

Tema 7. Gestión local de la información

Formación específica, cursos verano 2008
ETS de Informática Aplicada
Universidad Politécnica de Valencia

Plan de trabajo

- ◆ Gestión de directorios y ficheros
- ◆ Uso de archivos XML
- ◆ Creación y uso de bases de datos
- ◆ Soporte de Visual Studio a la gestión de datos

Apuntes previos

- ◆ Almacenamiento de objetos
 - Sistema de ficheros (soporte .NET CF)
 - Registro de sistema (sin soporte .NET CF)
 - Bases de datos (soporte .NET CF)
- ◆ Tamaño máximo de un objeto = la mitad de la memoria RAM disponible



GESTIÓN DE DIRECTORIOS Y FICHEROS

Ficheros en .NET

- ◆ Sistema de ficheros homogéneo
- ◆ Uso de las clases
 - System.IO.Directory
 - System.IO.DirectoryInfo
 - System.IO.File
 - System.IO.FileInfo

Determinar el directorio de ejecución de una aplicación

- ◆ Ejecutar ese código al cargar la aplicación (en el método Load_Form) y a partir de ese momento la gestión del directorio actual es mejor llevarla en una variable local

```
string directorio = Assembly.GetExecutingAssembly().GetModules()[0].FullyQualifiedName;  
if (directorio == "") directorio = "\\\";  
else directorio = directorio.Substring(0, directorio.LastIndexOf("\\\"));
```

Obtener los directorios y ficheros contenidos en un directorio dado

```
string [] Directorios = System.IO.Directory.GetDirectories(pathDirectorioActual, "*");  
foreach(string directorio in Directorios)  
{  
    //Código para gestionar el directorio  
}
```

```
string [] Ficheros = System.IO.Directory.GetFiles(pathDirectorioActual, "*");  
foreach(string fichero in Ficheros)  
{  
    //Código para gestionar el fichero  
}
```

Gestión de directorios

◆ DirectoryInfo: Información de un directorio

```
System.IO.DirectoryInfo d = new System.IO.DirectoryInfo (nombre_dir);  
d.CreationTime;           //fecha de creación  
d.Parent; d.Root;        //Nombre del directorio padre o raiz en su ruta de acceso
```

◆ Creación de un directorio

```
Alternativa 1: if (d.Exists == false) d.Create();  
Alternativa 2: Directory.CreateDirectory(nombre_dir);
```

◆ Creación de un subdirectorio

```
d.CreateSubdirectory(nombre_subdirectorio);
```

◆ Destrucción de un directorio

```
Alternativa 1: if (d.Exists == true) d.Delete();  
Alternativa 2: Directory.Delete(nombre_dir,true); //true indica borrado recursivo
```

Gestión de ficheros

◆ Información relativa a un fichero

```
System.IO.FileInfo f = new System.IO.FileInfo (nombre_fichero);  
f.Length;           //Tamaño del archivo  
f.CreationTime;    //Fecha de creación  
f.Extension;       //Extensión del archivo  
f.Directory;       //Directorio en el que se ubica el fichero
```

◆ Creación de un fichero

```
Alternativa 1: if (f.Exists()== false) f.Create();  
Alternativa 2: FileStream fs = File.Create(nombre_fichero)  
Alternativa 3: Cuando el fichero se abre en escritura
```

◆ Lectura de un fichero

```
Alternativa 1:StreamReader reader = new StreamReader(nombre_fichero);  
                textBox4.Text = reader.ReadToEnd();  
                reader.Close();  
Alternativa 2:FileStream f = f.OpenRead();  
                string line;  
                while((line = f.ReadLine()) != null) //Tratamiento de la línea  
                f.Close();
```

Gestión de ficheros

◆ Escritura en un fichero

```
StreamWriter escritor = new StreamWriter(nombre_fichero);  
escritor.Write("texto_a_escribir");  
escritor.Close();
```

◆ Borrado de un fichero

```
FileInfo f = new FileInfo(nombre_fichero);  
if (f.Exists == true) f.Delete();
```

Ejemplo: Ficheros y Directorios

Form1

Directorio actual

Nombre

Tamaño

Explorador Ver txt Gestion directorios/.

