

Dot Net Independent Study
Under
Prof. Paul Wolfgang
By
Harish Ramachandra

TABLE OF CONTENTS

TABLE OF CONTENTS	2
TABLE OF CONTENTS	2
1 INDEPENDENT STUDY.....	3
1.1 LIMITATIONS AND DRAWBACKS IN CONTEMPORARY MS PRODUCTS	3
1.2 DOES JAVA OR OTHER TECHNOLOGY RESOLVE SOME OF THE PROBLEMS LISTED ABOVE?.....	3
1.3 FEATURES OF THE .NET FRAMEWORK.....	4
2 DATA TYPES, PARAMETER PASSING AND METHODS	7
2.1 PARAMETER PASSING.....	7
2.2 POLYMORPHISM.....	7
3 CASE STUDY OF SIMULATION OF TOWERS OF HANOI.....	9
3.1 C# TOPICS INTRODUCED	9
3.2 BACKGROUND.....	9
3.3 ISSUES INVOLVED	9
3.4 DESIGN	9
3.5 IMPLEMENTATION	10
3.6 SCREEN SHOT	21
4 CASE STUDY OF A SIMPLE WEB SERVER.....	22
4.1 C# TOPICS INTRODUCED	22
4.2 BACKGROUND	22
4.3 ISSUES INVOLVED	22
4.4 DESIGN	23
4.5 IMPLEMENTATION	24
4.6 EVENTS IN JAVA	30
5 WEB SERVICES.....	32
5.1 WHY DO WE NEED WEB SERVICES?.....	32
5.2 WHAT EXACTLY ARE WEB SERVICES?.....	32
5.3 HOW DO I WRITE WEB SERVICES IN .NET?	32
5.4 HOW DO I CONSUME A WEB SERVICE?.....	33
5.5 A SAMPLE WEB SERVICE.....	35
5.5.1 University Portal	37
5.5.2 TranscriptService Code	40
5.5.3 RecoService.....	43
6 POWER POINT PRESENTATION OF .NET	45

1 INDEPENDENT STUDY

The 2 main features of the .Net platform are the **Common language runtime** and the **Unified set of base classes**. Before we go deep into either one of the two features it would be helpful to see limitations, drawbacks existing in contemporary MS products which this new frameworks hopes to answer.

1.1 LIMITATIONS AND DRAWBACKS IN CONTEMPORARY MS PRODUCTS

- Trade off between development time and performance.
- Lack off standardization in API calls over languages.
- Programmer responsible for type safety and garbage collection, the programmer was expected to know what he is doing.
- Code cannot describe itself; code gets compiled directly into x86 code from which you cannot make out anything without a documentation accompanying the executable class.
- Different application development experience if you were developing a web application (web forms etc) from stand-alone application running over windows.
- Lack of trust when downloading executable from the net.
- Meager support for incorporating web services, a lot of *plumbing* has to go if you have to make use of services (methods) residing in objects on other servers now.
- Current focus is on object-oriented approach to programming but with software getting more and more complex a language, framework supporting component oriented programming is desired.

1.2 DOES JAVA OR OTHER TECHNOLOGY RESOLVE SOME OF THE PROBLEMS LISTED ABOVE?

In this section I will mainly be concentrating on Java to see if it can address some of the issues listed above.

API calls remain the same irrespective of the type of application (applets, application, servlets) that you are developing.

Garbage collection is implemented in the Java virtual machine, garbage collector runs as a low priority thread and checks the memory for un referenced memory locations which has still been allocated or for dangling pointers (object reference variables) which point to nothing and nothing points to it , if the garbage collector finds any of these then it reallocates this space to free memory.

In java details about a class can be known at run time that is if we get the byte code of a class from the network then we can programmatically find and manipulate public fields and methods this has been illustrated by two small java classes that are part of this document. The name of the class is supplied to the *find* during run time through command line, *find* lists the name , public fields , public methods of the

supplied class. This shows that the byte codes have some metadata describing the class they were generated from.

Since Java applets run over an operating system abstraction, the Java virtual machine, which interprets the byte code, a tight security is enforced. The easiest way to find out this by coding a java applet which tries to do file I/O and running it in a browser, JVM detects this and throughs a security exception and terminates the applet. You can still do file I/O from applets but only after getting digital certificates.

Sun Microsystems' Java Technology Web Services Model is an exciting new standard put forward by JAVA for developing web services. These new set of API's will allow java applications to send messages in XML to other Java apps or servers, it would also be possible for supporting RPC calls using these API over the network using SOAP protocol. This technology is still evolving.

Java Bean technology can be used for developing, deploying and using reusable software components. A Java Bean component is self-describing and can expose its properties, methods and events to any development tool. So in case you buy a Bean from a third party your development tool can peep into the component code and expose its properties, methods and events allowing you to customize and use its functionality in your application by dragging and dropping to component.

1.3 FEATURES OF THE .NET FRAMEWORK.

Common Language Runtime: The common language runtime provides a code-execution environment that manages code targeting the .NET Framework. Code management can take the form of memory management, thread management, security management, code verification and compilation, and other system services.

Managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces security in a way that enables users to trust that although an executable attached to an e-mail can play an animation on screen or sing a song, it cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally feature-rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS. This means that managed code can consume other managed classes, types, and objects, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime ensures that the most common types of software issues are solved or eradicated completely. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management eliminates the two most common application errors, memory leaks and invalid memory references.

The runtime, coupled with the CTS, also accelerates developer productivity. For example, programmers can use their favorite development language, being absolutely assured that they can still take full advantage of the runtime, the class library, and components written in other languages by

other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, thus greatly easing the migration process for existing applications.

Although the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. A feature called Just-In-Time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Internet Information Services (IIS) and Microsoft® SQL Server. This enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers.

Unified set of Base classes: The .NET Framework class library is a collection of reusable classes, or types, that tightly integrate with the common language runtime. The class library builds on the object-oriented nature of the runtime, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the learning curve associated with using a new piece of code. In addition, third-party components can integrate seamlessly with the classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces, which you can use to develop your own collection classes. Your collection classes will then blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios.

