Para la descarga de nuestra herramienta para Guadalinex, sólo podemos optar por el método multiplataforma que no incluye el jdk. Por tanto debemos descargar e instalar primero el jdk 5.0 en su última actualización, para lo cuál accedemos a la zona de descarga del jdk en la que elegimos la descarga del jdk 5.0 actualización 12, exactamente igual que hicimos para Windows.

Ahora aceptamos los términos de la licencia y elegimos la versión de auto-extracción para Linux, y así podremos realizar la instalación en otros equipos si fuese necesario sin necesidad de estar conectados a Internet.

Una vez se haya descargado debemos abrir un Terminal y situarnos en el directorio en el que hayamos descargado nuestro fichero, hacer que el fichero sea ejecutable por medio de la orden chmod +x jdk-1_5_0_12-linux-i586.bin y posteriormente ejecutarlo mediante la orden ./jdk-1_5_0_12-linux-i586.bin para que comience el asistente y seguir los pasos por los que nos va guiando. Es muy sencilla y lo único que nos pregunta es el directorio en el que queremos que se instale el jdk.

Una vez hemos instalado correctamente el jdk, nos queda instalar nuestra herramienta para poder empezar a utilizarla. Lo primero que haremos será ir a la zona de descarga, aceptar los términos de la licencia y elegiremos la instalación base para la versión "**Studio Edition**" que es la versión completa. Para que comience la descarga se nos pedirá que nos identifiquemos con los datos usados en nuestro registro, que como te dijimos hemos hecho por tí y son los que te comentamos al principio del apartado 2, o puedes usar los tuyos propios si es que te has registrado.

Una vez descargado el fichero, para lo que debemos ser pacientes debido a su tamaño, procedemos a instalarlo. JDeveloper no tiene ningún instalador, sino que para instalarlo necesitarás simplemente descomprimir el fichero que te has bajado en un directorio cuyo nombre no contenga espacios en blanco (por ejemplo, /home/usuario/jdeveloper, que a partir de ahora nos referiremos a él como <jdev_home>). Una vez descomprimido, utilizamos un editor de texto para editar el fichero jdev_home>/jdev/bin/jdev.conf y así indicar la localización de nuestro jdk, por medio de la variable SetJavaHome. Por ejemplo, si hemos instalado el jdk en el directorio /home/usuario/jdk1.5_0_12, debemos editar el fichero y buscar la línea que haga referencia a la variable SetJavaHome y poner lo siguiente: SetJavaHome/home/usuario/jdk1.5.0_12.
Ya tenemos instalada nuestra herramienta, ahora sólo ejecutamos el programa <jdev_home>/jdev/bin/jdev, o creamos un lanzador en el escritorio por comodidad.

1 **Autoevaluación**

Respecto a la instalación de Oracle JDeveloper podemos afirmar...

- a) La instalación difiere mucho de cada Sistema Operativo y debemos bajarnos diferentes ficheros según el Sistema Operativo.
- b) El programa de instalación en Guadalinex es mejor que el de Windows ya que es más vistoso.
- c) No dispone de programa de instalación y simplemente debemos descomprimir el fichero que nos hemos bajado.
- d) El programa de instalación para Linux incluye el jdk y el de Windows no.

Comprobar

Marcar como leído

Unidad Didáctica II

Primeros pasos con Oracle Jdeveloper.

*CASO. **Víctor** ya ha descargado e instalado la herramienta en los dos Sistemas Operativos que utilizan en **SI Andalucía** sin ninguna complicación. Ya intuía que esto no sería complicado y lo que realmente está deseando es ponerse a trabajar con el entorno para comprobar las ventajas que le ofrece. Para ello cuenta con la inestimable colaboración de **Carmen** que ya había utilizado anteriormente esta herramienta y que además ha ido adelantando trabajo y leyendo documentación de la misma. **Carmen** propone a **Víctor** que lancen la aplicación y echen un vistazo a su aspecto para ir familiarizándose con la misma. A **Víctor** le parece una idea estupenda y propone a **Carmen** que vayan a por un café antes de empezar a realizar su primer programa en Java con JDeveloper.*

Me imagino que al igual que Víctor no habrás tenido ningún problema a la hora de descargar e instalar la herramienta que utilizaremos a partir de ahora. En el resto de esta unidad (y el resto del módulo) comprobarás las ventajas que ofrece esta herramienta y verás cóimo las posibilidades que nos brinda son casi ilimitadas.

Para ir familiarizándonos con el entorno, al igual que Carmen y Víctor, empezaremos echándole un vistazo y explicando el aspecto del mismo así como su manejo. Aunque muchas de sus funcionalidades las iremos descubriendo conforme vayamos avanzando materia.

Y como la mejor forma de ir cogiendo manejo con nuestro entorno es trabajando con él, lo que haremos seguidamente será realizar nuestro primer programa en Java, para lo que recurriremos al clásico "Hola Mundo!", que aunque muy simple nos sirve para empezar a manejar la herramienta.

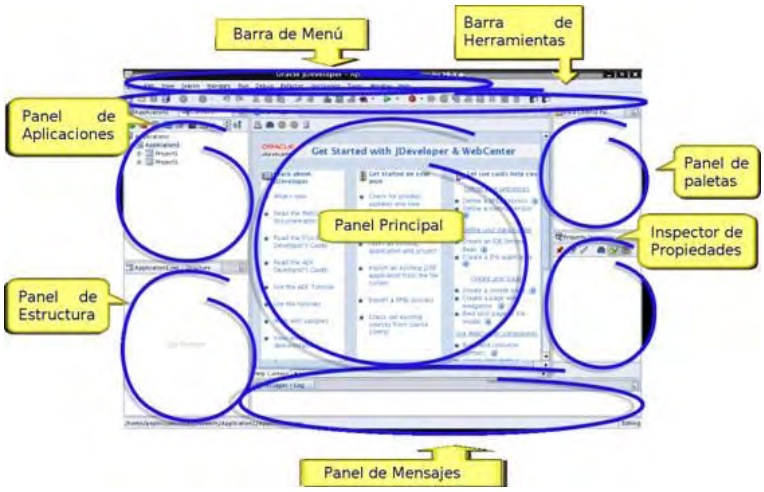
Posteriormente realizaremos algo un poco más elaborado, aunque tampoco para echar chispas, que nos permitirá adentrarnos en algunas funcionalidades que nos ofrece JDeveloper. Finalmente le pondremos una interfaz gráfica a dicho programa utilizando Swing.

Marcar como leído

Unidad Didáctica II

Familiarizándonos con el entorno.

Una vez hemos descargado e instalado nuestra herramienta, sólo nos queda lanzarla y empezar a trabajar con ella. Como vimos la instalación dependerá de la plataforma o Sistema Operativo que estemos utilizando. Sin embargo, su uso será idéntico en cualquier plataforma, de hecho como hemos podido comprobar descargamos lo mismo para usar en cualquier plataforma, lo único diferente era la máquina virtual de Java, pero el programa en sí no varía.



La imagen anterior corresponde a nuestra herramienta en ejecución. Nos encontramos con un entorno bastante amigable, el cuál esta estructurado de la siguiente forma:

1. **Barra de menús**: Como en cualquier aplicación, aquí encontraremos todas las posibilidades que nos ofrece la herramienta agrupadas por funcionalidades. Todas esas opciones las iremos descubriendo poco a poco conforme vayamos profundizando en el uso de la herramienta.
2. **Barra de herramientas**: En esta barra encontramos atajos a las funciones más comunes y más utilizadas, aunque siempre podremos acceder a ellas por medio de su opción de menú correspondiente. Por lo tanto podemos prescindir de ella. Por eso la herramienta nos permite que no se muestre mediante la opción "View | Toolbars | Main" (si está seleccionada nos la mostrará y de lo contrario no).
3. **Áreas de trabajo**: Las áreas de trabajo son cuatro (izquierda, inferior, derecha y central) y éstas contienen paneles. Los paneles sirven para mostrar información, realizar algún tipo de operación, mostrar información de forma estructurada, permitirnos introducir algún tipo de valores..., todo dependerá del tipo de panel. Los paneles se pueden minimizar, y cuando un área tiene todos sus paneles minimizados, ésta se encoge permitiendo ganar espacio al área de trabajo central, en la que se sitúa el panel principal. Aunque en un principio el panel central nos muestra la página de inicio con acceso a algunos temas de ayuda, luego será desde donde haremos la mayor parte de nuestro trabajo, ya que en él diseñaremos y codificaremos nuestras aplicaciones. En el panel central será donde se nos mostrarán los diferentes ficheros que creemos en nuestras aplicaciones. Podemos mostrar múltiples ficheros, y acceder a estos por medio de las pestañas que nos irán apareciendo. Cada una de estas pestañas corresponde a uno de los ficheros que tengamos abiertos y que podremos identificar por su nombre en dicha pestaña. Iremos viendo para qué nos sirven los demás paneles conforme vayamos haciendo uso de los mismos.

Ya hemos visto el aspecto que tiene nuestro entorno de trabajo. Antes de ponernos manos a la obra creando nuestra primera aplicación, debemos conocer los términos que éste utiliza para manipular las aplicaciones que queremos crear, ya que algunas contendrán muchos ficheros y tendremos que saber cómo agruparlos de una forma lógica para luego no volvernos locos buscándolos.

JDeveloper utiliza el término **Aplicación** como el nivel más alto a la hora de agrupar ficheros. Por tanto para empezar a trabajar lo primero que debemos hacer es crear una nueva aplicación. La estructura de una aplicación es almacenada en un fichero con extensión **.jws**, para posteriormente poder volver a abrirla y manipularla. Dicha estructura podemos verla en el panel de aplicaciones. A la hora de crear aplicaciones JDeveloper nos ofrece varias plantillas según el tipo de aplicación que queramos crear.

