Windows Mobile Programming

برنامه نویسی ویندوز موبایل

ویرایش: مرداد ۱۳۹۰

YousefAmiri.ir

در این مقاله، با مفاهیم زیر آشنا می شوید:

ایجاد نرم افزار برای دستگاه های موبایل توسط سی شارپ
ایجاد بانک اطلاعاتی از نوع SQLCE یا sdf
ذخیره، حذف و جستجوی رکوردهای جدول توسط C#
استفاده از DataSet و DatGrid برای نمایش داده‌ها
استفاده از SqlCeConnection و SqlCeCommand و اشیاء ADO.NET

معرفـــی

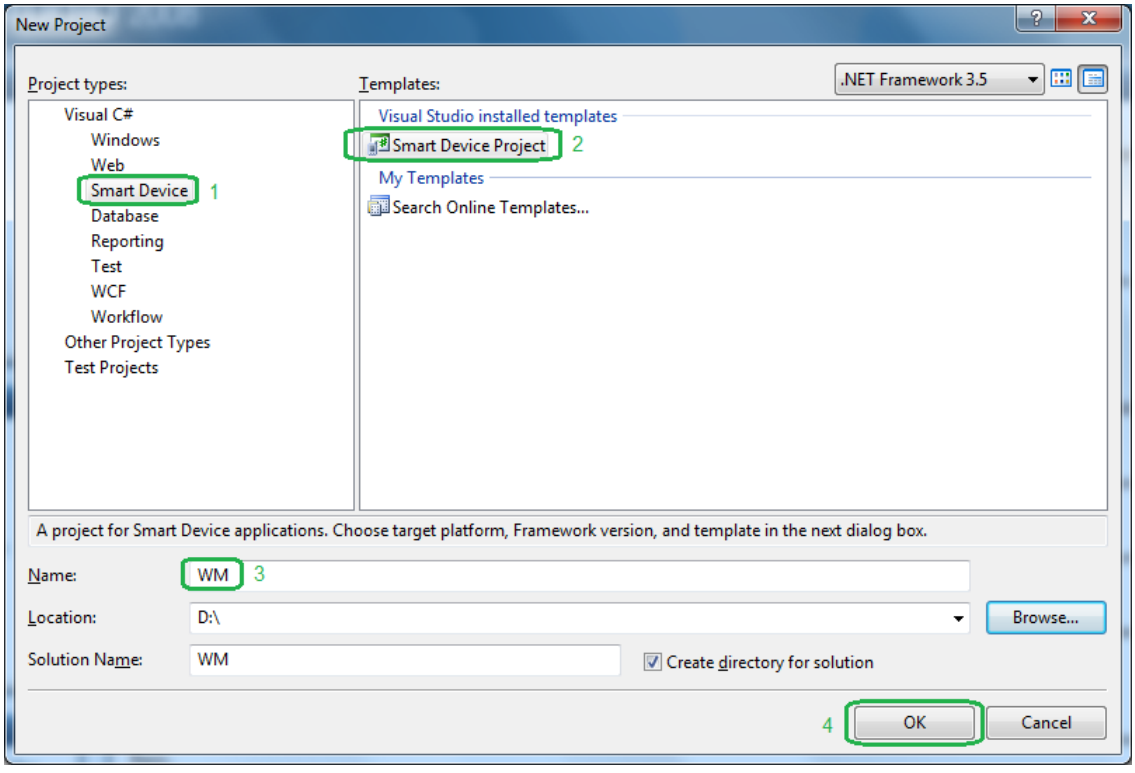
در ویژوال استودیو امکان برنامه نویسی برای دستگاه‌های قابل حمل از جمله Pocket PC و موبایل که سیستم عامل Windows Mobile دارند، فراهم شده است. البته باید در همان ابتدا با این دید شروع بکار نمایید که برنامه موردنظر شما متناسب با بستر اجرایی آن باشد. به عبارت دیگر، در این گونه دستگاه‌ها محدودیت ابعاد صفحه نمایش، سرعت پردازنده و حافظه وجود دارد. از طرف دیگر صفحه لمسی و اغلب بارکدخوان در اختیار شماست که باید رابط کاربر نرم افزار را متناسب با آن طراحی کنید.

در این مقاله، پروژه WM را بصورت گام به گام توسعه می دهیم که یک نرم افزار مبتنی بر بانک اطلاعاتی SQLCE است و داده‌های مربوط به کالاهای موجود در انبار را گردآوری و ذخیره می کند. ویژوال استودیو دارای شبیه ساز دستگاه‌های موبایل است و برنامه را در محیط آن اجرا می کند، بنابراین نیازی به داشتن یک دستگاه از این نوع با ویندوز موبایل ندارید.

