Innovative Technologies in a Systems Integration Curriculum: XML and Microsoft Visual Studio .NET

Alan Peslak (arp14@psu.edu)
Information Sciences and Technology, Penn State University
Dunmore, PA 18512, USA

Abstract

As a part of an upper-level systems integration course, the author incorporated instruction and projects in current integrative programming technologies, XML and Microsoft Visual Studio.NET. This paper provides an overall background on XML and IDEs and a review of the literature concerning their instructional implementations. A program of basic instruction is then reviewed and illustrated to provide a framework for the inclusion of these technologies in an information systems and sciences curriculum. Included is an active approach for learning server side programming in MS Visual Studio. The author includes exercises and evaluations of his implementation.

Keywords: programming, systems development, Integrated Development Environment, XML, Microsoft Visual Studio .NET, systems integration, .NET

1. INTRODUCTION

The importance of integrative technologies has risen to a level of prominence in recent years. Extensible markup language (XML) is suggested by Kay (2002) to have "become the primary means of defining, storing and formatting data in a multitude of areas, including documents, forms and databases." And the use of integrated development environments for programming and web development has become the standard method of operation for educators and professionals alike. Integrated development environments such as Microsoft Visual Studio offer significant productivity advantages in a production environment saving many hours of tedious and unproductive coding. Their use should be understood by today's students and graduates. Within this context, the inclusion of both of these integrative technologies as a part of an upper-level systems integration course seems essential. Beginning in 2003 and continuing in 2004 the author has included significant content on both XML and Microsoft Visual Studio.NET.

2. LITERATURE REVIEW

There has been little work published on the use of integrated development environments for pedagogical purposes. One of the few articles discussing how to include Microsoft.NET into information technology courses was prepared by Chaytor and Leung (2003). The authors see major advantages in using technologies that integrate web, programming, and wireless skills. The authors used a sequence of five courses through which the students are guided, beginning with visual programming, through dynamic web pages and concluding with XML and web services. The authors suggest this sequence and the use of the Microsoft Visual Studio .NET platform allow for progressive learning as well as preparing the students to be "industry-ready" by the time they complete their courses. The authors intend to add a service learning project to supplement the course and provide real-world experience.

Some of the significant difficulties encountered and benefits realized from their experience are noted below.

Difficulties

- Technical problems with IIS, FrontPage extensions and security
- Size and complexity of the IDE
- Understanding server side concept
- XML complexity
- Time

Benefits

- Motivation
- Lack of plagiarism
- Enjoyed the environment
- Creativity
- Ability to create "cool" interactive websites
- Relevant topics and leading-edge skills

Frank (2002) notes four advantages for a student to use an IDE for programming: reduce typing, improve organization, reduce errors, and improve debugging. He therefore recommends the use of an IDE for students. The one problem he does note is that IDEs are designed for commercial purposes and to improve productivity. This often involves code generation which Frank opposes. His article describes a methodology to prevent using these features in IDEs and require students to generate their own (rather than system generated) code. This is much more of a concern, in my opinion, in lower level courses to gain an understanding of programming steps and code. It is less of a problem once students understand programming code and are moving toward graduation and a work environment where understanding and use of IDEs will be required. Abuhejleh (2004) notes the importance of XML technologies and illustrates a teaching approach contrasting XML and relational technologies.

Ziegler and Crews (1999) suggest that commercial IDEs do not provide a proper environment for students learning programming. They suggest that these commercial IDEs are too complex, do not support problem solving, and have difficult to understand debugging capabilities. The authors discuss their simplified IDE FLINT under development at Western Kentucky which has been used successfully for introductory programming courses. Though it may be true that IDEs assume too much for introductory programming courses, many of these suggested shortcomings do not apply when students reach upper level status. My experience is that after basic programming in languages such as Java and C++, in traditional programming environments, students not only can understand but also can appreciate the benefits gained by IDEs. They do not have difficulty navigating the "complexity", they understand the debugging and problem solving is aided by other IDE tools.

Reis and Cartwright (2004) note that professional IDEs "work well in advanced courses". They note that the environment is often too complex for introductory courses. The complexity of learning language, algorithms, as well as a tool proves too complex. They also note that the professional IDEs do not have a simple interface because they are designed not as a learning tool but as a productivity tool with a dizzying array of features and benefits. Professional IDEs do not provide simple tools to avoid some initial syntax stumbling blocks which limit learning focus. Finally, the professional programs are large and sometimes too large for students' home computers. The authors have developed a lightweight Java plug-in for Eclipse known as Dr. Java that they suggest overcomes all of these shortcomings. Historically, Norman and Nunamaker (1989) found strong professional support for integrated development tools at the time and suggested their continued development as a means to improve productivity and performance.

At the other end of the spectrum, McKinney (2003) discusses the use of SoftWIRE from SoftWIRE technology as a further graphical front-end to drag and drop graphical, flow-chart images that can then generate actual VB code. The front end is suggested to be a "better" way to teach beginning programming courses since the emphasis is on the development of algorithms and procedures rather than learning complicated syntax. This is a different viewpoint from many traditional computer scientists who are already concerned with the simplification of programming through the IDE's already extensive graphical capabilities. Many of these instructors suggest that coding needs to be understood at the basic level. To understand what is truly happening at the lower level is essential according to these others so that efficient programs can be written and then properly comprehended and debugged. McKinney disagrees and suggests that even though millions of lines of traditional code exist, we will slowly convert from our legacy text-based programming to a graphics-based model. If this is so then the use of IDEs is not unwelcome, but an evolutionary step in programming development. Others working on lightweight environments for introductory programming courses include Kolling and Rosenberg (1996) with Blue.

Though Payne (2001) used an integrated development environment in his Java

programming course, he does recommend that students not use the IDE for at least one assignment so that they can fully understand documentation and syntax issues.

Lim (2002) discusses the teaching of web development technologies both past, present and future. Lim traces the evolution of web development instruction with its past reliance on HTML, Java and Java Script, CGI, and ADO through a future which includes active server pages XML, a web DB and web services. Though the author uses Java and JSP, the concepts and actual working code assignments can be accomplished and taught easily utilizing MS Visual Studio .NET, up to and including web services.

Microsoft presents many case studies of successful uses of Visual Studio.NET for productivity improvement. HP used the IDE to develop a unified digital imaging application for consumers. The firms estimated a 30% reduction in development time and a 25-50% reduction in future code maintenance due to using MS VS.NET (Microsoft, 2004).

As a result of the literature review and the author's experience, XML and IDEs are viewed as being important topics for our curriculum.

3. IMPLEMENTATION

Beginning in 2003, XML and Microsoft IDE Visual Studio .NET were included as a part of the author's Information Technology and Systems Integration courses. This paper focuses on the first of these courses, IST 420, Information Technology and Systems Integration I as it was presented in the Spring of 2004. Both XML and Visual Studio were significant portions of an overall introductory course on integrative technologies. The Official Course Description of IST 420 **INFORMATION TECHNOLOGY AND SYSTEMS INTEGRATION I** (3 credits) is: Introductory course on integration of information technology into different systems including the planning, development, and implementation of the integration.