For example, you can use the .NET Framework to develop the following types of applications and services:

- Console applications
- Scripted or hosted applications
- Windows GUI applications (Windows Forms)
- ASP.NET applications
- Web Services
- Windows 2000 and Windows NT services

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you are writing an ASP.NET application or Web Service, on the other hand, you use different classes, such as the Web Forms classes.

In the next sections I will try to go over each of the application types using C# as the language targeting the framework classes I will also try to use Java technologies to create the same type of applications, if possible, while doing this I will highlight the similarity/ difference and also tell in which technology I felt more comfortable in developing applications. The comments I will be making here are strictly my own.

Some words on Web services: Web services according to me can best be described as pieces of business logic which are frequently used, complex, and need to be highly secure. These web services are consumed by web applications. An e.g. would be credit card validation, suppose a credit card company X creates this service which allows credit card validation, you have a store and you want customers to purchase items via the internet then you could write a an asp or jsp script which would collect user information from a web page and call company X's web service to do the validation and billing for you. Web Services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other Web Services. As a result, Web Services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web Services applications. Both Web Forms and Web Services use Internet Information Server (IIS) as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of Web Services applications. Web Services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (a Web Service Description Language). The .NET Framework conforms to these standards to promote interoperability with non-Microsoft solutions.

I shall be going in depth into web services in the last few chapters. Since .NET lays a lot of emphasis on production and consumption of web services I thought it would be good to give a preface of it here.

2 DATA TYPES, PARAMETER PASSING AND METHODS

Support for primitive data types is almost same in both Java and C#.

2.1 PARAMETER PASSING

Java has only two ways of passing parameters , By value for primitive data types and By reference for objects. C# on the other hand offers more variety in parameter passing. C# has the following ways of passing parameter.

1. Normal Object passing is by reference
2. Normal primitive data type passing is by value.
3. Primitive data type's can be passed by reference by preceding the variable with ref keyword (**ref** should also be present in function signature).
4. An array is also treated as an object, it is also passed as reference
5. There is another type of parameter passing in which information is passed from the called method to calling method; this type of method passing is preceded by the keyword **out**. What actually happens is that an unassigned reference is created on the stack, the actual assignment has to take place in the called methods and when control returns to the calling program the contents of the stack are copied to the actual variable.
6. Another new method of passing parameters is the "**params**", this is usually a single dimension array, the novelty of this is that the calling program can pass any number of arguments to the called method, the called method could iterate through the params parameter for each argument. The changes made to params parameter won't be visible to the calling method.

All these different ways of parameter passing is illustrated in the sample programs, Method class.

Parameter passing in java is also illustrated with sample programs.

Static methods, which can be called without instantiation of a class , is supported by both Java and C#.

2.2 POLYMORPHISM

Refers to same name having different meaning,

Static Polymorphism: - This occurs during compilation time when the compiler has to choose one of the different implementation of the method , this it generally does by matching the signature of the calling method to all the implementation of the same method. Both Java and C# support this.

Dynamic Polymorphism: - This occurs during run time when the run time decides which version of the method to call depending on the object referenced by the reference variable (A parent reference type can hold reference to a child). Both Java and C# support but the important thing to note here is Java resorts to dynamic polymorphism always whereas C# resorts to dynamic polymorphism only if specified to do so (using keyword **virtual**). These things are illustrated in the sample code.

In Java whenever a private variables value was to be read or set then it could only be done by calling one of the appropriate methods. In C# the same thing can be done by using properties get for reading and set for writing into private variables.

3 CASE STUDY OF SIMULATION OF TOWERS OF HANOI

From now on my independent study is going to describe the features of .NET and C# with the help of a small project

3.1 C# TOPICS INTRODUCED

In this small project I have introduced C# *windows.Forms namespace*, which contains all the classes for GUI manipulation. I have also introduced *Threads namespace*, which contains classes for thread creation and manipulation.

3.2 BACKGROUND

Towers of Hanoi is a very good example of a recursive program. But like all recursive algorithms extremely hard to visualize what is actually happening. Our objective was to provide graphic illustration of actually movement of disks from one peg to another so as to help students understand the program better.

3.3 ISSUES INVOLVED

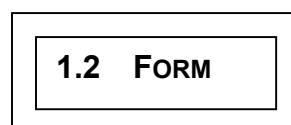
One of the major parts of this simulation is changing position of the disk at the right intervals of the program. Creating separate thread of execution for the simulation so that the application responds to user generated events even while the simulation is running.

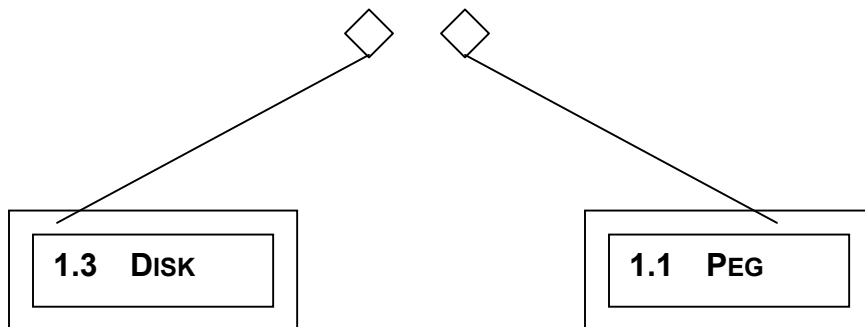
3.4 DESIGN

The application consists of 3 main classes

- The **disk** class, which is an abstraction for disks stacked up one over the other, they have properties and methods quite similar to actual disks (like color, size etc).
- The **peg** class, which is an abstraction for the “real world” peg with behavior to add disks to itself and also to remove them.
- The **form** class which acts as a container for all these components and contains the towers of Hanoi function , which runs as a separate thread from the main thread, this function is responsible for visual manipulation of the disks at proper steps in the towers of Hanoi algorithm.

Class diagram





An important design step in this application is to derive the **disk** and the **peg** class from the **Icomponent** class, it from this class all windows visual objects (buttons, textboxes etc) are derived. The advantage of deriving from the **Icomponent** is that I didn't have to write code for explicitly painting and re-painting the form canvas, a simple call **change Location** derived method of **disk** or **pegs** does all the job of erasing the previous location and repainting the object at a different position. In addition to this these objects (**disk** and **peg**) appear in the component toolkit present on the left side of the designer workplace, hence allowing for easier drag and drop.