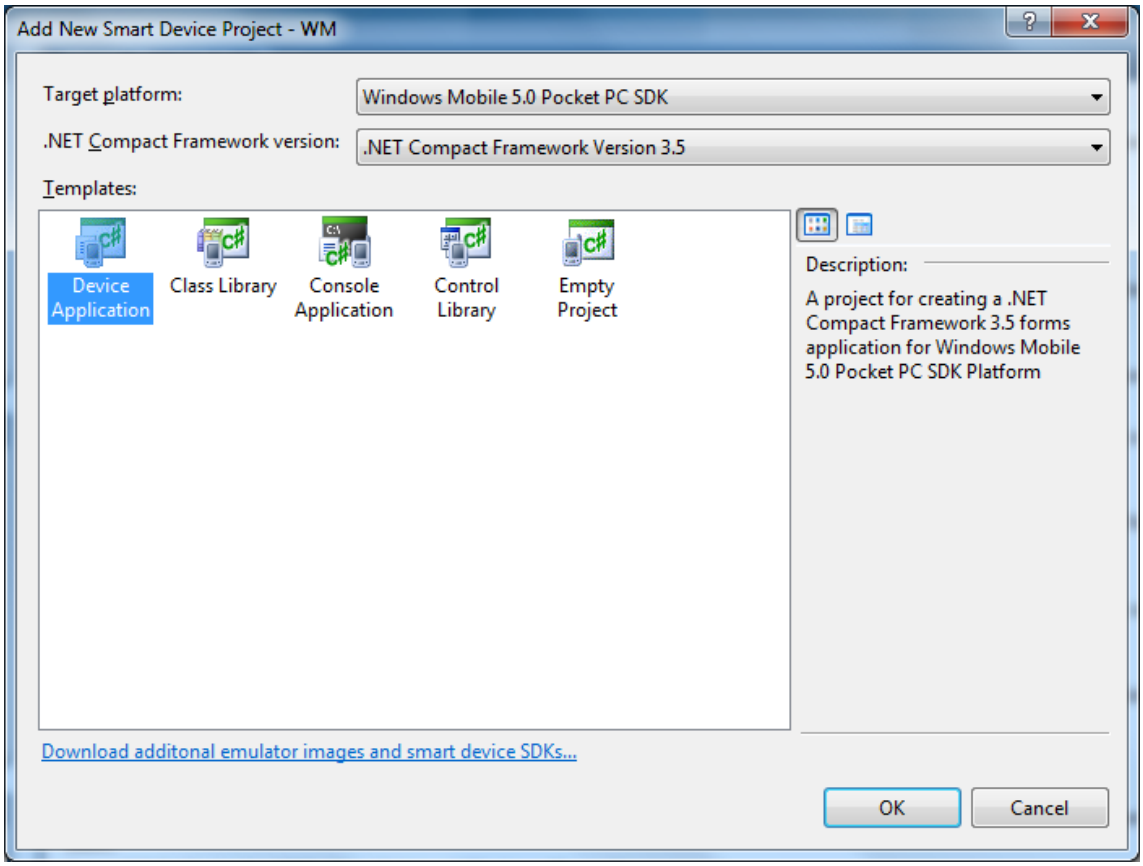
متأسفانه امکان استفاده از LINQ در این نوع پروژه‌ها در .NET Framework 3.5 فراهم نیست و برای ارتباط با دیتابیس از ADO.NET استفاده می کنیم.

ایجاد پروژه WM

مطابق شکل ۱ و ۲ پروژه جدیدی از نوع Smart Device در ویژوال استودیو و با نام WM ایجاد کنید.



شکل ۱ - ایجاد یک پروژه برای دستگاه‌های هوشمند



شکل ۲ – انتخاب نوع پروژه

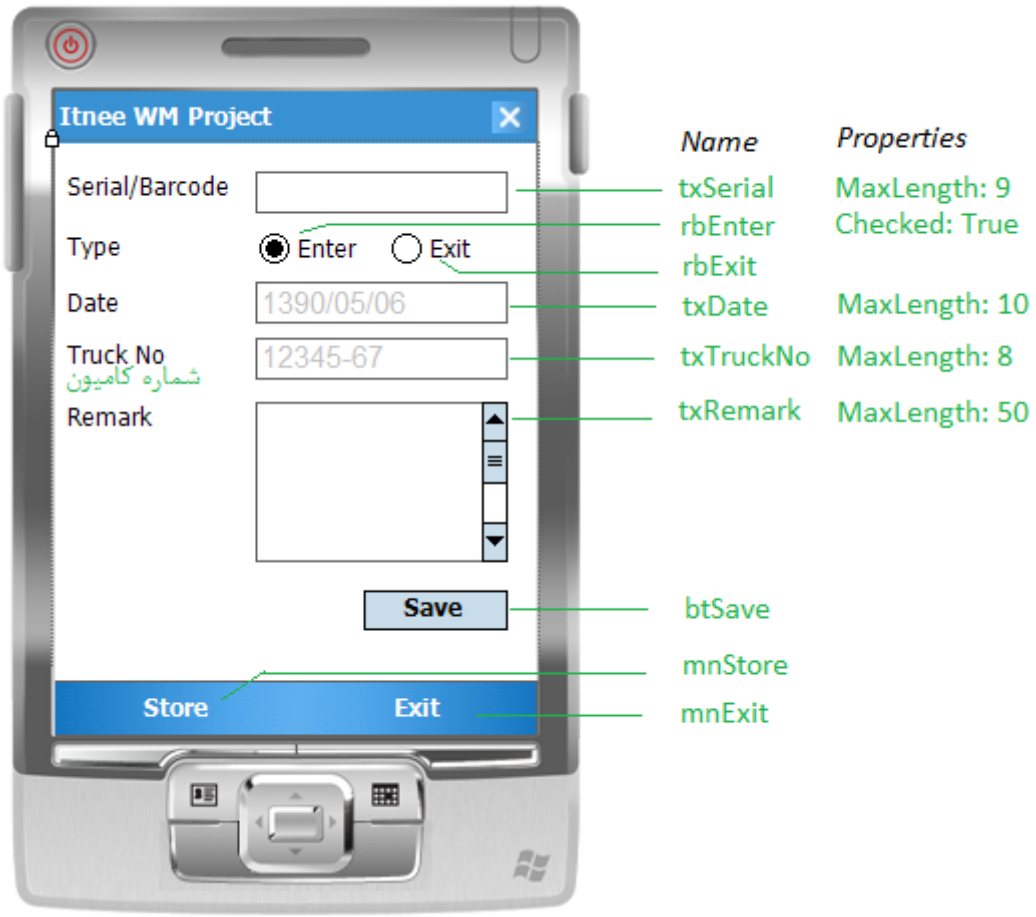
در این مرحله ویژوال استودیو یک فرم خالی در محیط شبیه ساز ایجاد می کند. آنرا به frMain تغییر نام دهید.