Terminar

Form1

Nombre archivo txt

Contenido archivo txt

Explorador Ver txt Gestion directorios/.

Terminar

Form1

Nombre subdirectorio

nombre relativo al directorio actual

Crear Eliminar

Nombre archivo

Nombre archivo txt

Contenido archivo txt

Crear Eliminar

Ver txt Gestion directorios/archivos

Terminar

ARCHIVOS XML

Información codificada en XML

- ◆ XML (Extensible Markup Language)
 - Representación textual y jerárquica de la información independiente de la plataforma
 - Ver ejemplo *Pedidos bebidas*
 - ◆ Elementos:
 - 1^{er} nivel catalog
 - 2^o nivel: category, product
 - 3^{er} nivel: name, salesunit, price
 - ◆ Atributos:
 - category => categoryid, categorianame
 - product => productid

XML en .NET CF

- ◆ *XmlReader* – lectura de un documento XML
 - Lectura hacia adelante
 - Proporciona información del último elemento leído
- ◆ *XmlWriter* – escritura de un documento XML
 - Escritura hacia adelante
 - Incluye información tras el último elemento escrito
- ◆ *XmlDocument* – Implementación .NET CF del modelo XML DOM
 - Guarda en memoria todo el documento XML utilizando una estructura de árbol
 - ◆ Capacidades de navegación y modificación avanzadas
 - ◆ Alto consumo de memoria

Características no soportadas

- ◆ Ciertas facilidades ofrecidas por .NET para el manejo de documentos XML no se soportan en .NET CF
 - Validación utilizando DTDs o XML Schemas
 - Xpath (XML Path Language), XSLT (Extensible Stylesheet Language Transformations)
 - La clase XmlDocument

La clase *XmlReader*

- ◆ Clase abstracta que permite leer un documento XML como una secuencia de nodos
 - Sólo se soporta la navegación *hacia adelante*
- ◆ Se instancia a través de las clases concretas:
XmlTextReader, XmlNodeReader

Instanciar la clase XmlTextReader

- ◆ Se toma la entrada de un *stream* que debe representar un documento XML

```
StreamReader catRdr = new StreamReader(  
    @"\\Archivos de Programa\\pedidobebidas\\catalog.xml");  
XmlTextReader catXML = new XmlTextReader(catRdr);  
//Lectura de datos XML  
//...  
catXML.Close();  
catRdr.Close();
```

Leer el contenido de un documento XML

◆ Uso del método *Read*

```
while (catXML.Read())
{
    MessageBox.Show("Tipo del nodo:" + catXML.nodeType.ToString() +
        "Nombre:" + catXML.Name +
        "Valor:" + catXML.Value);
}
```

Eliminar nodos innecesarios

◆ Espacios en blanco

```
catXML.WhitespaceHandling = WhitespaceHandling.None;
```

◆ Ir al siguiente nodo con contenido

```
catXML.MoveToContent();
```

Proceso de lectura del fichero

```

StreamReader catRdr = new StreamReader(@"Program Files\pedidosbebida\catalog.xml");
XmlTextReader catXML = new XmlTextReader(catRdr);
catXML.WhitespaceHandling = WhitespaceHandling.None;
catXML.MoveToContent();
string[] productItem = new string[3];
string prevNodeName = "";
while(catXML.Read())
{
    if (catXML.NodeType == XmlNodeType.Text)
    {
        switch (prevNodeName)
        {
            case "name":
                productItem[1] = catXML.Value;
                break;
            case "price":
                productItem[2] = catXML.Value;
                productsList.Items.Add(new ListViewItem(productItem));
                break;
        }
    }
    if (catXML.NodeType == XmlNodeType.Element)
    {
        prevNodeName = catXML.Name;
        if (prevNodeName == "product")
            productItem[0] = catXML.GetAttribute("productid");
    }
    else
        prevNodeName = "";
}
catXML.Close();
catRdr.Close();
}