The Course Objectives as defined in the syllabus are listed below.

Upon completion of the course, the student will be able:

To complete design and implementation of the following technologies at an introductory level

- XML
- Systems Analysis and Design
- SAP
- ASP.NET
- Enterprise Application Integration
- Service Oriented Architecture

In addition the following objectives are included

- To understand IT within an organization and enterprise;
- To understand the dynamics and associated issues when systems are introduced into practice;
- To foster an understanding of the role of IT in system integration;
- To recognize information technology integration issues in different systems;
- To acquire the fundamental organization, technology, and data modeling skills necessary for system integration;
- To understand the concept of XML for data integration;
- To understand enterprise resource planning in an organization;
- To understand enterprise application integration and the emerging role of web services.

Background for the course

What is systems integration?

- Integrated business solutions that access a common database, cross functional processes, and provide seamless operation. (example, ERP systems)

- Integration of distributed components across multiple servers and/or geographic locations via various middleware solutions. (example, CORBA, service oriented architecture/web services, ACID issues)

- Integration of complex business problems into comprehensive solutions (example, systems analysis and design, agile methods)

- Use of integrated environments for agile software development (example, Visual Studio .Net)

Why systems integration?

Provide high quality, low cost, efficient and effective business solutions.

"IT personnel are business people first." (Reich and Nelson, 2003)

Required Texts:

Concepts in Enterprise Resource Planning, Course Technology, Brady, Monk, Wagner ISBN 0619015934

Kalata, K. (2002) Introduction to ASP.NET. Course Technology. ISBN 0-619-06321-1

The Object Primer: Introduction to Techniques for Agile Modeling
A Ronin International White Paper, Scott W. Ambler (2001). Freely Available at http://www.ronin-intl.com/publications/objectPrimerAgileModeling.pdf

Schaum's Outline of Theory and Problems of Software Engineering. David Gustafson. (2002) McGraw-Hill. ISBN 0071377948

Additional online content.

All required. This is a two semester sequence course. Some texts will be used in both semesters.

XML

The initial assignment for XML technologies was to read the excellent outline prepared by Spiderpro, Kickstart XML Tutorial (Kampherbeek, 2001). With this background a basic exercise was then assigned.

> Please prepare an xml file and xsl style sheet to display the xsl.
> Please create a new 2 dimensional multi-row table with minimum 3 columns and 5 rows. Use new colors and a different table format.
>
> Sources for some of the html needed are http://www.davesite.com and http://www.w3schools.com
>
> Be creative.
>
> You may work together but you must submit your own unique work.
>
> Please submit a Word document with xml code, xsl code, and a screen print of the xsl displayed xml.

The exercise was well received and provided a good introduction to the integrative capabilities of XML. A post exercise survey yielded the following results.

XML survey
Q. What is your overall rating of this exercise?
0 - (1) 1=very poor
0 - (2) 2=poor
3 - (3) 3=average
6 - (4) 4=good
4 - (5) 5=very good
Average Response: 4.1 (4=good)
Responses: 13-Valid, 0-Blank, 13-Total

Code Sample 1 in the Appendix provides the XML and XSL code which illustrates an example of a successful solution. The screen shot of the result is shown in Figure 1.

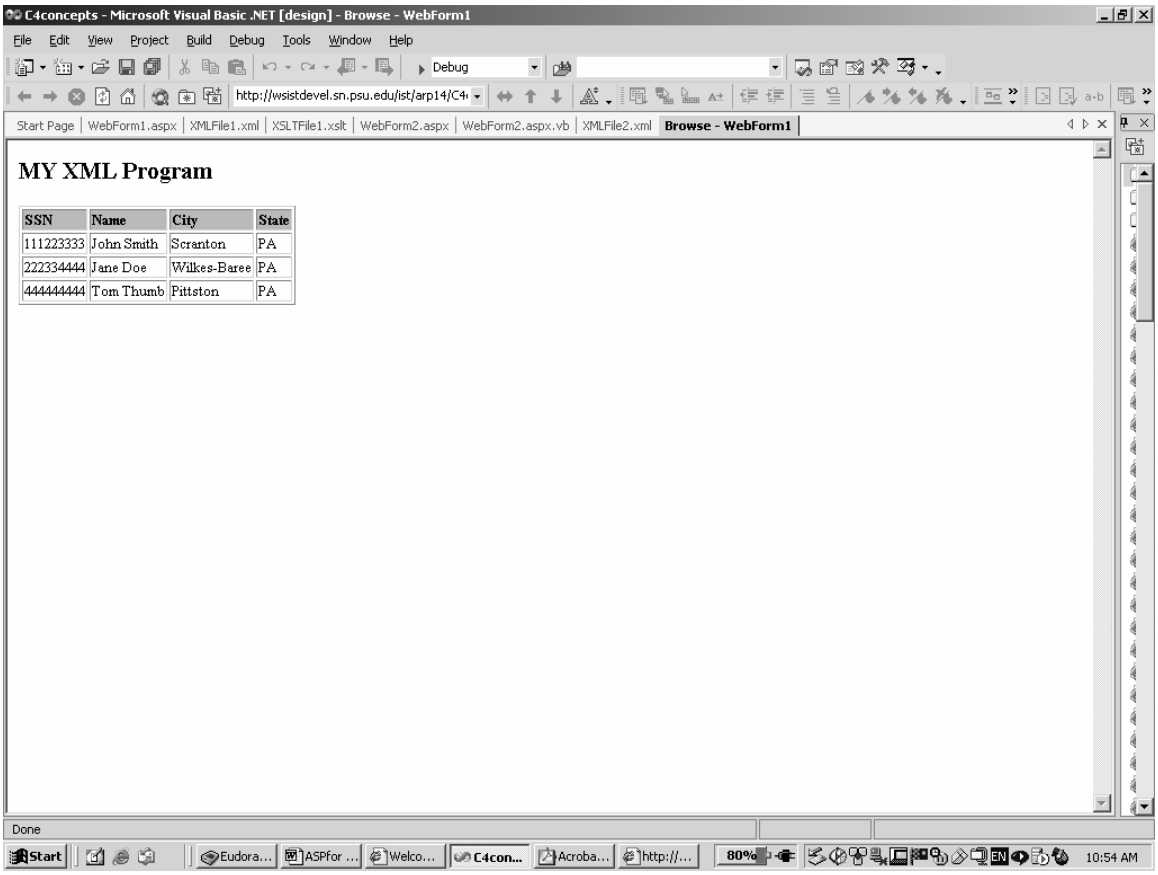


Figure 1 Actual XML code displayed from XSL Transform

In addition to this simple introduction, more complex exercises were incorporated from Chapter 4 of the Kalata (2003) text.