ایجاد منوی اصلی

از Toolbox روی MainMenu دوبار کلیک کنید تا به فرم اصلی اضافه شود. سپس آیتم‌های Store و Exit را به آن اضافه کنید. به آیتم Store گزینه Search را اضافه کنید و آنها را به ترتیب به mnStore، mnExit و mnSearch تغییر نام دهید.

طراحی فرم ثبت کالاهای ورودی یا خروجی

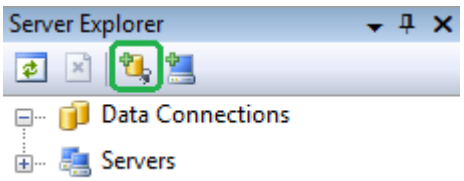
مطابق شکل ۳ فرم frMain را طراحی کنید. در اینجا برای جلوگیری از پیچیدگی فرم، از تعداد محدودی فیلد استفاده شده است.



شکل ۳ – طراحی فرم frMain

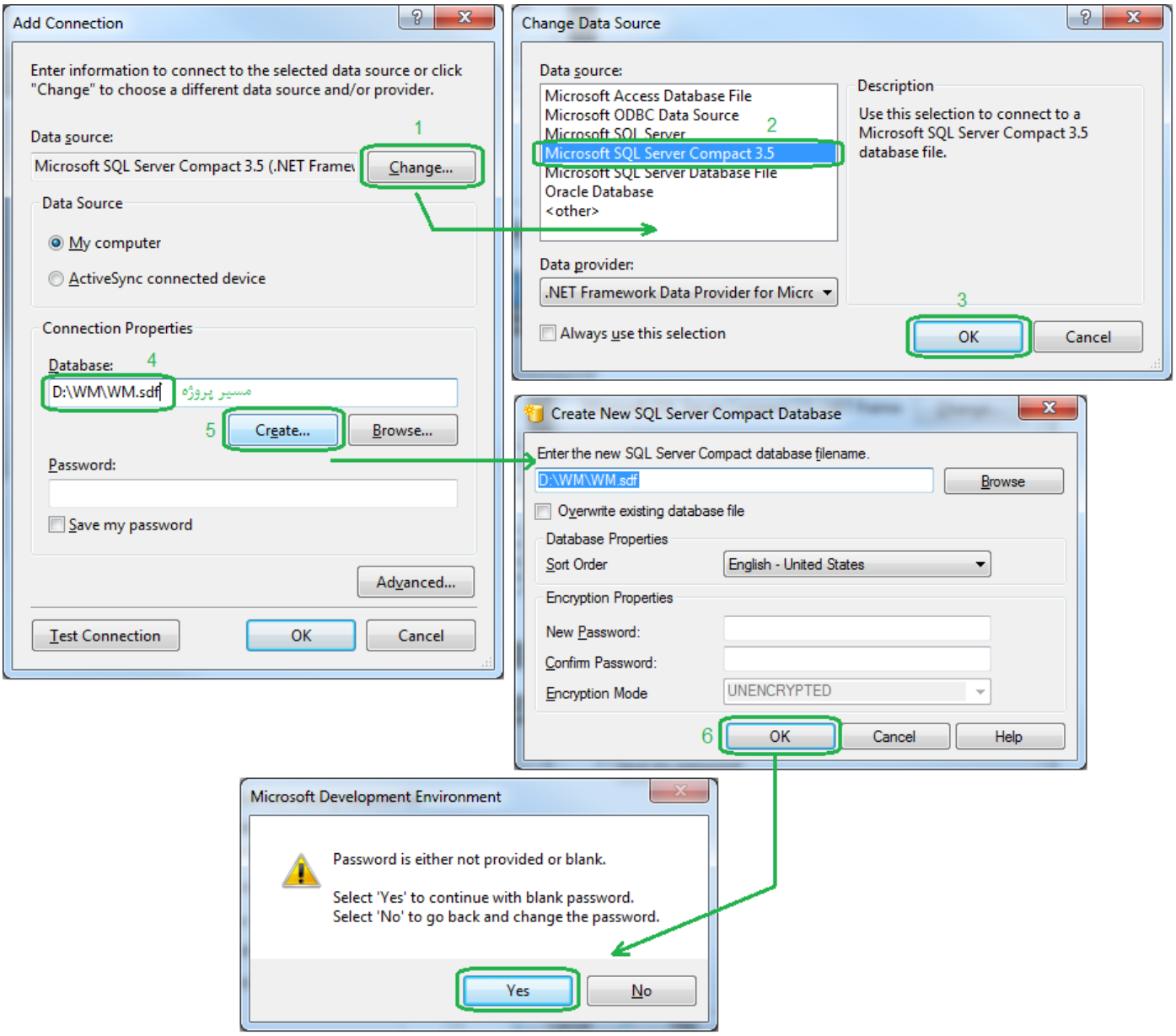
ایجاد بانک اطلاعاتی

بانک اطلاعاتی WM را با یک جدول به نام tStore طبق مراحل زیر ایجاد می کنیم. به منوی View بروید و Server Explorer را کلیک کنید. سپس روی آیکن Connect to Database کلیک کنید.