Una aplicación está compuesta de **Proyectos** que son organizaciones usadas para agrupar lógicamente ficheros relacionados. Un proyecto contendrá **ficheros fuente**, **paquetes**, **clases**, **imágenes** y cualesquiera **otros elementos** que nuestro programa pueda necesitar. Los Proyectos gestionan además las **Variables** de entornonecesarias en la compilación y ejecución de nuestros programas, y las **Opciones** que personalizan el comportamiento de nuestro entorno. En el panel de aplicaciones, los proyectos se muestran como el segundo nivel de la jerarquía.

Con todo lo visto hasta ahora estamos en disposición de poder crear nuestra primera aplicación. ¡Pues vamos al lío!

1 **Autoevaluación**

El panel de aplicaciones se encuentra en el área...

- a) Izquierda.
- b) Derecha.
- c) Inferior.
- d) Central.

Comprobar

Marcar como leído

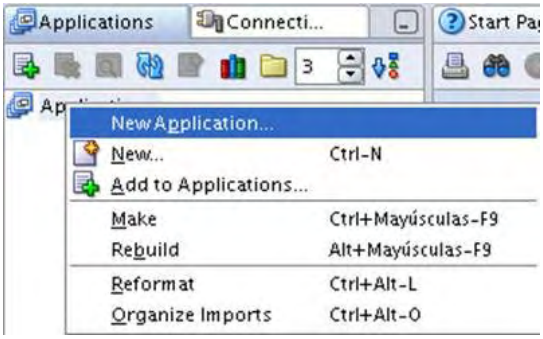
Unidad Didáctica II

Nuestra primera aplicación.

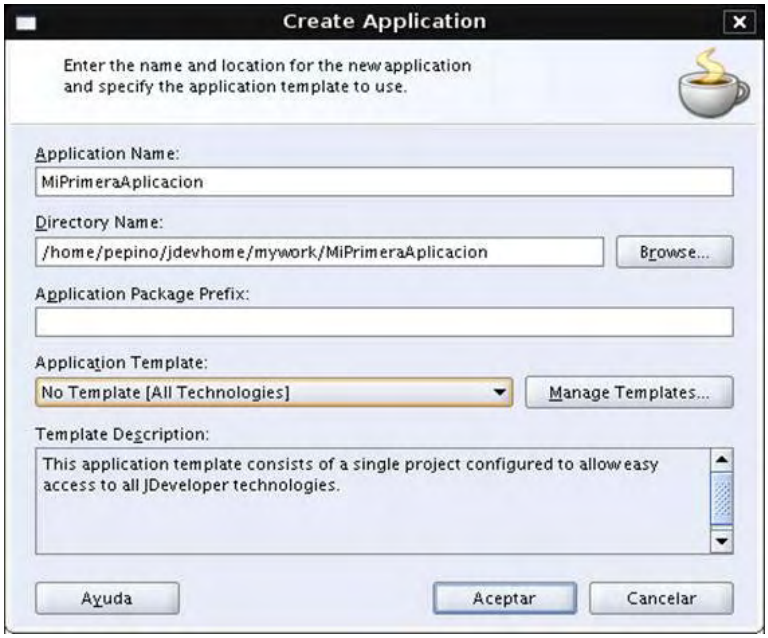
Ya hemos visto como es nuestro entorno en ejecución y con ello hemos perdido el miedo escénico que suele generar el enfrentarnos a una nueva herramienta. Pero para perder definitivamente ese miedo nada mejor que realizar nuestro primer programa y comprobar lo fácil que puede llegar a ser trabajar con JDeveloper.

La primera aplicación que haremos es el clásico "Hola Mundo", que seguro ya habéis realizado con otros entornos y sabéis que su complejidad es ínfima. Aunque es una aplicación muy simple es un buen punto de partida ya que nos permite comprender la filosofía de trabajo con nuestro entorno.

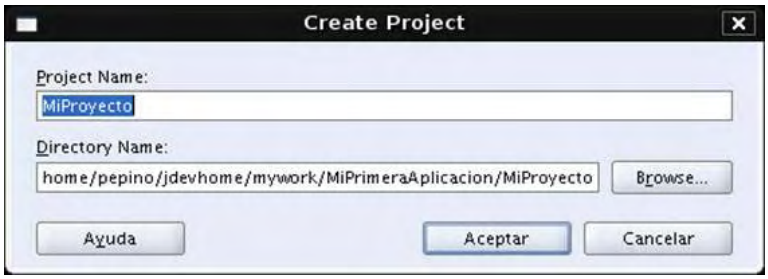
Para empezar a crear nuestro primer programa lo primero que debemos hacer es crear una aplicación en JDeveloper. Por lo que una vez lanzado el entorno nos situaremos en el panel de aplicaciones y pulsando el botón derecho del ratón nos aparecerá un menú contextual en el cuál elegiremos la opción "New Application".



JDeveloper lanzará un asistente que nos preguntará sobre el nombre de nuestra aplicación, nos indicará la ruta donde se creará el directorio que contendrá la aplicación. Podremos poner un prefijo al paquete de la aplicación y nos permitirá elegir una de las plantillas existentes. En principio nosotros simplemente le daremos nombre a la aplicación y todo lo demás lo dejaremos como viene por defecto, ya que la ruta apunta a nuestro directorio de trabajo, que será donde crearemos todas las aplicaciones, seguida por el nombre de la aplicación, el prefijo del paquete si no ponemos nada será el mismo que el nombre del proyecto y las plantillas de momento no nos interesan. El nombre que hemos elegido es "**MiPrimeraAplicación**"

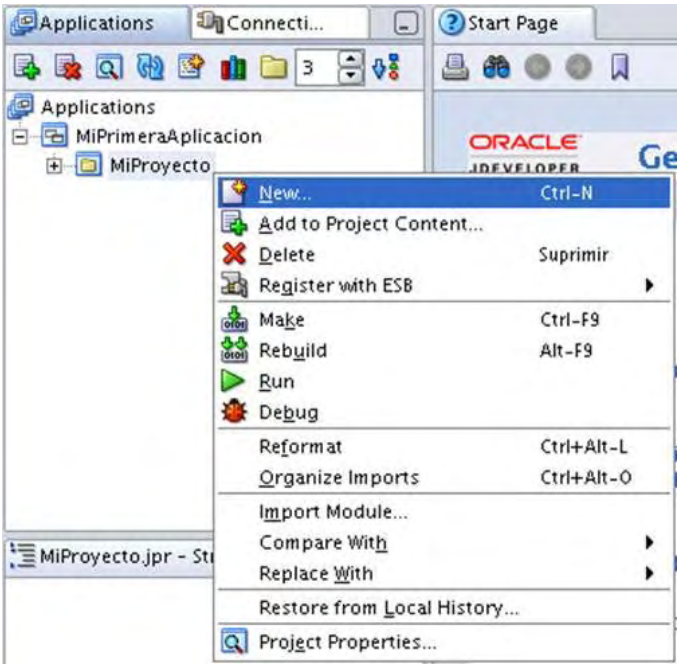


Como recordarás del punto anterior, la estructura que usaba JDeveloper era aplicaciones y proyectos, por lo que JDeveloper antes de terminar este asistente nos preguntará por el nombre que queremos darle a nuestro proyecto y si queremos cambiar el directorio. Nosotros nombraremos el proyecto como "**MiProyecto**" y el directorio lo dejaremos tal cual, ya que nos sigue interesando que sea ese.

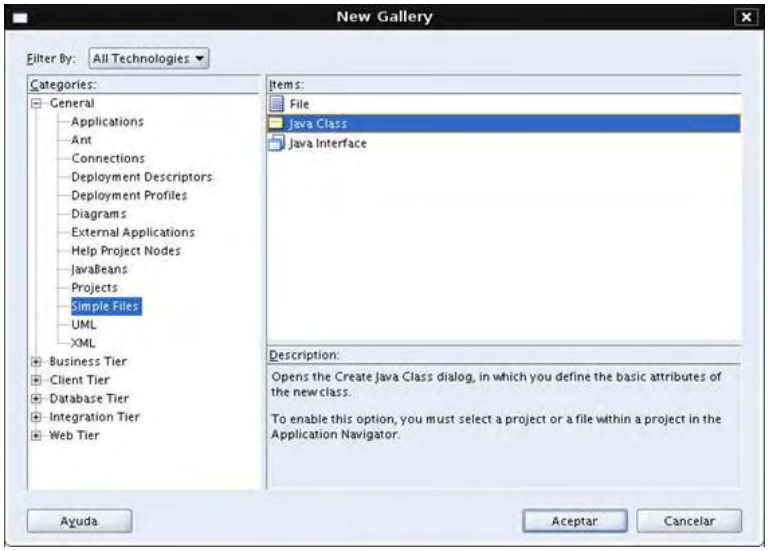


Ahora podemos examinar el panel de aplicaciones y comprobar la estructura que JDeveloper ha creado para que vayamos incluyendo ficheros en la misma.

Nuestro siguiente paso es añadir una clase Java al proyecto, para lo que nos situaremos sobre nuestro proyecto en el panel de aplicaciones y pulsando con el botón derecho del ratón se nos desplegará un menú contextual en el que deberemos elegir la opción "New".

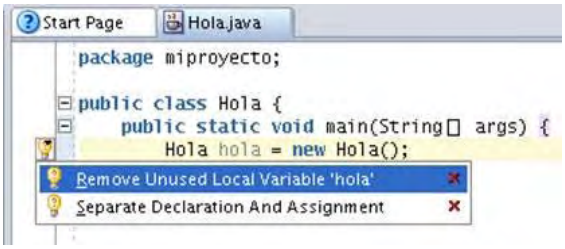


JDeveloper nos presentará una ventana que nos permitirá elegir qué queremos añadir. Como podemos ver está dividida en categorías y nosotros deberemos desplegar la categoría "General" y en ella elegimos "Simple File". Podemos ver que para ficheros simples nos muestra tres opciones, entre ellas la que a nosotros nos interesa, que es una clase Java.



