

Forum:[Hack Forums](#)

Thread: [Need to generate an Excel with data from a Database](#)

K???er - August 20, 2020, 3:18 pm

So I need to generate an Excel document with data from a database. There is a GUI where there is an option to choose between 2 classes. Then the program has to generate an Excel document with a worksheet for every student in that class. There are 4 tables, each with a foreign key. Now the problem is that I can't figure out how to generate a worksheet for every student. Can't really explain it clearly without showing the database and assignment itself.

EDIT: So I made everything work but I'm like 100% sure there is alot of stuff that can be rewritten much more nicely. I'm using DAOs and connections with Singleton and I think I've used way too much of them.

Would be nice if you could PM me if you think you could help me out to make the code look better and more efficient.

Thanks!

Shadow_Cloud - August 20, 2020, 11:36 pm

Post some code and we can have a look?

Hawk HF - August 21, 2020, 2:35 pm

(Aug 20, 2020 11:36 PM)Shadow_Cloud Wrote: [\[CITE\]](#) Post some code and we can have a look? Agreed. We're missing SO much info here that it's difficult to give advice.

ZestyVibes - August 23, 2020, 1:29 am

As has been said, its hard to know exactly what you are trying to achieve without knowing more about the surrounding code.

There are many ways to get data from a database, and there are many ways of writing an Excel file. Some methods of writing to Excel require interop and an actual running instance to be open. I will presume you would rather avoid that.

EPPlus is a great Excel library for C# which has no dependencies outside .Net and its free if not used commercially. There is also an older unmaintained verison under LGPL license.

[EPPlus \(new license\)](#)

[EPPlus \(old LGPL\)](#)

My general advice :

1. Get the data from the database using an appropriate technique and read it into a usable C# type you have created (e.g. StudentData)

I would use a simple SqlDataReader unless i was using a more complicated ORM layer to get data.

2. Write the data using EPPlus, making sure to add a new Worksheet to the Workbook for each student.

Example program to read student data from an Sql database and export a multi-worksheet .xlsx using EppPlus:

```
Code using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using OfficeOpenXml;
using OfficeOpenXml.Style;
```

```
namespace StudentDataExport
{
    internal class Program
    {
        private static void Main()
        {
            // Sql connection string
            var connectionString = "Data Source=(local);Initial Catalog=StudentDatabase;" + "Integrated Security=SSPI";

            // Get data from Sql DB in a usable format
            var studentData = ReadStudentData(connectionString);

            // Write data to excel file
            WriteToExcel(studentData);
        }

        private static List<StudentData> ReadStudentData(string connectionString)
        {
            var studentData = new List<StudentData>();

            var queryString =
                "SELECT StudentId, StudentName, StudentAddress, SeatNumber, AverageGrade FROM
                dbo.Students;";

            using (var connection = new SqlConnection(connectionString))
            {
                var command = new SqlCommand(queryString, connection);
                connection.Open();

                var reader = command.ExecuteReader();

                // Call Read before accessing data.
                while (reader.Read())
                {
                    var student = ReadRow(reader);
                    studentData.Add(student);
                    Console.WriteLine(student.ToString());
                }

                // Call Close when done reading.
                reader.Close();
            }
        }
    }
}
```

```

    return studentData;
}

private static StudentData ReadRow(IDataRecord record)
{
    return new StudentData
    {
        StudentId = (int) record[0],
        StudentName = (string) record[1],
        StudentAddress = (string) record[2],
        SeatNumber = (int) record[3],
        AverageGrade = (decimal) record[4]
    };
}

private static string WriteToExcel(List<StudentData> studentData)
{
    using (var package = new ExcelPackage())
    {
        foreach (var data in studentData)
        {
            //Add a new worksheet to the empty workbook for the student
            var worksheet = package.Workbook.Worksheets.Add(data.StudentName);
            //Add the headers
            worksheet.Cells[1, 1].Value = "ID";
            worksheet.Cells[1, 2].Value = "Name";
            worksheet.Cells[1, 3].Value = "Address";
            worksheet.Cells[1, 4].Value = "Seat";
            worksheet.Cells[1, 5].Value = "Average Grade";

            //Add the row
            worksheet.Cells["A2"].Value = data.StudentId;
            worksheet.Cells["B2"].Value = data.StudentName;
            worksheet.Cells["C2"].Value = data.StudentAddress;
            worksheet.Cells["D2"].Value = data.SeatNumber;
            worksheet.Cells["E2"].Value = data.AverageGrade;

            //format column headers;
            using (var range = worksheet.Cells[1, 1, 1, 5])
            {
                range.Style.Font.Bold = true;
            }

            // Lets set the header text
            worksheet.HeaderFooter.OddHeader.CenteredText = $"&24&U&\\"Arial,Regular Bold\\"
{data.StudentName}";
            // Add the page number to the footer plus the total number of pages
            worksheet.HeaderFooter.OddFooter.RightAlignedText =
                string.Format("Page {0} of {1}", ExcelHeaderFooter.PageNumber,
ExcelHeaderFooter.NumberOfPages);
            // Add the sheet name to the footer
            worksheet.HeaderFooter.OddFooter.CenteredText = ExcelHeaderFooter.SheetName;

```

```

        // Add the file path to the footer
        worksheet.HeaderFooter.OddFooter.LeftAlignedText =
            ExcelHeaderFooter.FilePath + ExcelHeaderFooter.FileName;
    }

```

```

    // Set some document properties
    package.Workbook.Properties.Title = "Student Export Data";
    package.Workbook.Properties.Author = "Your Name";

```

```

    var xlFile = new FileInfo("StudentExport.xlsx");

```

```

    // Save our new workbook in the output directory and we are done!
    package.SaveAs(xlFile);
    return xlFile.FullName;
}

```

```

}
}

```

```

public class StudentData

```

```

{
    public int StudentId { get; set; }
    public string StudentName { get; set; }
    public string StudentAddress { get; set; }
    public int SeatNumber { get; set; }
    public decimal AverageGrade { get; set; }

```

```

    public override string ToString()

```

```

    {
        return $"{StudentId}, {StudentName}, {StudentAddress}, {SeatNumber}, {AverageGrade}";
    }

```

```

}
}

```