```

```

<category categoryid="2" categoryname="Bebidas con alcohol">
  <product productid="2000">
    <name>Vino blanco Frances</name>
    <salesunit>Caja de 6 botellas</salesunit>
    <price>148.99</price>
  </product>
  <product productid="2001">
    <name>Vino tinto Californiano</name>
    <salesunit>Caja de 6 botellas</salesunit>
    <price>98.99</price>
  </product>
  <product productid="2002">
    <name>Cerveza alemana</name>
    <salesunit>Caja de 24 x 25ml botellin</salesunit>
    <price>29.99</price>
  </product>
</category>

```

La clase *XMLWriter*

◆ Instanciación

```
XMLTextWriter orderWriter =  
    new XMLTextWriter("order.xml", Encoding.ASCII);
```

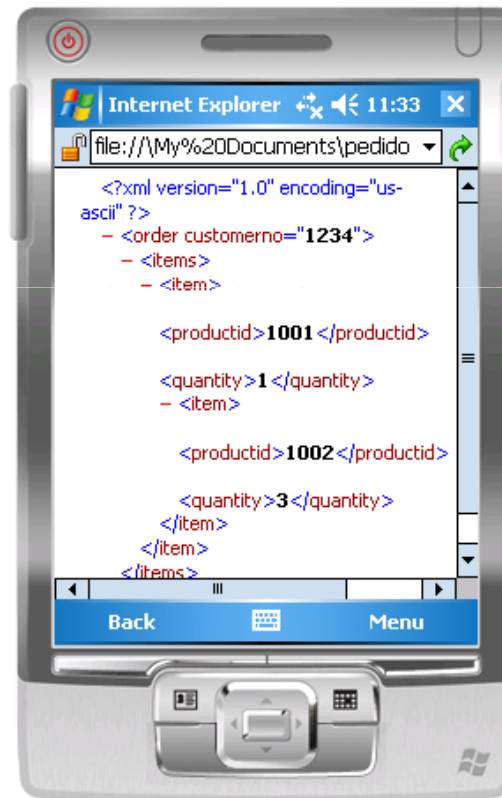
◆ Formato de los valores escritos

```
orderWriter.Formatting = Formatting.Indented;
```

Ejemplo de escritura

```
XmlTextWriter orderWriter = new XmlTextWriter(filename, System.Text.Encoding.ASCII);
orderWriter.Formatting = Formatting.Indented;
orderWriter.WriteStartDocument();
orderWriter.WriteStartElement("order");
orderWriter.WriteAttributeString("customerno", customerNo.Text);
orderWriter.WriteStartElement("items");
foreach(ListViewItem orderItem in orderlist.Items)
{
    orderWriter.WriteStartElement("item");
    orderWriter.WriteElementString("productid", orderItem.SubItems[0].Text);
    orderWriter.WriteElementString("quantity", orderItem.SubItems[1].Text);
    orderWriter.WriteEndElement();
}
orderWriter.WriteEndElement();
orderWriter.WriteEndElement();
orderWriter.WriteEndDocument();
orderWriter.Close();
```

Resultado



BASES DE DATOS

¿Por qué utilizar bases de datos en dispositivos móviles?

- ◆ Disponibilidad de la información en todo momento y todo lugar
- ◆ Conexiones inalámbricas actualmente caras y no ubicuas
- ◆ Emergencia del m-commerce
 - Diversidad de dispositivos móviles en el mercado (HW, SOP e interfaces heterogéneas)
 - Necesidad de ofrecer soluciones de almacenamiento interoperables
 - Acceso y manipulación eficiente de la información sin conexión a la red

Características a tener en cuenta

- ◆ Capacidad de almacenamiento del dispositivo
 - Todo el disponible / Limitado
- ◆ Prestaciones
 - La BD no debe ser un cuello de botella
- ◆ Escalabilidad
 - Incremento tamaño BD → Decremento prestaciones

Características a tener en cuenta

◆ Seguridad

- Autenticación → Evitar accesos no autorizados a la información de la BD
- Cifrado de datos
 - ◆ A nivel de comunicación
 - ◆ A nivel de la información almacenada