شکل ۴ – ایجاد بانک اطلاعاتی از طریق پانل Server Explorer

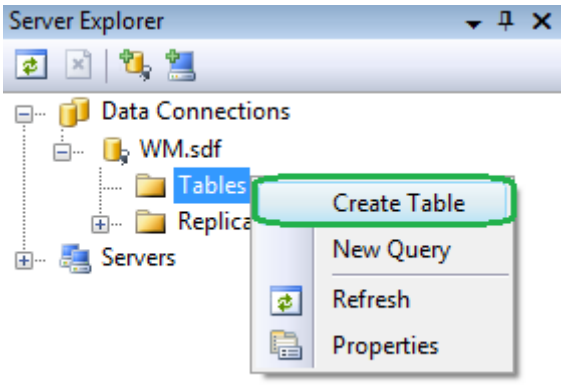
مطابق شکل ۵، در پنجره Add Connection نوع منبع داده (بانک اطلاعاتی) و نام و مسیر آنرا وارد کرده و دکمه Create را کلیک می کنیم. در این پروژه آموزشی، قصد نداریم برای بانک اطلاعاتی کلمه عبور تعریف کنیم.



شکل ۵ – مراحل ایجاد بانک اطلاعاتی از نوع sdf

ایجاد جدول tStore

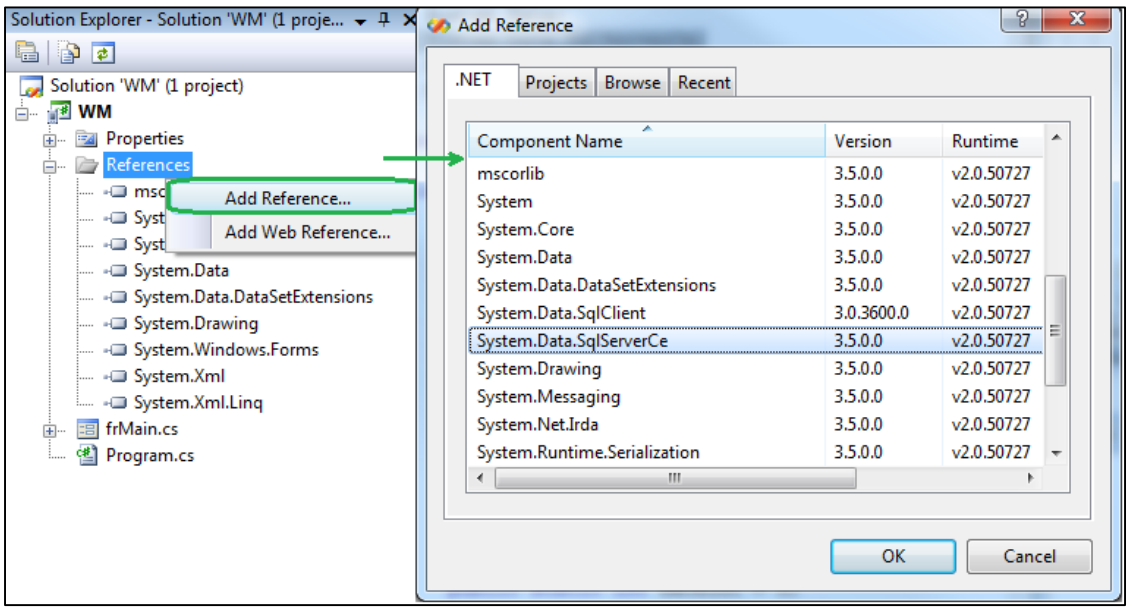
مطابق شکل ۶ در پانل Server Explorer ساختار درختی بانک اطلاعاتی WM را باز کنید، همانطور که مشاهده می کنید بانک های اطلاعاتی از نوع SQLCE بسیار ساده و کم حجم بوده و اشیاء متداول مانند View و ... را ندارند.