ASP.NET

The primary method of instruction was the text and text projects in the Course Technology text, Introduction to ASP.NET by Kalata (2003). The text provides good practical step by step instructions for the major Visual Studio topics and related three-tier, server side programming problems. Some of the topics covered with related exercises were:

- Console Applications
- Visual Basic .NET
- WebForms
- WebForm controls
- HTML Controls
- Solution Explorer
- Server Explorer
- Calendar Control
- Validation Controls
- XML and XSLT
- Data Binding
- Code Behind

In order to introduce the many topics, the author prepared simple exercises that highlighted features of the chapters in a simple work-along structure. The class took place in a setting where all students had access to computers and our server and I illustrated and they coded along these basic programs. The examples in Kalata were excellent and allowed good follow-up, but these exercises were simple enough for everyone to follow and also just illustrated the IDE construct.

Simple Chapter 3 Concepts such as understanding code behind were illustrated by a simple button and text box controls. The actual screen shots of this exercise where a button click changed text in a

textbox is shown in Figure 2. Figure 3 illustrates the use of a case statement and a validation control. Figure 4 shows the properties of the validation control. The actual code for each is shown in Code Sample 2 in the Appendix.

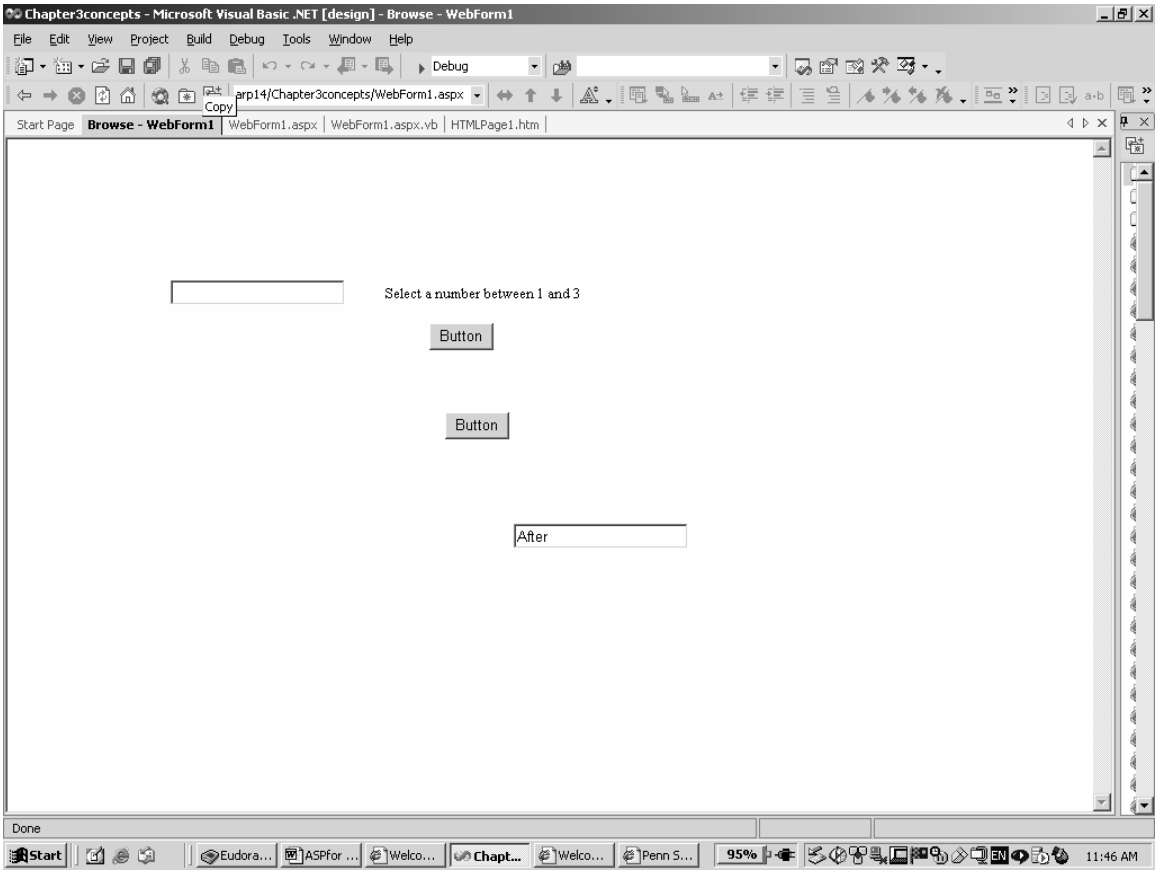


Figure 2 TextBox and Button

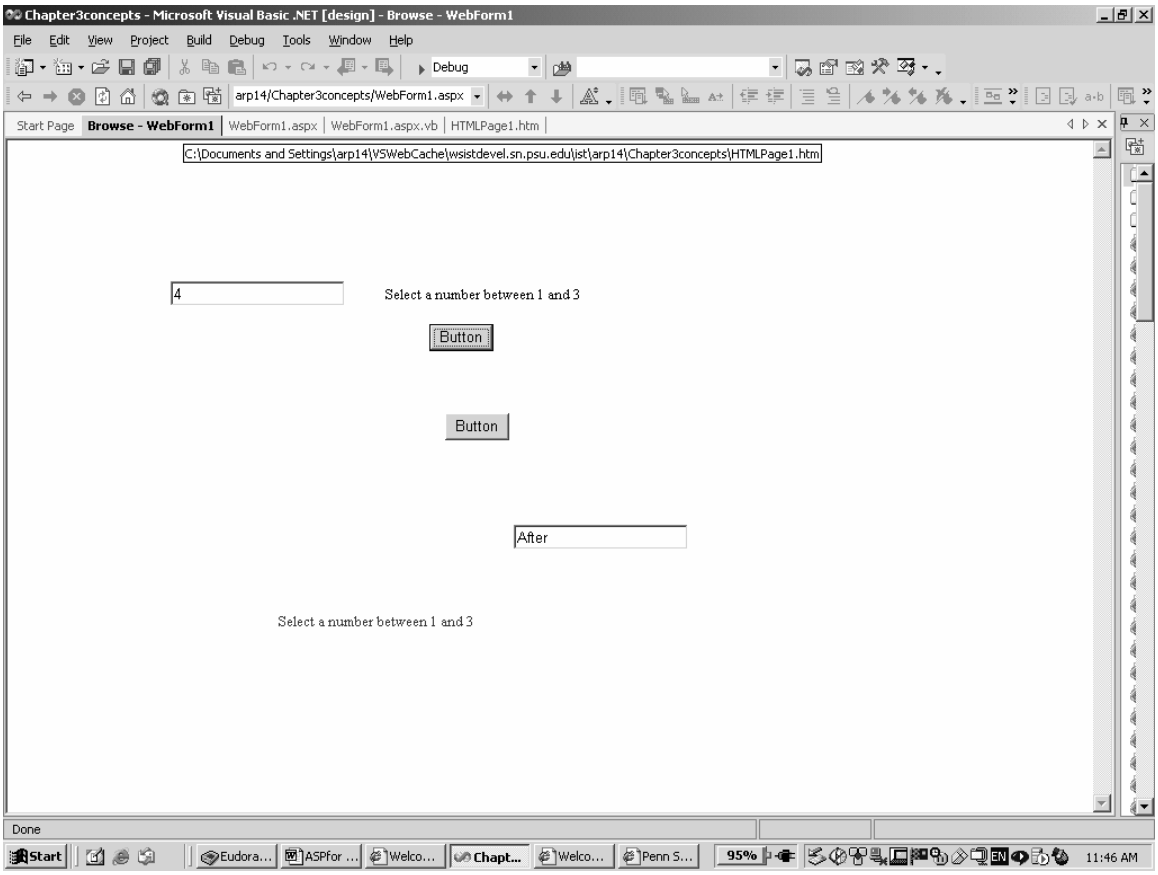


Figure 3 Case Statement and Validation Control

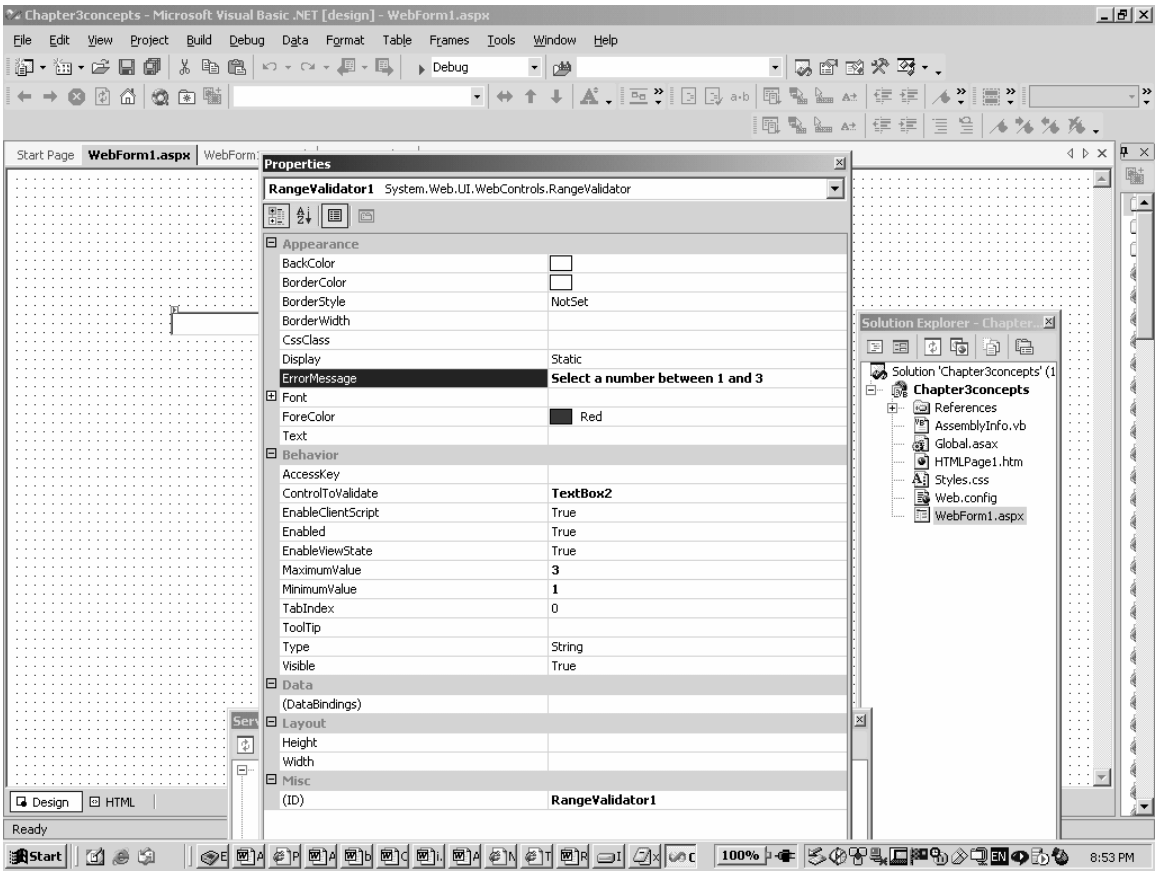


Figure 4 Validation Control

For the use of XML within the Visual Studio environment, a simple XML program and an XSL transformation file were required. The requirement was to develop an XML file, and a XSL format file and use the XML control within VS to relate the two. This is accomplished through the DocumentSource and TransformSource properties of the XML control. This is illustrated in Figure 3. and the displayed XML file as transformed by XSL is Figure 2. The actual XSL code prepared as well as the XML code are shown in Code Sample 3 in the Appendix.