Características a tener en cuenta

- ◆ Bajo consumo de recursos
 - Trabajo sobre procesadores lentos
 - Necesidad de huellas de memoria pequeñas
 - Uso de BD modulares, pero adaptadas a cada aplicación
 - Soporte nativo más eficaz
- ◆ Soporte de estándares
 - Mejor curva de aprendizaje y aplicabilidad de conocimientos a distintas plataformas
 - ODBC, JDBC → Gestión de BD relacionales con SQL

SQL Server CE: Introducción

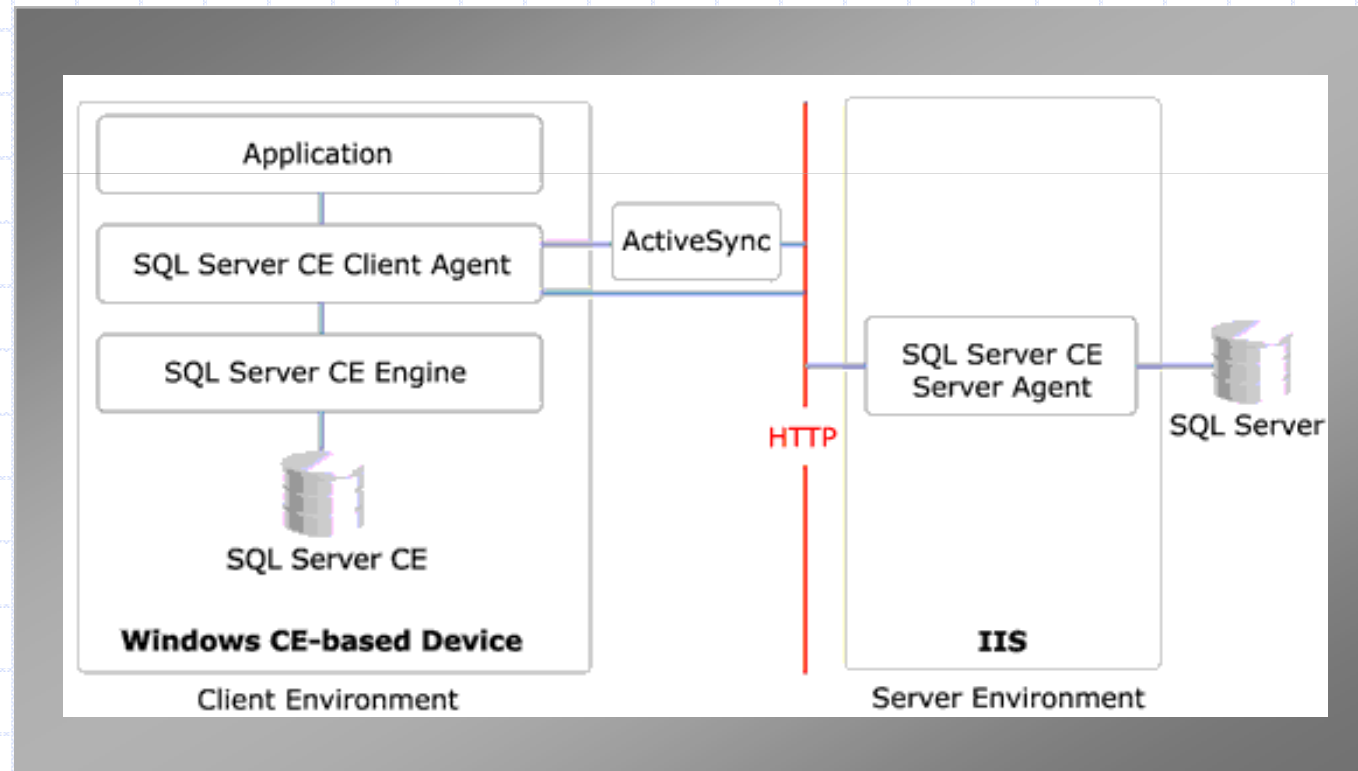
- ◆ No es un 'Server' como el resto de sus hermanos
 - Funciona in-process
- ◆ Tamaño compacto, adaptado al dispositivo
 - Entre 1.2 y 1.6 MB de disco dependiendo del dispositivo físico (tipo de procesador)
 - Soporte de hasta 2 Gb de almacenamiento
 - Soporte de blobs (binary large object) de hasta 1 Gb
 - En uso dependerá de la base de datos
 - ◆ Utiliza directorios temporales para guardar archivos

