The Importance of C# Within the Microsoft .NET Framework

A Thesis
in TCC 402

Presented to

The Faculty of the
School of Engineering and Applied Science
University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science in Computer Science

By

Rahul Gupta

March 22, 2004

On my honor as a University student, on this assignment I have neither given nor received unauthorized aid as defined by the Honor Guidelines for Papers in TCC Courses.

Signed ______________________________

Approved ______________________________ Date: ________
Technical Advisor: Alfred Weaver

Approved ______________________________ Date: ________
TCC Advisor: Peter Norton

Preface

I wish to thank Professor Grazioli and Professor Smith of the UVA McIntire School of Commerce for teaching me C# and allowing me to use materials they developed for their classes in the C# lab modules. I also thank my technical advisor, Professor Alfred Weaver of the UVA Computer Science department for his kind guidance and helpful advice. I would also like to thank the students in Professor Weaver’s CS551 class for their helpful evaluations. Finally, I would like to thank my TCC advisor, Professor Norton, for his guidance on this paper and for imparting the wisdom of economic writing to me.

Table of Contents

Glossary of Terms

Active Server Pages .NET (ASP.NET) - .NET language that makes complex web-enabled applications. Using ASP.NET developer can create highly-functional websites with the look and feel of MS-Windows applications.

byte code - intermediate format that Java compiles code into, which is then run on the Virtual Machine of each distinct machine operating system.

classes – code that defines objects in object oriented languages such as C++, Java, and C#.

Common Language Runtime (CLR) – facilitates *language independence* by bringing together code written in different programming languages and converting the code to *MSIL*.

compile – When software developer instructions on converted into instructions the machine can understand.

Framework Class Library (FCL) - a large library of classes. This allows the programmer to skip implementation of a variety of major functions via *reusable* components.

garbage collection - freeing unused memory for use by the program later. Implemented in Java and C#.

Graphical User Interface (GUI) – the interface that users interact with that facilitates the use of applications.

interpret – when the code is executed line-by-line at the program run-time, rather than all at once.

language independent – when programmers who are comfortable with different programming languages can contribute to the same software development effort by writing code in their favored languages.

Microsoft Intermediate Language (MSIL) – intermediate Microsoft language (similar to Java's *byte code*) that allows portability among various operating systems, programming languages, and features such as memory management and security.

object-oriented – languages that extend the *von Neumann* structure by adding support for structured objects. Objects represent a set of values and the functions or computations that can change those values.

platform independence - compiling code for use on machines of any architecture, such as the popular UNIX operating system.

reuse – when code is reused rather than recreated, e.g. using Web services.

side-effects – occur when it is not completely clear how statements affect the rest of the program.

Virtual Machine (VM) - code interpreter customized for every machine architecture and operating system. Java uses a Virtual Machine.

von Neumann languages - languages that are based on statements that change values in memory. Described as computing with *side-effects.*

Abstract

The Microsoft .NET software platform integrates various aspects of software development. In .NET applications can be written in a variety of programming languages and can be either stand-alone or web-based. A lab module for CS 453 Electronic Commerce Technologies was developed that instructs students on how to develop complex e-commerce websites using the programming language C# within the Microsoft .NET framework. This project and thesis provided insight into the power of C# and the .NET framework. Student evaluations were constructive and positive overall. There were many good suggestions for expansion of the C# lab module.

Chapter 1: Introduction

1.1 Background

The Microsoft .NET software platform integrates various aspects of software development. The .NET framework is installed on all machines that contain applications designed in the Visual Studio .NET Integrated Development Environment (IDE). This IDE allows software developers to create fully deployable applications. These applications can be written in a variety of programming languages and can be either stand-alone or web-based. The programming language C# (pronounced see-sharp) is a central language in .NET that evolved from the languages Java and C++. [3]

1.2 Problem Definition

C# and the Microsoft .NET framework has advantages over older languages that make it worth learning. Because there is very little support for C# and the .NET framework in the University of Virginia School of Engineering and Applied Science, few UVA students graduate with basic knowledge of this technology. This problem must be rectified so UVA graduates can be effective in the IT industry.

To address this problem, in this project a lab module for CS 453 Electronic Commerce Technologies was developed. It instructs students on how to develop complex e-commerce websites using C# within the Microsoft .NET framework. The lab module will introduce the unique features of C# and the .NET framework. Because the existing structure of CS 453 is being used, students may start at different levels depending on their knowledge of the languages and their skills. These difficulty levels are Beginner, Intermediate and Advanced. The Beginner level introduces the programming languages C# and ASP.NET and outlines the role that the Microsoft .NET framework plays in programming with these languages. The “Intermediate” and “Advanced” levels include more complex examples. This is a full breakdown of each lab module:

Beginner

- Getting Started: What is .NET and how does C# fit into the .NET framework?
- Unique functions/objects/syntax
- GUI elements/form designer

Intermediate

- Interacting with databases using C# and SQL
- Interacting with databases using ADO .NET
- Integration of C# with Excel

Advanced

- Creation of dynamic controls
- Integration of multiple languages in .NET
- Software reuse through Web services.

1.3 Rationale

The Microsoft .NET framework has major advantages over previous programming languages and environments. Applications written in .NET may be in any of several different programming languages. This functionality is what provides the .NET framework with *language interoperability*. Microsoft designed the .NET architecture to run on multiple platforms via the *.NET Framework*, although currently only MS-

Windows support exists. The framework manages the execution of applications and Web services, and provides many more functionalities including security enforcement and memory management. Programmers no longer face the memory management problems that plagued previous software development environments [2].

Because of these advantages, corporations are beginning to embrace .NET. They will need graduates who know how to use it. According to Dan Kasun of Microsoft's financial-services group, with .NET "development cycles can be reduced so a product that would normally take six months to develop could go out in one or two months. New standards for security, routing, reliable messaging and workflow are being implemented" [7]. University of Virginia students should therefore know how to apply the unique features of this technology.

1.4 Structure of Thesis

First, this thesis reviews literature on C#'s origins, and on the motivations for the creation of this language and framework. The evolution of the languages and programming paradigms from C++ and Java to C# is traced. The specific features of C# and the .NET framework are then examined. These features include language independence, security framework, memory management, web services, and database access. The thesis then methods used to derive the labs and the infrastructure problems encountered. Finally, the thesis concludes by presenting and analyzing the evaluations given by Computer Science students.

Chapter 2: Review of Relevant Literature

It is best to explain the importance of C# and .NET to the UVA Computer Science department by examining the evolution of programming languages leading up to C#. Please refer to the figure in Appendix 1 throughout the discussion.

2.1 Object Oriented Languages

Object-oriented languages evolved from *von Neumann languages,* which include C, Fortran, and Pascal. The von Neumann languages are based on statements that change values in memory, thus they are described as computing with *side-effects*. Object-oriented languages extend this von Neumann structure by adding support for structured objects. Each object represents a set of values and the functions or computations that can change them [5].

2.2 C++: An Object Oriented Language

C++ is the most widely used object-oriented language. Java and C# are descended from it [3]. Bjarne Stroustrup developed C++ at AT&T in the early 1980s. Stroustrup sought to combine two existing languages, C and Simula67. C is a von Neumann language that AT&T developed in the process of creating the UNIX operating system in the early 1970s [1]. The Norwegian Computing Centre designed Simula67, a language developed specifically for simulations [5]. To make C++, Stroustrup took almost all the features of C and added Simula67's ability to define objects. Stroustrup thus met his goals of creating a language structured around objects [1].

C++ was the most successful of the object-oriented languages, yet it lacked some useful features, including automatic *garbage collection* (freeing unused memory for use by the program later) and *platform independence* (*compiling* code for use on machines of any architecture, such as the popular UNIX operating system). Sun Microsystems addressed these shortcomings in the early 1990s through the development of Java [5], an object-oriented language now used widely.

2.3 Java: The Successor of C++ and Predecessor of C#

Java's *Virtual Machine* (VM) makes it platform independent. This virtual machine is a code interpreter customized for every machine architecture and operating system. Programmers need not worry about whether users will run their code on a Microsoft Windows operating system, or whether they will use Apple's Macintosh operating system because the users will install a VM customized for their operating system. This VM will automatically *interpret* the code, which is in an intermediate *byte code* format, properly on each distinct machine. Because Java performs garbage collection, developers no longer have to worry about freeing used memory or dealing with memory leaks. Java also simplifies *Graphical User Interface (GUI*) creation with its numerous GUI-focused *classes*. In Java's predecessor, C++, it is arguably much harder to create GUIs. Yet although Java solves several major deficiencies of C++, it has its own share of problems. Java VM must interpret intermediate byte code on the fly each time the users run the code, slowing execution speed. To correct this shortcoming, Microsoft designed C# and the .NET framework [5].

2.4 The .NET Architecture and C# Programming Language

Microsoft launched the .NET initiative in June 2000. Microsoft envisioned .NET as completely platform independent, like Java, and *language independent*. This means that programmers who are comfortable with different .NET-supported languages like C# can contribute to the same software development effort by writing code in their favored languages. .NET is geared towards software *reuse* using Web services. Web services are simply applications usually made for businesses that can be used over the World Wide Web [2]. For example, if Amazon.com wanted to create an automated way for Wachovia to pull financial information from its database, Amazon.com could develop a Web service for Wachovia to use. This Web service would then work for any other companies that require the same financial information from Amazon.com. Thus, the lab modules cover the creation and use of Web services extensively because this portability concept is central in the .NET framework.

The framework manages the execution of applications and Web services. It also provides many more functionalities including security enforcement and memory management. Thus, like Java, C# has automatic garbage collection. Microsoft developers have also designed .NET to be platform independent like Java. The *Common Language Runtime (CLR)* accomplishes this after the .NET IDE compiles code into the *Microsoft Intermediate Language (MSIL)*. MSIL is very similar to Java's intermediate byte-code as discussed above. Thus, MSIL allows portability among various operating systems, programming languages, and features such as memory management and security. If a developer creates a program with several different languages, the compiler converts the languages to MSIL and then uses CLR to bring them together. Finally, CLR makes the code machine-specific as it compiles the code into a native machine language for a specific platform [2]. This portability means that software developers create one complex tool that can be applied with only very minor modifications in many business applications running on different computer architectures.

There are several major .NET-compliant languages including C#, Visual C++ .NET, Visual Basic .NET, JScript, Perl, etc. Regardless of the language used, programmers use the *Framework Class Library (FCL*), a large library of classes. This allows the programmer to skip implementation of a variety of major functions via the reusable components of the FCL. C# applications are also combined with Microsoft *Active Server Pages .NET (ASP.NET*) to make complex web-enabled applications. Thus, the developer can create highly functional websites with the look and feel of MS-Windows applications [2].

2.5 The .NET Architecture and the UVA CS Department

C# is a very powerful language because it runs within the .NET framework and takes the best features of its predecessors C++ and Java. Thus, it is important for UVA students to learn C# and the central concepts behind .NET. Currently the UVA CS department is switching most of its curriculum to .NET in order to demonstrate the power of C# and the .NET framework to students. An introductory C# lab module could help them succeed.

Chapter 3: Methods

This section presents the research methods I used to create the C# lab modules. I also examine the computer infrastructure problems encountered in detail.

3.1 Research

I took CS453 E-Commerce Technologies in the fall 2003 semester. The class was taught by my technical advisor Professor Alfred Weaver. As I was completing the course, I thought that C#/ASP.NET would make a helpful addition to the class and would fit well with the existing labs. Professor Weaver approved the idea and agreed to be my technical advisor. Keeping the existing CS453 structure in mind, I split the lab modules into three main categories (Beginner, Intermediate, and Advanced) so students could learn at their own pace. Several of the previous CS453 labs dealt with material similar to what I included in the C# labs: creating web applications that sit on remote servers and that make use of powerful remote databases. I thought that code examples and illustrations would be the most effective way to convey the information.

3.2 Lab Content Decisions

I decided to introduce the basics of the C# language and Microsoft Visual Studio .NET IDE in the Beginner labs (see Appendix 2), and then move to more advanced topics such as web application development and database access in the Intermediate labs (see Appendix 3), and follow up with Web Services, language interoperability, and Microsoft Excel integration in the Advanced labs (see Appendix 4). Aside from teaching the basics of the language and .NET platform, I tried to concentrate on topics that are applicable in business today. Thus, the Intermediate and Advanced labs delve into topics that I believe will keep the students' interest and give them the knowledge to succeed in creating complex business applications. In each lab module, I have introduced the topic, provided code examples and illustrations, and followed up with an exercise for the reader to complete. I structured the exercises in such a way that the major topics in the lab module are integrated together in order to help the student understand why the topic is important. The student can then access the solutions to the exercises on a remote machine that we provide.

3.3 Materials

The materials requirements for this project are relatively small. I employed previous knowledge of C# and the Visual Studio .NET IDE as well as current published literature in the field toward this project. I used PCs running Windows XP and Windows 2000 while developing the lab modules and solutions. I used the Microsoft Visual Studio.NET IDE to create the exercise solutions and to provide illustrations in the lab modules. I used the MySQL database that ITC provides to students. The Computer Science Systems staff provided a Windows 2003 Server machine to us so we can host the exercise solutions as well as collect, execute, and grade student C# projects.

3.4 Infrastructure Problems and Resolutions

The Computer Science Department Systems staff is not equipped to handle C# ASP.NET applications on remote servers. There is also no provision in the CS department for the popular databases used with C#, Oracle and Microsoft SQL Server

2000. I was at an impasse when I encountered these problems. I thought back to the database systems used in the other CS453 lab modules, and I considered using the open-source MySQL server, since ITC and the CS Department provide full support of this database management system. I came across a successful way to use MySQL databases with C#. I installed a MyODBC driver, which provides an interface between the MySQL database and the database connectivity protocols Microsoft uses in the .NET platform.

Professor Weaver and I then investigated the implementation of the lab exercises. We determined that ITC and the CS department do not have an important component, Microsoft's Internet Information Services (IIS), installed on lab computers. It was not feasible to install IIS on lab computers because the software could open many security holes in the system. Thus, students completing the lab modules will have to install needed software on their own systems and work from home rather than ITC computer labs. I will further examine the effects of this shortly in the Results and Conclusions sections.

Chapter 4: Evaluation and Conclusion

The C# Lab Modules demonstrate that Microsoft's .NET platform and C# programming language are a revolutionary educational technology.

4.1 Importance Demonstrated

To write good software, Computer Science students must learn all the essential elements of the language they will use. Much is at stake because software is essential in safety-critical systems such as those used in flight control and in nuclear plants [6].

Microsoft has provided a baseline for developing secure and reliable applications though the Common Language Runtime (CLR). The CLR manages the compilation of all code produced in the Microsoft .NET IDE and compares the code with pre-defined security policies. The Microsoft Windows operating system and the .NET framework configuration define these policies. The software developer can therefore grant specific users privileges to view, compile, and execute code. The .NET framework compares the credentials of a user attempting to access code on a remote web server against the security zone restrictions defined in Microsoft's Internet Explorer on that remote server [4].

4.2 Evaluation Techniques

To evaluate the lab modules, Professor Weaver and I surveyed students with various proficiency levels in .NET. For the survey, see Appendix 5. Most were taking Professor Weaver's CS551 (Federated Trust Networks) class. Twelve students rated modules and suggested improvements. Most students also gave positive comments about

C# and the .NET framework. Many students were pleased with the structure and content of the labs. Students also gave good suggestions for future C# lab material. I incorporated specific suggestions into the labs when possible.

4.3 Outcomes and the Future

This project resulted in a full C# lab curriculum that I disseminated in Professor Weaver's CS551 class. Along with the documents making up each major section of the curriculum, I provided solutions to the exercises in the lab modules to students. The students can view, execute, and modify the code on their personal machines.

The lab modules are now on the CS551 Federated Trust Networks toolkit page. Professor Weaver will distribute them, and previous labs at Longwood University and to the Virginia community colleges. They will also be used in CS 453 (E-commerce Technologies) at UVA. Professor Weaver's team can consider additional .NET languages for future labs, such as Visual Basic .NET, because this lab provides the necessary background on the .NET framework.

Appendices

Appendix 1

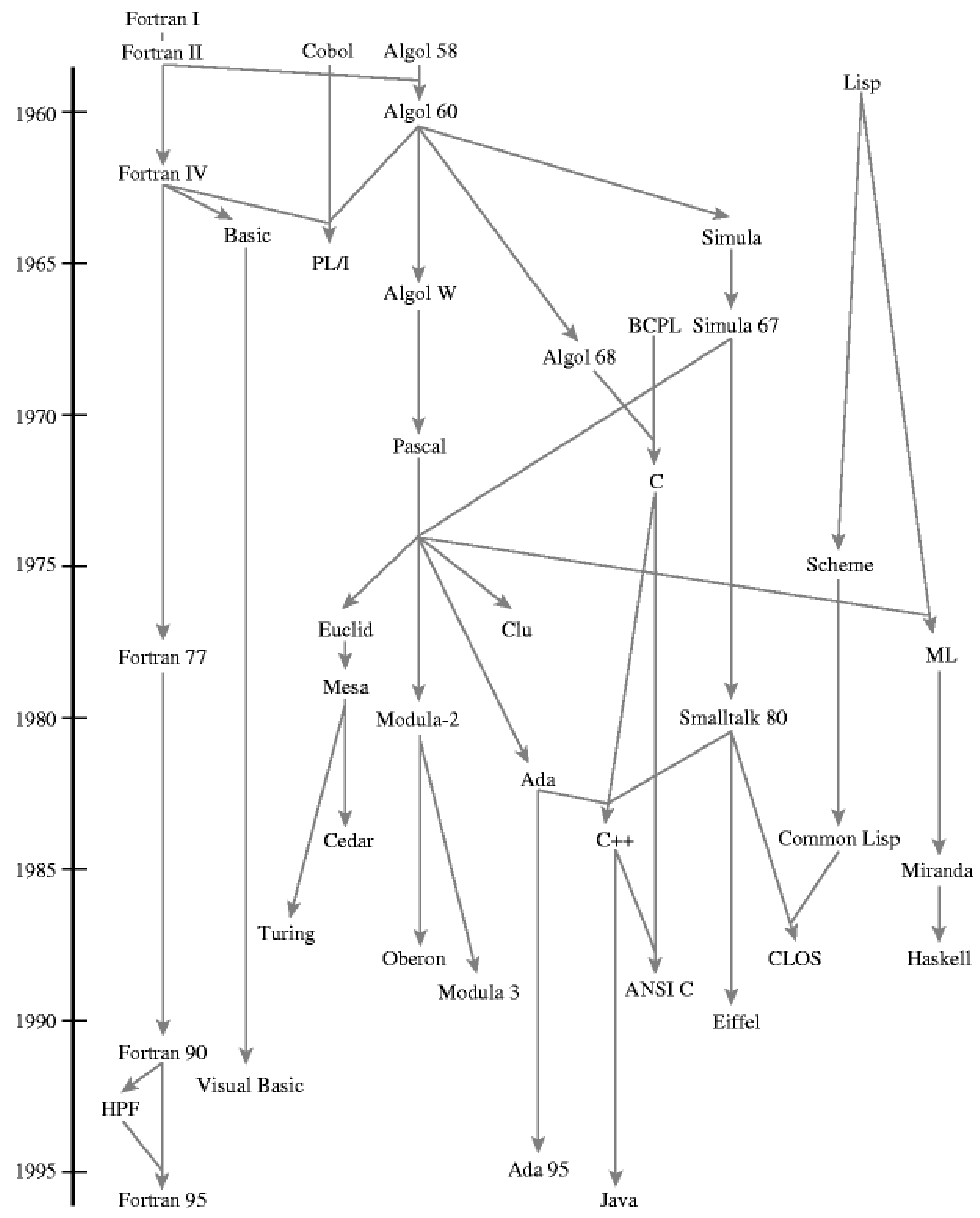


Figure 1. Evolution of programming languages [3]

Appendix 2

Beginning C#

Exercise B1 – Getting Started

What is .NET and how does C# fit into the .NET framework?
Microsoft .NET is a software platform that integrates various parts of software development and execution. The .NET framework is installed on all machines that will contain applications designed in the Visual Studio .NET Integrated Development Environment (IDE). This IDE allows software developers to create fully deployable applications. These applications can be written in a variety of programming languages (including C#) and can be either stand-alone or web-based. These applications can also be run a variety of devices including embedded devices such as cell phones.