In **C#** threads are implemented by specifying the function which has to be run as a separate thread during thread creation time. The threads are controlled by methods like **start()**, **stop()**, **suspend()**, **resume()** etc.

3.5 IMPLEMENTATION

Code for **disk** class

```
Using System;
using System.Windows.Forms;
using System.Drawing;

namespace towersofhanoi
{
    /// <summary>
    ///This class represents the disk in the towers of hanoi. The
    ///disk will be a visual component which can
    ///be positioned using IDE and also programmatically, this will
    ///allow us to move disks from one peg to another
    /// </summary>

    public class disk : UserControl
    {
        private System.Windows.Forms.Label disknumber;

        private void InitializeComponent()
        {
            this.disknumber = new System.Windows.Forms.Label();
            this.SuspendLayout();
            // disknumber
            this.disknumber.Location = new System.Drawing.Point(40, 10);
        }
    }
}
```

```
this.disknumber.Name = "disknumber";
this.disknumber.Size = new System.Drawing.Size(50, 10);
this.disknumber.TabIndex = 0;
this.disknumber.Text = "Disk#";
// disk
this.Controls.AddRange(new System.Windows.Forms.Control[] {
this.disknumber});
this.Name = "disk";
this.Size = new System.Drawing.Size(100, 20);
this.Load += new System.EventHandler(this.disk_Load);
this.ResumeLayout(false);
}

private void InitializeComponent(int scale)
{
    this.disknumber = new System.Windows.Forms.Label();
    this.SuspendLayout();
    // disknumber
    this.disknumber.Location = new System.Drawing.Point(32, 10);
    this.disknumber.Name = "disknumber";
    this.disknumber.Size = new System.Drawing.Size(50, 20);
    this.disknumber.TabIndex = 0;
    this.disknumber.Text = ""+(scale+1);
    this.disknumber.TextAlign =
    System.Drawing.ContentAlignment.TopLeft;
    // disk
    this.Controls.AddRange(new System.Windows.Forms.Control[] {
    this.disknumber});
    this.Name = "disk"+(scale+1);
    this.Size = new Size(15*scale+100,25);
    this.Load += new System.EventHandler(this.disk_Load);
    this.ResumeLayout(false);
}

public disk()
{
    this.InitializeComponent();
}

public disk(int scale)
{
    this.InitializeComponent(scale);
}

public void setColor(System.Drawing.Color temp)
{
    this.BackColor = temp;
}

private void disk_Load(object sender, System.EventArgs e)
{
}
}
```

```
}
```

Code for **peg** class

```
using System;
using System.Windows.Forms;
namespace towersofhanoi
{
    /// <summary>
    ///This is an abstractions for the Pegs over which the disks will be
    put.
    ///We make this a visual component so that is can be manipulated
    inside the /// IDE.
    /// </summary>

    public class Peg : UserControl
    {
        // this value gives the integer value from where the
        private int topofpeg;
        private void InitializeComponent()
        {
            // Peg
            this.Name = "Peg";
            this.Size = new System.Drawing.Size(8, 368);
        }

        public Peg()
        {
            this.InitializeComponent();
        }

        // this is to change color to distinguish between various pegs

        public void addDisk(disk temp)
        {
            int prevHeight = temp.Size.Height;
            temp.Location = new System.Drawing.Point(this.Location.X -
            temp.Size.Width/2)+this.Size.Width/2,topofpeg);
            topofpeg = topofpeg - prevHeight ;
        }

        public void removeDisk(disk temp)
        {
            topofpeg = topofpeg + temp.Size.Height;
        }

        public void setTopofpeg(int val)
        {
            topofpeg = val;
        }
    }
}
```

```
    }  
}
```

