

# La Guía de Delphi de Marco Cantù

A continuación incluimos un resumen de cada uno de los apartados y capítulos del libro “La Guía de Delphi de Marco Cantù” publicado por Danysoft en castellano [[www.danysoft.com](http://www.danysoft.com)].

## Apartado I: Unicode

---

La primera parte de este libro se centra en Unicode, el carácter de codificación internacional estándar que soporta por primera vez Delphi 2009. Los tres capítulos de este apartado introducen este tema, describen la implementación actual, y le guiarán en las tareas de importación y compatibilidad, respectivamente.

### **Capítulo 1: ¿Qué es Unicode?**

Unicode es el nombre del conjunto de caracteres internacionales que engloba los símbolos de todos los alfabetos escritos del mundo, de hoy y del pasado, y algo más. El estándar Unicode (formalmente denominado "ISO/IEC 10646") está definido y documentado por el Consorcio de Unicode, y contiene más de 100.000 caracteres. Su sitio web principal es: <http://www.unicode.org>

Como la adopción de Unicode es un elemento central de Delphi 2009 y son muchas las cuestiones a abordar, este capítulo se centra sólo en la teoría entre Unicode y las otras codificaciones de caracteres, mientras que el siguiente se centrará en los elementos clave de su aplicación en Delphi.

### **Capítulo 2: El Tipo Unicode String**

Como usted seguramente sabe, una de las nuevas, y mejores características de Delphi 2009, es la introducción de un nuevo tipo de cadena UnicodeString, que es también el formato ligado al tipo básico string. Cada vez que escribe "string" en su código, se está refiriendo de hecho a UnicodeString, mientras que en versiones anteriores de Delphi (con excepción de Delphi 1) se referían a AnsiString.

Junto con Char que se convierte en un alias de WideChar, este es un cambio muy significativo, que afecta a todo su código base. Es por eso que no será suficiente un solo capítulo para explicar todo lo que necesita saber. Voy a tratar la totalidad del nuevo tipo string, pero son muchos e inevitables los inconvenientes que surgen al importar código Delphi existente a la nueva versión del compilador Embarcadero, habilitada para Unicode, que trataré en el próximo capítulo.

### **Capítulo 3: Exportando a Unicode**

Disponer de soporte nativo Unicode en Delphi es un gran paso adelante, y el hecho de poder continuar utilizando el tipo de string significa que usted puede importar código existente simplemente a costa de recompilar. Este cambio es un gran cambio. Desde las llamadas a la API de Windows empleando punteros PChar sólo con el soporte matemático del puntero, hay muchas áreas de Delphi para las que puede esperarse que nuestra migración no será tan fácil y sencilla. En este capítulo se profundiza en estos y otros problemas posibles.

## Apartado II: Delphi 2009 y Su Compilador

---

Ahora que he cubierto completamente la nueva característica más importante de Delphi 2009, el soporte Unicode, ya es hora de centrarnos en todo el entorno de desarrollo, el compilador y las librerías en tiempo de ejecución. Esta sección trata lo más esencial y las características de bajo nivel, mientras que las novedades en la interfaz de usuario y la base de datos se cubrirán en las siguientes partes del libro.

### **Capítulo 4: Nuevas Características del IDE**

La 6ª encarnación del IDE Galileo tiene sólo un limitado conjunto de nuevas capacidades, si usted no tiene en cuenta el hecho de que todo se ha convertido a Unicode, lo que está probablemente lejos de ser trivial. Las mejoras más relevantes se relacionan con las nuevas ampliaciones del Project Manager y la habilidad de compartir opciones de proyectos entre los diferentes proyectos, utilizando las nuevas opciones de ficheros. Finalmente también se mejora, significativamente en Delphi 2009, la gestión de los recursos de ventanas.