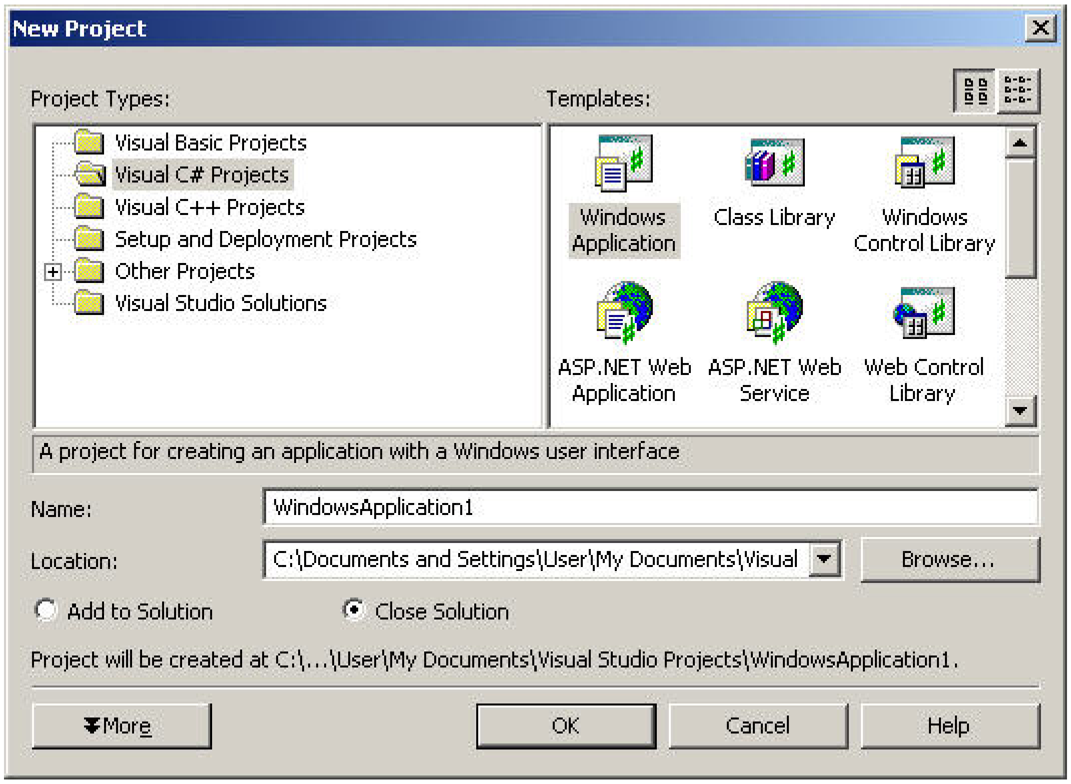
Major Features of .NET
.NET is based on a new software development model that actually allows the programmer to mix and match languages! This functionality is what provides the .NET framework with *language interoperability*. Thus, C# and Visual Basic.NET programmers, for example, can work together on the same project and successfully integrate their code. Language interoperability will be examined in detail later in the "Advanced" section of the lab. Also, the .NET architecture is designed to run on multiple platforms via the *.NET Framework*, although currently it only runs on Windows machines. The framework manages the execution of applications and Web services, and provides many more functionalities including security enforcement and memory management. Programmers no longer have to deal with various memory management issues that plagued previous software development environments. For example, when they allocate memory and then forget to return this memory to the system, it's okay because it will be taken care of by the automatic garbage collecting provided by .NET. Thus, programmers no longer have to worry about their programs consuming all available system memory, and they can instead on concentrate on the logic of their programs. One thing to note, however, is that the traditional use of pointers is no longer applied in .NET. There will be more on this later in the lab.

As discussed above, ideally .NET will be platform independent in the future. This is accomplished via the *Common Language Runtime (CLR)*, after programs are compiled in a two-step process. In the process of converting code to machine-specific instructions programs are first compiled into the *Microsoft Intermediate Language (MSIL)*. The reason the MSIL even exists is to allow portability among various operating systems, programming languages, and features such as memory management and security. Thus, this is the step in which code that is written in several different languages can be compiled together by the CLR. Finally, the code is made machine-specific as the CLR compiles it into a native machine language for a specific platform.

There are several major .NET-compliant languages including C#, Visual C++ .NET, Visual Basic .NET, JScript, Perl, etc. Regardless of the language used, programmers can make use of the Framework Class Library (FCL), which is a large library of classes. This allows the programmer to skip implementation of a variety of major functions via the reusable components of the FCL. Throughout the lab, we will be concentrating on creating C# applications combined with Microsoft Active Server Pages .NET (ASP.NET) in order to make our applications web-enabled.

Visual Studio .NET IDE Familiarization

First start up Visual Studio .NET and go to File -> New -> Project. This dialog will be shown:



A project for creating an application with a Windows user interface

Name: WindowsApplication1

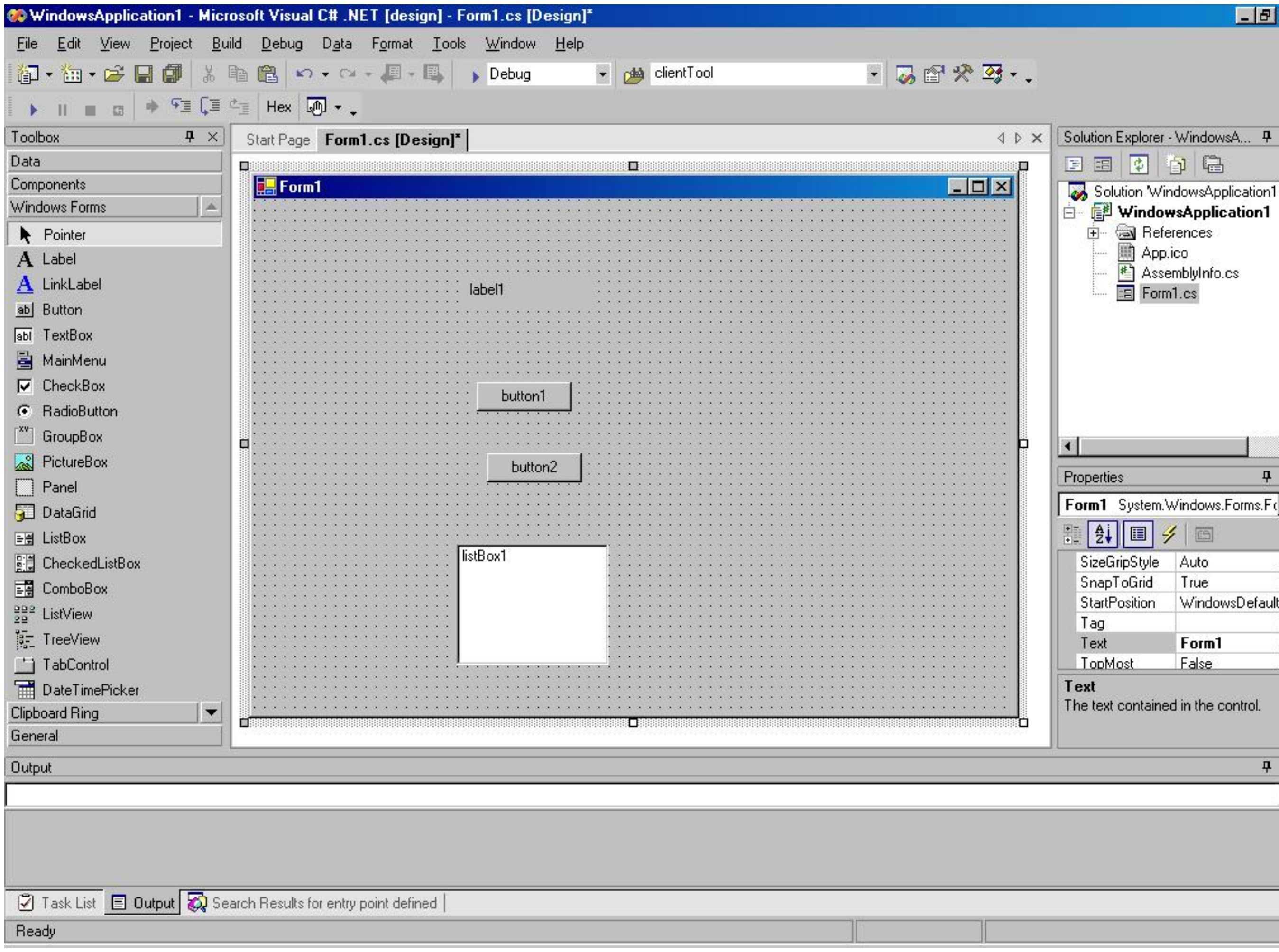
Location: C:\Documents and Settings\User\My Documents\Visual

Add to Solution [✓ Close Solution]

Project will be created at C:\...\User\My Documents\Visual Studio Projects\WindowsApplication1.

Make sure "Visual C# Projects" is selected on the left under "Project Types," and that "Windows Application" is selected on the right under "Templates." When we create web-enabled applications later in the lab we will select the option "ASP.NET Web Application" from the "Templates" options. You can optionally create a new name and location for the project. The main view of the VS.NET IDE will now appear.

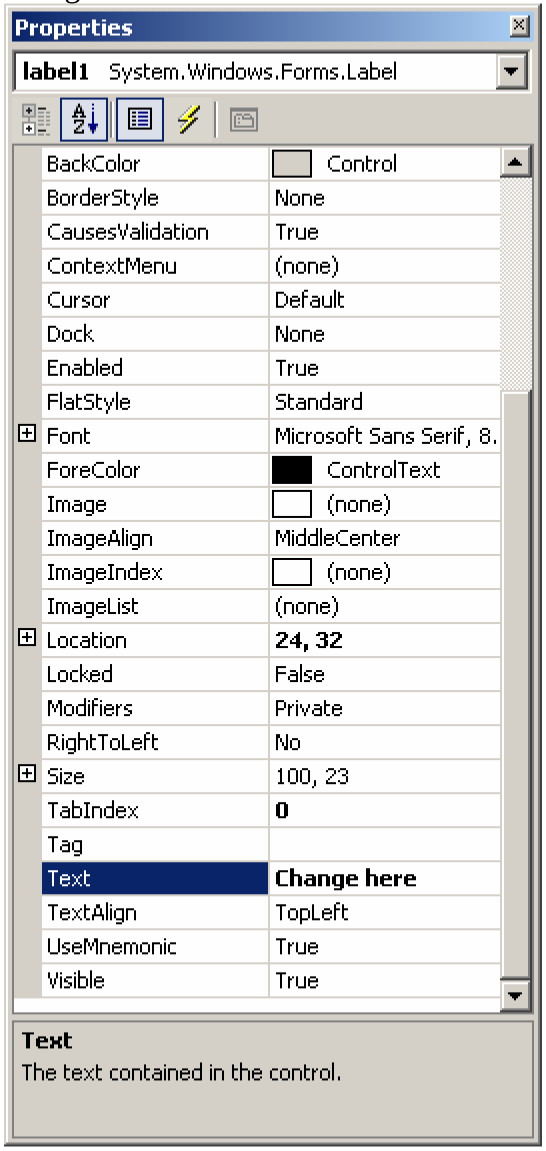
This screenshot illustrates the major components of the Visual Studio .NET (VS.NET) IDE:



The Graphical User Interface (GUI) elements available to the user are shown on the left docked toolbar. The form designer, which is at the center of the screen, is where these GUI elements are placed. This can be accomplished either by double-clicking the component in the toolbar or single-clicking the component (and keeping the mouse button pressed), and dragging the element onto the form. One can also single-click the element, and then single-click the component (and keep the mouse button pressed), and drag the cursor on the form, creating the exact location and size of the component. The details about the specific GUI elements shown will be the focus later in the lab, so for now don't worry about what each component does. On the upper right side is the "Solution Explorer". This basically contains all the files in the specific solution/project. VS.NET creates a higher level "solution," which can contain multiple "projects".

On the lower right side is the "Properties" box. This box contains all properties of the selected element. This is especially useful because the programmer will not have to create the code to change the elements' attributes later:

In here, one can modify such things as the form title, the text property of the object, the color of the forms/objects, etc. After adding a label to the form, its text property can be changed here:

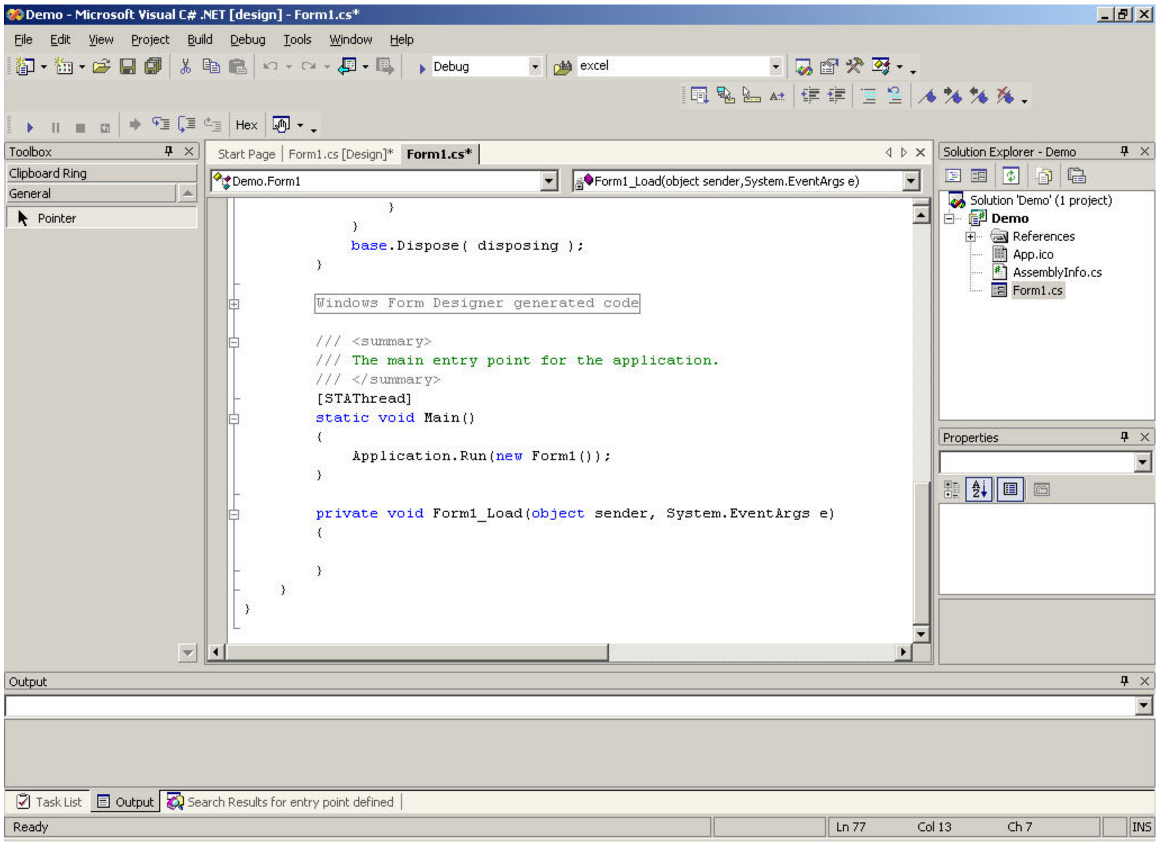


label1 System.Windows.Forms.Label

BackColor	Control
BorderStyle	None
CausesValidation	True
ContextMenu	(none)
Cursor	Default
Dock	None
Enabled	True
FlatStyle	Standard
⊞ Font	Microsoft Sans Serif, 8.
ForeColor	■ ControlText
Image	□ (none)
ImageAlign	MiddleCenter
ImageIndex	□ (none)
ImageList	(none)
⊞ Location	**24, 32**
Locked	False
Modifiers	Private
RightToLeft	No
⊞ Size	100, 23
TabIndex	**0**
Tag	
Text	**Change here**
TextAlign	TopLeft
UseMnemonic	True
Visible	True

Text
The text contained in the control.

The form or any element can also be double-clicked to access its specific code:

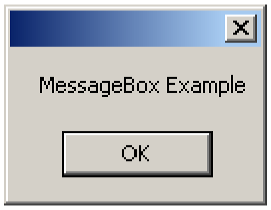


```
                    }
                }
                base.Dispose( disposing );
            }

            Windows Form Designer generated code

            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main()
            {
                Application.Run(new Form1());
            }

            private void Form1_Load(object sender, System.EventArgs e)
            {

            }
        }
    }
```

This code example shows the function for the form's load method called Form1_Load(). This is where the actual coding will take place.

To display text to the user, the following line can be added to the code in between the two brackets in Form1_Load:
MessageBox.Show("MessageBox test display");

This will result in the following message box being displayed to the user:


MessageBox Example

You may have noticed that when you typed the "." After the MessageBox, the IDE automatically showed you the options you could use. This is a very useful feature of VS .NET called *IntelliSense*. Basically, when you type a "." the IDE shows you all of the methods associated with the object immediately before the ".". This feature works on every object that has a method, and on every method that has submethods. For example, in `label1.Text.Clone` after typing the "." after "Text", more options will appear, and in this case "Clone" was selected.

Exercise

1. Open Visual Studio.NET, and create a new “Visual C# Windows Application” project. Call it “exB1”.
2. Change the background color of the form to red.
3. Change the title of the form to “Hello World Examples”
4. Add a label to the form, and make it read “Hello World!”
5. Add a MessageBox object that displays “Hello World!” when the form is loaded.

Source Used:

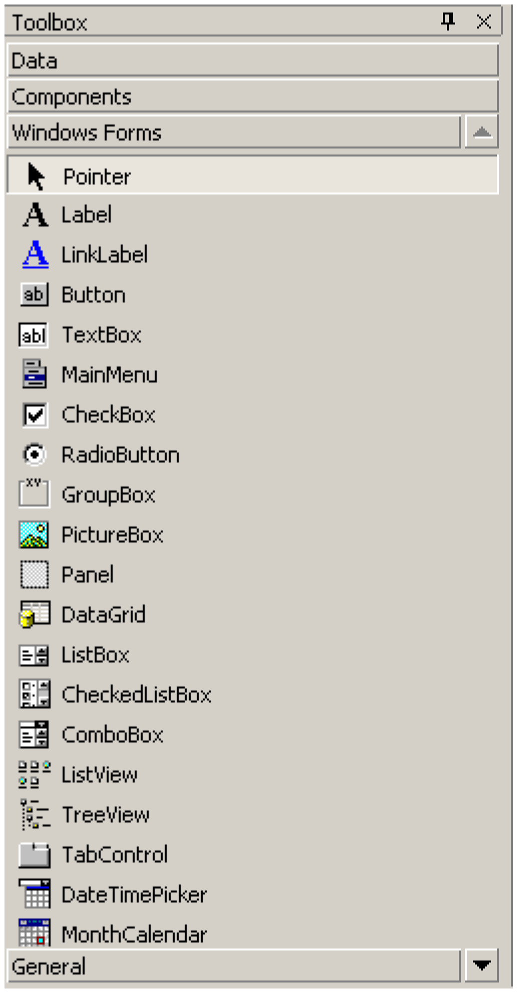
Deitel, H.M., P.J. Deitel, J. Listfield, T.R. Nieto, C. Yaeger, and M. Zlatkina. (2002). *C#: How to Program*. Upper Saddle River, New Jersey: Prentice-Hall Inc.