Code for **form1** class

```
using System;  
using System.Drawing;  
using System.Collections;  
using System.ComponentModel;  
using System.Windows.Forms;  
using System.Data;  
using System.Threading;  
  
namespace towersofhanoi  
{  
    /// <summary>  
    /// Summary description for Form1.  
    /// </summary>  
    ///  
  
    public class Form1 : System.Windows.Forms.Form  
    {  
        /// <summary>  
        /// Required designer variable.  
        /// </summary>  
        private int n;  
        private bool singlestep, slow, stop;  
        private Thread t;  
        private disk[] disklot;  
        private Color[] c = new  
  
        Color[] {Color.Beige, Color.Blue, Color.Chocolate, Color.DarkKhaki, Color.Dar  
        kGreen, Color.DarkViolet, Color.Turquoise, Color.Cyan, Color.DarkOrange, Colo  
        r.BlanchedAlmond};  
  
        private towersofhanoi.Peg Source;  
        private System.Windows.Forms.Label label1;  
        private towersofhanoi.Peg Auxillary;  
        private System.Windows.Forms.Label label2;  
        private towersofhanoi.Peg Destination;  
        private System.Windows.Forms.Label label3;  
        private System.Windows.Forms.MainMenu mainMenu1;  
        private System.Windows.Forms.MenuItem menuItem1;  
        private System.Windows.Forms.MenuItem menuItem2;  
        private System.Windows.Forms.MenuItem menuItem3;  
        private System.Windows.Forms.MenuItem menuItem4;  
        private System.Windows.Forms.MenuItem menuItem5;  
        private System.Windows.Forms.MenuItem menuItem6;  
        private System.Windows.Forms.Button move;  
        private System.Windows.Forms.MenuItem menuItem7;
```

```
private System.Windows.Forms.MenuItem menuItem8;
private System.Windows.Forms.MenuItem menuItem9;
private System.ComponentModel.Container components = null;

public void Initialize()
{
    this.InitializeComponent();
    n =4;
    this.InitializePegsDisks();
}

public Form1()
{
    // Required for Windows Form Designer support
    Initialize();
    // TODO: Add any constructor code after InitializeComponent call
}

/// <summary>
/// Clean up any resources being used.
/// </summary>

protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.label3 = new System.Windows.Forms.Label();
    this.menuItem8 = new System.Windows.Forms.MenuItem();
    this.menuItem9 = new System.Windows.Forms.MenuItem();
    this.menuItem4 = new System.Windows.Forms.MenuItem();
    this.menuItem5 = new System.Windows.Forms.MenuItem();
    this.menuItem6 = new System.Windows.Forms.MenuItem();
    this.menuItem7 = new System.Windows.Forms.MenuItem();
    this.menuItem1 = new System.Windows.Forms.MenuItem();
    this.menuItem2 = new System.Windows.Forms.MenuItem();
    this.menuItem3 = new System.Windows.Forms.MenuItem();
    this.mainMenu1 = new System.Windows.Forms.MainMenu();
    this.Source = new towersofhanoi.Peg();
}
```

```
this.Auxillary = new towersofhanoi.Peg();
this.move = new System.Windows.Forms.Button();
this.Destination = new towersofhanoi.Peg();
this.label1 = new System.Windows.Forms.Label();
this.label2 = new System.Windows.Forms.Label();
this.SuspendLayout();
// label3
this.label3.BackColor = System.Drawing.SystemColors.Control;
this.label3.Location = new System.Drawing.Point(744, 384);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(256, 24);
this.label3.TabIndex = 1;
this.label3.Text = "Destination";
this.label3.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
// menuItem8
this.menuItem8.Index = 5;
this.menuItem8.Text = "Resume";
this.menuItem8.Click += new System.EventHandler(this.menuItem8_Click);
// menuItem9
this.menuItem9.Index = 0;
this.menuItem9.Text = "Number of Disks";
this.menuItem9.Click += new System.EventHandler(this.menuItem9_Click);
//
// menuItem4
//
this.menuItem4.Index = 3;
this.menuItem4.Text = "SingleStep";
this.menuItem4.Click += new System.EventHandler(this.menuItem4_Click);
// menuItem5
this.menuItem5.Index = 6;
this.menuItem5.Text = "Reset";
this.menuItem5.Click += new System.EventHandler(this.menuItem5_Click);
this.menuItem6.Index = 7;
this.menuItem6.Text = "Exit";
this.menuItem6.Click += new System.EventHandler(this.menuItem6_Click);
this.menuItem7.Index = 4;
this.menuItem7.Text = "Stop";
this.menuItem7.Click += new System.EventHandler(this.menuItem7_Click);
// menuItem1
this.menuItem1.Index = 0;
this.menuItem1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
this.menuItem9, this.menuItem2, this.menuItem3, this.menuItem4, this.menuItem7
,
this.menuItem8, this.menuItem5, this.menuItem6});
this.menuItem1.Text = "Towers of Hanoi";
// menuItem2
this.menuItem2.Index = 1;
this.menuItem2.Text = "Play";
this.menuItem2.Click += new System.EventHandler(this.menuItem2_Click);
// menuItem3
this.menuItem3.Index = 2;
this.menuItem3.Text = "Slow";
this.menuItem3.Click += new System.EventHandler(this.menuItem3_Click);
// mainMenu1
this.mainMenu1.MenuItems.AddRange(new System.Windows.Forms.MenuItem[] {
this.menuItem1});
this.Source.BackColor = System.Drawing.Color.Black;
```

```
this.Source.Location = new System.Drawing.Point(152, 24);
this.Source.Name = "Source";
this.Source.Size = new System.Drawing.Size(8, 360);
this.Source.TabIndex = 0;
this.Auxillary.BackColor = System.Drawing.Color.Black;
this.Auxillary.Location = new System.Drawing.Point(512, 24);
this.Auxillary.Name = "Auxillary";
this.Auxillary.Size = new System.Drawing.Size(8, 360);
this.Auxillary.TabIndex = 0;
this.move.BackColor = System.Drawing.SystemColors.Control;