SQL Server CE: Componentes

- ◆ Proporciona componentes de dispositivo y componentes de servidor
 - Dispositivo
 - ◆ Motor de datos para ejecutar en Windows CE
 - ◆ Interface local para peticiones (queries)
 - ◆ Componentes para gestión de objetos (creación de base de datos, tablas, etc...)
 - Servidor
 - ◆ Extensiones de IIS para habilitar comunicación con el servidor corporativo

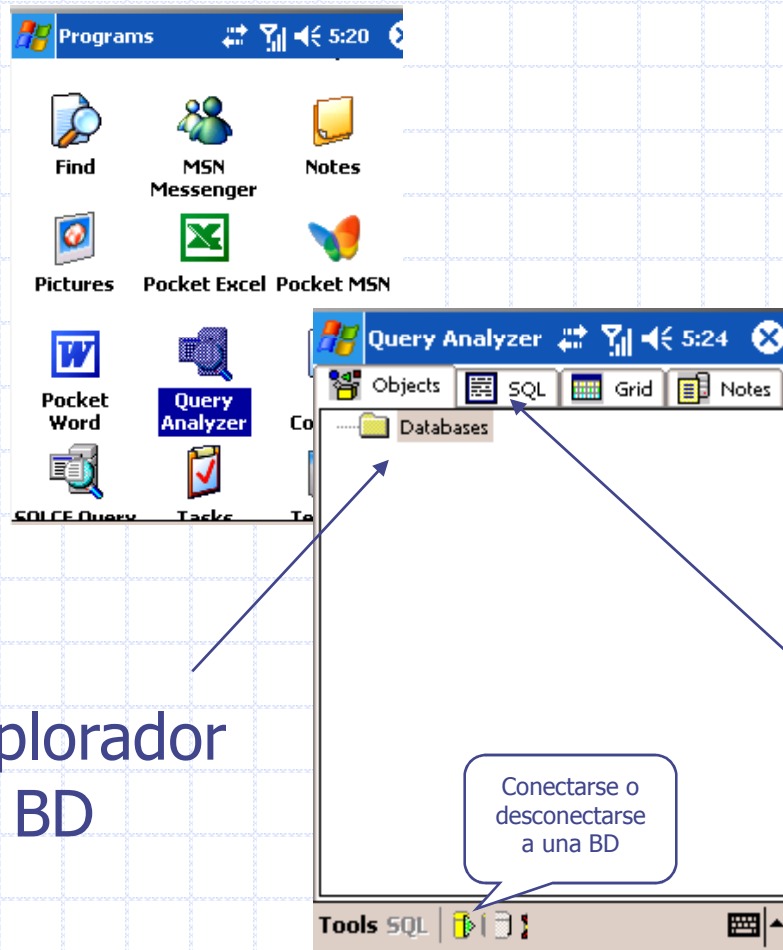
SQL Server CE: Componentes

◆ Arquitectura de los componentes



La utilidad QueryAnalyzer

- ◆ Se instala en el dispositivo al instalar el paquete Microsoft SQL Server CE



Ejemplo

- ◆ Pequeña agenda electrónica



Ejemplo

```
private void TestBBDD()
{
    if (!System.IO.File.Exists ("Test.sdf" )
    {
        SqlCeEngine engine = new SqlCeEngine ("Data Source = Test.sdf");
        engine.CreateDatabase ();
        SqlCeConnection conn = new SqlCeConnection ("Data Source = Test.sdf");
        conn.Open();
        SqlCeCommand cmd = conn.CreateCommand ();
        cmd.CommandText = "CREATE TABLE tblContactos (" +
            "id int IDENTITY ( 1, 1) PRIMARY KEY ," +
            "Nombre NTEXT ," + "Direccion NTEXT ," +
            "CP NTEXT ," + "Poblacion NTEXT ," +
            "Pais NTEXT ," + "Telefono NTEXT ," +
            "movil NTEXT ," + "Fax NTEXT ," +
            "email NTEXT ," + "web NTEXT ," +
            "cumpleaños datetime )";
        cmd.ExecuteNonQuery();
        conn.Close ();
    }
}
```