شکل ۶ – ایجاد جدول در بانک اطلاعاتی

ارتباط با بانک اطلاعاتی

برای ارتباط با بانک اطلاعاتی باید ریفرنس SQLServerCe را به پروژه اضافه نمایید. مطابق شکل ۷ از طریق پانل Solution Explorer آنرا به زیر شاخه References اضافه کنید.



شکل ۷ – اضافه کردن ریفرنس SQLServerCe

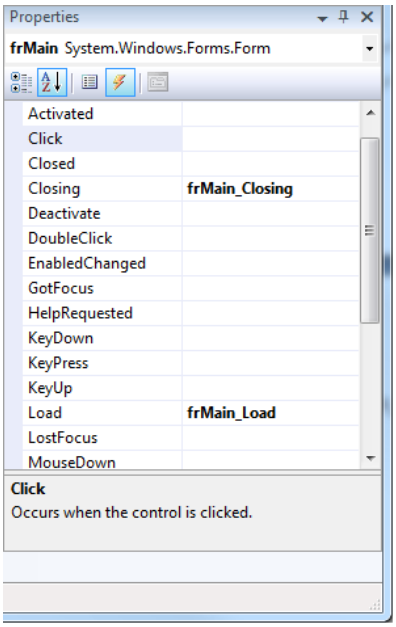
سپس فضاهای نامی زیر را به فرم frMain اضافه کنید:

```
using System.Data.SqlServerCe;
using System.IO;
using System.Reflection;
```

حال یک متغیر سراسری برای کانکشن بانک اطلاعاتی تعریف می کنیم تا در هنگام لود شدن فرم، کانکشن، ایجاد شده و در هنگام خروج از برنامه، بسته شود. بنابراین در ابتدای تعریف کلاس frMain دستور زیر را قرار دهید:

```
lCeConnection Cn;
```

فرم frMain را انتخاب و رویدادهای `frMain_Load` و `frMain_Closing` را با دوبار کلیک ایجاد کنید.



شکل ۸ – ایجاد دو رویداد **Load** و **Closing** برای باز کردن و بستن کانکشن در فرم اصلی

کد فرم frMain را بصورت زیر بنویسید:

```
using System.Data.SqlServerCe;
using System.IO;
using System.Reflection;

namespace WM
{
    public partial class frMain : Form
    {
        SqlCeConnection Cn;

        public frMain()
        {
         InitializeComponent();
        }

        private void frMain_Load(object sender, EventArgs e)
        {
             Cn = new SqlCeConnection("Data Source =" +
             Path.GetDirectoryName(Assembly.GetExecutingAssembly().GetName().CodeBase) +
                                      "\\WM.sdf");
             Cn.Open();
        }

        private void frMain_Closing(object sender, CancelEventArgs e)
        {
         Cn.Close();
        }

        private void mnExit_Click(object sender, EventArgs e)
        {
         Cn.Close();
         Application.Exit();
        }
    }
}
```

ذخیره در جدول tStore

روی دکمه btSave دوبار کلیک کنید و کد زیر را برای آن بنویسید:

```
private void btSave_Click(object sender, EventArgs e)
```

```
{
  //Check User Values (1)
  if (txSerial.Text == "" || txDate.Text.Length != 10 || txTruckNo.Text.Length != 8)
  {
    MessageBox.Show("Fill all fields.", "Save", MessageBoxButtons.OK, MessageBoxIcon.Exclamation,
                    MessageBoxDefaultButton.Button1);
    return;
  }

  try
  {
    int tmp = Convert.ToInt32(txSerial.Text);
  }
  catch
  {
    MessageBox.Show("Enter a valid digit.", "Save", MessageBoxButtons.OK,
    MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
    txSerial.Focus();
    return;
  }

  //Check Uniq Serail (2)
  var CmCount = new SqlCeCommand("SELECT COUNT(*) From tStore WHERE Serial=" + txSerial.Text,Cn);
  object objCount = CmCount.ExecuteScalar();
  if (objCount != DBNull.Value && Convert.ToInt32(objCount) != 0)
  {
    MessageBox.Show("Serial is already exist.", "Save", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation, MessageBoxDefaultButton.Button1);
    txSerial.Focus();
    return;
  }

  //Insert to tStore (3)
  var Cm=new SqlCeCommand("INSERT INTO tStore Values(@Serial,@Type,@Date,@TruckNo,@Remark)",Cn);
  Cm.Parameters.AddWithValue("@Serial", Convert.ToInt32(txSerial.Text));
  Cm.Parameters.AddWithValue("@Type", rbEnter.Checked);
  Cm.Parameters.AddWithValue("@Date", txDate.Text);
  Cm.Parameters.AddWithValue("@TruckNo", txTruckNo.Text);
  Cm.Parameters.AddWithValue("@Remark", txRemark.Text);
  Cm.ExecuteNonQuery();

  txSerial.Text = "";
  txSerial.Focus();
}
```