Beginning C#

Exercise B2 - GUI Elements/Form Designer

Overview

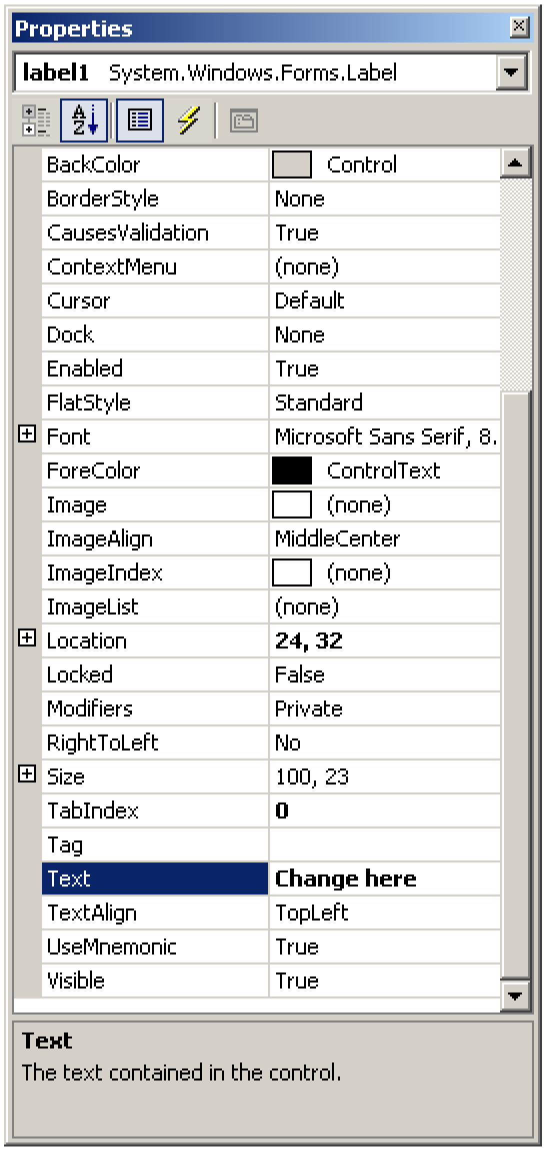
The previous section introduced the major components of the Visual Studio .NET IDE. To reiterate, the main component of the IDE is the form designer, which allows the user to create the visual components necessary for complex and aesthetically pleasing applications. These components are accessible via the "Toolbox" in the IDE:



Once the components are placed on the form using the instructions from the previous section, the attributes/properties of these components can be changed by using either the "Properties" box in the IDE or through code.

As a simple example, this is how you change the text property of a label (the text that is displayed to the user within the label).

Through the "Properties" box:



BackColor	▭ Control
BorderStyle	None
CausesValidation	True
ContextMenu	(none)
Cursor	Default
Dock	None
Enabled	True
FlatStyle	Standard
⊞ Font	Microsoft Sans Serif, 8.
ForeColor	■ ControlText
Image	▭ (none)
ImageAlign	MiddleCenter
ImageIndex	▭ (none)
ImageList	(none)
⊞ Location	**24, 32**
Locked	False
Modifiers	Private
RightToLeft	No
⊞ Size	100, 23
TabIndex	**0**
Tag	
Text	**Change here**
TextAlign	TopLeft
UseMnemonic	True
Visible	True

Text
The text contained in the control.

Through code:

```
        [STAThread]
        static void Main()
        {
            Application.Run(new Form1());
        }

        private void Form1_Load(object sender, System.EventArgs e)
        {
            label1.Text = "Change here";
        }
    }
```

Notice, that a default name of “label1” was given to the label that was added to the form. The next label will be called “label2,” the one after will be called “label3,” and so on. When you are working with multiple labels, these names can become very hard to keep track of. Therefore, it a good idea to specify a custom name for the object via the “Name” property. As shown above, this property can be set through either the “Properties” box in the IDE or through code. One of the most popular naming conventions is *firstNextLast* (e.g. buyNewCar).

Also, in the above code example, the Form_Load() method is displayed. This method basically describes what happens right before the form is displayed to the user. Thus a lot of code is typically run in the Form_Load method in addition to the event-handlers of buttons.

GUI elements

Now the specific properties of different GUI elements will be covered. There are some major features that they all share in common. One such feature is the “Name” property. Another widely-used feature is the “Visible” property, which allows a developer to hide specific GUI elements from the user. Also, keep in mind that Panels are very important objects because they are used to group different elements together and provide logical separation of components in applications.

We will start off talking about **Buttons**. You add buttons to a form as specified above and change the properties through the “Properties” box or through code, as with other elements. What makes buttons unique is that there is a specific event-handler associated with each button. Thus, when the button is clicked, all code that is placed inside this event-handler is executed. Specific properties of buttons that will come in handy include the ability to enable or disable a button (grey out) via the “Enabled” property, the ability to place an image on top of button via the “Image” property, and the ability to change the color of the buttons through the “BackColor” property.

Now we will talk about **Labels**. As you have seen from the previous section the main aspect of a label is its “Text” property which contains the text that will displayed to the user within the label.

Next on the agenda are **TextBoxes**. The “Text” property that is used is identical to that of the label. While this can be set via the properties, textboxes are usually assigned values for the “Text” property at run-time when the user of the program is inputting the text. Another major property of textboxes that you will learn to put to good use is its “BorderStyle.” This property will allow you to either keep your textboxes flat or give them a 3D look. Another feature that is widely used is the “Multiline” property which allows text to flow to more than one line, creating a scrollbar in the process. Also, “BackColor” comes in very handy.

Next we will talk about **RadioButtons**. Radio buttons are used primarily to allow the user to select of one of multiple objects. Any radio buttons that are in the same form and are not separated by panels are linked such that if any one radio button is selected, the other radio buttons cannot be. If groups of different radio buttons are desired so that each group has a different radio button selected, then Panels can be used to group them. The way to tell whether a particular RadioButton is selected is through the Boolean “Checked” property. It will be true if the box is checked and false otherwise.

Checkboxes, conversely, are designed so that the user can select multiple checkboxes. They are implemented similarly to RadioButtons, where several different checkboxes are created, and then multiple boxes can be selected. As with RadioButtons, the way to tell whether a particular checkbox is selected is through the Boolean “Checked” property. It will be true if the box is checked and false otherwise.

Finally, **PictureBox** objects will be discussed. These objects are used to display images in a form. The location of the image is selected through the “Image” property. Of particular importance is the “SizeMode” property. This property sets how the image will adjust to fit the size of the PictureBox element. The image within it can be centered, stretched, or automatically sized.

Exercise

1. Open Visual Studio.NET, and create a new “Visual C# Windows Application” project. Call it “exB2”.
2. Change the title of the form to “Calculator”.
3. The application should have a picture at the top (any image file will do).
4. In this form there should be two textboxes with corresponding labels that say, “Enter the first integer number” and “Enter the second integer number”.
5. There should also be a set of four radio buttons that allow the user to select “Add”, “Subtract”, “Multiply,” or “Divide.” These radio buttons should be placed in a panel separate from all other elements in the form.
6. There should be a button that says “Calculate”, and when selected, all calculations should take place.
7. Finally, there should be a label that displays the result as well as a label above it that says “The result:”.

Beginning C#

Exercise B3 - Common Syntax

Overview

All C# code syntax and constructs that are significantly different from C++ will now be presented. These are the topics that will be covered: Namespaces, Types, Conversion/Parsing functions, Arrays, Classes, DateTime objects, String Formatting, Try/Catch exception handling statements

Namespaces:

Each of the pre-built libraries that C# uses is defined in a namespace that contains all the classes in that library.
Examples:
System.console
System.data

The using directive for namespaces is just shorthand for writing out the full location of a particular method.
Example:

```
using System;
Console.WriteLine(“abc”);
```

If the using statement was left off, the following could be specified for the WriteLine() command:

```
System.Console.WriteLine(“abc”);
```

Types:

The following primitive types are used in C#:
int, bool, float, double, char, short, long, string, decimal

Decimal, in particular, is used when a monetary value is being specified
Example:

```
decimal Change = 5.50;
```

Conversion/Parsing functions:

In order to take the value of a number a user inputted, the Parse function must be used on the desired input type.
Example:

```
int myNum;
myNum = Int32.Parse(myTextBox.Text);
```

There are also Double.Parse(), Decimal.Parse(), etc. for the different types.

Arrays:

It is worth noting that there the array objects in C# are almost exactly like arrays in C++, you cannot dynamically change the size of the arrays. ArrayList objects on the other hand are closer to C++ vectors. The Array objects can be single or multi-dimensional. This is the common syntax for declaring Array objects:

Single dimensional arrays:

```
int[] myArray;
myArray = new int[20];

Alternately:
int[] myArray = new int[20];
```

Multidimensional arrays:

```
int[,] myMultiArray;
myMultiArray = new int[2,2];
```

Alternately:

```
int[] myMultiArray = new int[2,2];
```

Just as in C++, the array index begins at 0 and myArray[2] would access the third integer in the single dimension array declared above.

ArrayList objects:

```
//creating a new ArrayList with a size of 1
ArrayList myArrayList = new ArrayList(1);

//adds new elements to the ArrayList dynamically
myArrayList.Add(“new string1”);
myArrayList.Add(“new string2”);

//removes element from the ArrayList object
myArrayList.Remove(“new string1”);
```

Classes:

There are notable changes in the way classes are created. In C#, all methods are contained within a class. And there must be one class with a Main() method. There are still global variables that can be declared inside of the class, but outside of any methods (functions) in that class. Also, all classes can optionally be put into a namespace.

These different keywords specify the accessibility of elements in the class. Each keyword is appended to the beginning of the class when it is declared and implemented (details of this are shown in the example below):
Public: means the methods and objects are accessible both inside and outside the containing class
Private: means the methods or objects are limited to only the class they are contained in
Protected: means that access to the object is limited to the containing class or types derived from the containing class
Internal: means that access is limited to this program (assembly).

Also, when an object is declared as static, this means that there will be one instance of that object even if there are multiple instances of the class that object is contained in.

Also, just as in C++, by default, value types are passed into methods by value. The way to pass a value by reference is to add a “ref” parameter modifier. If you want pass in a ref parameter without first initializing it, use the “out” modifier.

Now instead of having to specify inspectors and mutators separately, they can be created in the same function body.
Example:

```
/* It is optional to put a namespace, but it helps with
organization when there are many different classes that can
be categorized */
namespace myNamespace
{
     public class myClass
     {
          private float myPrivateObject1;
          private static decimal myPrivateObject2;

          /*static constructor in which we pass a parameter
          by reference, and another parameter that is
          allowed in without being initialized */
          static myClass(ref float p1, out decimal p2)
          {
               myPrivateObject1 = p1;
               myPrivateObject2 = p2;
          }


          //demonstrates the get/set constructs
          public int myObject
          {
               get
               {
                    return hour;
               }

               set
               {
                    hour = value;
               }
          }

          /* notice that Main() is declared as static so
          that there is only one instance of Main */
          static void Main()
          {
               myClass myInstance = new myClass();

               //implicitly makes use of the get property
               myInstance.myObject = 5;

               //implicitly makes use of the set property
               int newNum = myInstance.myObject;
          }
     }
}
```

DateTime objects:

Another thing that is worth mentioning is the DateTime object:
Example:

```
DateTime myDate = new DateTime(2003,10,4);
myTextBox.Text = myDate.ToString();
```

This will display the following the myTextBox object:
10/4/2003 12:00:00 AM

Alternately, to display just the time one could use:
myTextBox.Text = DateTime.Now.ToLongTimeString();

This will display the following the myTextBox object:
9:09:39 AM (where this is replaced by the actual time)

One can also use a similar line of code to display just the date:

```
myTextBox.Text = DateTime.Now.ToLongDateString();
```

This will display the following the myTextBox object:
Saturday, October 04, 2003

Finally this code:

```
myTextBox.Text = DateTime.Now.ToShortDateString();
```

Will display the following the myTextBox object:
10/4/2003

String Formatting:

Also worth mentioning is the special mechanism that C# contains for column alignment and special characters (String.Format)
Example:

```
temp = 104.20;
String.Format( “Number is: {0,-12:C}”, temp);
```

This will display:

```
Number is         $104.20
```

The "C" in the above code added the money formatting ($ and .). The "-12" in the above code specified the 12 spaces after the words "Number is".

The “C” above is an example of a standard numeric format string. This property is not restricted to the String.Format() method as the following code example demonstrates:

```
double myNumber = 123456789;
Console.WriteLine(myNumber.ToString("C")); //Currency
Console.WriteLine(myNumber.ToString("E")); //Scientific
Console.WriteLine(myNumber.ToString("F")); //Fixed-point
Console.WriteLine(myNumber.ToString("N")); //Number
Console.WriteLine(myNumber.ToString("P")); //Percent
```

The output:

```
$123,456,789.00
1.234568E+008
123456789.00
123,456,789.00
12,345,678,900.00%
```

Also worth mentioning is the fact that in C# all objects are derived from the class Object, which defines ToString() as a virtual method. Thus, all objects can be converted to their string representation by using myObject.ToString();

Other methods worth looking into from the String class include: Length, CopyTo, ComapreTo, Equals, IndexOf, IndexOfAny, LastIndexOf, LastIndexOfAny, Concat, Replace, ToLower, ToUpper, Trim, and ToString.

Try/Catch exception handling statements:

A very useful feature in C# is try/catch exception handling. These are exception-handling blocks that basically allow one to divert any exceptions raised by the program into another block of code that you specify. In other words, if an exception occurs, you can alert the user and allow them to pick another course of action.

Example:

```
try
{
     int myNum = 5/0;
}
catch(DivideByZeroException)
{
     MessageBox.Show(“Divide by zero error!”);
}
```

Exercise

1. Open Visual Studio.NET, and create a new “Visual C# Windows Application” project. Call it “exB3”.
2. Change the title of the form to “Scheduler”.
3. The application should have the date displayed in a label at the top in the format Month/Day/Year (e.g. 10/4/2003).
4. There should be a textbox in the middle/top of the form that will allow the user to enter information to store.
5. The form should contain two drop down lists that allow the user to select a Week (Week 1, Week2, Week3, or Week4) and a Day (Sunday through Saturday).
6. There should be two buttons at the bottom of the form labeled “Retrieve Schedule” and “Store Schedule”.
7. When the user clicks “Retrieve Schedule” the application should pull out whatever is stored for the week and day the user selected in the drop down lists
8. If the user selected nothing in the drop down lists when the “Retrieve Schedule” is pressed, then a MessageBox must be displayed informing the user to select a week and day. This should be done using a try/catch exception handling.
9. There should be a class called “Day” that handles storage of information for a particular day. This class should have a private data member for the string itself that stores the information the user entered, a constructor, and a function that determines whether the day’s activities are empty. The day is empty if the string is empty.
10. There should be another class called “Week” that contains a private data member which is an array of seven “Day” objects. This class should contain a constructor and a function that counts how many days of the week are empty.
11. If there are no activities entered for a selected week and day, then the user should be able to enter in the information in the textbox in the middle, and click on “Store Schedule”. The application should then notify the user with a MessageBox

that the information has been stored. If activities do exist for the selected week and day, the user should be notified with a MessageBox.

12. There should be a label in between the textbox and buttons that informs the user how many days are empty in a given week after the user stores and retrieves the information for a certain week/day.

Source Used:

Deitel, H.M., P.J. Deitel, J. Listfield, T.R. Nieto, C. Yaeger, and M. Zlatkina. (2002). *C#: How to Program*. Upper Saddle River, New Jersey: Prentice-Hall Inc.

Appendix 3

Intermediate C#

Exercise I1 – Creating ASP .NET webforms and introducing SQL

Overview

Now that Windows Application development in Visual Studio .NET has been explained, we will concentrate on Web Application development via ASP.NET webforms using C# in the code-behind. The pages are displayed on the web using ASP.NET, and the program logic is created using C#. This is best illustrated with the following diagram:

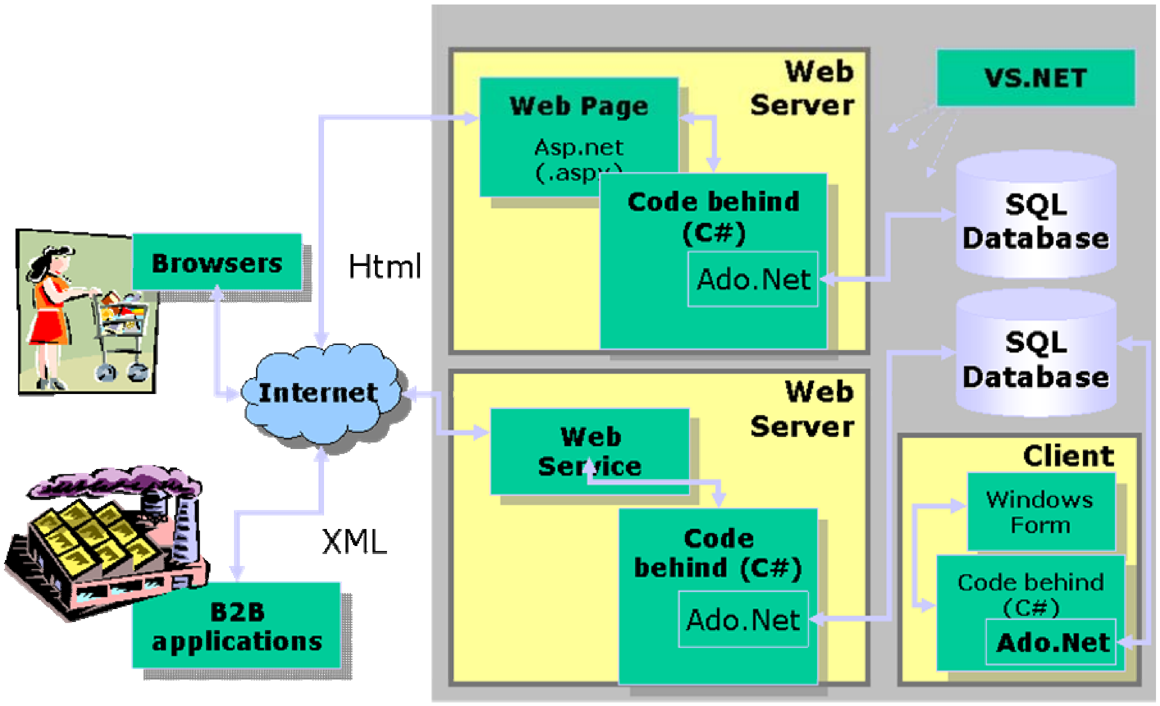


Diagram taken from http://gates.comm.virginia.edu/sg6m/Comm320/COMM320_HomePage.htm

What's going on:

- Clients' browser – sends a request for a page to a Web server, which then forwards the request to an ASP .NET server if request is for an .aspx page.
- ASP .NET server – during page's execution span, sends SQL commands to be processed by database systems, and sends HTTP requests to other Web servers.
- SQL Database server – responds to the request and provides access to shared data.
- Web server – sends the HTML generated by the page (after ASP .NET server finishes processing page).

- Client's browser – retrieves and displays this information from the Web server, and optionally sends requests to modify the information in the database tables.

MS IIS/ASP.NET Setup Instructions

For you to create and execute ASP.NET webapplications on your local machine, you must install Microsoft Internet Information Services (IIS). This comes with Windows 2000 Professional and Windows XP Professional. Refer to the following instructions to install and configure IIS
http://support.microsoft.com/default.aspx?scid=http://support.microsoft.com:80/support/kb/articles/Q312/0/73.asp&NoWebContent=1

If you install IIS after you have installed Visual Studio .NET, you will encounter the following error message when you try to create an ASP.NET project:
"Visual Studio .NET has detected that the specified Web server is not running ASP.NET

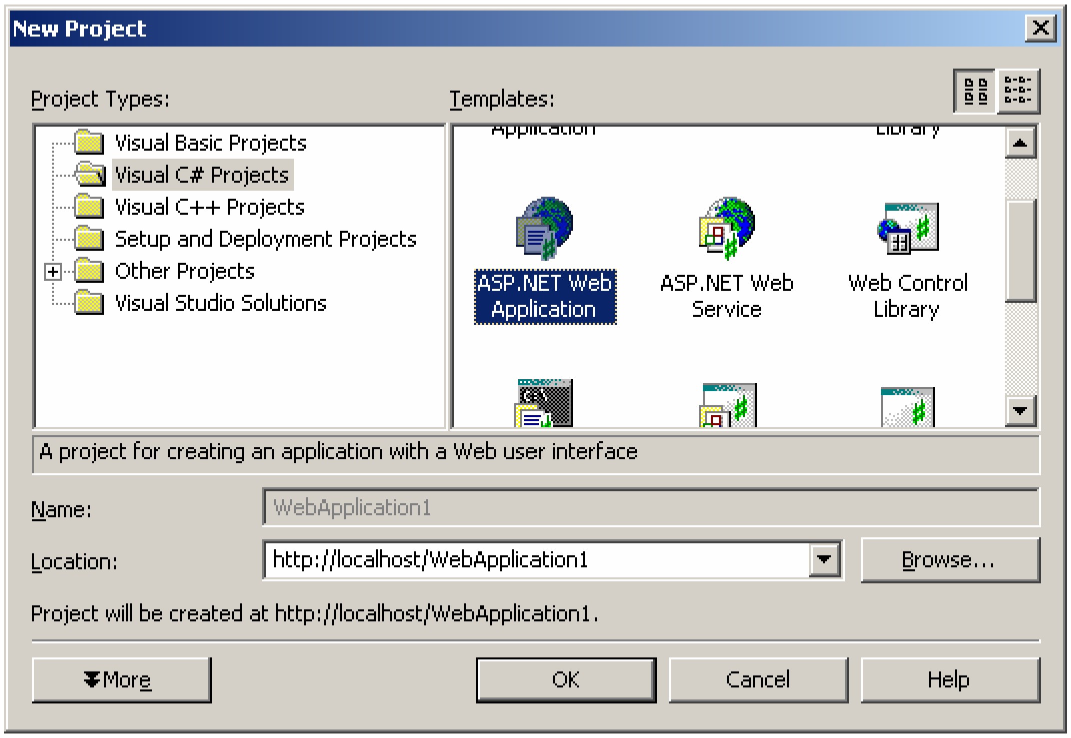
version 1.1. You will be unable to run ASP.NET Web applications or services."