Crea la BD

Construye la petición de crear una tabla

Realiza la petición

Cierra conexión con la BD

Operación SQL a realizar

Crear conexión
Con BD

Operación SQL a
realizar sobre la BD

Ejecución
de la operación

Leer
información

```
private int LoadData (string SQL)
{
    int index =0;
    MessageBox.Show (SQL);
    SqlConnection myConnection = new SqlConnection("Data Source = Test.sdf");
    SqlCommand myCommand = new SqlCommand(SQL, myConnection);
    try
    {
        myCommand.Connection.Open();
        SqlDataReader myReader =
            myCommand.ExecuteReader(CommandBehavior.CloseConnection);

        if(myReader.Read())
        {
            textNombre.Text = myReader.GetString(1) ; textDireccion.Text = myReader.GetString(2) ;
            textCP.Text = myReader.GetString(3) ; textPoblacion.Text = myReader.GetString(4) ;
            textPais.Text = myReader.GetString(5) ; textTelf.Text = myReader.GetString(6) ;
            textMovil.Text = myReader.GetString(7); textFax.Text = myReader.GetString(8) ;
            textMail.Text = myReader.GetString(9) ; textWeb.Text = myReader.GetString(10) ;
            textCumple.Text = myReader.GetDateTime (11).ToShortDateString ();
            index = myReader.GetInt32(0);
        }
        else
            MessageBox.Show ("No hay datos");

        if(myReader.Read())
            MessageBox.Show ("Se han encontrado mas de un resultado, refine su busqueda");

        myReader.Close();
    }
    catch (SqlCeException e)
    {
        HandleSqlCeException(e);
    }
    finally
    {
        myConnection.Close();
    }
    return index++;
}
```

Leer información

```
//Avanza
private void button2_Click(object sender, System.EventArgs e)
{
    int indice1 = LoadData ( "SELECT * FROM tblContactos Where ID = " +
        (indice+1).ToString ( ) );
    if (indice1 == 0 ) MessageBox.Show ("Se ha llegado al final de la lista");

    else indice++;
}
// Retrocede
private void button1_Click(object sender, System.EventArgs e)
{
    int indice1 = LoadData ( "SELECT * FROM tblContactos Where ID = " +
        (indice-1).ToString ( ) );
    if (indice1 == 0 ) MessageBox.Show ("Se ha llegado al principio de la lista");
    else indice --;
}
//Busca
private void button3_Click(object sender, System.EventArgs e)
{
    indice = LoadData ( "SELECT * FROM tblContactos Where nombre Like " +
        textBox2.Text + "%");
    if (indice == 0 ) MessageBox.Show ("No se ha encontrado coincidencia");
}
```

Añadir datos a la base de datos

```
private void button4_Click(object sender, System.EventArgs e)
{
    SqlConnection conn= new SqlConnection ("Data Source = Test.sdf");
    try{
        conn.Open();
        SqlCommand cmd = conn.CreateCommand ();
        cmd.CommandText = "INSERT INTO tblContactos " +
            "(Nombre, Direccion, CP, Poblacion, Pais, Telefono, Fax, email, web, " +
            " movil, cumpleaños) VALUES (" + textNombre.Text + ", " +
            textDireccion.Text + ", " + textCP.Text + ", " +
            textPoblacion.Text + ", " + textPais.Text + ", " +
            textTelf.Text + ", " + textFax.Text + ", " + textMail.Text +
            ", " + textWeb.Text + ", " + textMovil.Text + ", " +
            DateTime.Parse (textCumple.Text).ToString("dd/MM/yy") + ")";
        cmd.ExecuteNonQuery();
        conn.Close ();
    }
    catch (SqlCeException ex){
        conn.Close ();
        MessageBox.Show (ex.Message );
    }
}
```