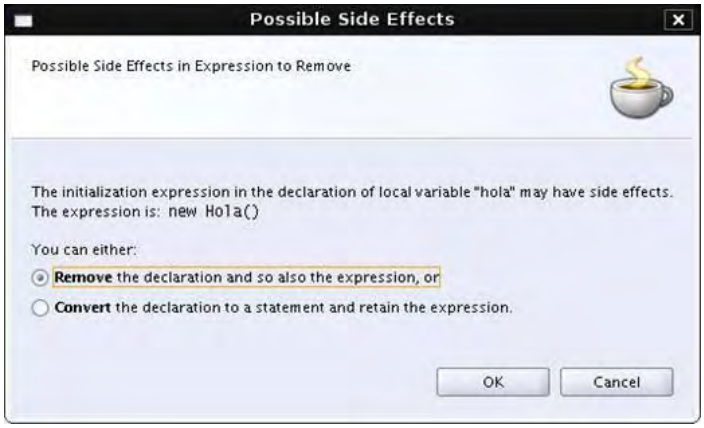
Seguidamente JDeveloper nos preguntará por el nombre de la clase, el paquete al que queremos que pertenezca y si heredará de alguna clase en especial. También tiene algunas opciones que permitirán que JDeveloper haga cosas por nosotros, como por ejemplo hacer dicha clase pública o no, generar un constructor por defecto para la misma o no y generar un método main para ella o no hacerlo. En nuestro caso nombraremos la clase como "**Hola**", el paquete al que pertenece lo dejaremos igual y como no heredera de ninguna clase en especial también dejaremos que extienda de "java.lang.Object". Entre las opciones deberemos marcar que sea pública y que genere un método main, pero no nos interesa que genere un constructor por defecto ya que no nos va a hacer falta.

Al pulsar sobre el botón "Aceptar", el editor de código se abrirá automáticamente mostrando el código generado. Recuerda que el editor de código está situado en el panel principal del área central. El código generado será el siguiente.

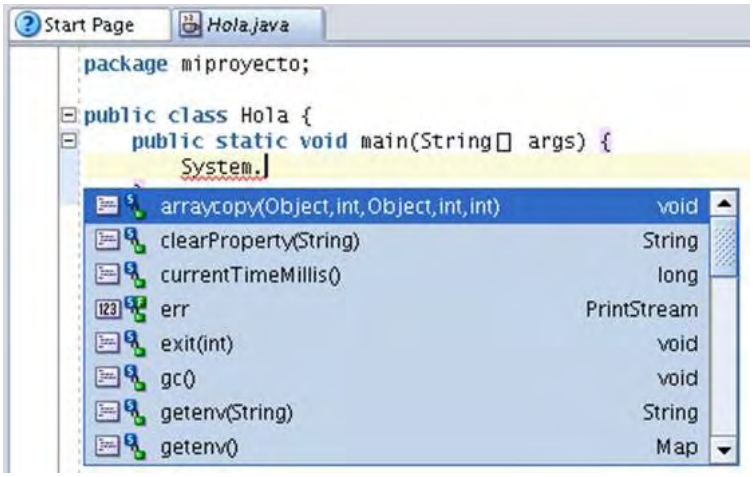


Como podemos ver el código generado es correcto, aunque hay algo que sobra y que no es necesario y es la línea que define una variable del tipo "Hola" y la inicializa. Como esta línea no es necesaria vamos a decirle a JDeveloper que la elimine (aunque también podríamos hacerlo nosotros a mano). Para ello nos situamos sobre dicha línea y vemos que nos aparece una bombilla amarilla en el margen izquierdo del editor de código, lo que nos indica que JDeveloper ha detectado que hay algo que se podría arreglar. Si pulsamos sobre la bombilla nos mostrará las opciones que él cree que podríamos llevar a cabo para corregir esta situación. Una de las opciones es borrar dicha variable, que será la que nosotros elegiremos.

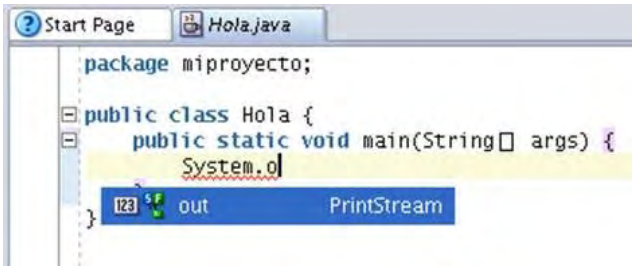
Al elegir esta opción JDeveloper nos avisa que esto puede tener efectos laterales y quiere que confirmemos lo que realmente queremos hacer. Como nosotros lo que queremos es borrarla, elegimos la opción "Remove".



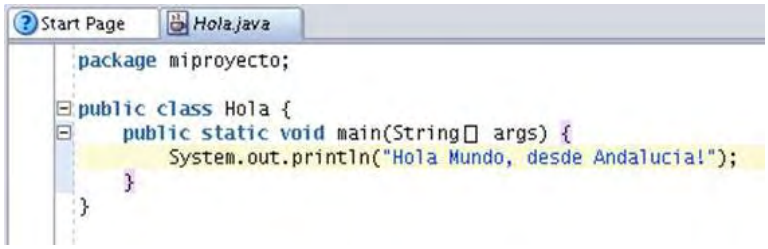
Podemos comprobar que la línea que no nos interesaba ha desaparecido, por lo que ahora deberemos insertar la sentencia que queremos que realice nuestro programa. Para ello escribimos "System." y JDeveloper nos muestra los métodos y/o variables que nos ofrece la clase "System".



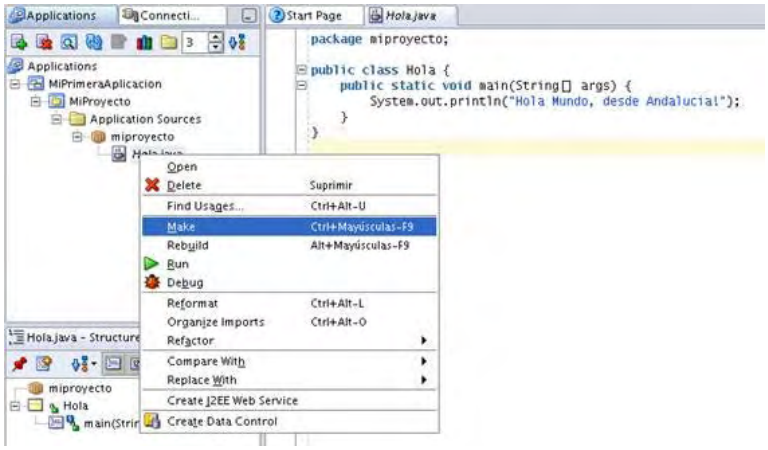
Si ahora pulsamos la letra "o" nos mostrará las opciones que empiezan por o.



Aceptamos la única que nos ofrece que es la que nosotros buscamos y al teclear el "." nos volverá a sugerir los métodos y/o variables de esta variable de clase. Pulsando la "p" buscamos el método "println()" que es el que deseamos y aceptamos. Como podemos ver JDeveloper ha generado la sentencia que buscamos y lo único que debemos hacer es introducir la cadena que queremos visualizar. Para ello al teclear la primera comilla, vemos que él autocompleta cerrando la misma y lo único que queda es introducir el texto que en nuestro caso será "Hola Mundo, desde Andalucía!". El código de nuestra clase quedará así.



Ya sólo nos queda compilar nuestro programa y ejecutarlo. Para llevar a cabo la primera operación debemos ir al panel de aplicaciones y sobre nuestro fichero "Hola.java" pulsamos el botón derecho del ratón para acceder al menú contextual y elegimos la opción "**Make**" para llevar a cabo la compilación.



Si todo va bien en el panel de mensajes se nos informará que la compilación se ha llevado a cabo con éxito.

Ahora sólo nos queda ejecutar nuestro programa, para lo que actuaremos de igual forma que en el paso anterior, pero en este caso elegimos la opción "Run". Podemos observar como en el área inferior aparece una nueva pestaña que nos mostrará la salida del proceso que estamos ejecutando. Podemos comprobar que el programa nos saluda como era de esperar y termina correctamente.

Bueno, como ves esto ha sido realmente fácil, ¿no? En el siguiente enlace puedes acceder a una presentación completa en la que se detalla todo el proceso seguido para realizar nuestra primera aplicación.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

Para realizar nuestra primera aplicación...

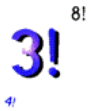
- a) Al ser una aplicación simple no hemos tenido que crear ni aplicación ni proyecto en el entorno.
- b) Hemos creado una aplicación en la que directamente hemos colocado los ficheros java.
- c) Aunque era una aplicación simple hemos tenido que crear una aplicación y dentro de ella un proyecto en el que hemos incluido los ficheros java.
- d) Al ser una aplicación simple sólo hemos tenido que crear un proyecto y en él hemos añadido ficheros.

Comprobar

Marcar como leído

Unidad Didáctica II

Profundizando en el entorno.



En el apartado anterior hemos podido comprobar lo fácil que ha resultado crear nuestra primera aplicación. Pero estarás pensando, ¿me hubiera costado mucho trabajo teclear ese fragmento de código en un editor de texto y compilarlo y ejecutarlo con el jdk? Efectivamente, debido a que nuestro programa era muy simple, en él no hemos podido apreciar la potencia que nos brinda JDeveloper. En este apartado vamos a explorar muchas de las posibilidades que nos ofrece Oracle JDeveloper para hacernos la vida más fácil a la hora de crear nuestras aplicaciones.