If you get the above error message or have other issues running ASP.NET, refer to instructions at this URL to repair your IIS Mappings:
http://support.microsoft.com/default.aspx?scid=kb;EN-US;q306005&GSSNB=1

If you have another ASP.NET/IIS problem, refer to this FAQ:
http://www.able-consulting.com/dotnet/aspnet/aspnet_faq.htm

Creation of Webforms

To create a webform instead of a standard Windows Application, start up Visual Studio .NET and go to File -> New -> Project. This dialog will be shown:



A project for creating an application with a Web user interface

Name: WebApplication1

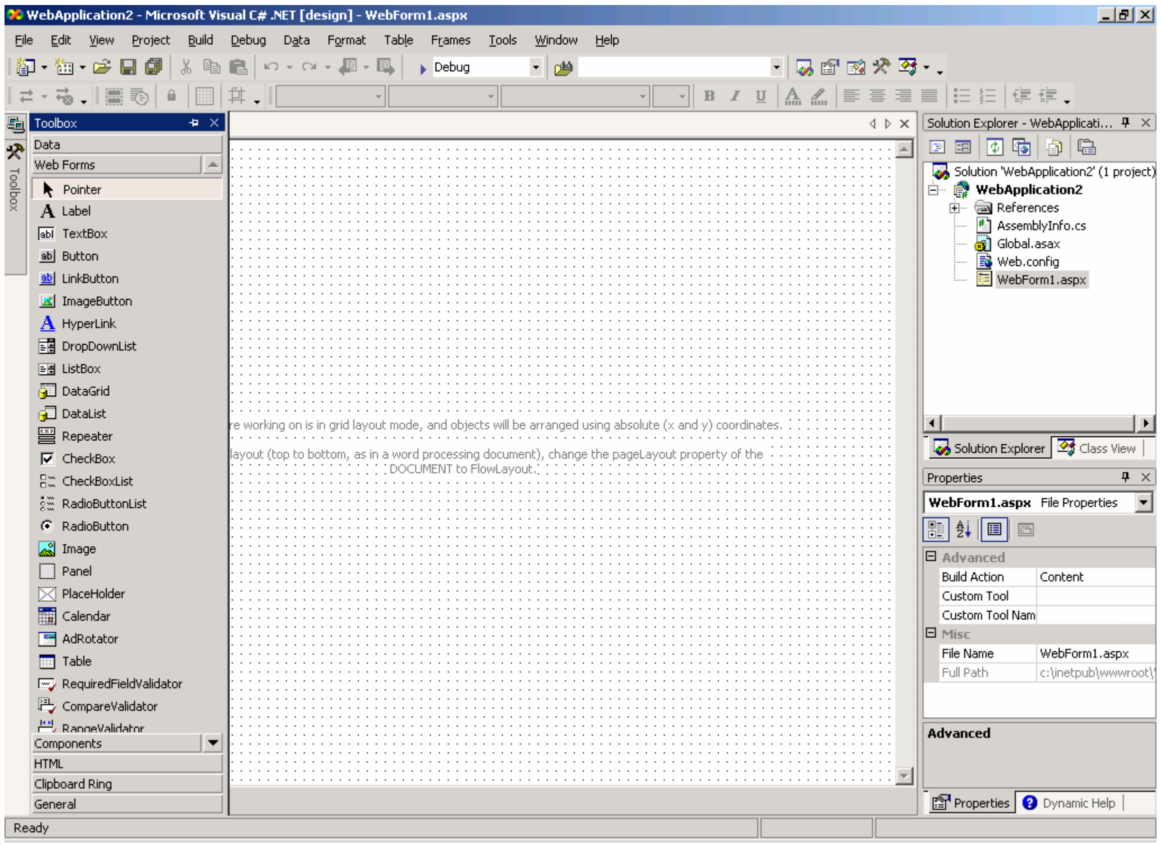
Location: http://localhost/WebApplication1

Project will be created at http://localhost/WebApplication1.

Make sure “Visual C# Projects” is selected on the left under “Project Types,” and that “ASP.NET Web Application” is selected on the right under “Templates.”

You can optionally create a new name and location (must be a server name if not “localhost”) for the project. In the “Location” dialogue, “localhost” means the local IIS server that has been installed on the machine.

Instead, if you have access to a remote server put this information in the “Location” box. The webform designer will now appear in the IDE:



re working on is in grid layout mode, and objects will be arranged using absolute (x and y) coordinates.

layout (top to bottom, as in a word processing document), change the pageLayout property of the DOCUMENT to FlowLayout.

Build Action	Content
Custom Tool	
Custom Tool Nam	
File Name	WebForm1.aspx
Full Path	c:\inetpub\wwwroot\

This is very similar to the view of the regular Windows Application form designer. What’s different is that the form is now an ASP .NET webform. When you drag components from the toolbar, they will be put into this webform. Just as before, you can double-click the components in the form in order to access code that will perform the actions. Instead of a Form_Load() method, however, all webforms have a Page_Load() method that can be accessed when the form itself is double-clicked. Any code that must run when the page is first loaded should be put here.

Also, most properties of the components in a Webform are the same as the components of a Windows Application. The major difference is the AutoPostBack attribute found in certain components in Webforms. This allows the change made to the object (e.g. dropdown list) to be sent to web server where the information stored can be processed.

We can create static pages, where the code contains all information that will be displayed in the page, but we want to eventually create powerful and complex dynamic web pages. Pages are dynamic when they have information taken from external sources, such as databases. Databases allow developers to quickly change the information being displayed without having to modify the basic layout of the page. Please refer to the diagram at the top again for an overview of how this all works. Database interaction techniques will now be explored.

Database Interaction with SQL Explained

Structured Query Language (SQL) is the standard language for defining, viewing, and manipulating information in databases. SQL is syntax-based and follows set rules about table structure. SQL is the only means by which to manage databases for almost all developers.

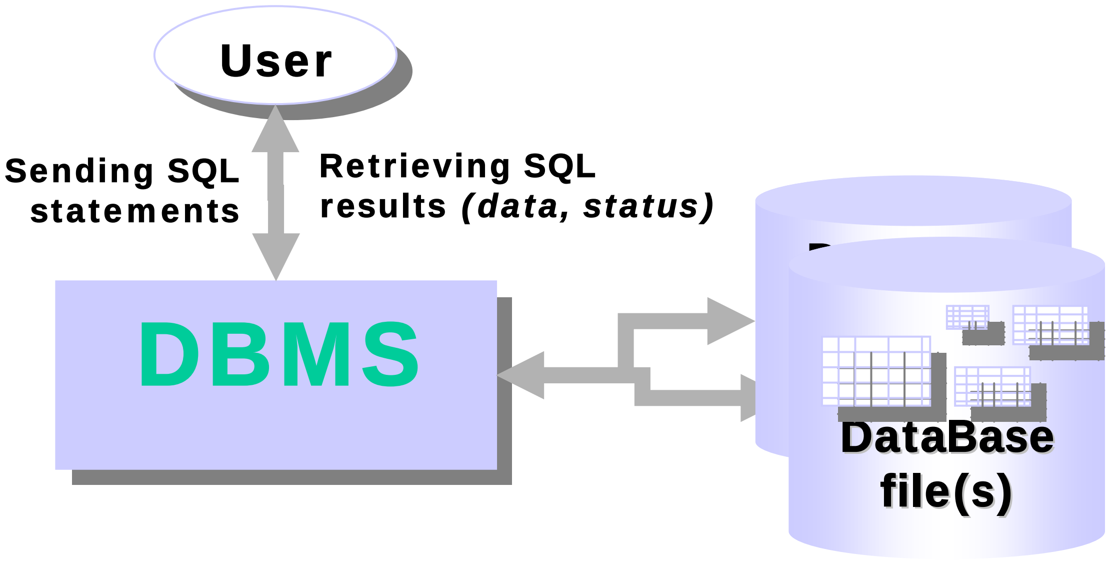


Diagram taken from http://gates.comm.virginia.edu/sg6m/Comm320/COMM320_HomePage.htm

Refer to the SQL section at http://iis.cs.virginia.edu/webweavers/ec%20labs/ for detailed information about the various SQL instructions that can be used.

Exercise

This exercise will demonstrate dynamic population of items in multiple dropdown lists based on results of previous dropdown lists. There will be a total of three dropdown lists that will be populated based on the selections of the previous lists. There will also be a label that displays the options of the user after all the dropdown lists have been selected.

1. Create a C# ASP.NET Web Application using the instructions above and call it “exI1WebApp”.
2. Create a dropdown list that allows users to select Pants, Shirts, Dresses, and Shoes. Based on the object selected different information will be accessible in the next dropdown list.
3. The next dropdown list will control color. It will display blue, black, red, and green as options for pants, shirts, and dresses. It will display only black, brown, and white as options for shoes.
4. The final dropdown list will allow the user to select from three sizes (small, medium, and large) for the pants, shirts, shoes, and dresses.
5. The user’s final selections will be displayed in a label after all options have been selected.

Sources Used:

1. COMM 320 (Professor Grazioli) website: http://gates.comm.virginia.edu/sg6m/Comm320/COMM320_HomePage.htm
2. Watson, Richard T. (2002). *Data Management: Databases and Organizations Third Edition*. New York, NY: John Wiley & Sons, Inc.

Intermediate C#

Exercise I2 – Interacting with MySQL databases using ADO .NET

Overview

Now that database access queries have been covered, we turn back to Visual Studio .NET. We will use ActiveX Data Objects (ADO) .NET components to connect with a MySQL database, execute the SQL queries, and display the data. ADO .NET is an object model that provides an API for database access.

Database Access and Setup

You have been provided with a MySQL account located on the CS servers. Go to the admin site located at this URL to create/view/modify tables:
http://cs-tl4.cs.virginia.edu/mysql/

If you would like to use the MySQL server provided free to all students by ITC, set up an account by completing the form found at:
http://dbm1.itc.virginia.edu/cgi-local/mysqlregisteruser.

Then to create, view, and modify tables on your ITC MySQL account, go to the phpMyAdmin located at this URL and login:
https://dbm1.itc.virginia.edu/phpMyAdmin/index.php3

ADO .NET Components

Depending on the database (Mysql, SQL Server 2000 or Oracle), the component names will be different, but they behave almost identically. We will use MySQL Odbc components for this discussion. The MyODBC driver must be downloaded and installed if not installed already in order to use an ODBC driver to access your ITC-provided MySQL database. Go to this URL http://www.mysql.com/downloads/api-myodbc-3.51.html. Under “Windows downloads” select the download link next to “Driver Installer”. Now run this file in order to install MyODBC. No additional configuration is required.

Overview of the components that will be used from namespace System.Data.Odbc:
OdbcConnection: Used to create the actual connection to the datasource
OdbcCommand: Represents the SQL command you wish to execute. This links to a specific OdbcConnection.
OdbcDataAdapter: Used to populate a DataSet with data from a datasource via a connection with an OdbcConnection.
OdbcDataReader: stores data from databases and allows users to work with all the data at one time or retrieve individual rows using the Read() method.

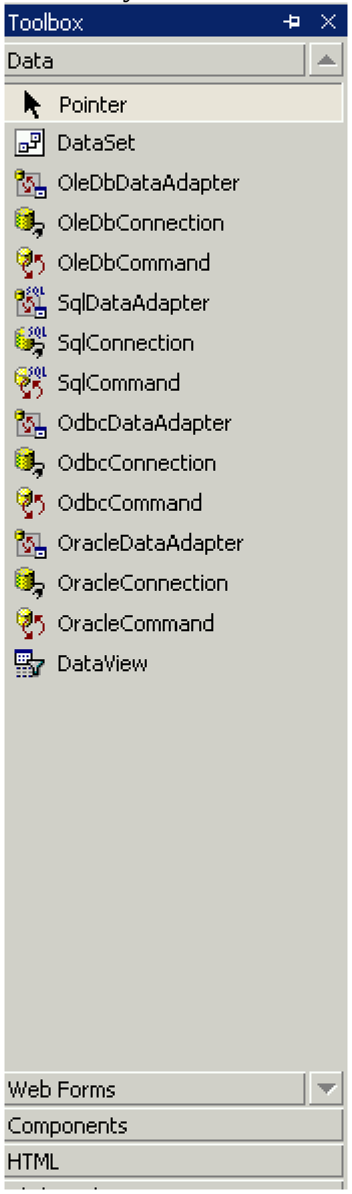
From namespace System.Data:
DataSet: This holds table information retrieved from an OdbcDataAdapter.

From System.Windows.Form:

DataGrid: This object is used in conjunction with the DataSet to display the contents of a table.

Adding and Setting Up ADO .NET Components in the Form

First a few of the components mentioned above must be added to the form. These objects are mostly included in "Data" panel in the toolbar:



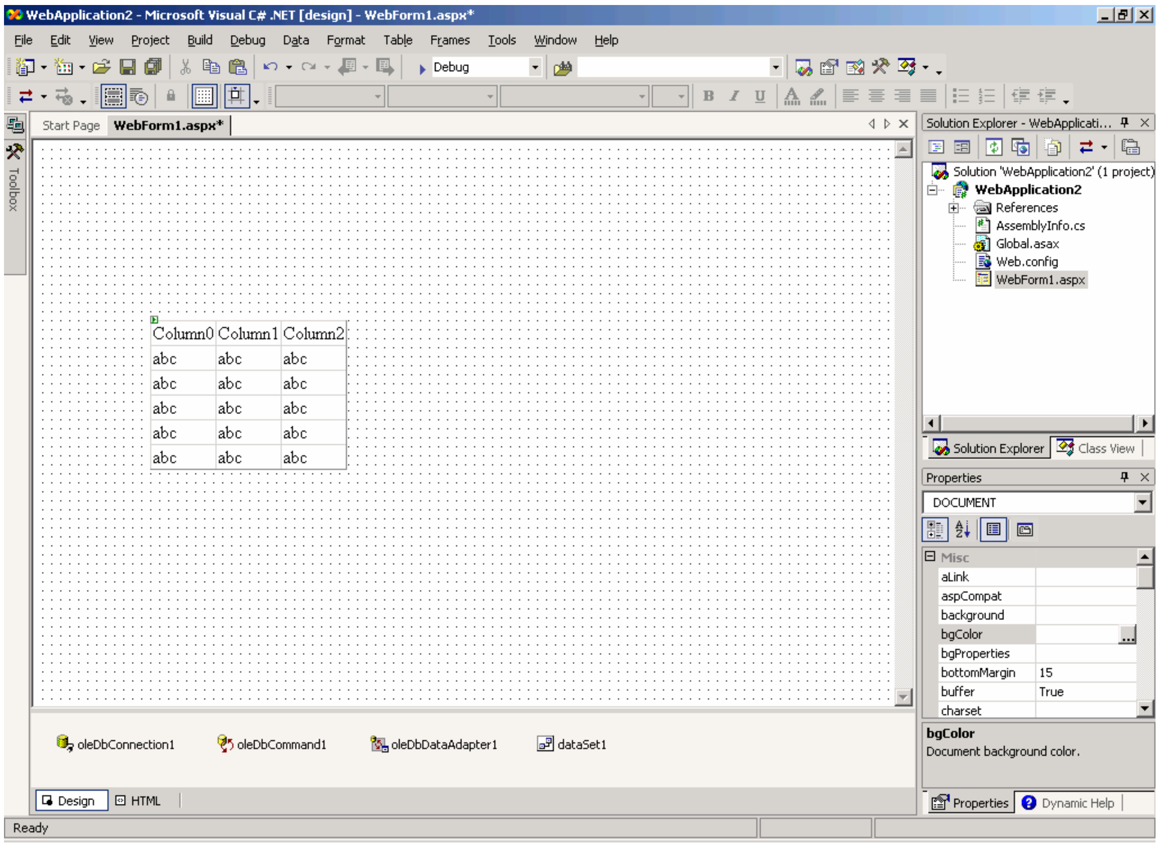
From this toolbar:

1. Add an OdbcConnection.
2. Go to the "ConnectionString" property of the OdbcConnection object and type "Stmt=;Option=131072;Pwd=*dbpw*;Driver=MySQL ODBC 3.51 Driver;Server= cs-tl4.cs.virginia.edu;Database=*dbname*;Uid=*emailid*;Port=3306". Replace the text in italics with the appropriate information (username, password, and database

name). Also, if you are using the ITC MySQL server instead of the CS one, change the server part of the string to “Server= dbm1.itc.virginia.edu”.

3. Add an OdbcCommand object to the form.
4. In the “Connection” property of the OdbcCommand object, expand the “Existing” category and select the connection that was created in step 1.
5. Set the “CommandText” of the OdbcCommand object to a SQL Select command that selects all items from your table that you want to display.
6. Add an OdbcDataAdapter to the form (press cancel when the Data Adapter Configuration Wizard shows up).
7. Set the “SelectCommand” property of the OdbcDataAdapter to the OdbcCommand that was created in step 3.
8. Add a DataSet object to the form (select “Untyped dataset” from the “Add DataSet” window that pops up).
9. Add a DataGrid object to the form (located back in “Web Forms” pane).
10. Set the “DataSource” property of the DataGrid object to the name of the DataSet that was created in step 8.

Your form will look similar to this:


WebApplication2 - Microsoft Visual C# .NET [design] - WebForm1.aspx*
File Edit View Project Build Debug Data Format Table Frames Tools Window Help
Debug
Start Page
WebForm1.aspx*
Toolbox
Solution Explorer - WebApplicati...
Solution 'WebApplication2' (1 project)
WebApplication2
References
AssemblyInfo.cs
Global.asax
Web.config
WebForm1.aspx
Solution Explorer
Class View
Properties
DOCUMENT
Misc
Design
HTML
Ready
Properties
Dynamic Help

Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

oleDbConnection1 oleDbCommand1 oleDbDataAdapter1 dataSet1

aLink	
aspCompat	
background	
bgColor	
bgProperties	
bottomMargin	15
buffer	True
charset	

bgColor
Document background color.

Now that the objects have been added to the form from the toolbar, the code can be created.

ADO .NET Code Example

Note that code was taken and modified from Professor Smith's website at http://mis1.comm.virginia.edu/dgs2m/comm327/fall2003.aspx.

The following code demonstrates the use of all these objects to display the information from the tables in a DataGrid using an OdbcDataAdapter:

```
OdbcConnection.Open();
OdbcDataAdapter.Fill(DataSet)
DataGrid.DataBind();
OdbcConnection.Close();
```

Alternatively, an OdbcDataReader object can be used instead of an OdbcDataAdapter and DataSet.

First, these are the steps to take in the form designer:

1. Create an OdbcConnection object.
2. Create an OdbcCommand object.
3. Set the "ConnectionString" in the OdbcConnection object to a new connection.
4. Set the "Connection" command in the OdbcCommand to this OdbcConnection.
5. Set the "CommandText" of the OdbcCommand object to a SQL Select command that selects all items from your table that you want to display.
6. Create a DataGrid object.

The following code demonstrates the use of an OdbcDataReader to display the information from the tables in the DataGrid:

```
OdbcConnection.Open();
System.Data.Odbc.OdbcDataReader myReader;
myReader =
OdbcCommand.ExecuteReader(System.Data.CommandBehavior.Close
Connection);
DataGrid.DataSource = myReader;
DataGrid.DataBind();
myReader.Close();
OdbcConnection.Close();
```

An OdbcDataReader can also be used to retrieve data in order to populate other elements of a form. A common element to use is a DropDownList object.

First, these are the steps to take in the form designer:

1. Create an OdbcConnection object.
2. Create an OdbcCommand object.
3. Set the "ConnectionString" in the OdbcConnection object to a new connection.
4. Set the "Connection" command in the OdbcCommand to this OdbcConnection.
5. Set the "CommandText" of the OdbcCommand object to a SQL Select command that selects all items from your table that you want to display.
6. Create a DropDownList object.