this.move.Location = new System.Drawing.Point(200, 8);
this.move.Name = "move";
this.move.Size = new System.Drawing.Size(48, 16);
this.move.TabIndex = 2;
this.move.Text = "Move";
this.move.Click += new System.EventHandler(this.button1_Click);
this.Destination.BackColor = System.Drawing.Color.Black;
this.Destination.Location = new System.Drawing.Point(880, 24);
this.Destination.Name = "Destination";
this.Destination.Size = new System.Drawing.Size(8, 360);
this.Destination.TabIndex = 0;
this.label1.BackColor = System.Drawing.SystemColors.Control;
this.label1.Location = new System.Drawing.Point(8, 384);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(288, 24);
this.label1.TabIndex = 1;
this.label1.Text = "Source";
this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
this.label2.BackColor = System.Drawing.SystemColors.Control;
this.label2.Location = new System.Drawing.Point(376, 384);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(272, 24);
this.label2.TabIndex = 1;
this.label2.Text = "Auxillary";
this.label2.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;
//
// Form1
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.BackColor = System.Drawing.SystemColors.Window;
this.ClientSize = new System.Drawing.Size(1028, 617);
this.Controls.AddRange(new System.Windows.Forms.Control[] {
this.move,
this.label3,
this.Destination,
this.label2,
this.Auxillary,
this.label1,
this.Source});
this.Menu = this.mainMenu1;
this.Name = "Form1";
this.Text = "Form1";
this.WindowState = System.Windows.Forms.FormWindowState.Maximized;
this.Load += new System.EventHandler(this.Form1_Load);
this.ResumeLayout(false);

}
```



```
#endregion

// This code is for stacking up the "n" disks on top of each other on the
// Source peg

private void InitializePegsDisks()
{
    singlestep = false;
    slow = false;
    stop = false;
    move.Visible = false;
    disklot = new disk[n];
    for (int i =0;i<n;i++)
    {
        disklot[i] = new disk(i);
        disklot[i].setColor(c[(i%10)]);
    }
    Source.setTopofpeg(Source.Size.Height);
    Auxillary.setTopofpeg(Auxillary.Size.Height);
    Destination.setTopofpeg(Destination.Size.Height);
    // Add disk controls to the form
    for (int i=0;i<n;i++)
    {
        this.Controls.Add(disklot[i]);
    }
    // Stack the disks on top of the Source peg
    for(int i=n-1;i>=0;i--)
    {
        Source.addDisk(disklot[i]);
    }

    t = new Thread( new ThreadStart(doJob));
}

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]

public void doJob()
{
    towers(n-1, Source, Destination, Auxillary);
}

public void towers(int n, Peg Source, Peg Destination, Peg Auxillary)
{
    if (n == 0 )
    {
        Source.removeDisk(disklot[n]);
        Destination.addDisk(disklot[n]);
        if (slow)
        {
            Thread.Sleep(1000);
        }
    }
}
```

```
    }
    if (singlestep)
    {
        t.Suspend();
    }
}

else
{
    towers(n-1, Source, Auxillary, Destination);
    Source.removeDisk(disklot[n]);
    Destination.addDisk(disklot[n]);
    if (slow)
    {
        Thread.Sleep(1000);
    }
    if (singlestep)
    {
        t.Suspend();
    }

    towers(n-1, Auxillary, Destination, Source);
}
}

public void removeAllDisk()
{
    for(int i=0;i<disklot.Length;i++)
    {
        disklot[i].Dispose();
    }
}

static void Main()
{
    Application.Run(new Form1());
}

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

private void menuItem2_Click(object sender, System.EventArgs e)
{
    slow = false;
    stop = false;
    singlestep = false;
    if (t.IsAlive)
        t.Resume();
    else
        t.Start();
}

private void reset()
```

```
{
    this.removeAllDisk();
    if(singlestep || stop)
        t.Resume();
    if (slow )
        t.Interrupt();
    t.Abort();
    this.InitializePegsDisks();
}

private void reset(int temp)
{
    this.removeAllDisk();
    if(singlestep || stop)
        t.Resume();
    if (slow )
        t.Interrupt();
    t.Abort();
    n = temp;
    this.InitializePegsDisks();
}

private void menuItem5_Click(object sender, System.EventArgs e)
{
    this.reset();
}

private void menuItem6_Click(object sender, System.EventArgs e)
{
    if(singlestep || stop)
        t.Resume();
    if (slow && !stop)
        t.Interrupt();
    t.Abort();
    this.Dispose();
}

private void menuItem4_Click(object sender, System.EventArgs e)
{
    stop = false;
    slow = false;
    singlestep = true;
    move.Visible = true;
    if (!t.IsAlive)
        t.Start();
}
```

```
private void button1_Click(object sender, System.EventArgs e)
{
    if (t.IsAlive)
        t.Resume();
}

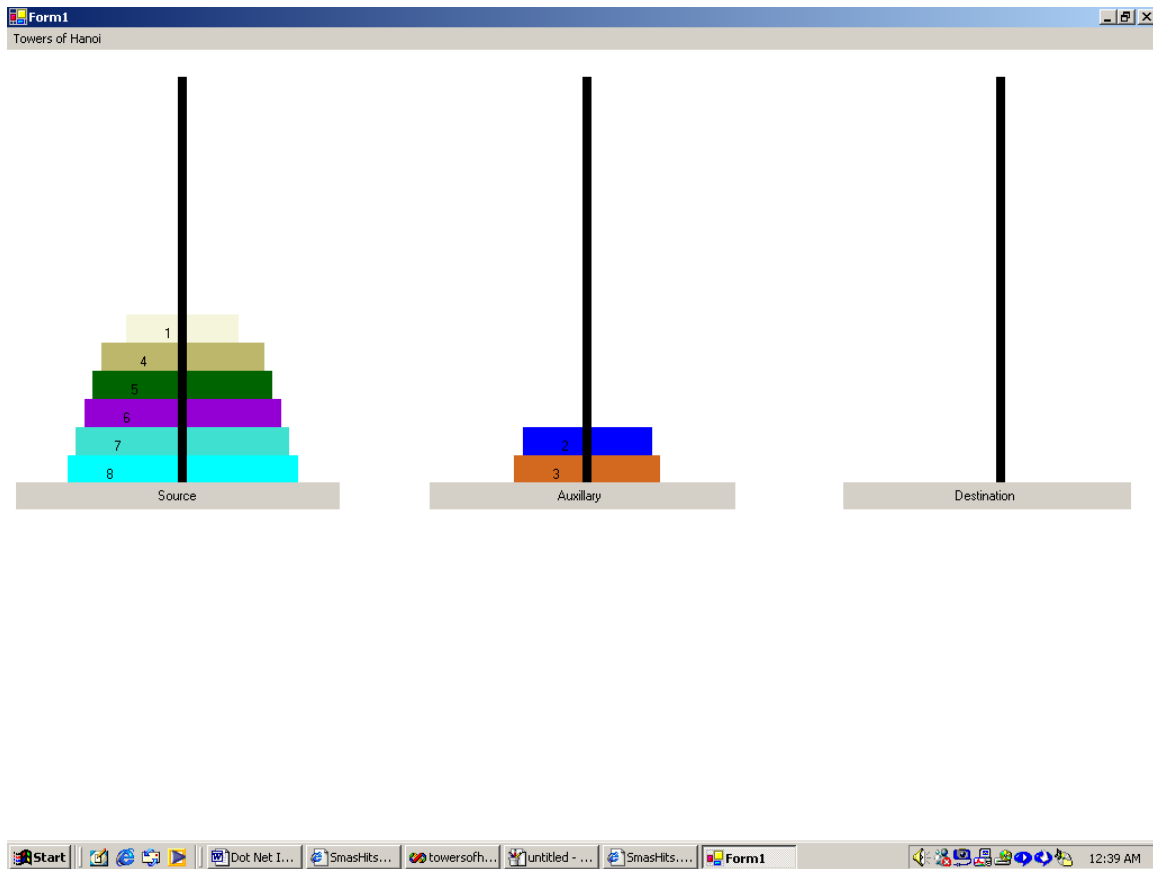
private void menuItem3_Click(object sender, System.EventArgs e)
{
    slow = true;
    stop = false;
    if (t.IsAlive)
        t.Resume();
    else
        t.Start();
}

private void menuItem7_Click(object sender, System.EventArgs e)
{
    singlestep = false;
    slow = false;
    stop = true;
    if (t.IsAlive)
        t.Suspend();
}

private void menuItem8_Click(object sender, System.EventArgs e)
{
    if(stop)
    {
        stop = false;
        if (t.IsAlive)
            t.Resume();
    }
    singlestep = false;
    slow = true;
}

private void menuItem9_Click(object sender, System.EventArgs e)
{
    int temp;
    Form2 dialog = new Form2();
    dialog.ShowDialog(this);
    if (dialog.DialogResult == DialogResult.OK)
    {
        temp = int.Parse(dialog.msgstr);
        this.reset(temp);
    }
}
}
```