Para poder aplicar los conceptos que vayamos exponiendo, nos basaremos en una pequeña aplicación que calcula el factorial de 10 y visualiza el resultado por pantalla. Conforme vayamos avanzando iremos modificando nuestro programa hasta crear un programa más reutilizable con muy poco esfuerzo. El ejemplo está planteado de una manera didáctica para que de este modo sirva para afianzar los conceptos expuestos. No pretende ser una lección magistral de programación orientada a objetos, ya que si se tratase de eso seguro que el problema se habría planteado de una forma muy diferente.

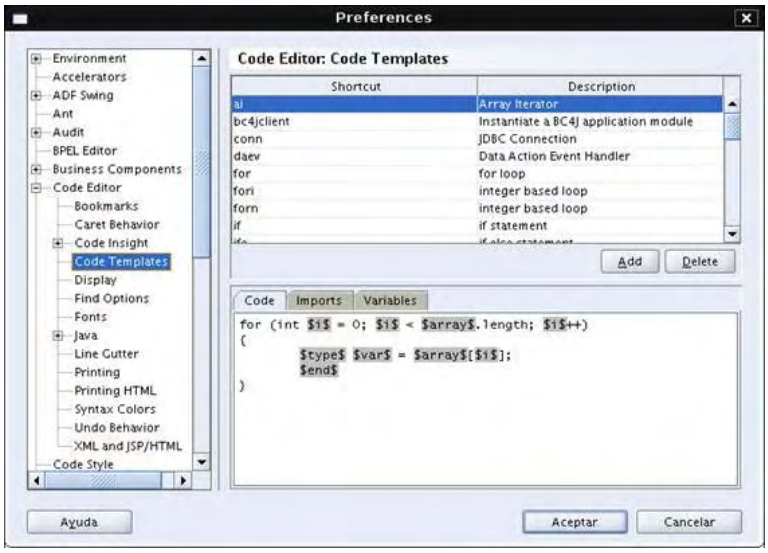
Marcar como leído

Unidad Didáctica II

Ayudándonos a codificar (I)

Ya hemos podido comprobar en nuestra primera aplicación como JDeveloper nos ayuda a la hora de codificar nuestras clases. En este punto vamos a repasar esas posibilidades y a exponer algunas nuevas que nos serán muy útiles en nuestro día a día como programadores. Veamos algunas de esas posibilidades.

- **Auto-completando código**. Esta característica ya la utilizamos en nuestra primera aplicación y nos permite desplegar una lista de métodos y/o variables de una clase conocida. Recuerda que la utilizamos a la hora de crear la sentencia System.out.println. Esta opción se activa cuando JDeveloper reconoce una clase conocida (ya sea del jdk o bien una creada por nosotros y a la que tenemos acceso). Esta opción podemos forzarla por medio de las teclas **Ctrl + Espacio**.
- **Asistente de código**. ¿Recuerdas la bombilla amarilla que nos aparecía en el margen izquierdo del editor de código y que utilizamos para eliminar una línea no deseada generada por JDeveloper? Pues esta es la funcionalidad denominada "asistente de código" y consiste en una serie de sugerencias que JDeveloper nos ofrece para solucionar problemas o para avisarnos de que algo podríamos cambiarlo. Las bombillas pueden ser de color amarillo indicando alguna sugerencia o de color rojo indicando que ha detectado un error.
- **Plantillas de código**. Las plantillas de código nos ayudan a escribir código más rápidamente ofreciéndonos atajos para sentencias que se usan con frecuencia en los programas. Para acceder a estas plantillas podremos pulsar Ctrl + Enter y JDeveloper nos mostrará la lista de plantillas que posee para que elijamos la deseada, o si conocemos su atajo, teclearlo y pulsar Ctrl + Enter y directamente JDeveloper nos lo expandirá. Por ejemplo, a la hora de crear la sentencia System.out.println podríamos haber tecleado sop + Ctrl + Enter y directamente nos la hubiera expandido. JDeveloper nos ofrece la posibilidad de definir nuestras propias plantillas accediendo a la opción de menú "**Tools | Preferences**" y eligiendo la categoría "**Code Editor**" y la opción "**Code Templates**".



En el siguiente enlace puedes comprobar como hemos utilizado las plantillas de código para realizar nuestro programa que calcula el factorial del número 10 y que será el que seguiremos a lo largo de este apartado para ir exponiendo algunas de las posibilidades que nos brinda JDeveloper para facilitarnos la vida.

Pulsa aquí para ver la simulación

Marcar como leído

Unidad Didáctica II

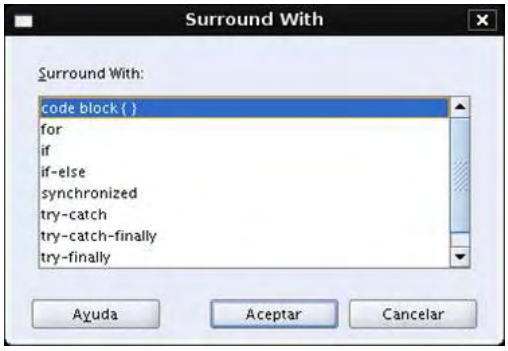
Ayudándonos a codificar (II)

Seguimos repasando las posibilidades que nos brinda JDeveloper y exponinedo algunas nuevas. En ese sentido, este epígrafe, como habrás deducido de su título, no es más que la continuación del anterior.

- **Generación de métodos de acceso**. Cuando trabajamos con la filosofía orientada a objetosjetos es aconsejable seguir las buenas prácticas de programación, por lo que las variables de instancia se recomienda que estén encapsuladas a través de métodos de acceso **get** y **set**. Para esto JDeveloper nos ofrece la opción de que una vez creada la variable, acceder al menú "**Source | Generate Accessors**" y que él los cree por nosotros. Para ello nos mostrará un asistente en el que podremos decirle de qué variables queremos generar dichos métodos y además podremos elegir entre generar ambos o sólo uno de ellos. Esta funcionalidad aún no es aplicable a nuestro ejemplo, pero te mostramos una presentación que demuestra su uso para una clase aún en construcción (esto ha sido sólo a modo de ejemplo, ya que este paso no lo tendremos en cuenta a la hora de seguir con nuestro programa de cálculo del factorial, por lo que desharemos los cambios hechos en nuestra aplicación para poder continuar). La presentación muestra cómo

crear una clase Factorial que contenga una variable de instancia privada resultado y en la cuál crearemos los métodos de acceso mediante la funcionalidad vista en este apartado.

Pulsa aquí para ver la simulación



- **Encerrar código**. ¿Cuántas veces te ha pasado que estás codificando y al escribir una sentencia recuerdas que ésta debe ir encerrada entre un bloque try - catch? Pues esto también es capaz de hacerlo JDeveloper por tí. Esta funcionalidad es accesible a través de la opción "**Source | Sourround With**" del menú principal. Al acceder a ella veremos cómo nos pregunta entre qué estructura queremos encerrar nuestro código seleccionado.
- **Reformatear código**. Otra tarea muy común cuando estamos desarrollando nuestras aplicaciones es tener que darle un formato legible a nuestro código, sobre todo cuando estamos haciendo pruebas y borramos fragmentos de código, introducimos otros nuevos, ... JDeveloper se encarga de hacer esto por nosotros y le da un formato legible a nuestro código, dando la anidación adecuada a cada bloque de código. A esta funcionalidad podemos acceder a través de la opción del menú "**Source | Reformat**" y ya verás como es muy cómoda para dejar nuestros desarrollos legibles.
- **Otras**. En la opción de menú "**Source**" puedes ver que hay otras opciones más triviales pero que a veces nos pueden venir bien. Podemos ver cómo podemos insertar o quitar comentarios a través de la opción "**Toggle Line Comments**". También podemos anidar un nivel más o quitarlo de un bloque seleccionado a través de las opciones "**Indent Block / Unindent Block**". También podemos acceder a la documentación de una clase del jdk a través de la opción "**Quick Javadoc**".

1 **Autoevaluación**

Dentro de las opciones que nos ofrece Oracle JDeveloper podemos afirmar que...

- a) Nos ofrece la posibilidad de reformatear nuestro código, poniendo la anidación adecuada a cada bloque de código.
- b) Nos ayuda a codificar mediante el uso de plantillas de código.
- c) Tiene un asistente de código que nos va sugiriendo posibles mejoras que podemos introducir en nuestro código.
- d) Todas las respuestas anteriores son correctas.

Comprobar

Marcar como leído

Unidad Didáctica II

Refactorización

La **refactorización** es el proceso que permite mejorar la calidad de nuestro código, casi siempre en busca de la **reusabilidad** y la **claridad**, sin modificar el comportamiento del mismo. Existen diversas técnicas de refactorización y son muy utilizadas en el desarrollo de software, ya que generalmente añadimos nuevas funcionalidades a nuestro código, las probamos y cuando estamos seguros de que funcionan correctamente entonces nos dedicamos a limpiar el código o a refactorizar.

JDeveloper nos ofrece varias técnicas de refactorización a las cuales podemos acceder por medio de la opción de menú "**Refactor**". Veremos en qué consisten y después las utilizaremos para hacer de nuestro programa que calcula el factorial del número 10 un programa más reutilizable y algo más claro (aunque evidentemente, debido a su sencillez, el programa es bastante claro en sí). Veamos primero las posibilidades que nos ofrece.