The following code demonstrates the use of an OdbcDataReader to display the information from the tables in the DropDownList:

```
OdbcConnection.Open();
System.Data.Odbc.OdbcDataReader myReader;
myReader =
OdbcCommand.ExecuteReader(System.Data.CommandBehavior.Close
Connection);
DropDownList.Items.Insert(0, “Select an item”)
int i = 1;
while (myReader.Read())
{
     DropDownList.Items.Insert(i, myReader.GetString(0));
     ++i;
}

myReader.Close();
OdbcConnection.Close();
```

Exercise

This exercise expands on the I1 Exercise by adding database interaction.

1. Create a C# ASP.NET Web Application using the instructions above and call it “exI2WebApp”.
2. Create a dropdown list that allows users to select Pants, Shirts, Dresses, and Shoes. Based on the object selected different information will be accessible in the next dropdown list.
3. The next dropdown list will control color. It will display blue, black, red, and green as options for pants, shirts, and dresses. It will display only black, brown, and white as options for shoes.
4. The final dropdown list will allow the user to select from three sizes (small, medium, and large) for the pants, shirts, shoes, and dresses.
5. The user’s final selections will be stored to a database table called “Selected” after all options have been selected and the “Store” button has been selected. The Selected table should contain 3 columns (Type, Color, Size), Type is the primary key and each column can take Varchar variables of size 20. One row will be updated every time the Store button is selected.
6. When the user clicks the “Retrieve” button, the information will be retrieved from the database table and displayed in a label.

Source Used:

1. COMM 327 (Professor Smith) website: http://mis1.comm.virginia.edu/dgs2m/comm327/fall2003.aspx
2. Deitel, H.M., P.J. Deitel, J. Listfield, T.R. Nieto, C. Yaeger, and M. Zlatkina. (2002). *C#: How to Program*. Upper Saddle River, New Jersey: Prentice-Hall Inc.

Intermediate C#

Exercise I3 – Integration of C# with Excel

Overview

One interesting feature in .NET is easy integration with Microsoft Applications. We will explore integration with Microsoft Excel because Excel is such a widely used Spreadsheet application, and databases and spreadsheets complement one another nicely. One thing to note is that we are returning to the creation of a Windows Application for this exercise, but the database connectivity described in the previous two exercises will be used.

Excel Components

The following diagram describes the components that will be used:

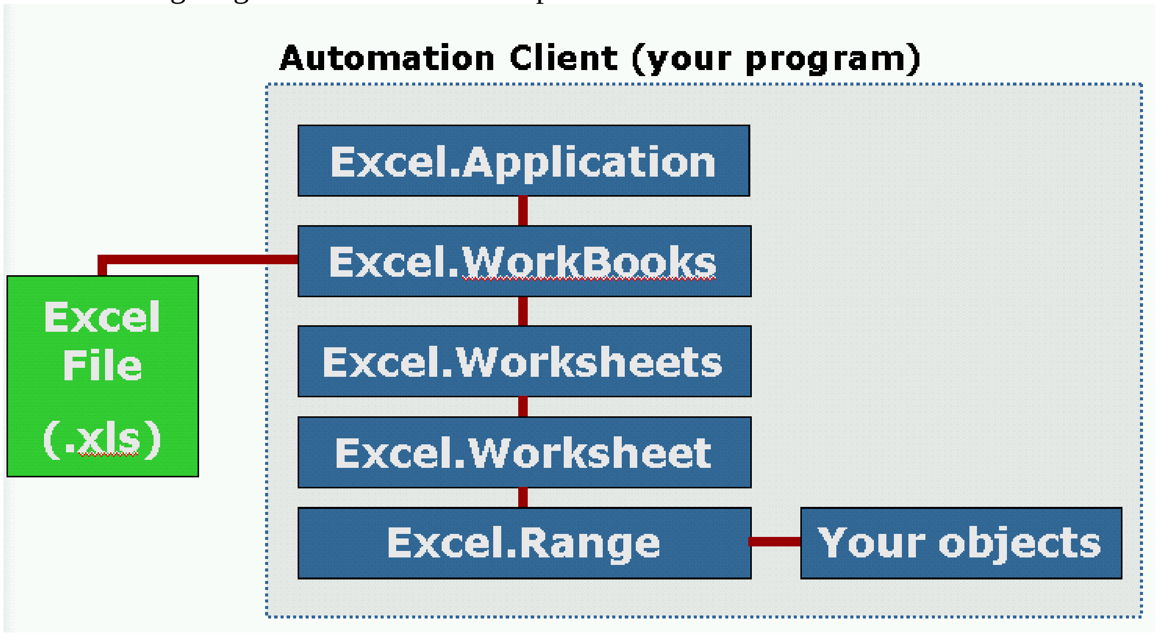


Diagram taken from http://gates.comm.virginia.edu/sg6m/Comm320/COMM320_HomePage.htm

This is an overview of how the whole system will work:

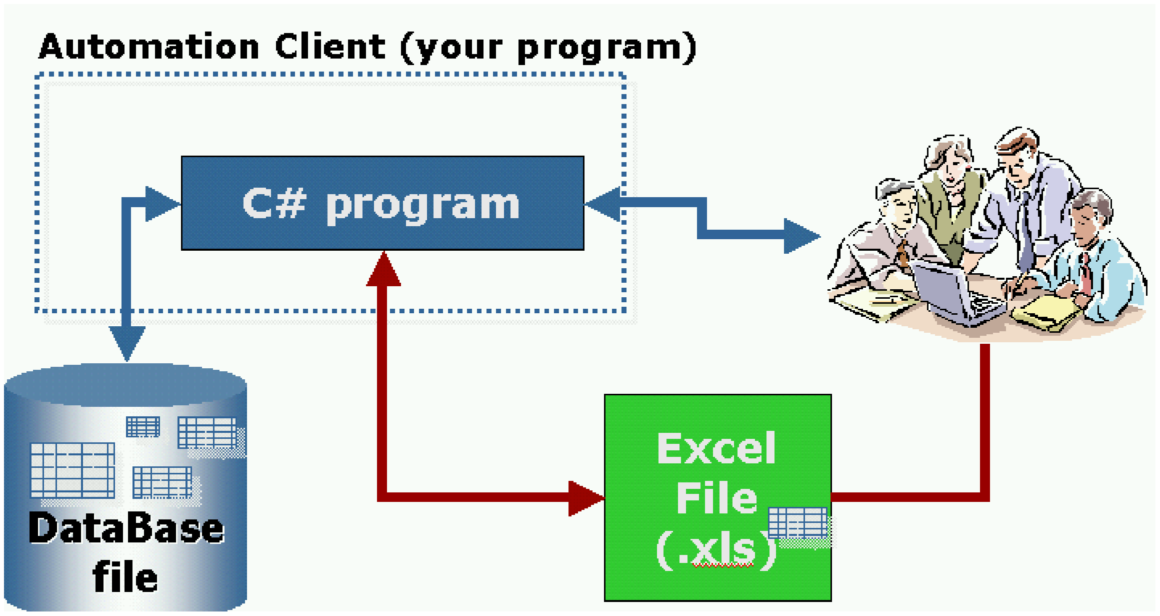
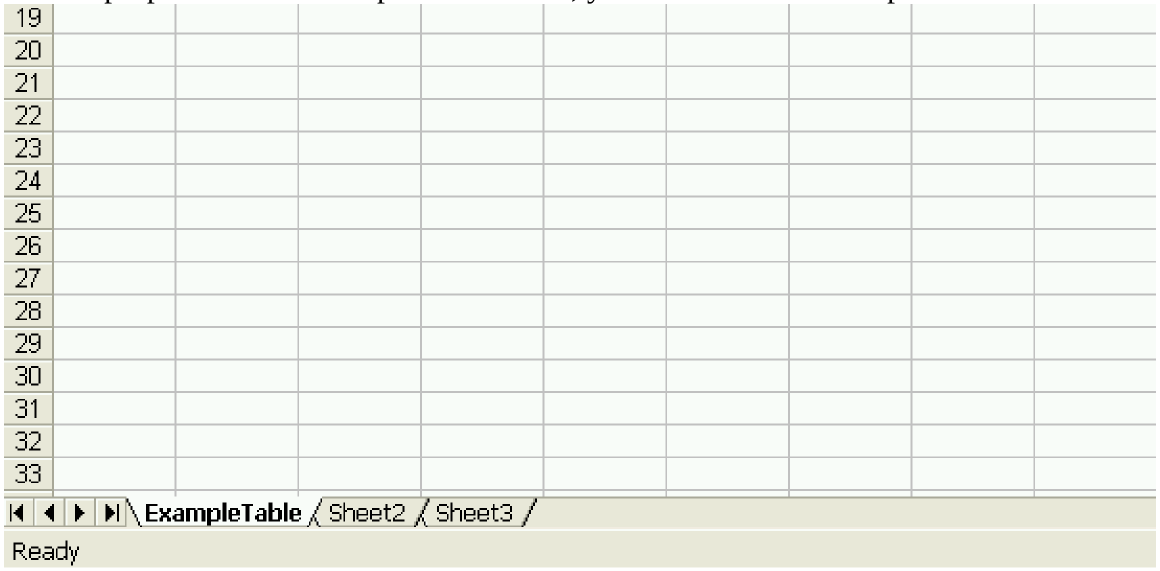


Diagram taken from http://gates.comm.virginia.edu/sg6m/Comm320/COMM320_HomePage.htm

Setting Things Up

This is the procedure to set an Excel workbook up for interaction with C#

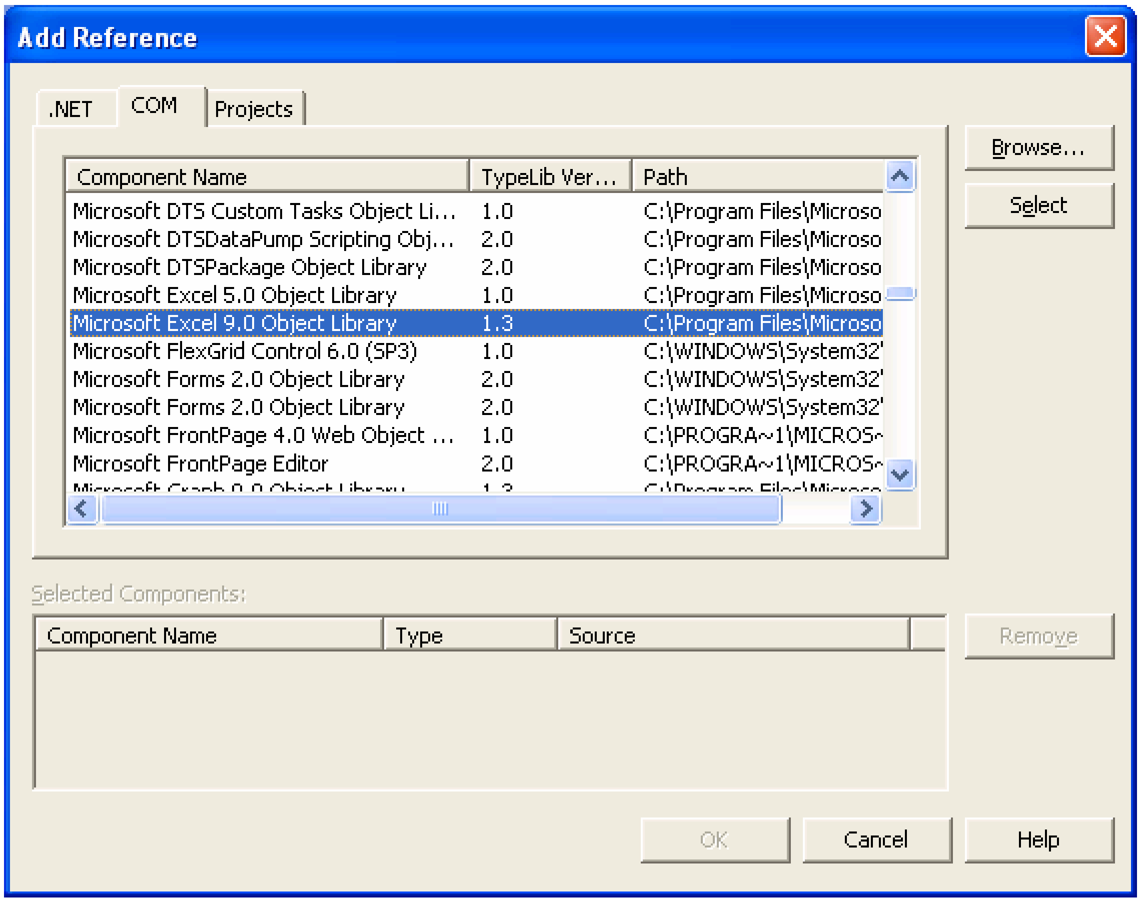
1. Create a new Excel spreadsheet file. For the purposes of the example code below, you can name it “testfile.xls”.
2. Change the worksheet name to something relevant from the standard “Sheet1”. For the purposes of the example code below, you can name it “ExampleTable”.



3. Now fill in any data you want for retrieval in your C# application and save your changes.

This is the procedure in Visual Studio .NET that must be followed in order to use Excel Integration:

1. Right click the “References” tab under the project name in the Solution Explorer on the upper right corner.
2. Select “Add Reference”.
3. Click on the COM tab at the top
4. Select “Microsoft Excel 9.0 Object Library”. Note: This may be 10.0 (Office XP) or 11.0 (Office 2003) depending on the version of Microsoft Excel you are using.



Component Name	TypeLib Ver...	Path
Microsoft DTS Custom Tasks Object Li...	1.0	C:\Program Files\Microso
Microsoft DTSDataPump Scripting Obj...	2.0	C:\Program Files\Microso
Microsoft DTSPackage Object Library	2.0	C:\Program Files\Microso
Microsoft Excel 5.0 Object Library	1.0	C:\Program Files\Microso
Microsoft Excel 9.0 Object Library	1.3	C:\Program Files\Microso
Microsoft FlexGrid Control 6.0 (SP3)	1.0	C:\WINDOWS\System32'
Microsoft Forms 2.0 Object Library	2.0	C:\WINDOWS\System32'
Microsoft Forms 2.0 Object Library	2.0	C:\WINDOWS\System32'
Microsoft FrontPage 4.0 Web Object ...	1.0	C:\PROGRA~1\MICROS~
Microsoft FrontPage Editor	2.0	C:\PROGRA~1\MICROS~

Component Name	Type	Source

Creating a C# Windows Application that Interacts with Excel

Note that code was taken and modified from Professor Grazioli’s website at
http://gates.comm.virginia.edu/sg6m/Comm320/COMM320_HomePage.htm

Now we can start coding! The following code will explain how to fill a spreadsheet from the data from a table, open and close the spreadsheet from your C# application, pull information out of the spreadsheet, and writing modifications made in your C# application to the spreadsheet.
Using the procedures in section 2 of this lab, create all the necessary components:
1. Create an OdbcConnection, OdbcCommand, and OdbcDataAdapter.
2. Set the “ConnectionString” in the OdbcConnection object.
3. Set the “Connection” command in the OdbcCommand to this OdbcConnection.

4. Set the “CommandText” of the OdbcCommand object to a SQL Select command that selects all items from your table that you want to put into the Excel spreadsheet.
5. Set the “SelectCommand” property of the OdbcDataAdapter to the OdbcCommand.
6. Create a new DataSet object and fill the DataSet using the OdbcDataAdapter.

Now enter the following code at the top of the code listing right after the class declaration. This code simply declares the Excel objects that will be used later in the code:

```
private Excel.Application myExcelObj;
private Excel.Workbook myWorkBook;
private Excel.Worksheet myWorkSheet;
private Excel.Range myRange;
```

The following code will fill your spreadsheet with the information from the database table you selected:

```
// Create a new application
myExcelObj = new Excel.Application();

//syntax: Open(Filename As String, [UpdateLinks],
[ReadOnly],
//[Format], [Password], [WriteResPassword],
//[IgnoreReadOnlyRecommended], [Origin],
//[Delimiter], [Editable], [Notify], [Converter],
//[AddToMenuRecentlyUsed]) As Workbook
//here I use the default values

//replace c:\localdata\testfile.xls with the path of your
//Excel spreadsheet file
//This works using Office XP (lib 10.0) and Office 2003
//(lib 11.0)
myWorkBook = myExcelObj.Workbooks.Open(xcelFileLocation, 0,
false, 5, "", "", true, Excel.XlPlatform.xlWindows, "\t",
true, true, 0, false, true, false);

//This works using Office 2000 (lib 9.0)
```

```
//myWorkBook = myExcelObj.Workbooks.Open(xcelFileLocation,
//0, false, 5, "", "", true, Excel.XlPlatform.xlWindows,
//"\t", true, true, 0, true);

//get the worksheet (replace “ExampleTable” with the name
//of you worksheet)
myWorkSheet = (Excel.Worksheet)
myWorkBook.Worksheets.get_Item("ExampleTable");
myWorkSheet.Activate();

//get the range. Where it says “A1”, put your starting
//cell, and where it says “D15” put you ending cell.
//Everything in between these two diagonal cells will be
//included in the range.
myRange = myWorkSheet.get_Range("A1", "D15");

//get the table. Replace “ExampleTable” with an appropriate
//table name
myDataTable = myDataSet.Tables["ExampleTable"];

//the rowCounter property tells the application which row
//of the spreadsheet to start on. In this example it is set
//to 2 in order to skip the first row that contains the
//names of the columns. Modify appropriately.
int rowCounter = 2;
foreach (System.Data.DataRow myRow in myDataTable.Rows)
{
     for (int colCounter=0; colCounter<
     myDataTable.Columns.Count;
          colCounter++)
     {
          myRange.set_Item(rowCounter, colCounter+1,
          myRow[(colCounter)].ToString());
     }
     rowCounter++;
}

//save, close, and quit
myWorkBook.Save();
myWorkBook.Close(false, null, null);
myExcelObj.Quit();
```

The following code allows you the open your spreadsheet:

```
myExcelObj = new Excel.Application();
myExcelObj.Visible = true;
```

```
//replace c:\localdata\testfile.xls with the path of your
//Excel spreadsheet file
//This works using Office XP (lib 10.0) and Office 2003
//(lib 11.0)
myWorkBook = myExcelObj.Workbooks.Open(xcelFileLocation, 0,
false, 5, "", "", true, Excel.XlPlatform.xlWindows, "\t",
true, true, 0, false, true, false);

//This works using Office 2000 (lib 9.0)
//myWorkBook = myExcelObj.Workbooks.Open(xcelFileLocation,
//0, false, 5, "", "", true, Excel.XlPlatform.xlWindows,
//"\t", true, true, 0, Missing.Value);

myWorkSheet = (Excel.Worksheet)

//get the table. Replace “ExampleTable” with an appropriate
//table name
myWorkBook.Worksheets.get_Item("ExampleTable");
myWorkSheet.Activate();
myForm.Activate();
```

The following code allows you to close your spreadsheet:

```
myWorkBook.Close(false, null, null);
myExcelObj.Quit();
```

The following code allows you to pull information from the spreadsheet and put it into your C# application (assuming you have textbox1, textbox2, and textbox3 objects-modify this accordingly):

```
//Create a new application
myExcelObj = new Excel.Application();

//replace c:\localdata\testfile.xls with the path of your
//Excel spreadsheet file
//This works using Office XP (lib 10.0) and Office 2003
//(lib 11.0)
myWorkBook = myExcelObj.Workbooks.Open(xcelFileLocation, 0,
false, 5, "", "", true, Excel.XlPlatform.xlWindows, "\t",
true, true, 0, false, true, false);

//This works using Office 2000 (lib 9.0)
```

```
//myWorkBook = myExcelObj.Workbooks.Open(xcelFileLocation,
//0, false, 5, "", "", true, Excel.XlPlatform.xlWindows,
//"\t", true, true, 0, Missing.Value);

//get the table. Replace “ExampleTable” with an appropriate
//table name
myWorkSheet = (Excel.Worksheet)
myWorkBook.Worksheets.get_Item("ExampleTable");
myWorkSheet.Activate();

//Grabs the data in the specified cells from the
//spreadsheet. Assumes that you want to write the data to
//textBox1, textBox2, and textBox3. Also assumes that you
//want to grab the data from the specified row and column
//numbers. Replace all accordingly.
// NOTE this works with Office XP (lib ~10.0) and Office
2003 (lib ~11.0)
myRange = myWorkSheet.get_Range("B2", "B2");
textBox1.Text =
(myRange.get_Value(Missing.Value)).ToString();
myRange = myWorkSheet.get_Range("B3", "B3");
textBox2.Text =
(myRange.get_Value(Missing.Value)).ToString();
myRange = myWorkSheet.get_Range("B4", "B4");
textBox3.Text =
(myRange.get_Value(Missing.Value)).ToString();

/* NOTE this works with Office 2000 (lib ~9.0)
 myRange = myWorkSheet.get_Range("B2", "B2");
 textBox1.Text = (myRange.Value).ToString();
myRange = myWorkSheet.get_Range("B3", "B3");
textBox2.Text = (myRange.Value).ToString();
myRange = myWorkSheet.get_Range("B4", "B4");
textBox3.Text = (myRange.Value).ToString(); */
```

Now assuming you have modified the data in the three textboxes, the following code will write the data from the textboxes back to the spreadsheet:

```
//get the range. Where it says “A1”, put your starting
//cell, and where it says “D15” put you ending cell.
//Everything in between these two diagonal cells will be
//included in the range.
myRange = myWorkSheet.get_Range("A1", "D15");

//Performs the modification of the data in the specified
//cells in the spreadsheet. Assumes that you want to write
//the data from textBox1, textBox2, and textBox3. Also
```

```
//assumes that you want to write the data from the
//textboxes to the specified row and column number. Replace
//all accordingly
myRange.set_Item(2, 2, textBox1.Text);
myRange.set_Item(3, 2, textBox2.Text);
myRange.set_Item(4, 2, textBox3.Text);

//save, close, and quit
myWorkBook.Save();
myWorkBook.Close(false, null, null);
myExcelObj.Quit();
SaveBtn.Text = "DONE";
```

Note: When using the application to interact with Excel, sometimes Excel processes will remain in memory even after the application has seemingly been closed. These extra Excel processes interfere with further functions in the application. Thus, after or before running the application, check the "Processes" list in Windows and end any running "EXCEL" processes (First make sure the actual application is closed).