3.6 SCREEN SHOT



4 CASE STUDY OF A SIMPLE WEB SERVER

4.1 C# TOPICS INTRODUCED

In this project I will be introducing the *.NET sockets namespaces*, which contains classes for standard socket interface. In addition to this, I will be introducing *delegate*, which are a nice (and fancy) way of implementing *call back*. The term *call back* refers to the property by which an object informs the object, which contains (container for it) it about an event. An easy way of understanding this is to think how a button will inform the form, which contains this button, about a mouse click on it (button).

4.2 BACKGROUND

The web server that I have coded for this chapter does what a web server is supposed to do, that is serve web pages!!!!!!!!!!!!. This web server just serves static html files. This seems trite so to make things a little bit more interesting I also display the connection history, that is from where did the request come from and also which file was requested. This web server will listen on port 1000. This server will serve pages in c:\mywebroot directory. Again the purpose of this project is to explore the language (in fact the features of the framework).

4.3 ISSUES INVOLVED

To implement any type of server we have to create a socket and bind it to a port, which is 1000 in our case. The net.sockets provides a specialized socket for this purpose called as TcpListener, we have to supply it with the port number to its constructor during object creation time. The acceptsocket returns a socket for each new connection. At this point if we start servicing then we make connections, which are pending wait a long time thus reducing the overall throughput, so what we do is create an object of type serviceconnection to service this connection. Each active serviceconnection (or in other words each active connection) is maintained in a collection (a list).

When service connection finishes it should inform the main class which in our case is the *form*, which should then remove this serviceconnection object from this list (since its no longer active) and at the same time display on the screen that this connection has been serviced.

How do we do this ????????

To this we use delegates, which can be considered as *type safe function pointers*. The important thing to remember is that delegate is a data type describing the type of data it can handle e.g.

```
public delegate void removeme(serviceconnection c);
```

This statement declares that removeme as a delegate , the variables of this will contain references to function which have signature *func_name(serviceconnection)*.

This statement has to be public otherwise it serves little or no use because this delegate has to be visible to the container class.

This is how the container object informs the contained object of which function to call back when its job is done

```
serviceconnection.removeem rm = new serviceconnection.removeem(this.removeive);  
// created a object of type removeem delegate and assign the //reference of function removeive to it.
```

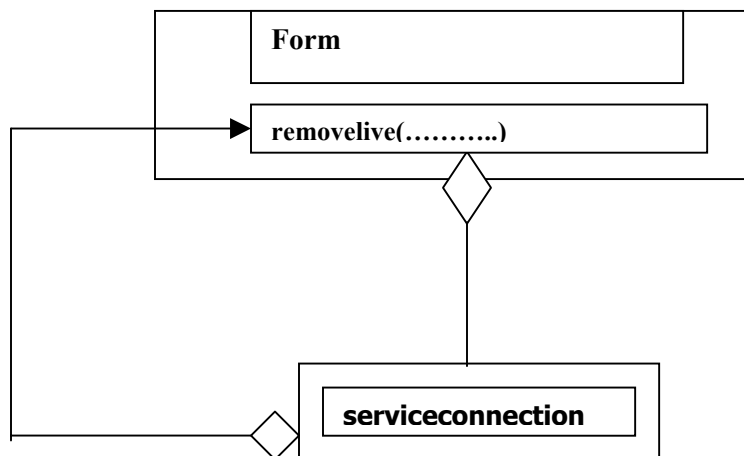
```
serviceconnection tempser = new serviceconnection();  
// pass the delegate to the serviceconnection object  
tempser.service(skt,rm);
```

4.4 DESIGN

The project contains 2 classes

- The **Form** class which contains visual objects with which you can control the operation of the server, it also contains a text area which displays the connection history. This class contains the socket, which listens to port 1000 and instantiates a serviceconnection object to service every new connection.
- The **serviceconnection** class, which does the actual job of reading from the file and writing it to the socket. After it has finished the job of writing the file it informs its container that the client has been serviced.

Class Diagram



4.5 IMPLEMENTATION

The **serviceconnection** class

```
using System;
using System.Net.Sockets;
using System.IO;
using System.Collections;
using System.Threading;

namespace simplewebserver
{
    /// <summary>
    /// Summary description for serviceconnection.
    /// </summary>
    public class serviceconnection
    {

        private Socket skt;
        private Thread t;
        private string file,identify;
        public delegate void removeme(serviceconnection c);
        private serviceconnection.removeme informbreak;
        // This is the private function which does the job of reading from the
        // file and sending it to the client;
        private void readfile(string file,StreamWriter stw)
        {
            try
            {
                file = "c:\\mywebroot\\"+file;
                FileStream rfs = new
                    FileStream(file, FileMode.Open, FileAccess.Read);
                StreamReader str = new StreamReader(rfs);
                char [] buffer = new Char[rfs.Length];
                str.Read(buffer, 0, (int) rfs.Length);
                stw.Write(buffer);
            }
            catch(System.Exception e)
            {
                stw.Write(" Error == "+ e.Message);
            }
        }

        public void doJob()
        {
            try
            {
                NetworkStream nws = new NetworkStream(skt);
                StreamReader str = new StreamReader( nws);
            }
        }
    }
}
```



```
        StreamWriter stw = new StreamWriter(nws);
        file = (str.ReadLine().Replace("GET /", "").Replace("
HTTP/1.1", ""));
        readfile(file, stw);
        stw.Flush();
        skt.Close();
    }
    catch(System.Exception e)
    {
    }
    // inform master that I have finished servicing
    informbreak(this);
    // Stop the thread after the service is over
    t.Abort();
}

public void service(Socket tempskt, serviceconnection.removeme proc)
{
    this.skt = tempskt;
    this.informbreak= proc;
    t = new Thread(new ThreadStart(this.doJob));
    identify = skt.RemoteEndPoint.ToString();
    t.Start();
}

public string remotehost()
{
    return (file+"-----"+identify);
}

public serviceconnection()
{
    //
    // TODO: Add constructor logic here
    //
}

public void stopThread()
{
    if (t.IsAlive)
        t.Abort();
}
}
}
```