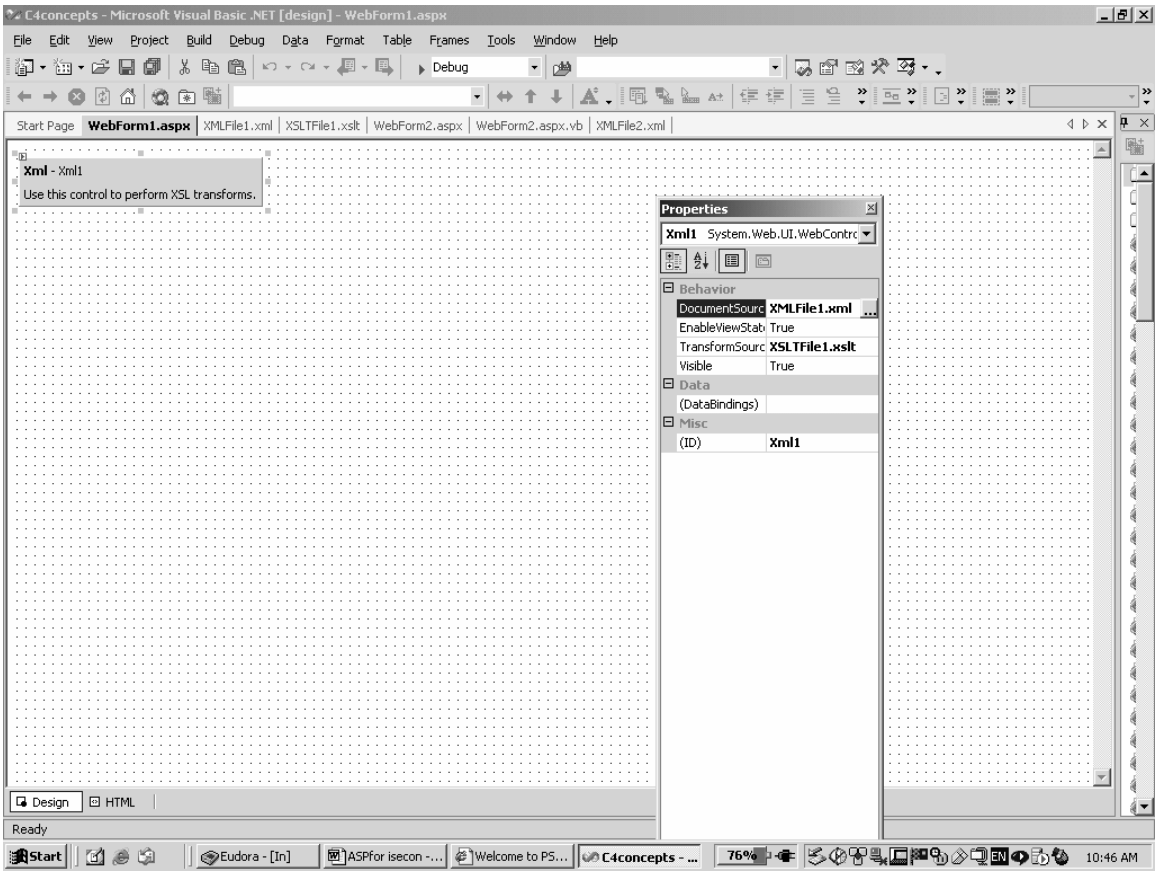


Figure 5 Displayed XML exercise

To illustrate the calendar control and Adrotator controls a simple WebForm was prepared. The AdRotator is shown without pictures due to copyright concerns. The calendar control was used with a simple text box and was programmed to display Tax Day in the box if April 15 was clicked (Figure 6). Again these are simple examples to strip away all excess complexity and allow everyone to understand the environment and the controls only. The calendar control code is shown in the Appendix as Code Sample 4.

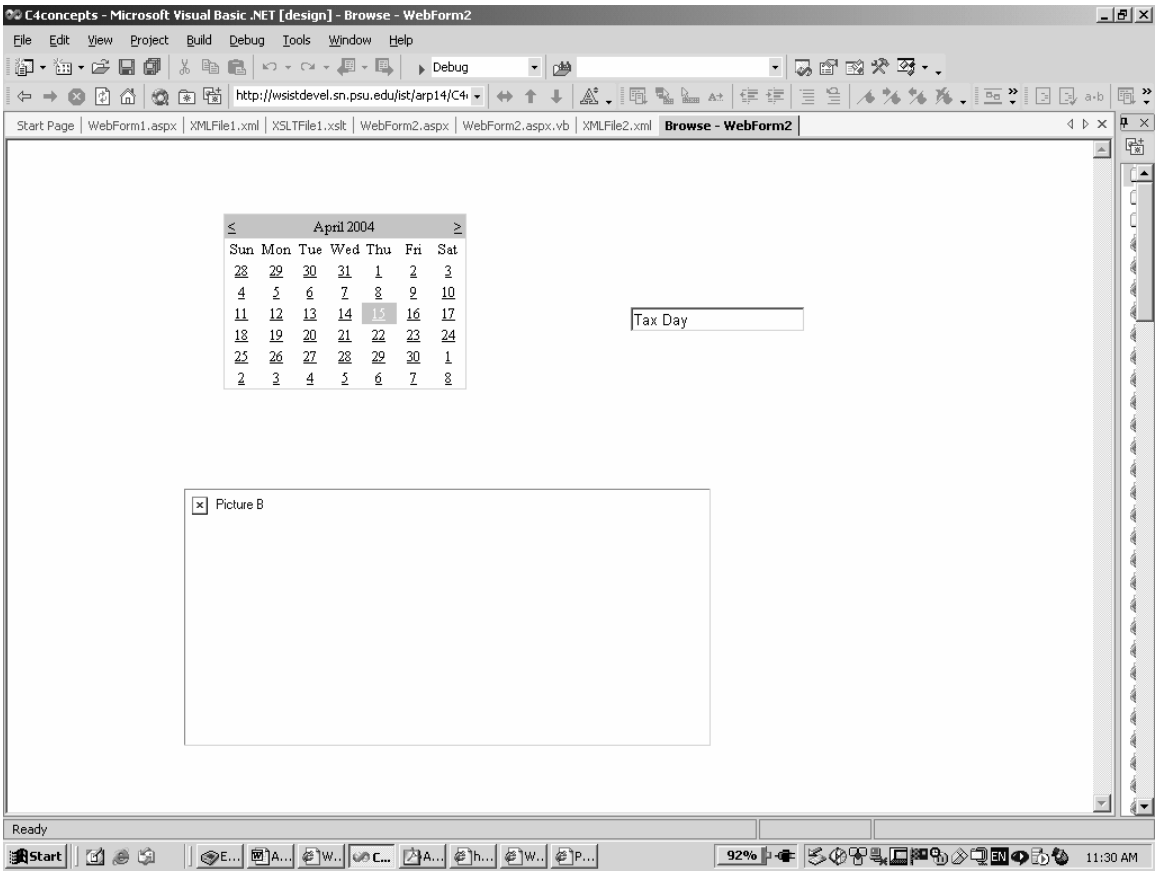


Figure 6 Calendar Control and AdRotator Control exercise

Golf program

As a final "programming" assignment in the IST 420 course, the author developed a problem which incorporated may of the concepts reviewed – a golf handicap program.

Requirements of The Golf program

1. ASP.NET web application
2. Main web page
3. Links to at least three options:
a. Calendar control that results in display of current tournament based on date selected
b. Calculation of handicap differential based on course, score, slope rating
c. Calculation of handicap index based on number of valid scores

4. Use any of the application languages, VB.NET, C++, J#, C#

5. Use an array for valid scores

To gather information about how to accomplish this, you may discuss with class members but all programs must be your work.

You will need to find out PGA rules for handicaps. These are available on the Internet.

A word document with screen prints of input and output as well as copied and pasted application code is due April 19 prior to class in ANGEL drop box. Code on the server is frozen at this time as well.

Sample code is shown in the Appendix as Code Sample 5. Screen shots from the Golf Program are shown at the end of the Appendix.

ASP.NET Results
The overall assessment of the ASP.NET exercises including text and in-class was mixed.
The reuslts of the ASP.NET assignments survey were:

Q. What is your overall rating of this exercise?

0 - (1) 1=very poor

3 - (2) 2=poor

5 - (3) 3=average

3 - (4) 4=good

2 - (5) 5=very good

Average Response: 3.3 (3=average)
Responses: 13-Valid, 0-Blank, 13-Total

Some of the negative responses upon review were due to many of the technical difficulties that we had during the class. These technical issues included switching from localhost to a separate server, logon authentication problems, and debugging difficulties with MS Visual Studio. All issues were resolved by the end of class but appear to have affected some results. The author expects future results to be more positive.