1. **Rename**. Renombra una clase, un método, un parámetro, una variable..., actualizando todas sus referencias.
2. **Move**. Mueve métodos de una clase a otra.
3. **Inline**. Reemplaza una variable o una constante por su valor literal, o la llamada a un método por los contenidos del mismo.
4. **Delete Safely**. Chequea antes de borrar para asegurarse de que lo que vas a eliminar no está siendo usado en cualquier otro lugar. Puede ser usado con clases, métodos y campos.
5. **Duplicate**. Duplica una clase.

6. **Make Static**. Convierte un método de instancia en un método de clase.
7. **Pull Member Up / Down**. Mueve la declaración de métodos o campos arriba o abajo en la jerarquía de clases y actualiza sus referencias.
8. **Extract Interface**. Crea una interfaz para cualquiera de los métodos públicos de la clase actual e implementa dicha interfaz para la clase actual.
9. **Extract Superclass**. Crea una superclase de la clase actual con los métodos y campos que le indiquemos.
10. **Use Supertype Where Posible**. Reemplaza todas las ocurrencias de una clase por su superclase cuando sea posible.
11. **Convert Anonymous to Inner Class**. Convierte una clase anónima a una clase nombrada.
12. **Change Method**. Nos permite cambiar la firma de cualquier método e incluso combinar dos métodos sobrecargados en uno sólo.
13. **Encapsulate Field**. Encapsula el acceso a un campo por medio de métodos de acceso y reemplaza todas las referencias a ese campo por dichos métodos de acceso (a diferencia de lo que hacía la funcionalidad "**Generate Accessors**" que simplemente generaba los métodos).
14. **Introduce Field / Variable / Parameter / Constant**. Reemplaza una expresión seleccionada por un campo, una variable local, un parámetro o una constante.
15. **Extract Method**. Crea un nuevo método a partir del fragmento de código seleccionado, añadiendo los parámetros que sean necesarios y devolviendo un valor si posteriormente se utiliza alguna variable del código seleccionado. Si se utiliza más de una variable no podrá crear una función que devuelva más de un valor y nos informará del error.
16. **Replace Constructor With Factory Method**. Crea un método de fábrica para crear la clase y convierte el actual constructor en privado.

Ahora que sabemos las técnicas que JDeveloper nos ofrece para refactorizar, vamos a aplicarlas a nuestro programa que calcula el factorial de 10. Para ello, lo primero que podemos apreciar es que todo lo hacemos en el método **main**, y nos gusta más tener un método que calcule el factorial y nos devuelva el resultado. Por tanto vamos a aplicar la técnica "**Extract Method**" para llevar a cabo dicha operación. Vemos como hacerlo en la siguiente presentación.

Pulsa aquí para ver la simulación

Podemos apreciar que el programa es un poco más modular pero muy poco reutilizable, ya que el método siempre calcula el factorial de 10. Para que el método calcule el factorial de cualquier entero que le pasemos, aplicaremos la técnica "**Introduce Parameter**", para que el 10 que controla el bucle lo sustituya por un parámetro que se le pase al método. Una vez hecho esto el nombre de nuestro método no tiene mucho sentido, por lo que lo cambiaremos por factorial. Después de realizar esto vemos que JDeveloper ha modificado la llamada a nuestro método y le pasa como argumento el número 10, ya que debemos recordar que la refactorización en ningún momento cambia el comportamiento de nuestro programa. Veamos esto en una presentación.

Pulsa aquí para ver la simulación

Por último hemos pensado que estaría mejor tener una clase llamada Factorial que sería la encargada de calcular el factorial de un número. Para ello creamos la clase vacía y mediante la técnica de refactorización "**Move**", movemos el método de nuestra clase CalcularFactorial10 a la clase Factorial. Podemos apreciar cómo a la llamada al método se le antepone el nombre de nuestra nueva clase. Por último renombraremos el nombre del método que ahora se podría llamar calcular por estar dentro de la clase Factorial. Finalmente ejecutamos nuestra aplicación y comprobamos que efectivamente hace lo mismo que antes y sin embargo su código es más claro, más legible, más reutilizable, en fin que es un código de más calidad y todo ello con muy poco esfuerzo.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

La técnica de refactorización consiste en:

- a) Mejorar nuestro código sin modificar su comportamiento.
- b) Sacar factores de nuestros métodos.
- c) En Oracle JDeveloper aún no podemos utilizar refactorización.
- d) Permite mejorar nuestro código y corregir los errores del mismo.

Comprobar

Marcar como leído

Unidad Didáctica II

Depuración de aplicaciones.

Seguro que ya has depurado más de una aplicación con otros entornos de trabajo. Las posibilidades que ofrece JDeveloper son muy parecidas a las posibilidades que nos ofrece cualquier otro entorno, por lo que todo esto te debería resultar familiar, pero lo refrescaremos un poco.

Podemos acceder a las opciones de depuración a través de la opción de menú "**Debug**". En ella podemos encontrar como hemos dicho anteriormente las opciones típicas (que casi todas están también en la barra de herramientas) y las más importantes son las siguientes.

1. **Debug Project / Debug File**. Con esta opción podemos empezar una sesión de depuración de un proyecto o de un fichero particular.
2. **UI Debug Project**. Esta opción permite depurar proyectos que contienen interfaces gráficas.
3. **Debug with Diagram Project**. Por medio de esta opción podremos depurar proyectos y seguir su diagrama de secuencia si es que éste lo hemos creado anteriormente.

4. **Toggle Breakpoint**. Esta opción nos permite poner y quitar puntos de ruptura. Otra forma más directa de hacer esto es pulsando sobre el margen izquierdo del editor de código y veremos cómo nos aparece un punto rojo en dicho margen. Si queremos quitarlo simplemente debemos volver a pulsar sobre el mismo.
5. **Pause / Resume**. Nos permite parar momentáneamente la depuración o ir hasta el final de la depuración ignorando los puntos de ruptura.
6. **Step Over / Into / Out / to End of Method**. Son las opciones típicas de depuración que nos permiten depurar línea a línea y si esta línea es un método lo ejecuta entero (**Step Over**), o se mete dentro y sigue depurándolo línea a línea (**Step Into**). Una vez en el método tenemos la opción de salirnos de él (**Step Out**) o de ir hasta el final del mismo (**Step to End of Method**).
7. **Run to Cursor**. Continúa la depuración hasta llegar a la línea donde esté situado el cursor en el editor de código.

Cuando comenzamos una **sesión de depuración** nos aparecen **nuevos paneles**:

- Un panel nuevo situado en el área izquierda que representa la pila (**Stack**) de nuestro programa.
- Una nueva pestaña en el área inferior representando el panel de depuración propiamente dicho, en el que se nos irá informando de los sucesos de la depuración.
- Un panel también en el área inferior nos muestra los puntos de ruptura.
- Y un panel, en el área inferior, pero en la parte derecha, en el que podremos ver las variables de nuestro programa, los datos del mismo e incluso evaluar expresiones. Veamos una pequeña demostración de una sesión de depuración para nuestra aplicación.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

Cuando creamos una sesión de depuración...

a) Podemos ver el estado de las variables pero nunca modificarlas.
b) Podemos ver el valor de las variables pero no podemos evaluar expresiones entre variables.
c) Podemos movernos entre puntos de ruptura pero nunca podemos movernos hasta donde está situado el cursor.
d) Podemos ver el estado de las variables, modificarlas y evaluar expresiones entre variables.

Comprobar

Marcar como leído

Unidad Didáctica II

Generando documentación.

Como ya debes saber, en Java es posible generar documentación para nuestras clases y métodos de una forma rápida y sencilla. Esto es posible mediante le herramienta "**Javadoc**" que genera de forma automática un grupo de archivos HTML en base a una sintaxis de comentarios insertados en el código fuente. Recuerda que comentar un código es algo importante ya que nos ayuda a recordar el cometido de una clase o un método.

Los comentarios de los que Javadoc generará documentación deben empezar por la cadena /** y deben finalizar por la cadena */. Dentro de estos delimitadores podemos poner texto descriptivo, etiquetas HTML o etiquetas que le indiquen a la herramienta el significado de ese comentario. Recuerda que las etiquetas debían ir precedidas por el símbolo de la arroba y podríamos usar las siguientes:

1. **@author.** Indica el autor de la clase que estamos comentando o del método.
2. **@deprecated.** Indicará que el método o clase ha sido reemplazado por otro en las nuevas versiones.
3. **@exception.** Nos informará de las excepciones que puede generar.
4. **@param.** Con esta etiqueta informamos del significado de nuestros parámetros.
5. **@return.** Informará del valor devuelto por el método que estamos comentando.
6. **@see.** Es posible insertar una referencia a otra clase o método de utilidad, que tiene relación con éste y que aconsejamos que se consulte también.
7. **@since.** Versión desde la que incluimos el método o clase.
8. **@trhows.** Sinónimo de @exception.
9. **@version.** Indica la versión del método o clase.

JDeveloper nos permite situarnos en una clase o un método y por medio de la opción de menú "**Source | Add Javadoc Comment**" nos añade los delimitadores y si es un método nos incluirá las etiquetas @param y @return, si son aplicables. Decir que para que Javadoc genere documentación de un método, éste debe ser público. Finalmente podremos generar la

documentación de nuestro código comentado por medio de la opción de menú "**Run | Javadoc ...**". Después podremos acceder a la misma desde nuestro navegador preferido o desde el entorno. JDeveloper crea un directorio dentro del directorio de nuestro proyecto que como debes recordar estaba dentro del directorio de nuestra aplicación. El directorio creado se llama javadoc y en él se incluyen todos los ficheros generados. Veamos una demostración de cómo podemos documentar nuestra clase Factorial utilizando comentarios al estilo Javadoc.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

Para generar documentación automática en JDeveloper:

- a) Eso no podemos hacerlo ni con JDeveloper ni con Java.
- b) Podemos hacerlo usando una sintaxis muy extraña utilizada sólo por Oracle.
- c) Podemos utilizar comentarios al estilo Javadoc y a partir de ellos se generará la documentación.
- d) Oracle JDeveloper utiliza una herramienta inteligente que es capaz de generar documentación en castellano y en inglés de nuestras clases sin necesidad de que nosotros pongamos comentarios.