The **Form** class

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Threading;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace simplewebserver
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private ArrayList live_connection = new ArrayList();
        private Thread t;
        private TcpListener mainsock ;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.ListBox connection_view;
        private System.Windows.Forms.Button Start;
        private System.Windows.Forms.Button Stop;
        private System.ComponentModel.IContainer components = null;

        public Form1 ()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();
            // Create a new TCP socket object
            mainsock= new TcpListener(1000);

            t = new Thread(new ThreadStart(this.doJob));
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            mainsock.Stop();
            t.Abort();
            if( disposing )
            {

```

```
        if(components != null)
        {
            components.Dispose();
        }
    }

    base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
    this.Start = new System.Windows.Forms.Button();
    this.labell1 = new System.Windows.Forms.Label();
    this.Stop = new System.Windows.Forms.Button();
    this.connection_view = new
        System.Windows.Forms.ListBox();
    this.SuspendLayout();
    this.Start.Location = new System.Drawing.Point(168, 40);
    this.Start.Name = "Start";
    this.Start.TabIndex = 6;
    this.Start.Text = "START";
    this.Start.Click += new
        System.EventHandler(this.Start_Click);

    this.labell1.Location = new System.Drawing.Point(32, 72);
    this.labell1.Name = "labell1";
    this.labell1.Size = new System.Drawing.Size(448, 24);
    this.labell1.TabIndex = 4;
    this.labell1.Text = "Status of connection";
    this.labell1.Click += new
        System.EventHandler(this.labell1_Click);

    this.Stop.Enabled = false;
    this.Stop.Location = new System.Drawing.Point(280, 40);
    this.Stop.Name = "Stop";
    this.Stop.TabIndex = 7;
    this.Stop.Text = "STOP";
    this.Stop.Click += new
        System.EventHandler(this.Stop_Click);
    this.connection_view.Location = new
        System.Drawing.Point(32, 96);
    this.connection_view.Name = "connection_view";
    this.connection_view.Size = new System.Drawing.Size(472,
        95);
    this.connection_view.TabIndex = 5;
    //
    // Form1
    //
    this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
    this.ClientSize = new System.Drawing.Size(512, 197);
    this.Controls.AddRange(new System.Windows.Forms.Control[]
{
```

```
        this.Stop,
        this.Start,
        this.connection_view,
            this.label1});
    this.Name = "Form1";
    this.Text = "A Simple web server";
    this.Load += new System.EventHandler(this.Form1_Load);
    this.ResumeLayout(false);
}
#endregion

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

private void removelive(serviceconnection tv)
{
    this.connection_view.Items.Add("Disconnected
        "+tv.remotehost());
    live_connection.Remove(tv);
}

private void addlive(serviceconnection tv)
{
    this.connection_view.Items.Add("Connected
        "+tv.remotehost());
    live_connection.Add(tv);
}

// This method runs in a thread and it responsible for
// continuously
// listening for client connection

public void doJob()
{
    while(true)
    {
        Socket skt = mainsock.AcceptSocket();
        // create a delegate pointing to the function which
        // should be triggered when
        // the connection has been service
        serviceconnection.removeme rm = new
            serviceconnection.removeme(this.removelive);
        serviceconnection tempser = new
            serviceconnection();
        // service the current request
        tempser.service(skt, rm);
        addlive(tempser);
    }
}
}
```

```
        }
    }

    public static void Main()
    {
        Application.Run(new Form1());
    }

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

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

    private void Start_Click(object sender, System.EventArgs e)
    {
        mainsock.Start();
        t = new Thread(new ThreadStart(doJob));
        t.Start();
        Start.Enabled = false;
        Stop.Enabled = true;
    }

    private void Stop_Click(object sender, System.EventArgs e)
    {
        mainsock.Stop();
        t.Abort();
        Start.Enabled = true;
        Stop.Enabled = false;
        object[] temp = new object[live_connection.Count];
        temp = live_connection.ToArray();
        for(int i=0;i<live_connection.Count;i++)
            ((serviceconnection)temp[i]).stopThread();
        live_connection.RemoveRange(0,live_connection.Count);
    }

    private void labell1_Click(object sender, System.EventArgs e)
    {
    }
}
}
```

4.6 EVENTS IN JAVA

The interface

```
public interface myeventlistener {  
    public void eventfired();  
}
```

The class generating the events

```
public class eventgenerator {  
    myeventlistener evl;  
  
    /** Creates new eventgenerator */  
    public eventgenerator() {  
    }  
  
    // event listener register with event generator  
    public void addlistener(myeventlistener temp)  
    {  
        evl = temp;  
    }  
  
    private void broadcast() {  
        if (evl != null)  
            evl.eventfired();  
    }  
  
    public void run()  
    {  
        for(int i=0;i<10;i++)  
        {  
            if(i%2 == 0)  
            {  
                System.out.println("From eventgenerator , I am now firing my event");  
                broadcast();  
            }  
            System.out.println(i);  
        }  
    }  
}
```

```
}
```

container class

```
public contain{  
  
    public contain () {  
        abc = new eventgenerator();  
        abc.addlistener(this);  
        abc.run();  
    }  
  
    public void eventfired() {  
        System.out.println(" Yes I got the event from eventgenerator");  
    }  
  
    public static void main (String[] args)  
    {  
        new contain();  
    }  
}
```

output from running this