Exercise

Create a C# Windows application (called "exI3") that contains five buttons and three textboxes. Leave the textboxes blank and label the buttons according to their functions:

Button	**Function**
1.	Retrieve data from table
2.	Fill spreadsheet with data from table
3.	Open the spreadsheet
4.	Close the spreadsheet
5.	Retrieve data from spreadsheet
6.	Save form data to spreadsheet

Create a spreadsheet file in Microsoft Excel called "exI3.xls". Change the name of the active worksheet to "Clothes". Fill the top three cells (A1, A2, A3) with "Type", "Color", and "Size". The first button will retrieve the contents of the "Selected" table created in I2. The second button will fill the spreadsheet with that data. The third button will open the "exI3.xls" spreadsheet in Microsoft Excel. The fourth button will click Microsoft Excel. The fifth button will populate the three textboxes with the information from the corresponding cells from "exI3.xls". The sixth button will store whatever has been entered into the textboxes into the corresponding cells in "exI3.xls".

Source Used:

1. COMM 320 (Professor Grazioli) website:
http://gates.comm.virginia.edu/sg6m/Comm320/COMM320_HomePage.htm

Appendix 4

Advanced C#

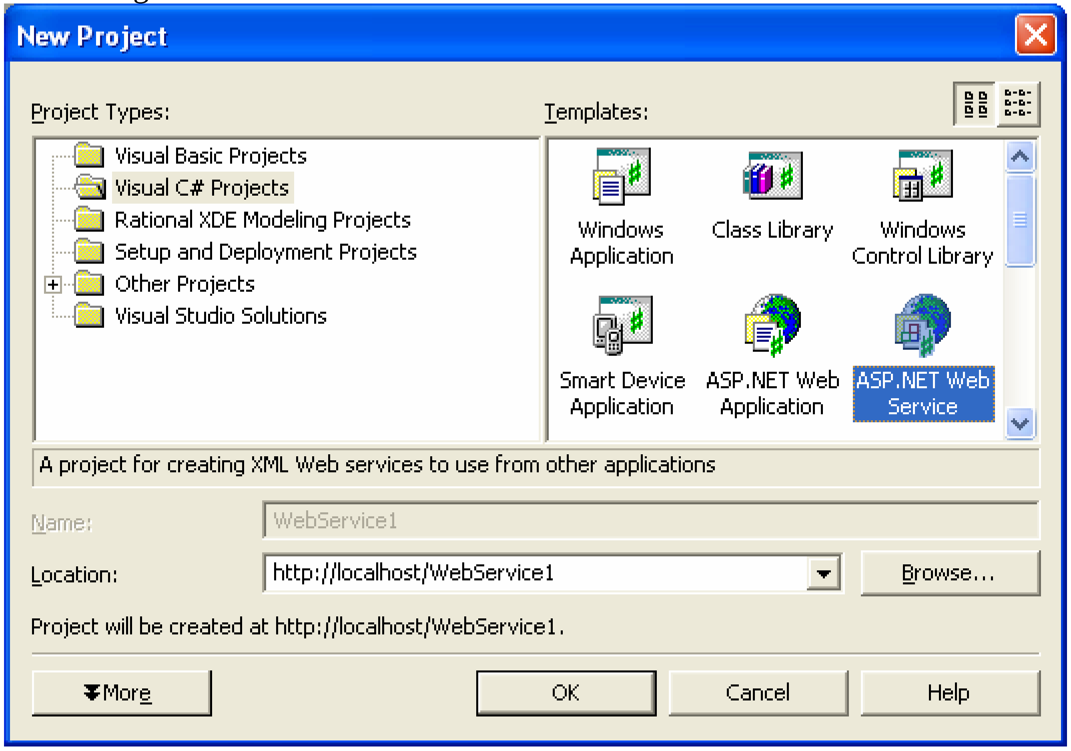
Exercise A1 – Creating an XML Web Service Using C#

Overview

A Web Service is a reusable piece of code that generally sits on a remote server so it can be called by various web applications. A common place that Web Services are used is in interfaces among different companies. For example, if Company B wants to access the data from Company A's database server, Company A can create a Web Service that provides a standard interface for Company B to access the database. Later, if Company C wants to access the database, Company A simple has to direct Company C to the existing Web Service.

Creation of Web Services

To create a Web Service, start up Visual Studio .NET and go to File -> New -> Project. This dialog will be shown:



A project for creating XML Web services to use from other applications

Name: WebService1

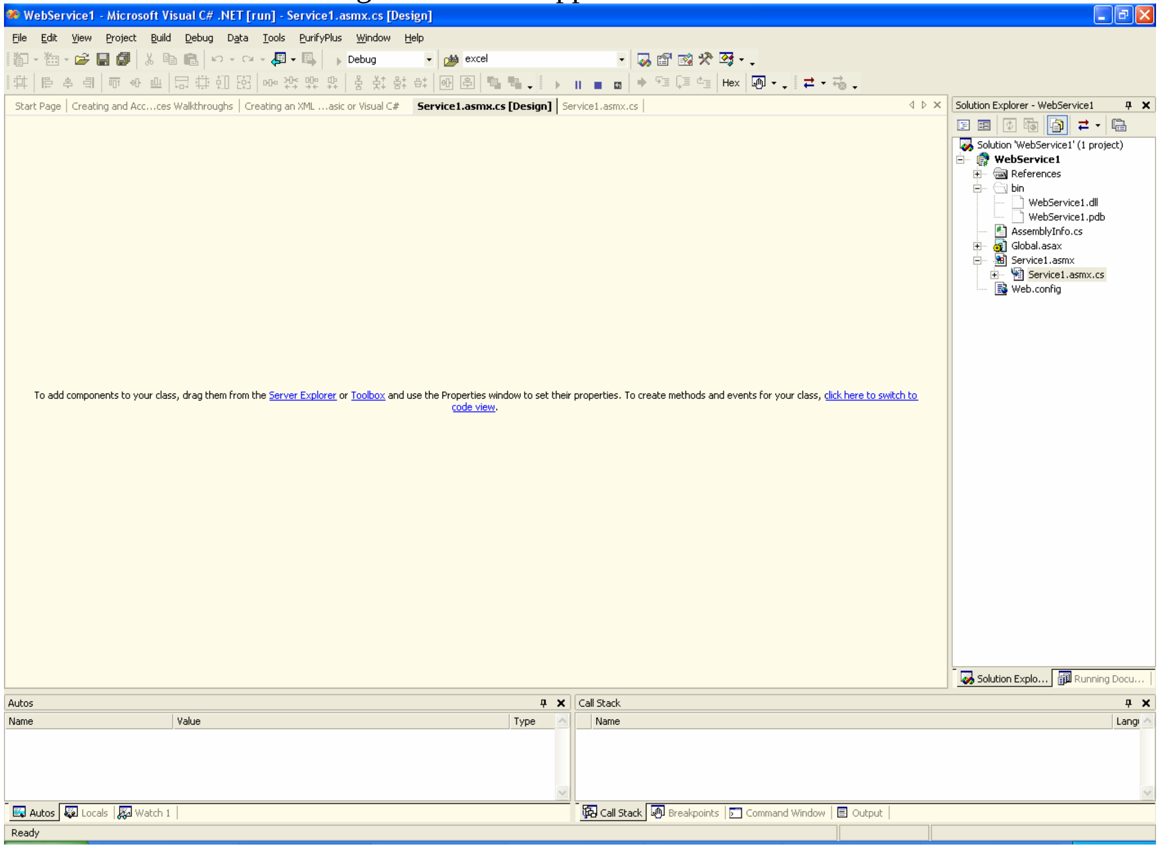
Location: http://localhost/WebService1

Project will be created at http://localhost/WebService1.

Make sure "Visual C# Projects" is selected on the left under "Project Types," and that "ASP.NET Web Service" is selected on the right under "Templates."

You can optionally create a new name and location (must be a server name if not "localhost") for the project. In the "Location" dialogue, "localhost" means the local IIS server that has been installed on the machine. This comes with Windows 2000 Professional and Windows XP Professional, and must be installed for local execution. To install this go to Control Panel and select Add/Remove Programs -> Add/Remove Windows Components -> Internet Information Services (IIS). Make sure all components of IIS have been selected before installing.

Instead, if you have access to a remote server put this information in the “Location” box. The Web Service form designer will now appear in the IDE:


WebService1 - Microsoft Visual C# .NET [run] - Service1.asmx.cs [Design]
File Edit View Project Build Debug Data Tools PurifyPlus Window Help
Debug
excel
Hex
Start Page | Creating and Acc…ces Walkthroughs | Creating an XML …asic or Visual C# | Service1.asmx.cs [Design] | Service1.asmx.cs
Solution Explorer - WebService1
Solution 'WebService1' (1 project)
WebService1
References
bin
WebService1.dll
WebService1.pdb
AssemblyInfo.cs
Global.asax
Service1.asmx
Service1.asmx.cs
Web.config
Solution Explo… | Running Docu…
Autos
Name | Value | Type
Call Stack
Name | Lang
Autos | Locals | Watch 1
Call Stack | Breakpoints | Command Window | Output
Ready

To add components to your class, drag them from the Server Explorer or Toolbox and use the Properties window to set their properties. To create methods and events for your class, click here to switch to code view.

From here, change the name of the form to “TemperatureConverter”. Now double click this window to access the code behind. We will now create a simple Web Service using code from Microsoft Visual Studio .NET Help “Walkthrough: Creating an XML Web Service Using Visual Basic or Visual C#”.

First locate the line that reads:

```
public class TemperatureConverter : System.Web.Services.WebService
```

Above this line, add the following code:

```
[System.Web.Services.WebService(
              Namespace="http://Walkthrough/XmlWebServices/",
              Description="A temperature conversion service.")]
```

Adding the WebService attribute allows us to give the Web Service a description that will be displayed to the user as well as a namespace that will aid us when we create multiple Web Services in the same project.

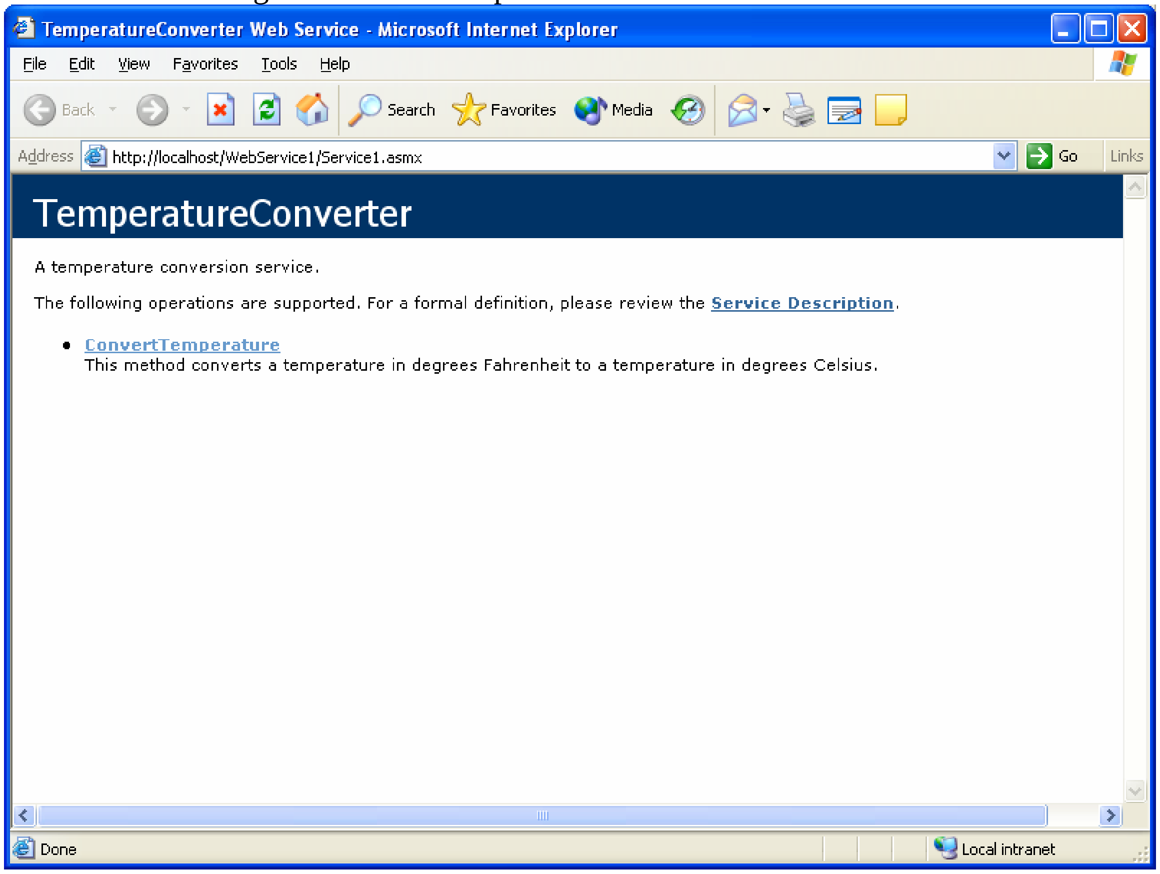
We will now add the method that actually performs the conversion. Add the following code to the TemperatureConverter class:

```
[WebMethod(Description="This method converts a temperature in " +
       "degrees Fahrenheit to a temperature in degrees Celsius.")]
public double ConvertTemperature(double dFahrenheit)
{
   return ((dFahrenheit - 32) * 5) / 9;
}
```

This method (called a WebMethod in the context of Web Services) takes the Fahrenheit temperature and plugs it into an equation and outputs the Celsius answer. Note that is necessary for the WebMethod attribute to be attached to method in order for it to be associated with the Web Service we are creating. We have also provided a description for the WebMethod to be displayed to the user.

Now we will run the Web Service. First, right-click the file “Service1.asmx” in the Solution Explorer and click “Set as Start Page”. Now press “F5” to build and execute the Web Service. If there are any syntactical errors, they will show up exactly like a regular windows or web application and you simply correct your C# code.

Notice the following window comes up:


TemperatureConverter Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebService1/Service1.asmx
Go Links
Done
Local intranet

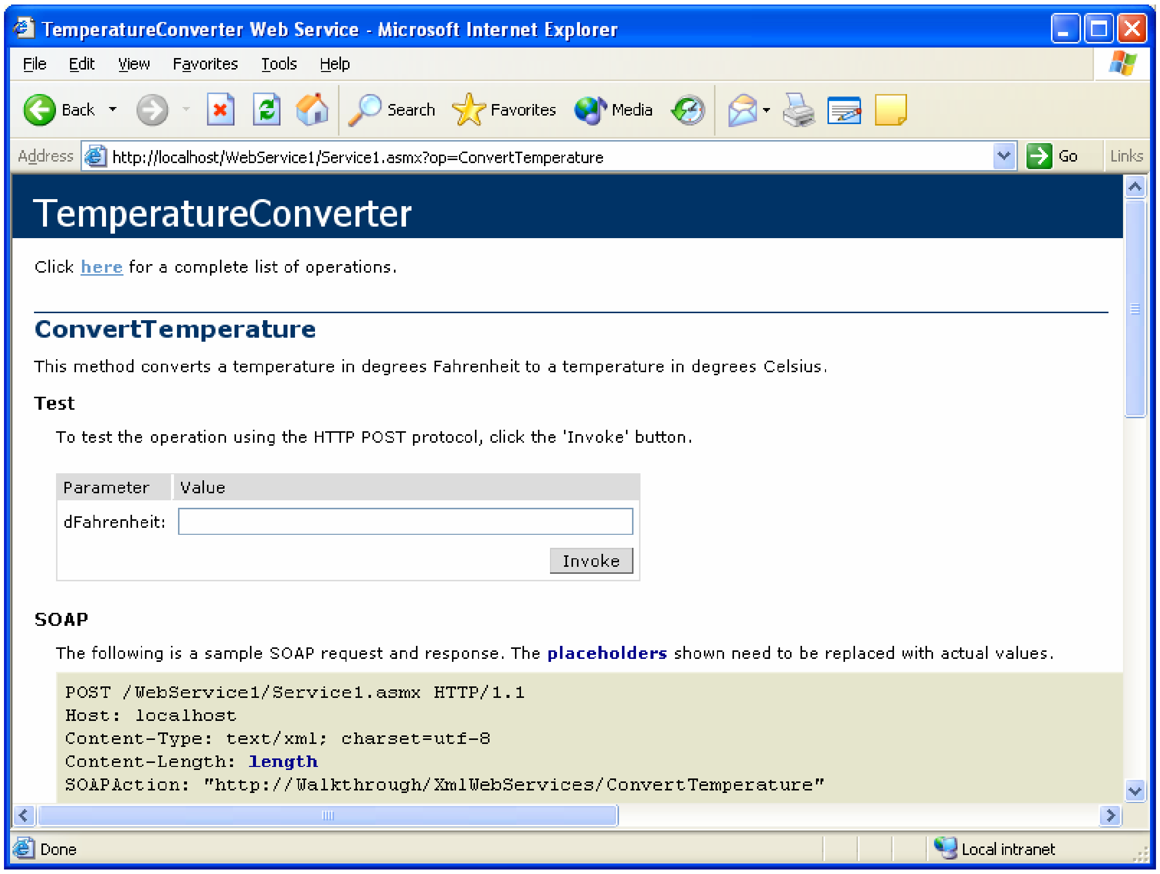
TemperatureConverter

A temperature conversion service.

The following operations are supported. For a formal definition, please review the Service Description.

- ConvertTemperature
  This method converts a temperature in degrees Fahrenheit to a temperature in degrees Celsius.

Notice the displaying of the description for the Web Service we specified “A temperature conversion service.” Also notice that we can select our “ConvertTemperature” class because we added the WebMethod attribute to it. Go ahead and select “ConvertTemperature”. You are taken to this page:


TemperatureConverter Web Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebService1/Service1.asmx?op=ConvertTemperature
Go Links
Done
Local intranet

TemperatureConverter

Click here for a complete list of operations.

ConvertTemperature

This method converts a temperature in degrees Fahrenheit to a temperature in degrees Celsius.

Test

To test the operation using the HTTP POST protocol, click the 'Invoke' button.