Comprobar

Marcar como leído

Unidad Didáctica II

Gestión de tareas.

Una práctica bastante común a la hora de programar (incluso en otras facetas de la vida), es dejar cosas a medio hacer e ir al grueso del problema, para luego ir maquillando nuestro código con los pequeños detalles que hemos dejado pendientes. El problema que tiene esto es que luego se nos olvidan las cosas y de todo lo que pensamos en su momento sólo nos acordamos de la mitad. Una posible solución a esto sería anotarlo en el código por medio de un comentario, pero cuando nuestras aplicaciones empiezan a ser extensas y compuestas de muchos ficheros fuente, el encontrar esos comentarios puede resultar una tarea bastante compleja y al final seguro que nos dejamos alguna pendiente.

JDeveloper nos ofrece una funcionalidad muy adecuada para este tipo de quehaceres. Permite que insertemos comentarios en línea comenzando con la etiqueta TODO (del inglés "to do", que significa para hacer, o por hacer) seguida de un comentario y esto automáticamente creará una tarea pendiente de ejecución. Posteriormente podremos acceder al panel de tareas, por medio de la opción "**Task Window**" del menú "**View**" y en él podremos ver las tareas pendientes de llevar a cabo y si pulsamos sobre ellas el cursor se nos situará sobre el principio de la línea en la que definimos la tarea. La ventaja que nos ofrece esta funcionalidad es poder ver de un vistazo todas las tareas que tenemos pendientes e incluso poder ordenarlas por antigüedad, prioridad, etc. En la siguiente presentación podemos ver cómo utilizar esta funcionalidad que nos ofrece nuestro entorno.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

La ventana de tareas nos muestra:

- a) Las tareas que JDeveloper hace por nosotros.
- b) En JDeveloper no existe esa ventana.
- c) Nos muestra las tareas pendientes que hemos ido anotando en nuestro código. CORRECTA.
- d) Nos muestra las tareas que JDeveloper cree que debemos llevar a cabo para implementar las clases.

Comprobar

Marcar como leído

Unidad Didáctica II

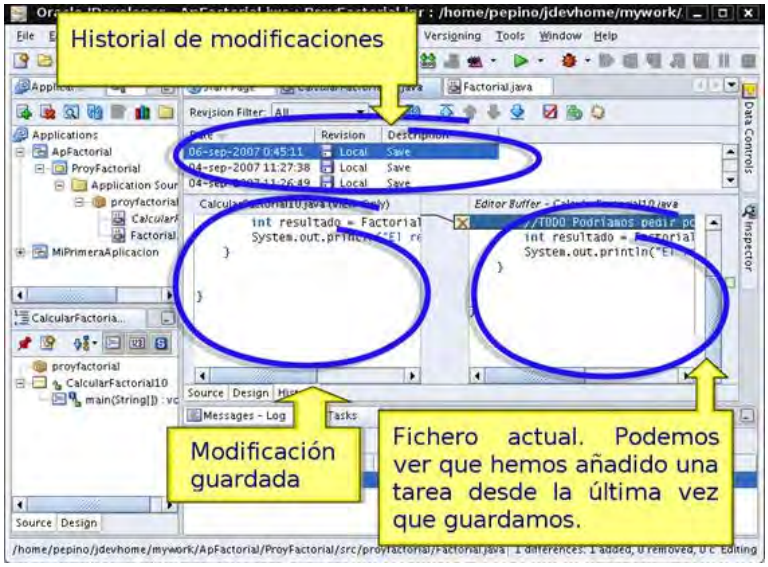
Historial de modificaciones.

¿Cuántas veces no se te ha ocurrido una idea feliz, que crees que va a mejorar tu código, que ya funcionaba y cuando te decides a llevarla a cabo modificando el código, te das cuenta que no es tan buena como pensabas y además no funciona?

Encima, en medio de todos esos cambios y con la seguridad de que la idea iba a ser la panacea de la programación, le diste a grabar los cambios. Cuando descubres que la feliz idea no era tal y que además has machado tu código que funcionaba, te entran ganas de aporrear la pantalla (sin que ella tenga ninguna culpa de tus supuestas ideas felices).

Para solucionar estas situaciones indeseables que a todos nos han pasado y a veces hemos perdido mucho tiempo en el intento, JDeveloper nos ofrece una funcionalidad que puede hacer que nuestro estado de nervios no sufra cambios indeseados. Para ello, JDeveloper va guardando todos los cambios significativos que hacemos en nuestro código. Para ello nos ofrece una lista de cambios y nos muestra las modificaciones entre nuestro fichero fuente actual y la modificación seleccionada, ofreciéndonos la posibilidad de dar marcha atrás en cualquier momento.

Para acceder a esta funcionalidad, en el editor de código disponemos de tres pestañas; "**Source**", "**Design**" y "**History**". Esta última pestaña es la que nos permite acceder a dicha funcionalidad. Si la seleccionamos podremos acceder a una pantalla como la siguiente:



Como puedes ver, tenemos una lista de todas las modificaciones realizadas en nuestro código fuente. Si seleccionamos una de ellas podremos apreciar las diferencias entre el fichero fuente actual y la modificación guardada por JDeveloper. Por ejemplo, en la pantalla anterior vemos cuál es la última modificación realizada en nuestro código fuente, que no era otra que añadir una tarea pendiente de realizar (//TODO)

1 **Autoevaluación**

El historial de modificaciones...

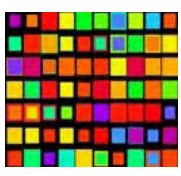
- a) Permite que veamos las diferentes modificaciones que hemos ido realizando en nuestros ficheros, pero no permite dar marcha atrás a una de ellas.
- b) En JDeveloper no existe el historial de modificaciones.
- c) Permite que veamos las diferentes modificaciones que hemos ido realizando en nuestros ficheros y permite dar marcha atrás a cualquiera de ellas.
- d) Todas las respuestas anteriores son falsas.

Comprobar

Marcar como leído

Unidad Didáctica II

Dotando de una interfaz gráfica a nuestra aplicación.



En este apartado se indican cuáles son los pasos necesarios para dotar de una interfaz gráfica simple a nuestra aplicación, pero en ningún momento se pretende explicar cuáles son los componentes que forman una interfaz gráfica, cuáles son sus propiedades, métodos o eventos.

> ***Recuerda que se te recomendaba encarecidamente que repasases los conocimientos adquiridos en el módulo de Programación en Lenguajes Estructurados o que te leyeses los tutoriales recomendados.***

Aquí sólo se expondrá la manera en que se llevan a cabo estas tareas en JDeveloper sin pretender en ningún momento dar una explicación exhaustiva, ya que por un lado deberías conocer y por otro irás practicando a lo largo de todo este módulo.

Dado que nuestro caso de estudio calcula el factorial de 10, pero que mediante la refactorización lo hemos podido hacer un poco más reutilizable, realizando una clase Factorial que es capaz de calcular el factorial de cualquier número entero, la idea será:

- dotar de una interfaz gráfica a nuestra aplicación
- que nos pregunte el entero del que queremos calcular el factorial y
- que al pulsar un botón nos informe del resultado de dicho cálculo.

Para ello diseñaremos un JFrame que incluirá dos JLabel; uno para indicar que introduzcamos un valor y otro en el que mostraremos el resultado del factorial de dicho valor. También incluirá un JTextField en el cuál introduciremos el valor del que queremos calcular el factorial. Por último tendrá un JButton que al pulsarlo calculará el factorial del valor introducido en el JTextField haciendo uso de nuestra clase Factorial y lo visualizaremos en el segundo JLabel. Para ello deberemos implementar el evento ActionPerformed de nuestro JButton, con el siguiente código:

```
private void jButton1_actionPerformed(ActionEvent e) {
      int valor = 0, factorial;

        try {
            valor = Integer.parseInt(JTextField.getText());
            } catch (NumberFormatException nfe)
            {
                JTextField.setText("");
                R.setText("Debes introducir un entero");
            }
            factorial = Factorial.calcular(valor);
            R.setText("El factorial de " + JTextField.getText() + " es... " + factorial + ".");
    }
}
```

Una vez que tenemos nuestro JFrame diseñado, simplemente deberemos añadir una nueva aplicación a nuestro proyecto que se encargue de lanzar este JFrame recién diseñado. Veamos una demostración de cómo llevar a cabo esta tarea que debería ser pan comido para ti si has seguido las recomendaciones expuestas anteriormente.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

Para dotar de una interfaz gráfica a nuestra aplicación debemos:

a) Simplemente crear un nuevo Frame para la misma.
b) Crear un nuevo Frame y lanzar ésta desde una aplicación cliente.
c) Con JDeveloper sólo podemos crear programas de consola.
d) Sólo podemos crear interfaz gráfica por medio de applets.

Comprobar

Marcar como leído

Unidad Didáctica II

Despliegue de aplicaciones.

Cuando nuestra aplicación consta de un solo fichero podemos compilarlo y a la hora de ejecutarlo hacerlo mediante el fichero **.class** generado. Pero cuando nuestra aplicación consta de varios ficheros, ya sean clases u otros recursos, eso ya no es tan trivial. Para que nuestra aplicación pueda ser ejecutada con éxito en cualquier entorno tendremos que empaquetarla en un fichero que tenga una

estructura adecuada y contenga la información necesaria para poder ejecutarla. Esto podemos hacerlo mediante la funcionalidad conocida como despliegue de la aplicación.