### **Capítulo 5: Genéricos**

El fuerte control de tipo que provee Delphi es muy útil para mejorar la exactitud del código, un tema que tiendo a destacar en mis libros de iniciación. Un fuerte control de tipo, sin embargo, también puede ser una molestia, si usted no dispone de un procedimiento o una clase que pueda actuar con diferentes tipos de datos. Esta cuestión está dirigida por una nueva característica del lenguaje Object Pascal, agregado recientemente a lenguajes similares como C# y Java, llamados *genéricos*. Esto es lo que escribí en 1994 en un libro sobre C++:

*Ahora puede declarar una clase, sin especificar el tipo de dato de uno o más de sus miembros: esta operación se puede retrasar hasta que un objeto de esa clase es realmente declarado. Del mismo modo, puede definir una función sin especificar el tipo de uno o más de sus parámetros hasta que la función sea llamada.*

Ahora, 14 años después, esta función llega a Object Pascal. Como puede imaginar estoy emocionado de disponer de genéricos (que se llaman templates en C++) en Delphi, aunque debo decir que he sido testigo de diferentes excesos de esta función que yo no entendía plenamente en ese momento. Dudo que los genéricos sean excesivos en Delphi sino más bien todo lo contrario: existe el riesgo de que una actualización tan importante del lenguaje pase casi inadvertida. En este capítulo se tratará de profundizar en el tema, demostrándole el valor de los genéricos en Delphi y cómo pueden ser aplicados incluso en la programación visual estándar.

### **Capítulo 6: Métodos Anónimos**

Desde hace mucho tiempo, el lenguaje Delphi tiene tipos de procedimiento (es decir, tipos que declaran punteros a procedimientos y funciones) y métodos puntero (es decir, tipos que declaran punteros a métodos). Aunque puede ser que rara vez usted los haya utilizado directamente, éstas son las características clave con las que trabaja cada desarrollador de Delphi. De hecho, los tipos punteros a métodos son la base para los manipuladores de eventos de la VCL: cada vez que declara un manipulador para un evento, incluso para un simple Button1Click usted de hecho está declarando un método que se conectará a un evento (en este caso, el evento OnClick) utilizando un método puntero.

Los métodos Anónimos extienden esta característica permitiéndole pasar un código determinado a un método como un parámetro, en lugar de como el nombre de un método definido en otro lugar. Esta no es, sin embargo, la única diferencia. Lo que hace muy distinto los métodos anónimos a otras técnicas, es la forma en que gestiona la vida de las variables locales.

La definición anterior iguala la característica llamada “closures” de otros lenguajes, como por ejemplo JavaScript. Si los métodos anónimos de Delphi, son de hecho “closures”, ¿Cómo es que Embarcadero se refiere a ellos usando un término diferente? La razón es que C++ Builder ha estado utilizando el término “closures” para lo que llamamos en Delphi manipuladores de eventos, así que nos encontraríamos con una característica diferente con el mismo nombre, y esto podría crear confusiones. Además, el lenguaje C# utiliza la expresión métodos anónimos para un mecanismo similar al de Delphi, por lo que tiene sentido utilizar el mismo apodo.

Si los métodos anónimos son un tipo de característica nueva para Delphi, han girado alrededor durante muchos años en diferentes formas y con diferentes nombres en otros lenguajes de programación, lenguajes notablemente más dinámicos. He tenido una amplia experiencia con los “closures” en JavaScript, en particular con la librería jQuery y la utilización de AJAX. La correspondiente función en C# se llama, un delegado anónimo.

Pero no quiero dedicar más tiempo a comparar los “closures” y sus técnicas relacionadas con los diversos lenguajes de programación, sino más bien describir en detalle cómo funcionan estos en Delphi 2009.

## ***Capítulo 7: Más Cambios del Lenguaje y de la RTL***

En los primeros capítulos de Unicode, y en los dos últimos sobre características del lenguaje, he tratado la mayoría de las novedades del compilador y también he presentado un gran número de nuevas clases de la RTL, desde TCharacter y TEncoding hasta los nuevos contenedores genéricos. Puede encontrar una lista completa en la sección “Resumen de Nuevas Unidades y Nuevas clases RTL” al final de este capítulo. Aquí voy a mostrar otras nuevas características del compilador y de las áreas de la RTL que no encajan en ninguno de los capítulos anteriores.