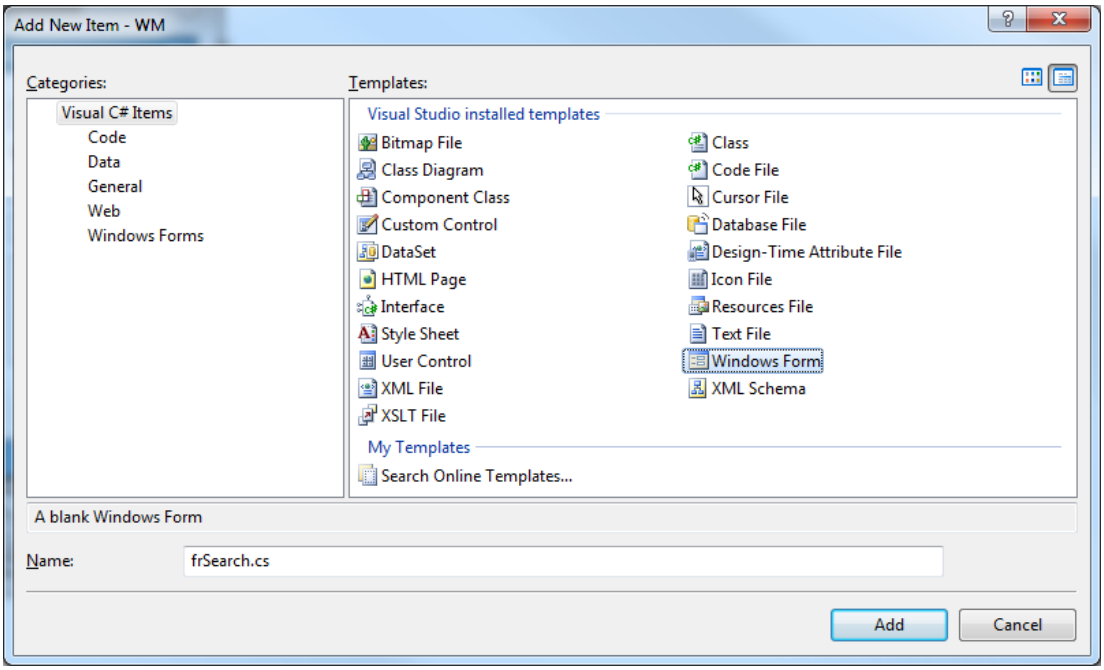
بخش ۱ کد بالا، مقادیر وارده کاربر را کنترل می کند و اگر سریال، غیر عدد باشد، به کاربر پیام هشدار می دهد.

بخش ۲ کد بالا، سریال وارده را در جدول جستجو می کند و درصورتیکه قبلا وجود داشته باشد، پیام می دهد.

بخش ۳ کد بالا، عملیات اصلی (ذخیره) را انجام می دهد.

ایجاد فرم جستجو

از منوی Project گزینه Add Windows Form... را انتخاب کنید. سپس مانند شکل ۹ یک فرم جدید به پروژه اضافه کنید.



شکل ۹ – ایجاد فرم frSearch

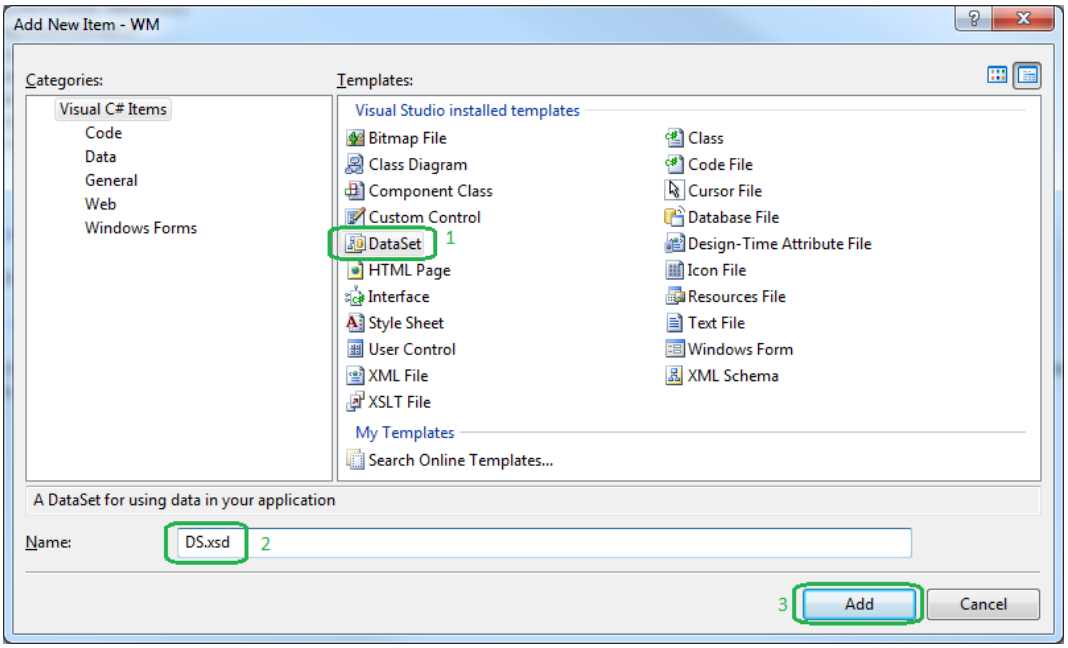
در فرم frMain و منوی اصلی، روی گزینه Search دوبار کلیک کنید و کد زیر را برای آن بنویسد:

```
private void mnSearch_Click(object sender, EventArgs e)
{
    var f = new frSearch();
    f.ShowDialog();
}
```