El despliegue es una de las etapas finales del desarrollo de software. En ella, se copian todos los archivos necesarios para ejecutar una aplicación del entorno de desarrollo al de producción. El resultado de la etapa de despliegue es un archivo empaquetado que contiene todas nuestras clases, librerías..., y además contiene un descriptor de despliegue en el que se especifican todos los detalles necesarios para que nuestra aplicación se pueda ejecutar. Todo esto se empaqueta en un fichero que suele tener la extensión "**.jar**" o "**.war**" y el cuál puede a su vez ir comprimido.

Para llevar a cabo el despliegue de nuestra aplicación con JDeveloper, simplemente deberemos:

- Crear un nuevo perfil de despliegue para nuestro proyecto mediante la opción "**Deploy | New Deployment Profile**" del menú "**Run**".
- Nos preguntará el tipo de despliegue que queremos hacer, que por ahora usaremos el despliegue a un fichero "**.jar**", ya que los otros tipos de despliegues son para aplicaciones Web.
- Después nos preguntará por el nombre del fichero y debemos indicar cuál es la clase principal de nuestro proyecto que será la que se ejecutará.
- También nos da la opción de que el fichero se comprima y de que podamos meter recursos adicionales en el mismo.

Una vez hecho esto, vemos cómo en nuestro proyecto aparece una carpeta denominada "**Resources**" y dentro de la misma aparece nuestro fichero de despliegue. Ahora sólo nos queda llevar a cabo el despliegue en sí pulsando con el botón derecho del ratón sobre el fichero y eligiendo la opción "**Deploy to JAR File**" del menú contextual que nos aparecerá. El fichero generado ya estará listo para su uso en cualquier plataforma que tenga instalado el jdk y bien configuradas las variables PATH y CLASSPATH. Para ejecutarlo simplemente deberemos ejecutar la orden "java-jar nombre_fihcero.jar" y nuestra aplicación empezará a ejecutarse. Veamos una demostración de cómo realizar todo el proceso.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

El despliegue de aplicaciones consiste en:

a) Generar un fichero en el que irán todas las clases compiladas necesarias para ejecutar nuestra aplicación.
b) Compilar todos los ficheros de nuestra aplicación.
c) Compilar todos los ficheros de nuestra aplicación y comprimirlos.
d) Consiste en pasar nuestra aplicación de un proyecto a otro.

Comprobar

Marcar como leído

Unidad Didáctica II

Reutilizando código por medio de los JavaBeans.

CASO. Víctor está un poco harto de tener que repetir código que ya tiene implementado en otras aplicaciones o de hacer las mismas comprobaciones cuando por ejemplo utiliza un JTextField en el que sólo quiere que se introduzca un número. Cree que esto es una pérdida de tiempo que tan preciado es, debido a las apretadas agendas de entrega de proyectos que tienen últimamente en SI Andalucía. Está seguro que debe haber una forma de no perder este tiempo y para ello se le ocurre preguntar a María si ella conoce alguna forma de poder solucionar este problema.

***María** responde a **Víctor** encantada y le comenta que para eso están los JavaBeans y que, la verdad, hacen la vida mucho más fácil al programador, ya que precisamente permiten solucionar el problema que a **Víctor** le trae por la calle de la amargura. **Víctor** le dice a **María** que por favor le explique en qué consisten y que le de una pequeñas nociones para que él pueda empezar a utilizar los JavaBeans en sus proyectos y que éstos puedan ser compartidos por todo el equipo de **SI Andalucía**. **María** lo ve una muy buena idea y está dispuesta a enseñar a **Víctor** a utilizar esta tecnología. Primero le explica en qué consiste y luego pasará a diseñar un ejemplo.*

Seguro que a ti te ha pasado muchas veces lo que a Víctor; que has tenido que repetir código que ya tenías implementado en otras aplicaciones, hacer las mismas comprobaciones, etc. En este apartado vamos a mostrar cómo se puede solucionar ese problema haciendo uso de nuestro entorno y los JavaBean.

Para ello,

- Primero veremos qué es realmente un JavaBean.
- Seguidamente pasaremos a diseñar un JavaBean simple que nos ayudará a entender la filosofía de trabajo con ellos.

- Lo instalaremos en nuestro entorno para que pueda ser utilizado en cualquier otro proyecto y
- Finalmente haremos uso del mismo en nuestro caso de estudio del cálculo del factorial.

Marcar como leído

Unidad Didáctica II

¿Qué es un JavaBean?

Un JavaBeans no es más que un componente software reutilizable e independiente de la plataforma.

Para definir un JavaBean simplemente diseñamos una clase, tratando de **encapsular su implementación** y **mostrando al exterior solamente los métodos y propiedades que son públicos**. Sólo se muestra aquello que forma parte del servicio que el bean ofrece al exterior. A la hora de diseñar nuestro JavaBean debemos seguir una serie de reglas para hacer que éste pueda ser utilizado en cualquier entorno de trabajo. Un JavaBean puede ser utilizado en varios contextos, como componentes utilizados en una aplicación cliente con entorno gráfico o en aplicaciones Web representando normalmente entidades o reglas de negocio. Nosotros nos centraremos en los primeros y en unidades siguientes podrás comprobar el uso de los segundos.

Los JavaBeans gráficos son componentes que pueden ser manipulados visualmente por un entorno de desarrollo igual que si de un componente Swing se tratase. Para ello a la hora de diseñarlos debemos seguir unas pautas que permitan a la herramienta gráfica conocer las propiedades de nuestro componente y los eventos que es capaz de lanzar. Una vez diseñado podremos instalarlo en nuestro entorno y utilizarlo como cualquier otro componente a través de la paleta de componentes arrastrándolo a nuestra aplicación.

Veamos cuáles son las reglas que debemos seguir a la hora de diseñar un JavaBean:

- **La clase debe implementar la interfaz** Serializable. Esta regla no es obligatoria, pero es aconsejable si queremos que nuestro JavaBean pueda guardar información entre sesiones.
- **Definir un constructor por defecto**. Esta regla tampoco es obligatoria, pero también es conveniente si queremos que nuestro entorno pueda crear instancias de nuestro JavaBean, sobre todo cuando tenemos propiedades que inicializar.
- Para **informar al entorno sobre las propiedades del JavaBean** hay dos métodos: primero escribir un método setXXX y otro getXXX para cada propiedad; en segundo lugar usar archivos **BeanInfo**. Nosotros nos centraremos en el primer método: para cada propiedad "XXX" crearemos un método getXXX que devuelve el valor de la propiedad y un método setXXX que cambia el valor de la propiedad. Los IDE deducen la propiedad a partir de esta convención de nombres. Las propiedades de un JavaBean pueden tener la siguiente naturaleza:
  - **Propiedad simple**: Una propiedad simple es una propiedad representada por un tipo de dato básico como puede ser un entero, un boolean, un carácter, una cadena, ... Para estas propiedades dispondremos de un método get que nos devolverá ese valor simple y un método set que nos permitirá modificar el mismo. Estas propiedades serán las que trataremos en este punto.
  - **Propiedad indexada**: Una propiedad indexada representa un array de valores, por lo que además de los métodos get y set que permiten devolver el array completo y modificar el array completo, también debemos implementar un método get que recibirá como parámetro un índice y nos devolverá el valor del elemento que ocupa esa posición dentro del array y un método set que tendrá como parámetros un índice y un valor y permitirá modificar el elemento que ocupa dicho índice por el valor pasado como parámetro. Veamos un pequeño ejemplo de una propiedad llamada valores que es un array de enteros. En estas propiedades no profundizaremos en esta unidad.

```
private void jButton1_actionPerformed(ActionEvent e) {

      int valor = 0, factorial;


        try {

            valor = Integer.parseInt(JTextField.getText());

            } catch (NumberFormatException nfe)

            {

                JTextField.setText("");

                R.setText("Debes introducir un entero");

            }
```

```
            factorial = Factorial.calcular(valor);
            R.setText("El factorial de " + JTextField.getText() + " es... " + factorial + ".");
    }
}
```

- **Propiedad ligada**: Los objetos de una clase que tienen una propiedad ligada notifican a otros objetos (**listeners**) interesados, cuando el valor de dicha propiedad cambia, permitiendo a estos objetos realizar alguna acción. Cuando la propiedad cambia, se crea un objeto (**event**) que contiene información acerca de la propiedad (su nombre, el valor previo y el nuevo valor), y lo pasa a los otros objetos (**listeners**) interesados en el cambio. Estos JavaBean los diseñaremos utilizando la plantilla "**Customizer**" en vez de la plantilla "**Bean**" de JDeveloper, aunque en estas propiedades tampoco nos detendremos en esta unidad.
- **Propiedad restringida**: Una propiedad restringida es similar a una propiedad ligada salvo que los objetos (**listeners**) a los que se les notifica el cambio del valor de la propiedad tienen la opción de vetar (**veto**) cualquier cambio en el valor de dicha propiedad. En estas propiedades tampoco nos detendremos.

1 **Autoevaluación**

Para diseñar un JavaBean...

a) No tenemos porque hacer nada, ya que el entorno lo hace todo por nosotros.
b) Debemos crear como mínimo dos constructores, uno por defecto y otro que acepte como parámetro una cadena que representará el nombre de nuestra aplicación.
c) Como mínimo debe tener propiedades simples y propiedades indexadas.
d) Debemos generar métodos de acceso para las propiedades de nuestro JavaBean.

Comprobar

Marcar como leído

Unidad Didáctica II

Diseñando un JavaBean.

Ya hemos visto las reglas que debemos seguir para diseñar un JavaBean, pero nada mejor que ponerlas en práctica para comprender bien cómo debemos llevar a cabo nuestro diseño, ¿no crees?.