## Apartado III: La VCL y Las Bases de Datos

---

La parte principal de la “Experiencia Delphi “ se refiere a su librería clave, la Librería de Componentes Visuales (VCL), y en particular su subconjunto centrado en el desarrollo con bases de datos, incluyendo cliente/servidor y las arquitecturas de 3-capas. Muchos de los controles visuales y algunos de los componentes de acceso a bases de datos han recibido una importante actualización en Delphi 2009, y son objeto de esta tercera parte del libro, que abarca también COM, los nuevos controles Ribbon, y la nueva arquitectura en múltiples niveles DataSnap 2009.

### **Capítulo 8: Mejoras En la VCL**

Aunque quizá no tan importante como el soporte Unicode u otros cambios del compilador, la VCL en Delphi 2009 recibe una serie de pequeñas pero importantes mejoras (algunas de las cuales habían sido solicitadas hacía tiempo por los usuarios de Delphi). En este capítulo, voy a centrarme en las mejoras realizadas y nuevos controles añadidos, mientras que el siguiente lo dedicaré al nuevo componente Ribbon.

La VCL es una de las piedras angulares de Delphi y su arquitectura ha contribuido significativamente al éxito de esta herramienta. Aún hoy, mirando la interfaz de usuario de otros entornos diseñados después de muchos años por grandes empresas de TI, la VCL destaca considerablemente.

Con los cuatro nuevos componentes (BalloonHint, ButtonedEdit, CategoryPanelGroup, y LinkLabel), y además el soporte Ribbon y sus innumerables pequeñas mejoras, la VCL ha visto una significativa actualización en Delphi 2009. Algunas de estas actualizaciones son específicas para Windows XP o Windows Vista y, sobretodo, aumentan el soporte para Vista en la VCL desde Delphi 2007.

### **Capítulo 9: Soporte COM en Delphi 2009**

Durante muchos años, la tecnología de Modelo de Objetos de Componentes (COM) ha estado en la base del sistema operativo Windows, como una forma de permitir que las aplicaciones hablasen con el SO y a los programas entre si. COM es para Windows el único modelo de objetos que funciona de forma nativa a través de distintos lenguajes. El hecho de que la programación COM estuviera lejos de ser fácil y que proporcionara unos fundamentos no tan sólidos, ha sido una de las razones que ha mencionado Microsoft para abandonarla y pasar a un modelo de objetos gestionados como el de la plataforma .NET de Microsoft.

A pesar de que Microsoft dijese que COM estaba obsoleto cuando anunció .NET, la tecnología COM todavía sigue siendo muy importante en el uso de muchas aplicaciones Windows y está lejos de desaparecer. COM proporciona también una forma sencilla para que las aplicaciones Win32 trabajen junto con .NET. En cualquier caso, no quiero ahondar aquí en la historia de COM o en una comparación con .NET. Sólo quiero centrarme en el hecho de que, aunque no creciera y no se utilice más para la comunicación entre diferentes ordenadores, COM se encuentra todavía en el corazón de muchos programas Windows. Además, tradicionalmente Delphi hace muy fácil escribir servidores COM y utilizar los ya existentes, por lo que muchos desarrolladores Delphi se basan en esta tecnología.

Dada esta introducción, es relevante señalar que existen importantes cambios en el soporte COM proveído por Delphi 2009. En particular, el papel de las bibliotecas de tipos ha sido minimizado y hay un nuevo tipo de archivos de código fuente, ficheros restringidos IDL, que asumen un papel central para el desarrollo COM en esta versión de Delphi.

## **Capítulo 10: Ribbon**

A comienzos de los ochenta, junto con la revolución del PC, IBM decidió tratar de estandarizar las reglas de la interfaz de usuario (en la época de MS-DOS) de las aplicaciones PC. Esta especificación se llamó Common User Access (CUA) y Microsoft la abanderó durante muchos años en los programas DOS y en las versiones iniciales de Windows, hasta el punto de que llegó a ser una forma natural de interactuar con la mayoría de los programas. Se sabe que los comandos de Copiar y Pegar se encuentran en el menú Edición o que hay que buscar en el menú Archivo para guardar o cargar un documento, así que al menos ya sabes algo de antemano de cada nuevo programa.