مراحل ایجاد DataGrid در فرم frSearch برای نمایش داده ها و جستجوی کالا

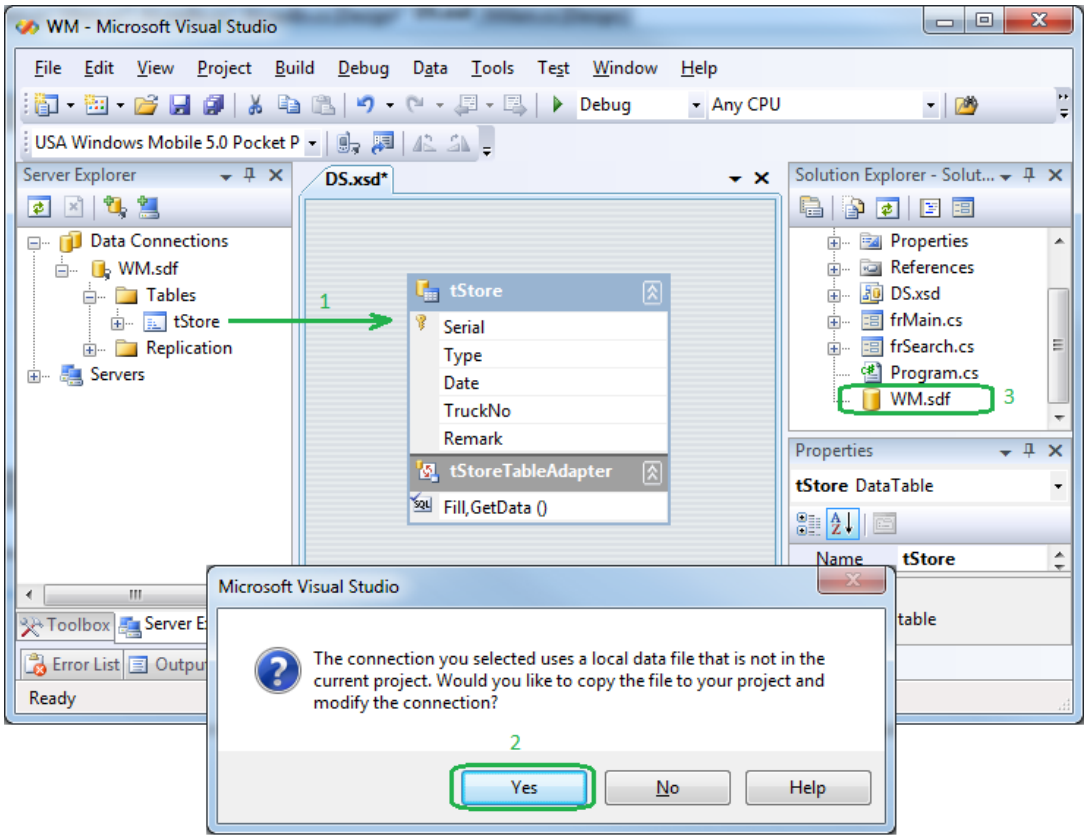
۱. ایجاد DataSet
۲. کشیدن جدول بانک اطلاعاتی به داخل دیتاست
۳. کشیدن جدول از داخل دیتاست بر روی فرم frSearch

از منوی Project گزینه Add New Item را کلیک کنید. سپس DataSet را انتخاب و مانند شکل ۱۰ آنرا نامگذاری کنید:



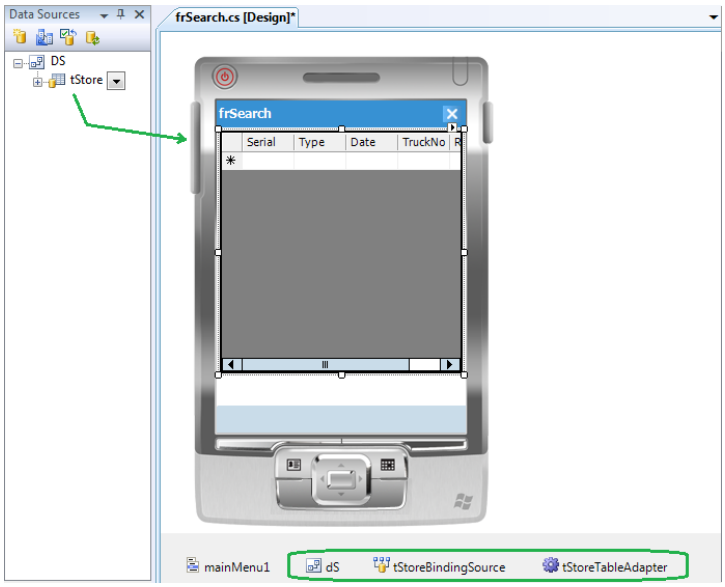
شکل ۱۰ – ایجاد دیتاست DS

جدول tStore را به داخل دیتاست بکشید. ویژوال استودیو به شما اطلاع می دهد که فایل بانک اطلاعاتی به پروژه اضافه نشده است. که با کلیک بر روی دکمه Yes این فایل را مطابق شکل ۱۱ به پروژه اضافه می نماید.