Gestión de Excepciones

```
private void DisplaySQLCEErrors(SqlCeException ex)
{
    SqlCeErrorCollection errorCollection = ex.Errors;

    StringBuilder bld = new StringBuilder();
    Exception inner = ex.InnerException;

    if (null != inner)
    {
        MessageBox.Show("Inner Exception: " + inner.ToString());
    }
    // Enumerate the errors to a message box.
    foreach (SqlCeError err in errorCollection)
    {
        bld.Append("\n Error Code: " + err.HResult.ToString("X"));
        bld.Append("\n Message   : " + err.Message);
        bld.Append("\n Minor Err.: " + err.NativeError);
        bld.Append("\n Source    : " + err.Source);

        // Enumerate each numeric parameter for the error.
        foreach (int numPar in err.NumericErrorParameters)
        {
            if (0 != numPar) bld.Append("\n Num. Par. : " + numPar);
        }

        // Enumerate each string parameter for the error.
        foreach (string errPar in err.ErrorParameters)
        {
            if (String.Empty != errPar)
                bld.Append("\n Err. Par. : " + errPar);
        }

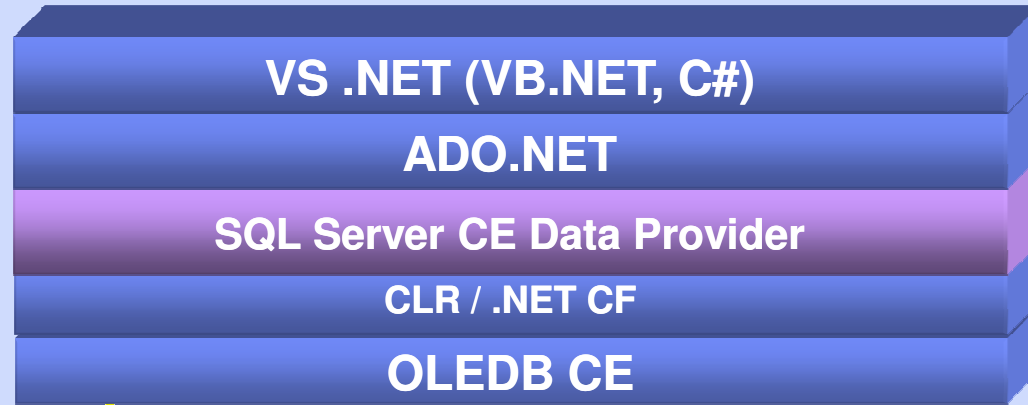
        MessageBox.Show(bld.ToString());
        bld.Remove(0, bld.Length);
    }
}
```

SQL Server CE

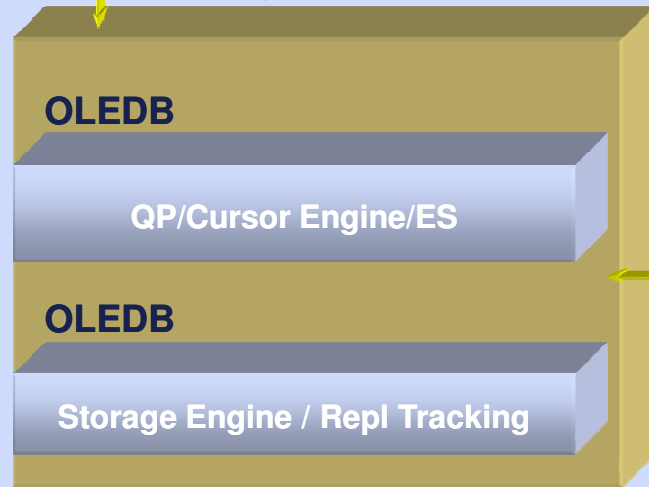
- ◆ Métodos para sincronización
 - Remote Data Access (RDA)
 - ◆ Base de datos local
 - ◆ Modelo Push/Pull al servidor SQL remoto
 - ◆ Seguimiento de modificaciones opcional
 - Merge Replication
 - ◆ Uso de motor de replicación para sincronizar
 - ◆ Modelo de suscripción