Parameter	Value
dFahrenheit:	

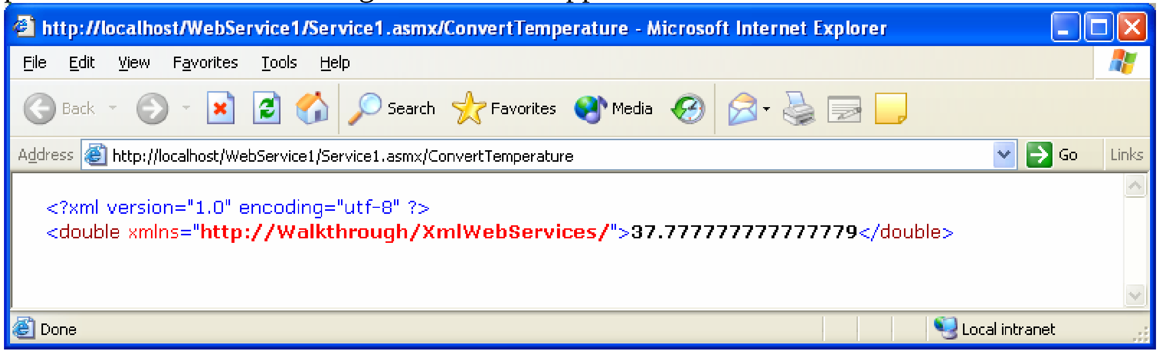
Invoke

SOAP

The following is a sample SOAP request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /WebService1/Service1.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://Walkthrough/XmlWebServices/ConvertTemperature"
```

Notice that we can input our “dFahrenheit” variable in the textbox. If you type “100” and press “Invoke” the following window will appear:


http://localhost/WebService1/Service1.asmx/ConvertTemperature - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebService1/Service1.asmx/ConvertTemperature
Go Links
Done
Local intranet

```
<?xml version="1.0" encoding="utf-8" ?>
<double xmlns="http://Walkthrough/XmlWebServices/">37.777777777777779</double>
```

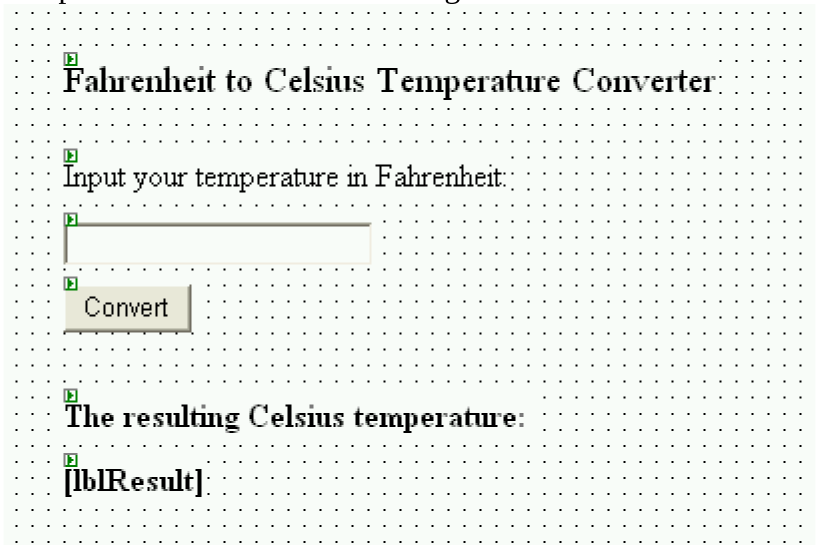
This is the result of our function displayed in XML. The Fahrenheit temperature we entered has been converted to Celsius.

Now we will use this Web Service in an existing Web Application using our own interface.

Using an XML Web Service Using C#

We will now access our Web Service using code from the Microsoft Visual Studio .NET Help "Walkthrough: Accessing an XML Web Service Using Visual Basic or Visual C#".

First create a Web Application using the steps described in the Intermediate C# lab. Now add a textbox, some labels, and button to the webform. Set the "Text" properties of the components to mirror the following:



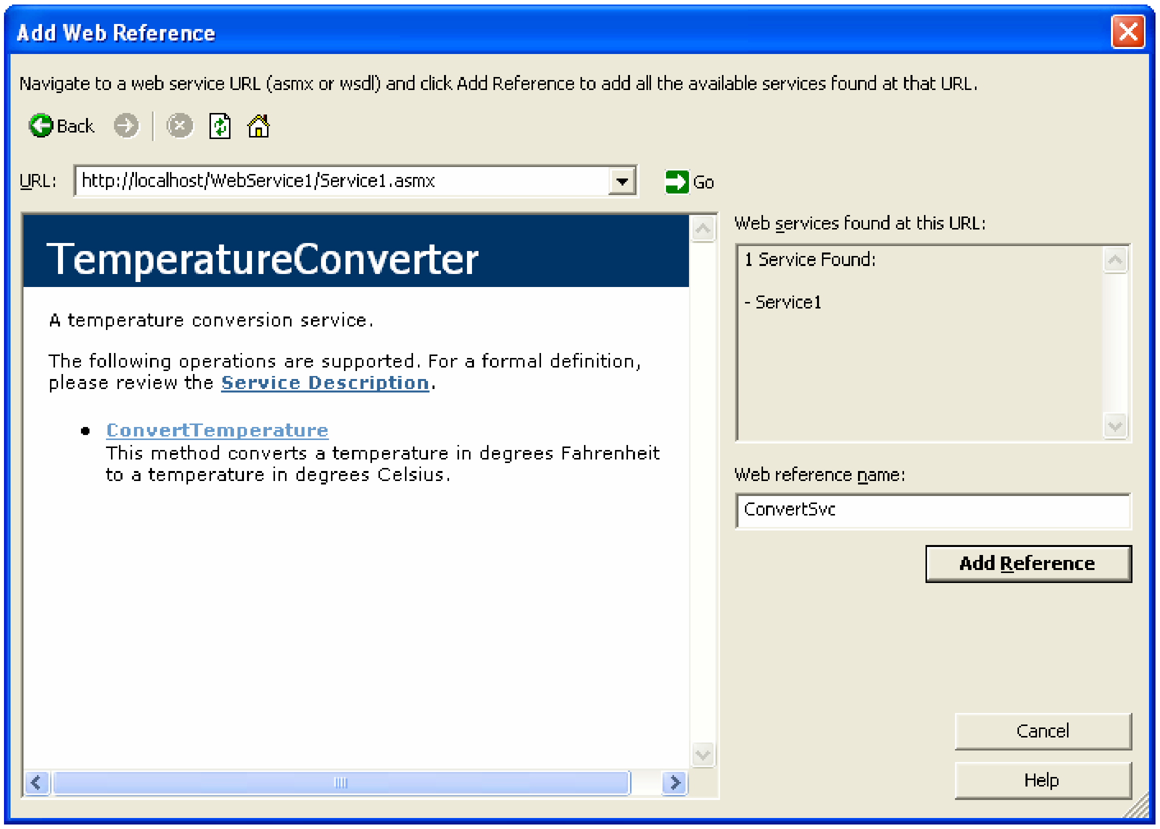
Fahrenheit to Celsius Temperature Converter

Input your temperature in Fahrenheit:

The resulting Celsius temperature:

[lblResult]

Now we need to add a reference to our existing Web Service. On the Project menu, select "Add Web Reference". On the resulting screen enter "http://localhost/WebService1/Service1.asmx" next to "URL" and select "Go". Now put "ConvertSvc" under "Web reference name". The window should now look like this:


Add Web Reference

Navigate to a web service URL (asmx or wsdl) and click Add Reference to add all the available services found at that URL.

URL: http://localhost/WebService1/Service1.asmx

TemperatureConverter

A temperature conversion service.

The following operations are supported. For a formal definition, please review the Service Description.

- ConvertTemperature
  This method converts a temperature in degrees Fahrenheit to a temperature in degrees Celsius.

Web services found at this URL:

1 Service Found:

- Service1

Web reference name: ConvertSvc

Click “Add Reference”

In the Solution Explorer you will notice that your Web Reference has been added under the name “ConvertSvc”. The Web Reference we just added gives the web application the means to communicate with the XML Web service and to locate it at run time. It does this by creating a proxy class that interfaces with the XML Web service and creates a local representation of the XML Web service for us to work with called “ConvertSvc”.

We will now create an instance of the XML Web service's proxy class so we can access the methods of the Web Service we created. When our web application calls these methods, Visual Studio .NET provides the communications between our web application and Web Service.

We will basically take the value we provide in the textbox, and make a call to the Web Service ConvertTemperature method using the proxy class. We will then display the resulting value returned from the XML Web service in the label.

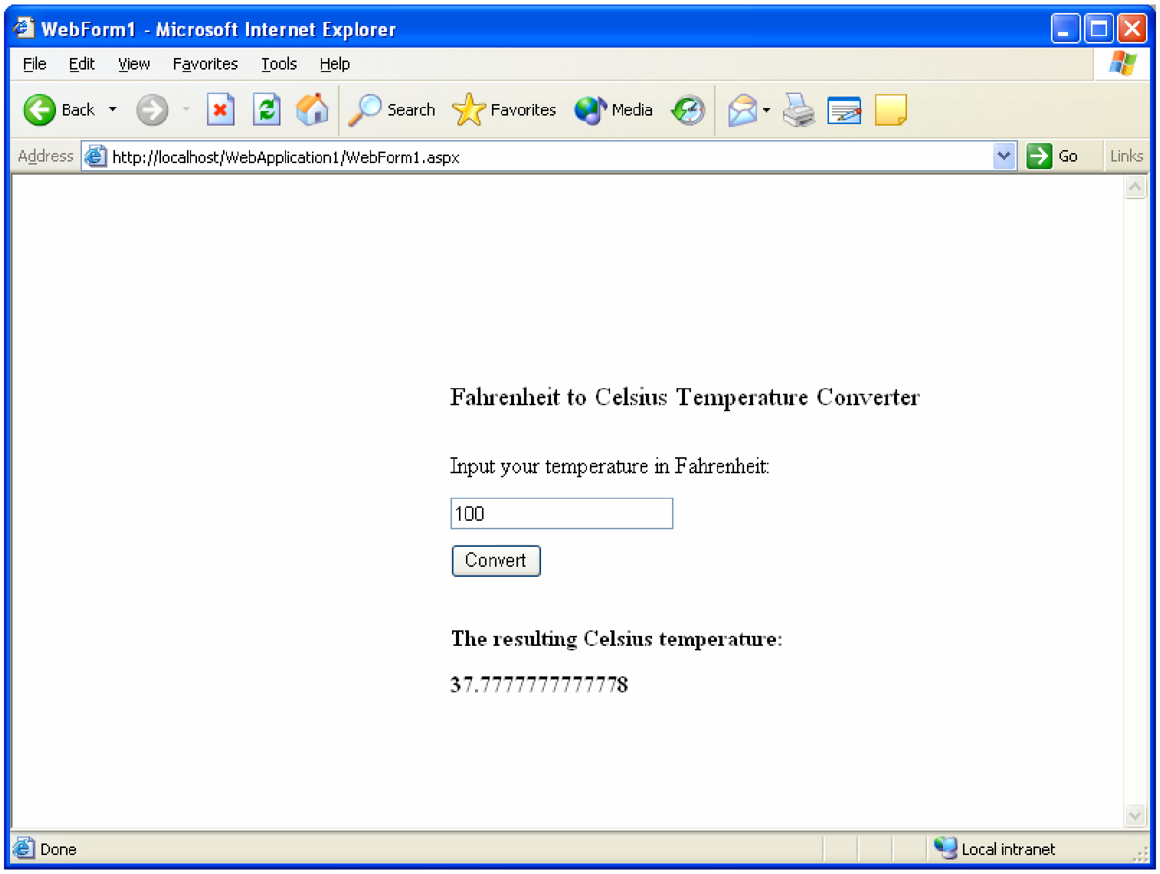
Double click the Convert button and add the following code to its event handler:

```
//Create new instance of the Web Service called ws
ConvertSvc.TemperatureConverter ws = new
ConvertSvc.TemperatureConverter();
```

```
//take the value out of the textbox and convert it to a double number
//and put the result in the variable dFahrenheit
double dFahrenheit = Convert.ToDouble(TextBox1.Text);

//pass the number to the method “ConvertTemperature” of our Web Service
//instance and put the result it returns in the variable dCelsius
double dCelsius = ws.ConvertTemperature(dFahrenheit);

//take the value of dCelsius and display it in our label
Label1.Text = dCelsius.ToString();
```

Change the names of the components accordingly in the code. Now set the file “WebApplication1.aspx” as the Start Page and execute the application.

In the resulting page enter “100” and click “Convert”. The following page will be shown:


WebForm1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebApplication1/WebForm1.aspx
Go Links
Done
Local intranet

Fahrenheit to Celsius Temperature Converter

Input your temperature in Fahrenheit:

100

Convert

The resulting Celsius temperature:

37.7777777777778

We have successfully interfaced with our Web Service in our web application. Thus, any number of applications can make use of our Web Service.

Exercise

1. Create a Web Service that has four methods: Add, Subtract, Multiply, and Divide. Each method should take two double-precision numbers and perform the

appropriate operation on the numbers. Be sure to include a description for the Web Service. Name the service form “Calculator”.

2. Create a C# ASP.NET web application that makes use of this Calculator Web Service. This web application must contain at a minimum two textboxes for the inputs, four buttons (one for each function), and a label that displays the output.

Sources Used:

1. Microsoft Visual Studio .NET Help “Walkthrough: Creating an XML Web Service Using Visual Basic or Visual C#”.
2. Microsoft Visual Studio .NET Help “Walkthrough: Accessing an XML Web Service Using Visual Basic or Visual C#”.

Advanced C#

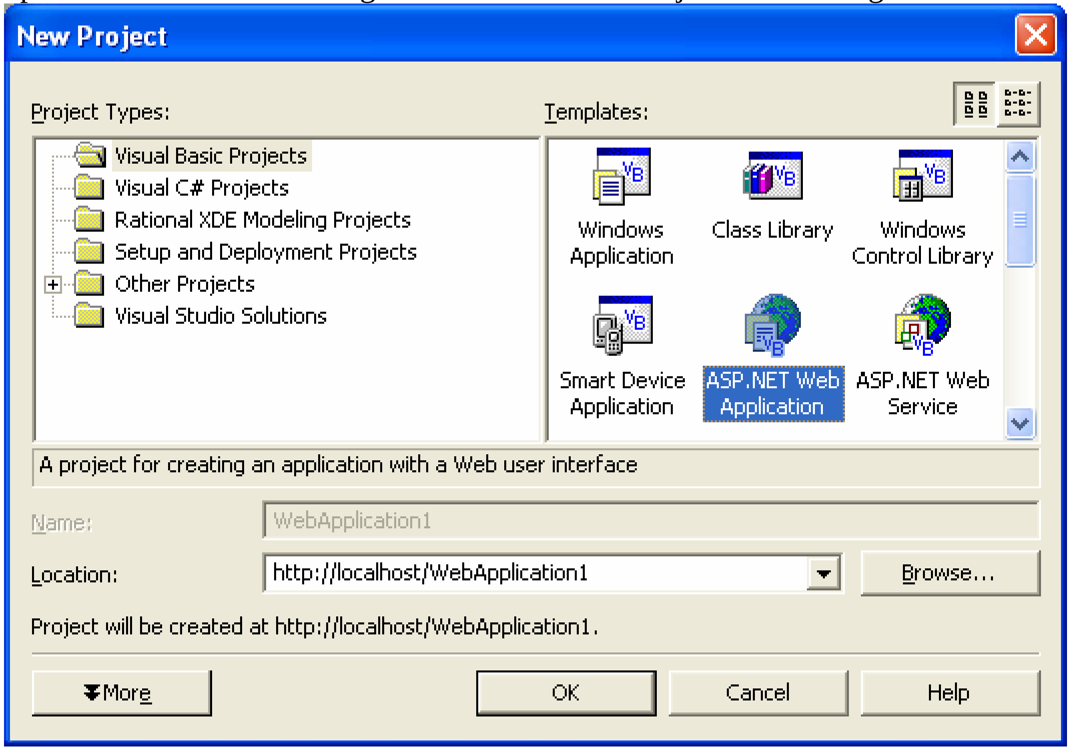
Exercise A2 – Language Interoperability in C# using Web Services

Overview

One of the powerful features of .NET and Web Services is language-interoperability. We can write code in different languages that communicate with each other! In this case, we are going to create a Visual Basic.NET web application that uses the C# Web Service TemperatureConvert we created in Exercise A1.

Creation of VB.NET Webform

To create a Visual Basic .NET Web Application instead of a C# Web Application, start up Visual Studio .NET and go to File -> New -> Project. This dialog will be shown:



New Project

Project Types:
- Visual Basic Projects
- Visual C# Projects
- Rational XDE Modeling Projects
- Setup and Deployment Projects
- Other Projects
- Visual Studio Solutions

Templates:
- Windows Application
- Class Library
- Windows Control Library
- Smart Device Application
- ASP.NET Web Application
- ASP.NET Web Service

A project for creating an application with a Web user interface

Name: WebApplication1

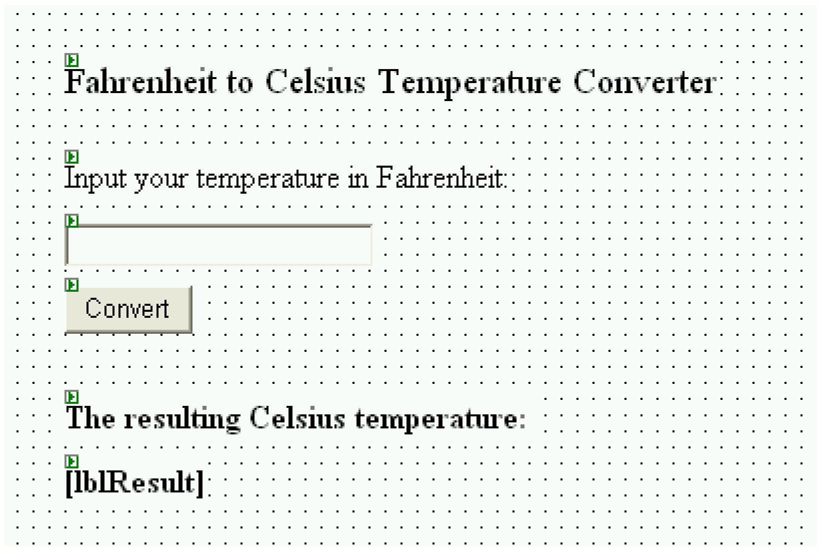
Location: http://localhost/WebApplication1

Project will be created at http://localhost/WebApplication1.

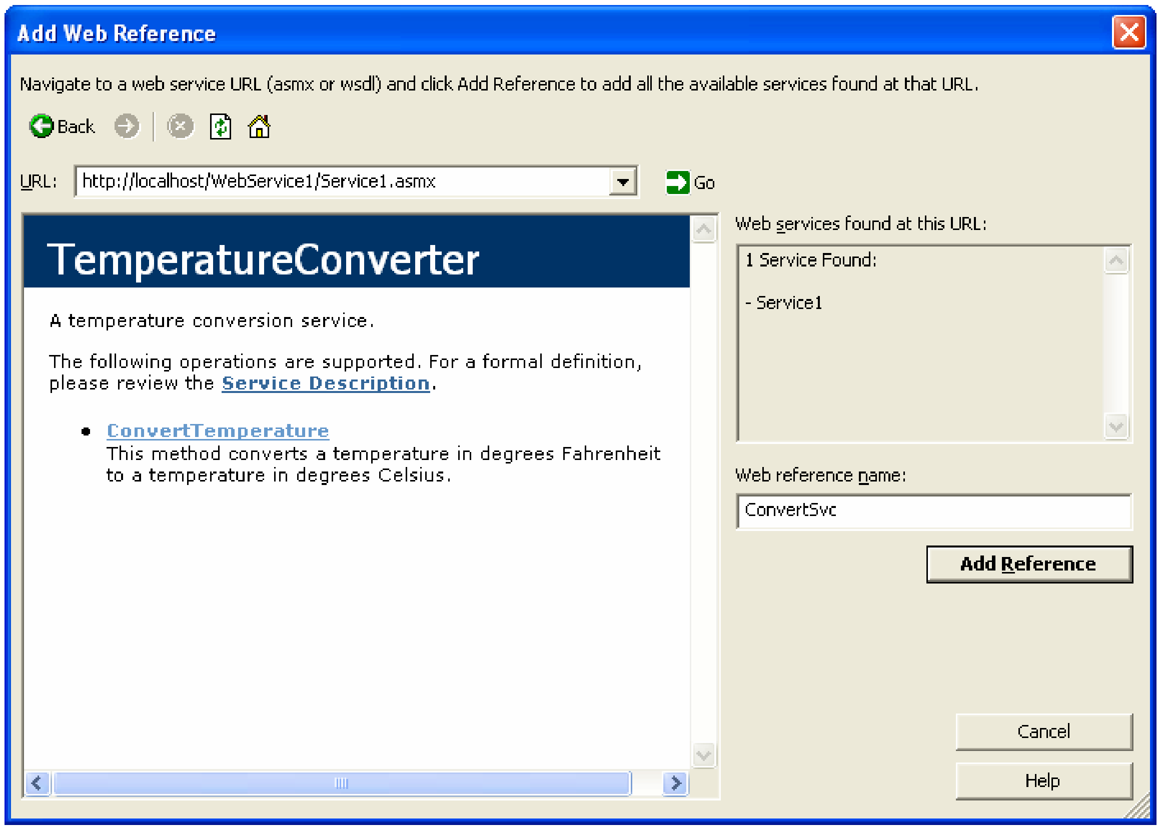
Make sure "Visual Basic Projects" is selected on the left under "Project Types," and that "ASP.NET Web Application" is selected on the right under "Templates."

You can optionally create a new name and location (must be a server name if not "localhost") for the project. In the "Location" dialogue, "localhost" means the local IIS server that has been installed on the machine. This comes with Windows 2000 Professional and Windows XP Professional, and must be installed for local execution. If you have access to a remote server put this information in the "Location" box. The webform designer will now appear in the IDE.

Now add a textbox, some labels, and button to the webform. Set the "Text" properties of the components to mirror the following:


Fahrenheit to Celsius Temperature Converter
Input your temperature in Fahrenheit:
Convert
The resulting Celsius temperature:
[lblResult]

Now we need to add a reference to our existing C# Web Service. On the Project menu, select “Add Web Reference”. On the resulting screen enter “http://localhost/WebService1/Service1.asmx” next to ”URL” and select “Go”. Now put “ConvertSvc” under “Web reference name”. The window should now look like this:



Navigate to a web service URL (asmx or wsdl) and click Add Reference to add all the available services found at that URL.

URL: http://localhost/WebService1/Service1.asmx

TemperatureConverter

A temperature conversion service.

The following operations are supported. For a formal definition, please review the Service Description.

- ConvertTemperature
  This method converts a temperature in degrees Fahrenheit to a temperature in degrees Celsius.

Web services found at this URL:

1 Service Found:

- Service1

Web reference name: ConvertSvc

Click “Add Reference”

In the Solution Explorer you will notice that your Web Reference has been added under the name “ConvertSvc”.

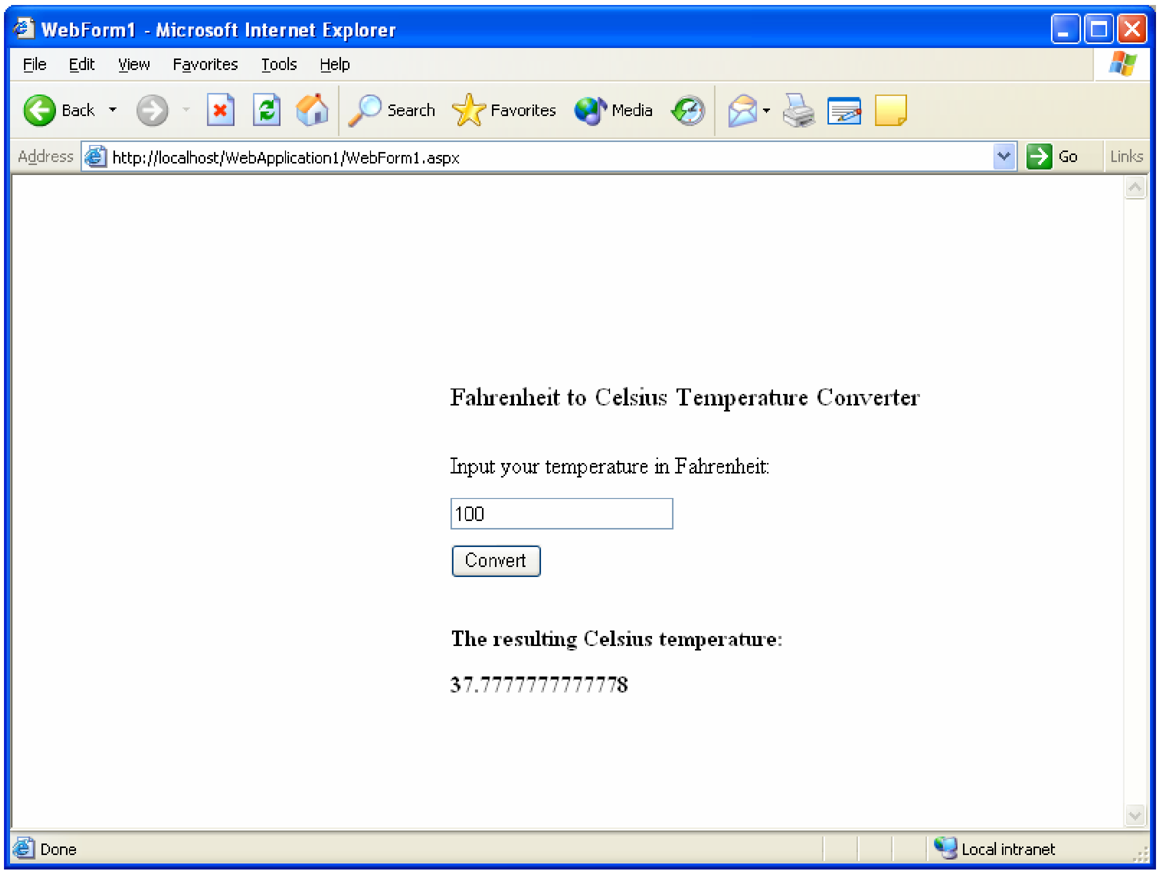
Just as in the C# Web Application from Exercise A1, we will take the value we provide in the textbox, and make a call to the C# Web Service ConvertTemperature method using the proxy class. We will then display the resulting value returned from the XML Web service in the label.

Double click the Convert button and add the following code to its event handler:

```
'Create new instance of the Web Service called ws
Dim ws As New ConvertSvc.TemperatureConverter

'take the value out of the textbox and convert it to a double number
and put the result in the variable dFahrenheit
Dim dFahrenheit = Convert.ToDouble(TextBox1.Text)

'pass the number to the method "ConvertTemperature" of our Web Service
instance and put the result it returns in the variable dCelsius
Dim dCelsius = ws.ConvertTemperature(dFahrenheit)

'take the value of dCelsius and display it in our label
Label1.Text = dCelsius.ToString()
```

Change the names of the components accordingly in the code. Now set the file “WebApplication1.aspx” as the Start Page and execute the application.

In the resulting page enter “100” and click “Convert”. The following page will be shown:


WebForm1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://localhost/WebApplication1/WebForm1.aspx
Go Links
Done
Local intranet

Fahrenheit to Celsius Temperature Converter

Input your temperature in Fahrenheit:

100

Convert

The resulting Celsius temperature:

37.7777777777778

Thus, we have successfully interfaced with our C# Web Service with our Visual Basic .NET web application. Thus, Web Services make mixing and matching languages in .NET simple.

Exercise

1. Create a Visual Basic.NET ASP.NET web application that makes use of the C# Web Service from Exercise 1. This web application must contain at a minimum two textboxes for the inputs, four buttons (one for each function), and a label that displays the output.

Sources Used:

1. Microsoft Visual Studio .NET Help “Walkthrough: Creating an XML Web Service Using Visual Basic or Visual C#”.
2. Microsoft Visual Studio .NET Help “Walkthrough: Accessing an XML Web Service Using Visual Basic or Visual C#”.

Advanced C#

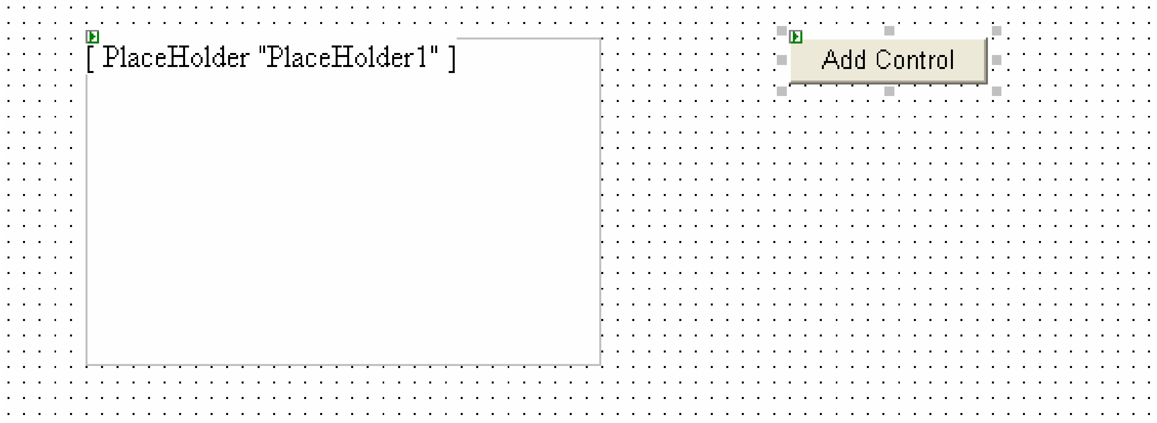
Exercise A3 – Dynamic Control Creation in Web Applications using C#

Overview

Dynamic controls are useful in various applications, most notably survey generation applications. In C#/ASP.NET webforms, dynamic control creation is accomplished through the use of PlaceHolders.

Creation/Use of PlaceHolder Controls

Add the following items to the web form: panel, PlaceHolder (place in the panel), and button (label it Add Control). Your page will look similar to this:

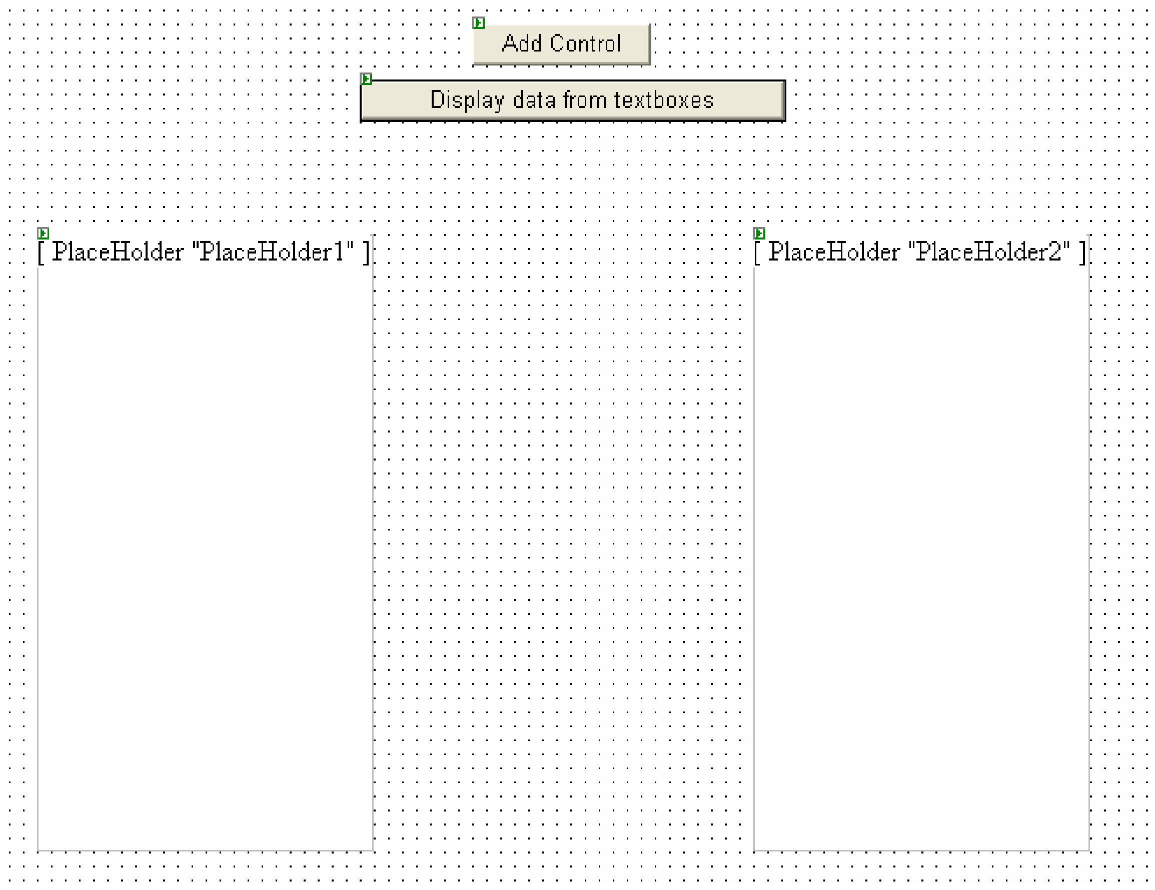


Now add the following code to the click event of the button:

```
PlaceHolder1.Controls.Add(new TextBox());
```

If you run the application and click on the “Add Control” button, a new TextBox will added to the form each time you click the button.

Now what if you want to add textboxes dynamically to a form and get the text from the boxes after the user has entered text in them? Add one more panel and placeholder combination as well as one button to the form. Make your page look similar to this:


Add Control
Display data from textboxes
[PlaceHolder "PlaceHolder1"]
[PlaceHolder "PlaceHolder2"]

Add the following code to the Page_Load() method:

```
for (int i = 1; i <= 5; ++i)
{
	TextBox newTB = new TextBox();
	newTB.ID = "TextBox" + i;
	PlaceHolder1.Controls.Add(newTB);
}
```

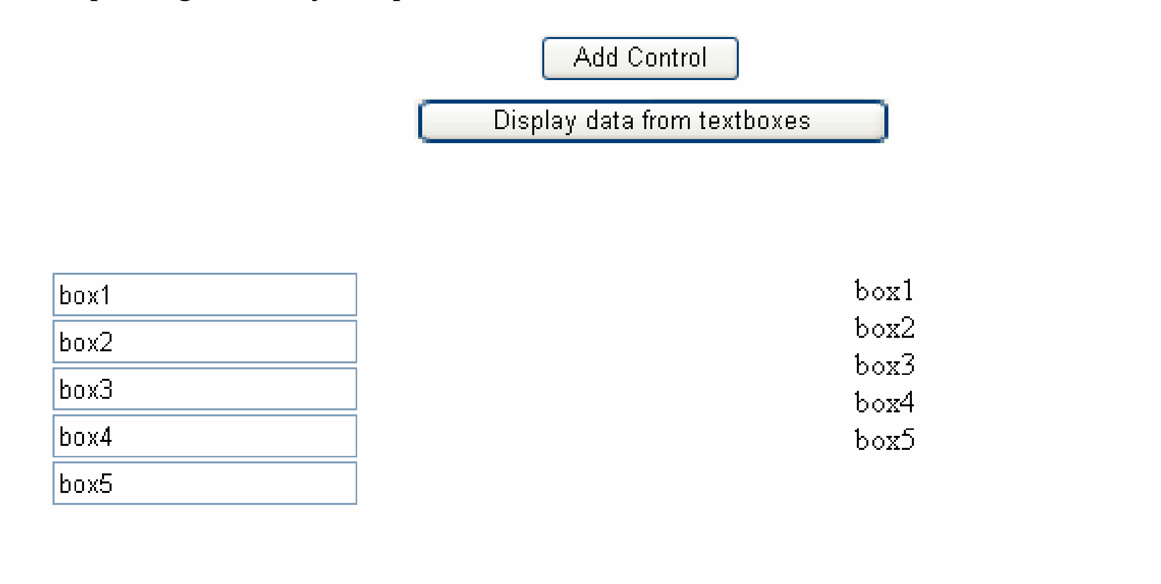
This will add and display five textboxes in PlaceHolder when the form is loaded. Note how each textbox is assigned a unique ID.

Add the following code to the "Display data from textboxes button":

```
for (int i = 1; i <= PlaceHolder1.Controls.Count; ++i)
{
	PlaceHolder2.Controls.Add(new LiteralControl(
	((TextBox)(this.FindControl( "TextBox" + i ))).Text));
	PlaceHolder2.Controls.Add(new LiteralControl("<BR>"));
}
```

One literal control for the contents of each textbox is displayed in PlaceHolder2. The unique ID that was assignment in Page_Load() is used to find the text to put into each literal control. The second LiteralControl is to add a page break between each display.

Run the project (set the main form as the Start Page), and the output should be similar to this depending on what you input:



Now you know how to create dynamic controls in C#/ASP.NET.

Exercise

Create a Visual Basic.NET ASP.NET web application that contains four buttons. The buttons perform the following four functions (and are labeled accordingly):

1. Display CheckBox
2. Display RadioButton
3. Display TextBox
4. Display Label

These controls are created dynamically via PlaceHolder(s)- the decision of whether to make one PlaceHolder or four is left to you.

Appendix 5

C# Lab Evaluation Survey

1. Using a scale of 1-5, please tell us how proficient you were in C# before you looked at our C# labs.

1—no knowledge
2—knew a little but had not programmed in C# before
3—had done some C# programming
4—experienced C# programmer
5—expert C# programmer

2. Please evaluate the labs you reviewed. For each lab, first tell us how much of the lab you completed by circling one of these codes.

A—did not review it
B—read it, but did no programming
C—read it and programmed at least one exercise
D—read it and programmed all exercises.

Then, for those labs that you did review, please rate on a scale of 1-5 how helpful that lab was in learning the subject material.

1—no value whatsoever
2—minimal value; it exposed me to the concept, but I still don't understand it
3—worthwhile; I understand the concept and can program it
4—valuable; I am now fluent in that concept
5—excellent; now I am the master of that concept

Then indicate whether you looked at the "answer" portion of that particular lab.

Lab	Your degree of review of this lab—circle one	Your rating of this lab—circle one	Did you use "answers"?
B1 - Getting Started	A B C D	1 2 3 4 5	Yes No
B2 - GUI Elements/Form Designer	A B C D		
B3 - Common Syntax	A B C D		
I1 - Creating ASP .NET webforms and introducing SQL	A B C D		
I2 - Interacting with MySQL databases using ADO .NET	A B C D		
I3 - Integration of C# with Excel	A B C D		

A1 - Creating an XML Web Service Using C#	A B C D		
A2 - Language Interoperability in C# using Web Services	A B C D		
A3 - Dynamic Control Creation in Web Applications using C#	A B C D		

3. With reference to any particular lab, how would you suggest that it be improved? Please comment on whether any part of the lab was unclear, or what portions might need elaboration or a better explanation.

Lab number (e.g., B1, I2, A3)	Suggestions for improvement
B1	
B2	
B3	
I1	
I2	
I3	
A1	
A2	

A3	

5. Do you think C# and the .NET platform are unique/effective enough to warrant studying in comparison with other languages adapted for remote servers/databases (yes/no)?

Yes No Don’t know

6. Are there C# and/or .NET topics not covered in the lab that you think should be covered, either within an existing exercise or in an entirely new exercise? If so, what topics?

7. What else would you like the author of the lab exercises to know?

Bibliography

[1] Deitel, H.M., P.J. Deitel, J. Listfield, and T.R. Nieto. (2001). *e-Business & e-Commerce: How to Program*. Upper Saddle River, New Jersey: Prentice-Hall Inc.

[2] Deitel, H.M., P.J. Deitel, J. Listfield, T.R. Nieto, C. Yaeger, and M. Zlatkina. (2002). *C#: How to Program*. Upper Saddle River, New Jersey: Prentice-Hall Inc.

[3] Fairbanks, Andrew, Justin LeCam, and Tyler McCabe. (2003, May). *Microsoft .NET: Understanding the.NET Framework and Developing Applications Using .NET*. The Journal of Computing in Small Colleges, Volume 18 Issue 5.

[4] Ruest, Nelson and Danielle Ruest. (2003, August). *Raising the Bar on Windows Security*. Windows Server System Magazine.

[5] Scott, Michael L. (2000). *Programming Language Pragmatics*. San Francisco, CA: Morgan Kauffmann Publishers.

[6] Storey, Neil. (1996). *Safety-Critical Computer Systems*. New York: Addison Wesley Longman.

[7] Suszynski, Marie E. (2003) *Microsoft: .NET Makes Integrating Systems Faster and Easier.* A. M. Best Company, Inc. (Comdex).