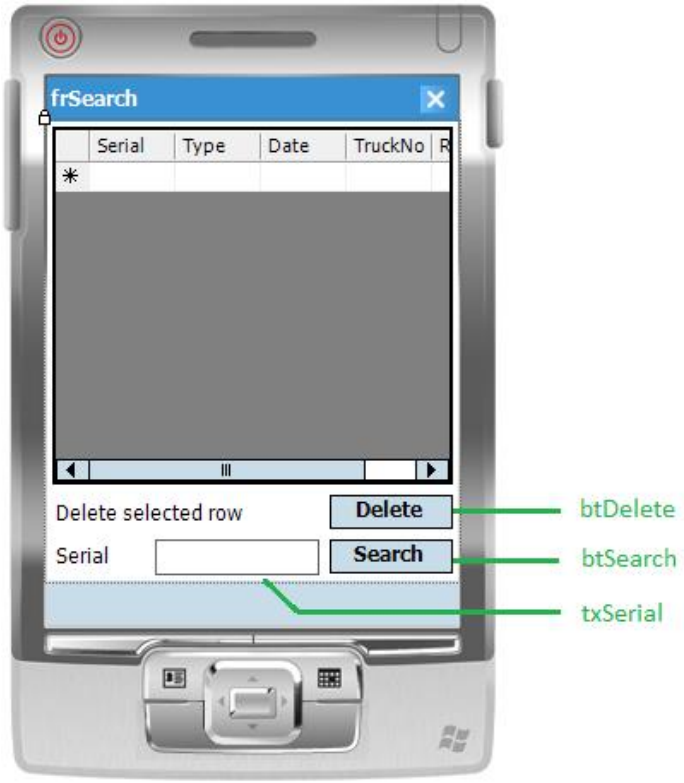
شکل ۱۱ – اضافه کردن جدول tStore به دیتاست و بانک اطلاعاتی به پروژه

از منوی Data گزینه Show Data Sources را کلیک کنید تا پانل Data Sources باز شود. سپس جدول tStore را به داخل فرم frSearch بکشید.



شکل ۱۲ – ایجاد DatGrid برای جستجو

فرم frSearch را مطابق شکل ۱۳ طراحی کنید



شکل ۱۳ – طراحی فرم جستجو

کد فرم جستجو را مانند زیر بنویسید:

```
using System.Data.SqlServerCe;
using System.IO;
using System.Reflection;

namespace WM
```

```
{
    public partial class frSearch : Form
    {
        public frSearch()
        {
            InitializeComponent();
        }

        private void frSearch_Load(object sender, EventArgs e)
        {
            if (DSUtil.DesignerUtil.IsRunTime())
            {
                tStoreTableAdapter.Fill(dS.tStore);
            }

            if (tStoreBindingSource.Count == 0)
                btDelete.Enabled = false;

        }

        private void btDelete_Click(object sender, EventArgs e)
        {
            //Get Current Row
            object[] obj = ((DataRowView)tStoreBindingSource.Current).Row.ItemArray;

            if (MessageBox.Show("Are you sure you want to delete " + obj[0].ToString() + "?",
                "Delete", MessageBoxButtons.YesNo, MessageBoxIcon.Question,
                MessageBoxDefaultButton.Button2) == DialogResult.Yes)
            {
                SqlCeConnection Cn = new SqlCeConnection("Data Source =" +
                Path.GetDirectoryName(Assembly.GetExecutingAssembly().GetName().CodeBase) +
                                        \\WM.sdf");
                Cn.Open();
                var Cm=new SqlCeCommand("DELETE FROM tStore WHERE Serial="+obj[0].ToString(),Cn);
                Cm.ExecuteNonQuery();
                tStoreBindingSource.RemoveCurrent();
                if (tStoreBindingSource.Count == 0)
                    btDelete.Enabled = false;
                Cn.Close();
            }
        }

        private void btSearch_Click(object sender, EventArgs e)
        {
            if (txSerial.Text != "")
                tStoreBindingSource.Filter = "Serial=" + txSerial.Text;
            else
                tStoreBindingSource.Filter = "";
            btDelete.Enabled = (tStoreBindingSource.Count > 0);
        }
    }
}
```

با لود شدن فرم، داده های جدول tStore به داخل گراید ریخته می شود. اگر جدول خالی باشد دکمه Delete غیرفعال می شود.

با فشردن دکمه Delete، ردیف جاری گراید، در آرایه obj ریخته می شود که برای دسترسی به هر سلول در ردیف جاری، کافیست اندیس آنرا ذکر کنید. برای نمونه `obj[0].ToString()` شماره سریال از ردیف انتخاب شده را برمی گرداند.

برای جستجو در گراید از خاصیت Filter شیء BindingSource استفاده می کنیم. این شیء به گراید متصل است و هر عملی روی آن انجام شود در گراید منعکس می گردد.

پروژه را اجرا کنید، چند رکورد ثبت کنید و سپس در فرم جستجو آنها را حذف یا جستجو کنید. توجه داشته باشید.