Incluso aunque hubo muchos agregados a CUA, incluyendo barras de herramientas, accesos rápidos, barras de comandos y más, su estructura principal se mantuvo durante alrededor de 20 años. Microsoft rompió esta regla por primera vez en Office 2007 (y parcialmente en Windows Vista) con la definición de un nuevo paradigma de interfaz de usuario, llamado Fluent User Interface. Este interfaz es generalmente conocido como el interfaz “Ribbon” debido a su elemento visual principal.

No quiero profundizar en el debate acerca de si ha sido un buen movimiento o no (tengo algunas dudas), más bien quiero centrarme en el hecho de que esta interfaz de usuario y la serie de reglas rigurosas, no son fáciles de implementar sin algunos componentes visuales listos para utilizar. Delphi 2009 incluye este conjunto de controles visuales que son el objeto de este capítulo. Los cuales han sido desarrollados en Embarcadero por Jeremy North.

## **Capítulo 11: Datasets Y dbExpress**

Con toda la atención mostrada en Unicode y en las nuevas características del lenguaje Delphi, es posible que tenga la impresión de que hay pocas novedades en Delphi 2009 para los desarrolladores de base de datos. Esta impresión es completamente equivocada. No solamente disponemos ahora de base de datos con pleno soporte Unicode, en comparación con el apoyo parcial de metadatos de versiones anteriores, sino que también hay varias novedades en dbExpress, incluido una nueva versión de DataSnap que voy a cubrir, en el Capítulo 12. Aquí, en cambio, voy a centrarme en los rasgos esenciales de TDataSet y en las clases relacionadas, el mejorado dbExpress, y ahondar en varios temas asociados.

## **Capítulo 12: DataSnap 2009**

Durante mucho tiempo Delphi ha incluido una tecnología para crear aplicaciones de datos “multi-tier”. Inicialmente conocida como MIDAS y más tarde como DataSnap, esta tecnología estaba basada en COM, incluso aunque la conectividad remota se estableciese a través de sockets o HTTP, en vez de DCOM. Durante algún tiempo incluso soportó CORBA. Una versión ligeramente modificada, que se incluía con la conectividad SOAP.

Delphi 2009 todavía incluye el DataSnap clásico, pero además aporta una nueva y reluciente tecnología de aplicaciones remotas y “multi-tier” también. Está parcialmente basada en la arquitectura dbExpress. Esta nueva tecnología todavía se conoce como DataSnap, pero para evitar confusiones normalmente es referida como “DataSnap 2009”.

## **Conclusión**

En este capítulo he cubierto una de las más significativas actualizaciones en términos de la librería de componentes de Delphi 2009, la nueva arquitectura DataSnap para construir aplicaciones “multi-tier” sin tener que recurrir a COM. Se puede usar DataSnap 2009 para la programación orientada a bases de datos pero también para invocar fácilmente métodos de servidor.

Ya que este es el último capítulo del libro, no hay sección “A continuación” solo una pequeña conclusión acerca del producto. No tengo mucho más que añadir al material presentado, lo que espero es que le ayude a comprender las nuevas características de Delphi 2009 y a apreciar esta nueva versión del producto.

Delphi 2009 es seguramente una versión fuera de serie, con un increíble número de nuevas características que me hizo escribir más de 400 páginas. Con un nuevo propietario (Embarcadero Technologies) y una buena versión con soporte para cualquier idioma (gracias a Unicode), podemos esperar que Delphi tenga una larga vida y una grata codificación, ¡en cualquier parte del mundo!

---

Para más información o adquirir el libro contacte con Danysoft en el 902 123146 [o en el +34 916638683 internacional], o visite [www.danysoft.com/embarcadero](http://www.danysoft.com/embarcadero) .