Our web server

Due to security concerns, we were unable to use localhost server for our ASP.NET programs at our campus. As an alternative a small server was set up for us to host our server side programs. In conjunction with this we were provided access to a SQL Server database for any database issues.

4. PERSONAL OBSERVATIONS OF XML AND IDEs

There are several lessons that were learned from attempting to incorporate innovative integrative technologies into our IST curriculum. They centered primarily on proper infrastructure and proper methods.

Get the technology right early.
Perhaps the most frustrating experience on the part of users was the problems we encountered with infrastructure. The Kalata text uses localhost for the server side of most ASP.NET programs. Due to security concerns, we were unable to use localhost in our labs or classrooms and thus everything had to be translated from localhost to our webserver host. In addition, there were significant problems with authorization due to the need for FrontPage extensions. In the end, it took several weeks to properly authorize and configure all students. This comes on top of trying to learn the IDE itself. Take the time to test out all users and infrastructure well in advance of the course so that all attention and frustration can be directed only at the IDE, not the technical glitches. The students will develop an overall negative approach to the IDE if technical problems are their first experiences. First impressions are hard to break.

Keep initial concept lessons simple.
The Kalata text is an excellent text providing complex examples and innovative exercises but the examples are too complex to work through in class. As I have suggested, simple stripped-down examples of the key concepts in each chapter can be worked through by all in class. Then the progression can be made to the full examples in the text. The author is a believer that concepts need to be understood without additional overhead.

Build slowly to complexity.
The next step after the basic examples is the text walkthroughs. Then exercises at the end of the chapters provide good variations from the text examples. Only after this progression has taken place can the students' progress to an unstructured problem on their own.

Illustrate and work together initially.
As noted, the author has experienced that working together on small exercises can lead to a shared learning experience in which everyone is enriched. The interaction and feedback and difficulties of all can be shared and everyone including the instructor can learn. In addition, there is opportunity for those who are more familiar with the IDE or XML to point out their experiences. Often the students will have explored areas where event he instructor has not traveled. Most Microsoft programs including Visual Studio are complex multi-threaded programs that have multiple ways of accomplishing tasks and many hidden and undiscovered shortcuts and techniques. Encourage students to explore Visual Studio and all will be rewarded.

Incorporate problem solving and analytical research.

Instead of focusing on significant overhead code that can be automated, Visual Studio allows the programmers to focus on the business problems. You should encourage students to undertake programs that require analysis and critical thinking skills. The golf program is one simple example that encourages practical problem solving with an application that all can understand.

Expect creativity.
Programmers tend to be overly logical types who have difficulty utilizing the left side of their brains. But problems that will be faced in the real world require creativity in devising solutions as well as technical skills in solving them. Expect creative solutions to unstructured problems. Allow students some leeway to condition the left sides of their brains. The results are often pleasantly surprising.

You can never overemphasize the importance of syntax in XML.
Coming from HTML world most students tend to be lax in syntax in programming in an XML world. Certainly the Visual Studio itself has been criticized for encouraging programming laziness. But when you need to code XML syntax is paramount and many students do not comprehend the depth of debugging that can result. One student had a particularly vexing problem that turned out to be wwww instead of www in the header statement <xsl:stylesheet version ="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

Make sure you display all possible error messages in Visual Studio.
One of the problems we had in debugging our programs was the default Server Error that was received from the IDE for nearly any problem. When using a hosted webserver, make sure in the Web.config file that <customErrors mode="RemoteOnly" /> is changed to "Off", so that true error messages are displayed.

Emphasize the importance of IDEs.
Most students don't understand or care about productivity but in an era of outsourcing they should and you need to make them care. Business will only pay for essential costs and not any more. Productivity gains are significant through the use of IDEs and jobs could remain in their home location if productivity gains are achieved.

More time on Visual Basic .NET is required.
One of the last insights the author received was that even though many students had programming courses in C++ and Java, significant time still needed to be spent on the nuances of Visual Basic. Most students are still novices programming and need time to adjust to the change in language. The author assumed too much and experienced significant difficulties with arrays in Visual Basic .NET.

5. CONCLUSION

With the successful second teaching of the integrative technologies of XML and Microsoft Visual Studio on an introductory level the author looks forward to moving to a full three-tiered approach using ASP.NET. Future coursework will fully illustrate the database connection with ADO.NET and the Access and SQL Server connections. The golf program will be utilized to incorporate database storage and the calculation of a handicap index which will require storage and array sorting. These assignments will be incorporated into the second course of our IST Systems Integration sequence. The importance of XML and IDEs in systems development both now and in the future has been identified. The approach presented here provides a base level of skill in utilizing these innovative technologies.

6. REFERENCES

Abuhejleh, 2004, "XML Technologies and/or Relational Databases: A Classroom Experience." Information Systems Education Journal, *2* (19). http://isedj.org/2/19/. ISSN: 1545-679X. (Also appears in *The Proceedings of ISECON 2003: §3231.* ISSN: 1542-7382.)

Chaytor, Louise and Soleda Leung, 2003, "How to Creatively Communicate Microsoft.NET Technologies in the IT Curriculum." Proceeding of the 4th conference on Information technology education**,** pp. 168-173.

Eastmond, Dan, 2000, "Enabling student accomplishment online: an overview of factors for success in web-based distance education." Journal

of Educational Computing Research, 23 (4), pp. 343-358.

Frank, Ronald, 2002, "An Inherent Conflict in Using IDEs in Computer Language Courses." The Proceedings of ISECON 2002, v 19 (San Antonio): §221d. ISSN: 1542-7382.

Kalata, Kathleen, 2003, Introduction to ASP.NET, Course Technology, Boston, MA.

Kampherbeek, Jan, 2001, "KickStart TutorialXMLversion1.0." http://www.spiderpro.com/ebooks/kickstart_tutorial_xml.pdf

Kay, Ronald, 2004, "Quickstudy: XSL.", Computerworld, http://www.computerworld.com/developmenttopics/development/webdev/story/0,10801,92787,00.html

Kolling, Michael and John Rosenberg, 1996, "An Object-Oriented Program Development Environment for the First Programming Course." Proceedings of the 27th SIGCSE Technical Symposium on Computer Science Education, pp. 83-87.

Lim, Billy, 2002, "Teaching Web Development Technologies: Past, Present, and (Near) Future." Journal of Information Systems Education, 13 (2). pp. 117-123.

McKinney, Alfred, 2003, "A Recent Radical Graphical Approach to Programming". The Journal of Computing in Small Colleges. 18(6), pp. 28-34.

Microsoft Corporation, 2004, "HP: HP Creates Software for Unified User Experience Using Microsoft .NET Technology." http://www.microsoft.com/resources/casestudies/CaseStudy.asp?CaseStudyID=14965

Norman, Ronald and Jay Nunamaker, Jr., 1989, "Integrated Development Environments: Technological and Behavioral Productivity Perceptions." Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, pp. 996-1003.