Para ello vamos a diseñar un JavaBean que heredará de un JTextField, pero que nos permitirá acotar el número de caracteres que podemos introducir y que además sólo nos permitirá introducir dígitos en el mismo. De esta forma resultará ideal para que el usuario introduzca campos numéricos.

Para ello, nuestro JavaBean tendrá una propiedad simple (además de las que hereda del jTextField) que denominaremos **longitud** y que restringirá la longitud en caracteres que podremos introducir en dicho campo. Además queremos que nuestro componente sólo acepte dígitos, ya que queremos utilizarlo en nuestro caso de estudio, en el cuál aceptamos números para calcular su factorial.

- Lo primero que debemos hacer es crear un nuevo proyecto dentro de nuestra aplicación (también podríamos haber creado una nueva aplicación) y
- en el mismo creamos un JavaBean por medio de la opción "**New**" en la que elegimos la categoría "**JavaBean**" y dentro de ella la opción "**Bean**".
- Elegimos el nombre que le queremos poner, que podría ser JNumFieldAcotado y elegimos la clase de la que heredará, que en nuestro caso será .

- Una vez creado, vemos que lo único que ha hecho JDeveloper es crear una clase que hereda de JTextField y que tiene un constructor por defecto vacío. Te preguntarás por qué JDeveloper no ha hecho que nuestra clase implemente la interfaz Serializable, y la verdad es que no lo ha hecho ya que al heredar de la clase JtextField, ésta ya implementa esa interfaz.
- Como queremos disponer de una propiedad que definirá la longitud en caracteres que podremos insertar en nuestro componente, definiremos una propiedad dentro de nuestra clase de tipo int cuyo nombre sea longitud.
- Ahora generamos los métodos de acceso para dicha propiedad (recuerda que podemos hacerlo mediante la opción "**GenerateAccessor**" del menú "**Source**").
- Ya sólo nos queda decirle a nuestro componente cómo debe comportarse, para lo cuál definiremos cómo debe actuar ante el evento **KeyTyped,** que es el evento que se lanza cuando pulsamos una tecla sobre nuestro componente. El código del método que se lanza cuando se produce este evento podría ser el siguiente:

```
private void this_keyTyped(KeyEvent e) {
        char c = e.getKeyChar();
        if ((c >= '0' &#38;&#38; c <= '9') || (c == KeyEvent.VK_DELETE) ||
            (c == KeyEvent.VK_BACK_SPACE) || (c == KeyEvent.VK_ENTER)) {
            if (longitud > 0 &#38;&#38; getText().length() >= longitud) {
                Toolkit.getDefaultToolkit().beep();
                System.out.println("Sobrepasada longitud");
                e.consume();
            }
        }
        else {
            Toolkit.getDefaultToolkit().beep();
            e.consume();
        }
    }
```

¡¡Ya tenemos nuestro JavaBean implementado!!

Veamos cómo podemos usarlo en nuestras aplicaciones, para lo cual primero debemos instalarlo en nuestro entorno de trabajo. En el siguiente enlace puedes ver una presentación en la que se muestra cómo hemos fabricado nuestro componente JNumFieldAcotado.

Pulsa aquí para ver la simulación

1 **Autoevaluación**

Para implementar un JavaBean...

- a) JDeveloper nos ofrece una plantilla llamada "Bean" que creará nuestra clase y un constructor por defecto.
- b) Debemos crear métodos de acceso para nuestras propiedades.
- c) Debemos implementar la interfaz "Serializable".
- d) Todas las respuestas anteriores son correctas.

Comprobar

Marcar como leído

Unidad Didáctica II

Instalando nuestro JavaBean en JDeveloper.

¿Qué nos queda por hacer? Ya hemos implementado el JavaBean, pero habrá que instalarlo para que JDeveloper "se entere", y nos permita usarlo en las próximas aplicaciones que desarrollemos. ¿Cómo lo hacemos?

Para instalar nuestro componente en el entorno:

- Lo primero que debemos hacer es desplegar el proyecto en el que se encuentra a un fichero JAR como ya hemos visto en el punto dedicado al despliegue de aplicaciones en esta misma unidad. Recuerda que lo primero era crear un perfil de despliegue en el que indicábamos que queríamos desplegar a un fichero JAR y seguidamente desplegábamos nuestro proyecto.
- Una vez hecho esto debemos integrar este fichero JAR recién desplegado en las librerías de nuestro entorno. Para ello accedemos a la opción "**Manage Libraries**" del menú "**Tools**". Bajo la categoría "**Users**" añadimos una nueva entrada,

eligiendo el fichero JAR que acabamos de crear y nombrándolo como JNumFieldAcotado.

- Sólo nos queda instalar el componente en la Paleta de Componentes, para lo cual elegimos la opción "**Configure Palette**" del menú "**Tools**". Añadimos una nueva página a la paleta que podemos nombrar como "**Componentes de Usuario**" y en la que iremos instalando todos los JavaBeans que vayamos creando. En esta nueva página añadimos el nuevo componente. Aquí te mostramos una presentación de todo el proceso que hemos seguido para instalar nuestro componente en la paleta de componentes de nuestro entorno.

Pulsa aquí para ver la simulación

¡¡Ya tenemos nuestro componente instalado!!

En consecuencia, ya podremos utilizarlo en cualquier aplicación que estemos diseñando. Veamos en el siguiente apartado cómo podemos utilizarlo en nuestra aplicación que calculaba el factorial de un número.

1 **Autoevaluación**

A la hora de instalar un JavaBean en nuestro entorno...

a) En JDeveloper no podemos instalar nuevos componentes.
b) Debemos desplegar nuestro componente a un fichero JAR, instalarlo en las librerías del entorno y finalmente añadirlo a la paleta de componentes.
c) Simplemente añadiremos nuestra clase a la paleta de componentes.
d) Todas las respuestas anteriores son falsas.

Comprobar

Marcar como leído

Unidad Didáctica II

Utilizando el JavaBean en nuestra aplicación.

¡¡Llegó la hora de sacarle provecho a tanto trabajo!! Prepárate a ver tu primer JavaBean en acción, ya que vamos a usarlo en nuestra aplicación. ¿Cómo podemos hacer uso de él?

> ***La utilización de nuestro componente recién creado es idéntica a la utilización de cualquier otro componente de la paleta de componentes. Simplemente debemos arrastrarlo a nuestra ventana, darle el valor deseado a las propiedades del mismo y compilar nuestra aplicación.***

¿Esperabas algo más complicado? Pues siento decirte que los JavaBeans no se idearon para complicar las cosas, sino para simplificarlas, como acabamos de decirte, y como vas a ver a continuación.

Para entender mejor el proceso, sustituiremos en el caso de estudio del cálculo del factorial de un número, el componente JTextField por nuestro nuevo componente JNumFieldAcotado. Para ello:

- Primero eliminamos el JTextField de nuestra JFrame y en su lugar arrastramos desde la paleta de componentes el componente que hemos creado, para lo cuál deberemos seleccionar la página "Componentes de Usuario" que creamos para su instalación.
- Después buscaremos la propiedad **longitud** que hemos creado y le daremos el valor 2 para impedir que podamos introducir números con una longitud mayor a 2.
- Como en nuestra aplicación utilizamos los nombres por defecto que iba asignando JDeveloper a los componentes, para que todo funcione correctamente deberemos cambiar las referencias a jTextField1 por jNumFieldAcotado1, que es el nombre que habrá asignado JDeveloper al componente añadido a nuestra ventana. Para ello:
  - primero cambiaremos el nombre asignado por JDeveloper a nuestro componente a jTextField1
  - posteriormente aplicaremos refactorización y renombraremos esta variable para que así JDeveloper haga el trabajo de cambiar todas las referencias por nosotros.

Puedes ver los pasos seguidos en la siguiente presentación.

Pulsa aquí para ver la simulación

Con el uso de los JavaBean terminamos esta Unidad. Espero que te haya sido de utilidad y que hayas perdido el miedo a usar Oracle JDeveloper. Como dije anteriormente, hemos pretendido utilizar un ejemplo que pudiésemos ir modificando e ir aplicando sobre él los conceptos expuestos durante toda la Unidad. Sabemos que dicho ejemplo podría mejorarse, y sabemos mejorarlo... ;-)

Por ejemplo, podría mejorarse en cuanto a programación dirigida a objetos. También incluso tiene fallos, como que dados ciertos valores para calcular su factorial, éstos sobrepasan la precisión de un entero, pero se perseguía el carácter didáctico del mismo y no complicarlo con otros aspectos que no fuesen la utilización de nuestro entorno, pero te animamos a que practiques incluyendo otras propiedades útiles al componente, como la que acabamos de mencionar, u otras que puedas necesitar en tus aplicaciones.

1 **Autoevalución**

Para utilizar un JavaBean que anteriormente hemos creado e instalado en el entorno debemos...

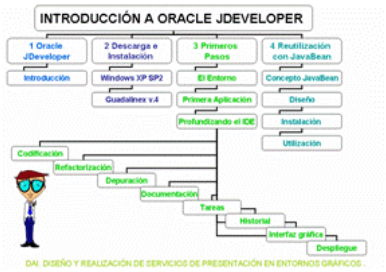
- a) Añadirlo también como proyecto a nuestra aplicación.
- b) Simplemente debemos arrastrarlo desde la paleta de componentes como si de un componente estándar se tratase.
- c) Debemos añadirlo también al paquete de nuestra aplicación.
- d) No es posible utilizar JavaBeans creados por nosotros con JDeveloper.

Comprobar

Marcar como leído

Unidad Didáctica II

Mapa Conceptual de la Unidad.



Marcar como leído