Accessing Data Using System.OleDb

Source **http://www.csharphelp.com/archives/archive132.html**

This simple application demonstrates several aspects of object-oriented programming in C#. It builds a simple class called "Batters" with several fields and then populates their values by retrieving data from an Access Database. In my research on how to retrieve data using System.OleDb I found that there wasn't any one help file that demonstrated how to put it all together. I hope this helps.

```
// BEGIN C# CODE
using System;
using System.Text;
using System.Data;
using System.Data.OleDb;

public class Batter
{
  // Declare private fields.
  private string firstName;
  private string lastName;
  private char bats;
  private int ab;
  private int runs;
  private int hits;
  private int doubles;
  private int triples;
  private int homers;
  private int rbis;
  private int walks;
  private int ks;
  private int sb;

  // Constructor without supplied arguments
  public Batter()
  {
    this.firstName="";
    this.lastName="";
    this.bats = ' ';
    this.ab = 0;
    this.runs = 0;
    this.hits = 0;
    this.doubles = 0;
    this.triples = 0;
    this.homers = 0;
    this.rbis = 0;
    this.walks = 0;
    this.ks = 0;
    this.sb = 0;
  }

  // Constructor with all arguments supplied
```

```
public Batter(string firstName, string lastName, char bats, int ab,
              int runs, int hits, int doubles, int triples, int homers,
              int rbis, int walks, int ks, int sb)
{

  this.firstName = firstName;
  this.lastName = lastName;
  this.bats = bats;
  this.runs = runs;
  this.ab = ab;
  this.hits = hits;
  this.doubles = doubles;
  this.triples = triples;
  this.homers = homers;
  this.rbis = rbis;
  this.walks = walks;
  this.ks = ks;
  this.sb = sb;
}

// Properties with Get and Set accessors for private access fields
public string FirstName
  {
    get
    {
      return firstName;
    }
    set
    {
      firstName = value;
    }
  }

  public string LastName
  {
    get
    {
      return lastName;
    }
    set
    {
      lastName = value;
    }
  }

public char Bats
{
  get
  {
    return bats;
  }
  set
  {
    bats = value;
  }
}
```

```
public int AB
{
  get
  {
    return ab;
  }
  set
  {
    ab = value;
  }
}

public int Runs
{
  get
  {
    return runs;
  }
  set
  {
    runs = value;
  }
}

public int Hits
{
  get
  {
    return hits;
  }
  set
  {
    hits = value;
  }
}

public int Doubles
{
  get
  {
    return doubles;
  }
  set
  {
    doubles = value;
  }
}

public int Triples
{
  get
  {
    return triples;
  }
  set
  {
    triples = value;
    }
  }

  public int Homers
  {
    get
    {
      return homers;
    }
    set
    {
      homers = value;
    }
  }

  public int RBIs
  {
    get
    {
      return rbis;
    }
    set
    {
      rbis = value;
    }
  }

  public int Walks
  {
    get
    {
      return walks;
    }
    set
    {
      walks = value;
    }
  }

  public int Ks
  {
    get
    {
      return ks;
    }
    set
    {
      ks = value;
    }
  }

  public int SB
  {
    get
    {
      return sb;
    }
    set
    {
      sb = value;
    }
  }

  // Overrided ToString method from System.Object which formats the player
  // info in a string suitable for console output.
  public override string ToString()
  {
    StringBuilder strb = new StringBuilder(500);
    strb.Append("Batting Statistics");
    strb.Append("\n=========================");
    strb.Append("\nName: " + firstName + " " + lastName);
    strb.Append("\nBats: " + bats);
    strb.Append("\nAB: " + ab);
    strb.Append("\nRuns: " + runs);
    strb.Append("\nHits: " + hits);
    strb.Append("\nDoubles: " + doubles);
    strb.Append("\nTriples: " + triples);
    strb.Append("\nHomers: " + homers);
    strb.Append("\nRBIs: " + rbis);
    strb.Append("\nWalks: " + walks);
    strb.Append("\nKs: " + ks);
    strb.Append("\nSB: " + sb);
    return strb.ToString();
  }

  public static void Main()
  {
    // Instantiates b as a new Batter object
    Batter b = new Batter();

    // Stores connection string and sql select statement as strings
    string strConnection = "Provider=Microsoft.Jet.OLEDB.4.0;";
    strConnection += " Data Source=c:\\mlb.mdb;";
    strConnection +=" user id=; password=;";
    string strCommand = "SELECT * FROM Batters";

    OleDbConnection conn = new OleDbConnection(strConnection);
    OleDbDataAdapter adapter = new OleDbDataAdapter();
    adapter.SelectCommand = new OleDbCommand(strCommand, conn);
    try
    {
      conn.Open();
      Console.WriteLine("The connection is open");
      DataSet ds = new DataSet();
      adapter.Fill(ds);

      // Ideally you would load several batter records from your
      // database and loop through an array of batter objects.
      // This program assumes only one record exists in your
      // data set.
      foreach(DataTable dt in ds.Tables)
        foreach(DataRow dr in dt.Rows)
        {
          b.FirstName = Convert.ToString(dr["FirstName"]);
```

```
          b.LastName = Convert.ToString(dr["LastName"]);
          b.Bats = Convert.ToChar(dr["Bats"]);
          b.AB = Convert.ToInt32(dr["AB"]);
          b.Runs = Convert.ToInt32(dr["Runs"]);
          b.Hits = Convert.ToInt32(dr["Hits"]);
          b.Doubles = Convert.ToInt32(dr["Doubles"]);
          b.Triples = Convert.ToInt32(dr["Triples"]);
          b.Homers = Convert.ToInt32(dr["Homers"]);
          b.RBIs = Convert.ToInt32(dr["RBIs"]);
          b.Walks = Convert.ToInt32(dr["Walks"]);
          b.Ks = Convert.ToInt32(dr["Ks"]);
          b.SB = Convert.ToInt32(dr["SB"]);
        }
      Console.WriteLine("Data was retrieved");
    }
    catch(OleDbException e)
    {
      Console.WriteLine("Error: {0}", e.Errors[0].Message);
    }
    finally
    {
      string connState = conn.State.ToString();
      if ( connState == "Open")
      {
        conn.Close();
        Console.WriteLine("The connection has been closed\n");
      }
      else
        Console.WriteLine("The connection was never open\n");
    }

      Console.WriteLine(b.ToString());
      // Code to keep console window open after program execution.
      // This is valuable if you are building your code from an IDE
      // like VS.NET or SharpDevelop.
      Console.Write("\nPress ENTER to continue");
      Console.Read();
  }
}
// END C# CODE
```

Because I wanted to limit the scope of this demo application, you will have to do a few simple tasks before the program will work.

1. **Create an Access database named "mlb.mdb" and save it to "c:\mlb.mdb".**
2. **Create a simple table named "Batters".**
3. **Create the following fields in the "Batters" table:**
    - **FirstName (Text)**
    - **LastName (Text)**
    - **Bats (Text)**
    - **AB (Numeric)**
    - **Hits (Numeric)**
    - **Runs (Numeric)**
    - **Doubles (Numeric)**

- **Triples (Numeric)**
- **Homers (Numeric)**
- **RBIs (Numeric)**
- **Walks (Numeric)**
- **Ks (Numeric)**
- **SB (Numeric)**

Now simply add the stats of your favorite ballplayer. When you are all done, run the application. Your console output should look something like this:

```
The connection is open
Data was retrieved
The connection has been closed
```

***~~~ End of Article ~~~***