Payne, Michael, 2001, "An Advanced Web Java Class's Hardware and Software Needs." The Proceedings of ISECON 2001, v 18 (Cincinnati): §18b.

Reich Blaize and Kay Nelson, 2003, "In Their Own Words: CIO Visions About the Future of In-House IT Organizations." The Data Base for Advances in Information Systems, 34 (4), pp. 28-44.

Reis, Charles and Robert Cartwright, 2004, "Taming a Professional IDE for the Classroom." Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education, pp. 156-160.

Ziegler, Uta and Thad Crews 1999, "An Integrated Program Development Tool for Teaching and Learning How to Program." Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education, pp. 276-280.

7. APPENDIX

Code Sample 1 XML and XSL

XSL Code

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
        <html>
        <body>
                <h2> MY XML Program </h2>
                <table border="1">
                <tr bgcolor ="#9acd32">
                        <th align="left"> Number </th>
                        <th align="left"> Manager </th>
                        <th align="left"> Name </th>
                        <th align="left"> Totalprice </th>
                </tr>
                <xsl:for-each select="sales">
                <tr>
                <td><xsl:value-of select="shop/number"/></td>
                <td><xsl:value-of select="shop/manager"/></td>
                <td><xsl:value-of select="product/name"/></td>
                <td><xsl:value-of select="product/totalprice"/></td>

                </tr>
                </xsl:for-each>
                </table>
        </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

XML Code

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="xsl3.xsl"?>
<sales>
        <shop>
                <number>
                        100
                </number>
                <manager>
                        Tom Thumb
                </manager>
        </shop>
        <product>
                <name>
                        broccoli
                </name>
                <totalprice>
```

```
                    10
                </totalprice>
        </product>
</sales>
```

Code Sample 2 Case and Validation

TextBox and Button Code

```
   Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
      TextBox1.Text = "Before"

      'Put user code to initialize the page here
   End Sub

   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
      TextBox1.Text = "After"
   End Sub
```

Case Statement and Validation Code

```
   Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
      Select Case TextBox2.Text
         Case 1
            TextBox2.Text = "Case 1"
         Case 2
            TextBox2.Text = "Case 2"

         Case 3
            TextBox2.Text = "Case 3"

      End Select
   End Sub
End Class
```

Code Sample 3 XML and XSL code

XML Code

```
<?xml version="1.0" encoding="utf-8" ?>
<NewDataSet>
        <employee>
                <ssn>111223333</ssn>
                <name>John Smith</name>
                <city>Scranton</city>
                <state>PA</state>
```

```
        </employee>
        <employee>
                <ssn>222334444</ssn>
                <name>Jane Doe</name>
                <city>Wilkes-Baree</city>
                <state>PA</state>
        </employee>
        <employee>
                <ssn>444444444</ssn>
                <name>Tom Thumb</name>
                <city>Pittston</city>
                <state>PA</state>
        </employee>
</NewDataSet>
```

XSLT

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version ="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
        <html>
        <body>
                <h2> MY XML Program </h2>
                <table border="1">
                <tr bgcolor ="#9acd32">
                        <th align="left"> SSN </th>
                        <th align="left"> Name </th>
                        <th align="left"> City </th>
                        <th align="left"> State </th>
                </tr>
                <xsl:for-each select="NewDataSet/employee">
                <tr>
                <td><xsl:value-of select="ssn"/></td>
                <td><xsl:value-of select="name"/></td>
                <td><xsl:value-of select="city"/></td>
                <td><xsl:value-of select="state"/></td>

                </tr>
                </xsl:for-each>
                </table>
        </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

Code Sample 4 Calendar Control

```
Private Sub Calendar1_SelectionChanged(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Calendar1.SelectionChanged
    If Calendar1.SelectedDate.ToShortDateString() = "4/15/2004" Then
       TextBox1.Text = "Tax Day"
    Else : TextBox1.Text = Calendar1.SelectedDate.ToShortDateString()
```

```
    End If

  End Sub
```

Code Sample 5 Golf Code

DIFFERENTIAL CODE

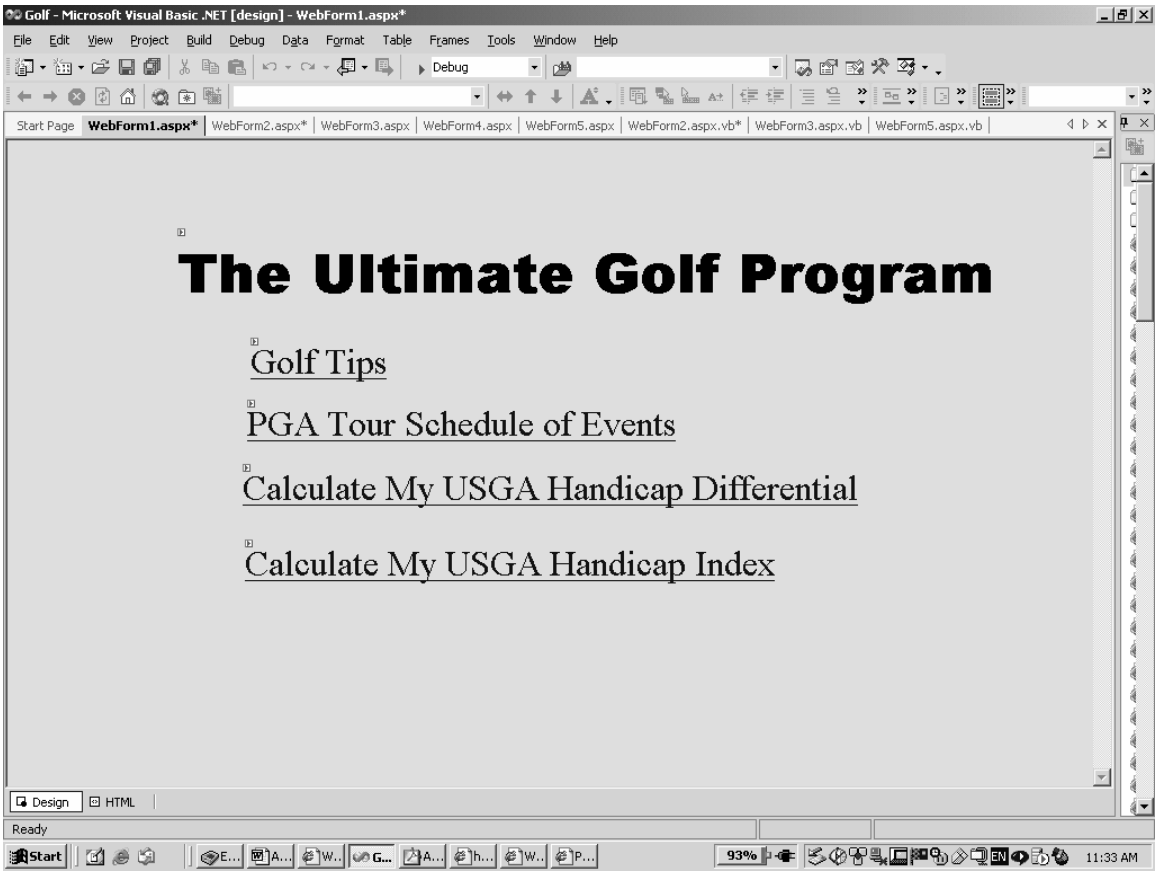
```
Imports System.Data.OleDb
Public Class WebForm5
  Inherits System.Web.UI.Page