SQL Server CE

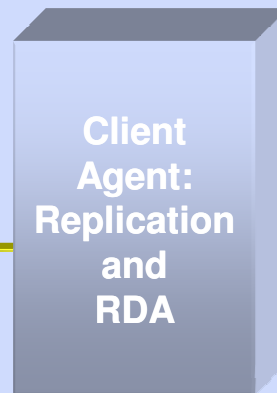
.NET CF / Managed Stack



SQL CE Edition v2.0



Data Provider

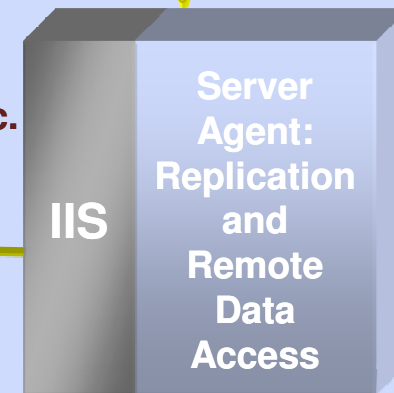
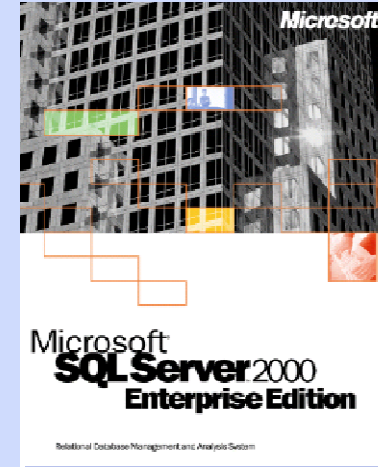


CLIENT

802.11b,
CDPD,
GSM,
CDMA,
TDMA, etc.

HTTP

Occasionally
Connected



IIS

SERVER

SOPORTE DE VISUAL STUDIO A LA GESTIÓN DE DATOS

Demo

- ◆ Crear un origen de datos
=> Explorador de servidores
- ◆ Crear un proyecto
- ◆ Agregar al proyecto un origen de datos
=> Orígenes de datos
- ◆ Dejar caer el origen de datos en el formulario

Localización del runtime necesario

- ◆ En función de lo que se tenga instalado
 - Visual Studio 2005 => *Unidad:*\Program Files\Microsoft Visual Studio 8\SmartDevices\SDK\SQL Server\Mobile\v3.0
 - SQL Server 2005 => *Unidad:*\Program Files\Microsoft SQL Server 2005 Mobile Edition\Device\Mobile\v3.0
- ◆ En función del sistema para el que se desarrolle
 - Mobile 2003 y Windows CE 4.0 => \wce400\armv4
 - Windows CE 5.0 y Windows Mobile 5.0 => \wce500\ *procesador* que puede ser {armv4i, mipsii, x86, etc.}

Ficheros que definen el runtime y sirven para su instalación

- ◆ El más importante:
 - `qlce30.plataforma.wce5.procesador.cab`
donde *plataforma* es *phone* para smartphone, *ppc* para Pocket PC, and *en_blanco* para Windows CE
- ◆ Soporte a la replicación:
 - `sqlce30.repl.plataforma.wce5.procesador.cab.`
 - Contiene soporte para RDA y Replicación.
- ◆ Soporte de desarrollo, incluyendo el Query Analyzer tool, que se localiza en:
 - `sqlce30.dev.ENU.plataforma.wce5.procesador.cab`

Más información

- ◆ Consultar la MSDN de Visual Studio
 - <http://msdn.microsoft.com/es-es/library/ms229335.aspx>
- ◆ SQL Server Compact Edition (SSCE) books online
 - <http://msdn.microsoft.com/en-us/library/ms173053.aspx>