  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
    Dim score As Single
    Dim course As Single
    Dim slope As Single
    Dim hcap As Single
    score = (TextBox1.Text())
    course = (TextBox2.Text())
    slope = (TextBox3.Text())
    hcap = ((score - course) * 113) / slope
    TextBox4.Text() = hcap
  End Sub

  Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button2.Click
    Dim CS As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\arp14\My
Documents\db5.mdb;"
    Dim cmd As New OleDbCommand("INSERT INTO hitable (course, hi)VALUES('" &
TextBox5.Text & "','" & TextBox4.Text & "')", New OleDbConnection(CS))

    cmd.Connection.Open()
    cmd.ExecuteNonQuery()
    cmd.Connection.Close()
  End Sub
End Class
```

INDEX CODE

```
Public Class WebForm3
  Inherits System.Web.UI.Page

  Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
MyBase.Load
    'Put user code to initialize the page here
  End Sub
```

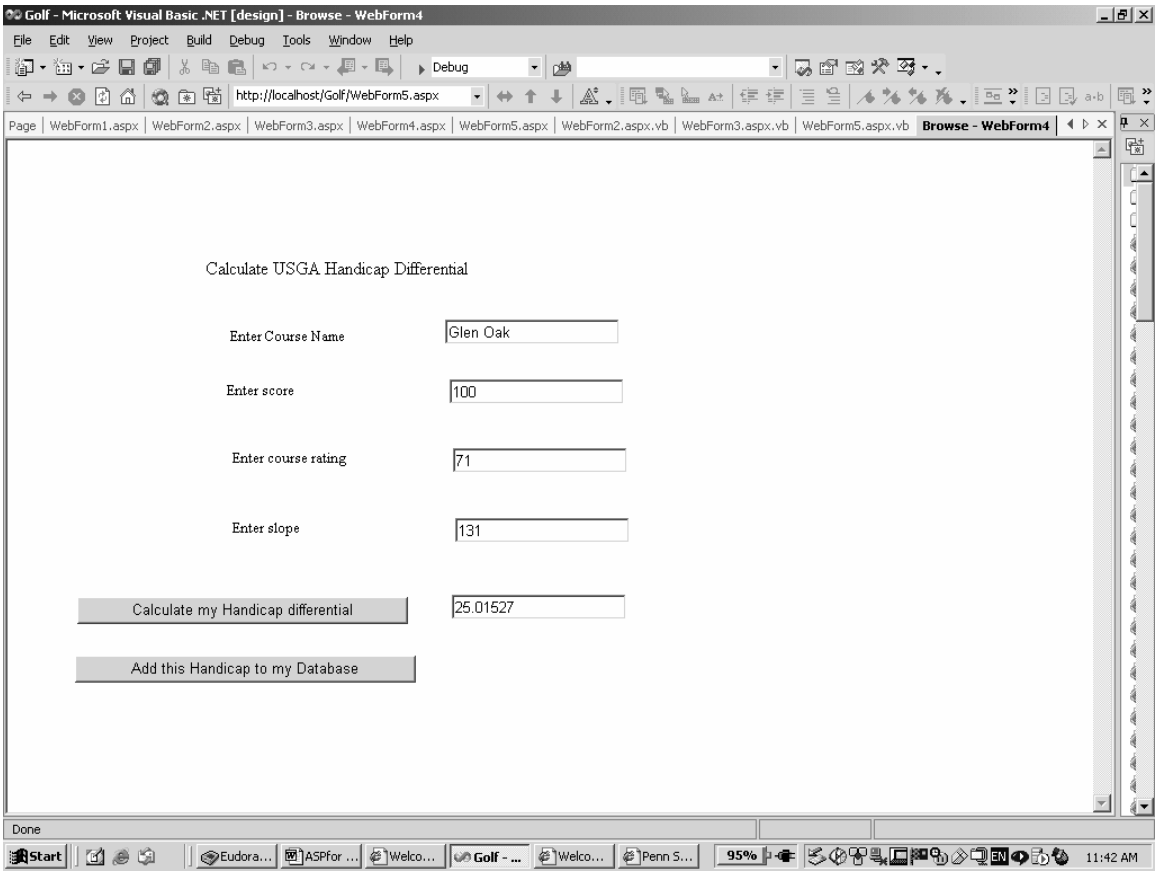
```
   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
Button1.Click
      Dim CS As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\arp14\My
Documents\db5.mdb;"
      Dim objCN As New OleDb.OleDbConnection(CS)
      objCN.Open()
      Dim mySQL As String = "select hi from hitable"
      Dim objCM As New OleDb.OleDbCommand(mySQL, objCN)
      Dim objDR As OleDb.OleDbDataReader
      objDR = objCM.ExecuteReader()
      Dim MyCat(50) As String
      Dim MyDoub(50) As Double
      Dim i As Integer
      i = 0
      While objDR.Read()
         i = i + 1
         MyCat(i) = objDR("hi")
         MyDoub(i) = CDbl(MyCat(i))
      End While

      Array.Sort(MyDoub)

      Dim total As Double

      total = 0
      Dim count1 = 0

      Select Case i
         Case 0 To 4
            TextBox1.Text = "Not enough scores"
         Case 5 To 6
            count1 = 1
         Case 7 To 8
            count1 = 2
         Case 9 To 10
            count1 = 3
         Case 11 To 12
            count1 = 4
         Case 13 To 14
            count1 = 5
         Case 15 To 16
            count1 = 6
         Case 17
            count1 = 7
         Case 18
            count1 = 8
         Case 19
            count1 = 9
         Case Is >= 20
            count1 = 10
      End Select
      Dim count2 = 0
      While count2 < count1
```

```
        count2 = count2 + 1
        total = total + (MyDoub(50 - (i - count2)))
    End While
    total = total / CDbl(count2)
    total = total * 0.96
    Dim trunc As Integer
    total = total * 10 - 0.5
    trunc = CInt(total)
    total = CDbl(trunc)
    total = total / 10
    TextBox1.Text = total
  End Sub


End Class
```



Golf Program – generic initial screen

Golf program - Golf tips



Golf program – differential screen