```
From eventgenerator , I am now firing my event  
Yes I got the event from eventgenerator  
0  
1  
From eventgenerator , I am now firing my event  
Yes I got the event from eventgenerator  
2  
3  
From eventgenerator , I am now firing my event  
Yes I got the event from eventgenerator  
4  
5  
From eventgenerator , I am now firing my event  
Yes I got the event from eventgenerator  
6  
7  
From eventgenerator , I am now firing my event  
Yes I got the event from eventgenerator  
8  
9
```

5 WEB SERVICES

Web Services are units of Application logic, which can be targeted programmatically. Suppose you start building an application and realize that you don't have to write all the code from scratch because there are already small blocks of code distributed around the world, which you can use. If the blocks of code you wanted to use in your application were implemented as web services then you could use them in your application as if they were classes residing on your own system.

5.1 WHY DO WE NEED WEB SERVICES?

- No need to reinvent the wheel makes software reuse feasible. Allows for structured programming in web applications.
- A trusted party must do privileged tasks like credit card validation but this functionality is used by many applications. In such cases the trusted party could implement credit card validation as a web services for other developers to consume.
- To allow a wide variety of incompatible systems connected on the Internet use and share application logic.
- To some extent can help prevent abuse of intellectual property.

5.2 WHAT EXACTLY ARE WEB SERVICES?

Web Services enable people to use code or rather application logic residing on remote system as if they were residing on their own. So is that all? What's so great about web services then?

The problem seems innocuous but there are a whole lot of issues involved some of which are.

The language in which the web service is written in may be different then the language which consuming of this service is using. Different languages are rarely compatible in their data-types.

The parameters that are passed to and fro from the service may be complex and may need information to describe itself.

How will people on the Internet know about services being offered?

Etc.....

5.3 HOW DO I WRITE WEB SERVICES IN .NET?

Writing web services in .NET is really easy. The class, which contains method(s), which you want to expose to other, should derive from “System.Web.Services.WebService”. In addition to this the methods/services that you want expose should be preceded by the keyword [Web Method].

That is all there is to it, creating your web services is that simple. Visual studio behind the scenes does a whole of work by integrating your web service with IIS.

Web Services can be accessed by their URL.

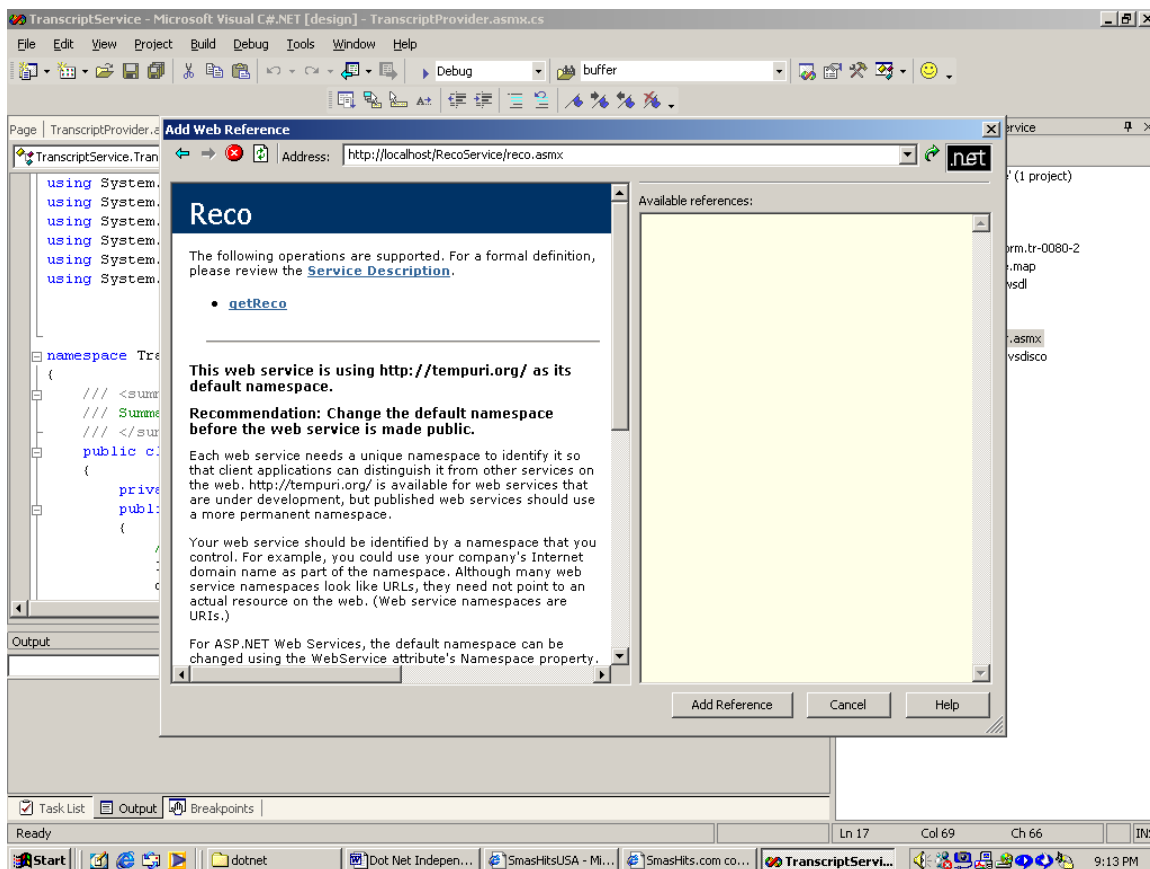
5.4 HOW DO I CONSUME A WEB SERVICE?

There are a few things that you have to do before you can start using a web service. You have to download a SOAP contract file from the URL of the web service. The wsdl (SOAP contract file) describes in XML syntax the exposed methods, the means to call it, the data-types it takes and returns etc.

The wsdl is converted to a proxy class (a stub) which contains the interfaces of the web methods and knowledge of how to call the actual web service. This proxy class is used in our application to access the web service.

A proxy class can be thought of as a layer which abstracts the parsing of raw XML and supplying to our code the relevant data.

Snapshot of adding a proxy



A look at a proxy class.....

```
//-----  
// <autogenerated>  
// This code was generated by a tool.  
// Runtime Version: 1.0.2914.16  
//  
// Changes to this file may cause incorrect behavior and will be lost if  
// the code is regenerated.  
// </autogenerated>  
//-----  
  
namespace TranscriptService.edu.temple.dorm.tr_0080_2 {  
    using System.Diagnostics;  
    using System.Xml.Serialization;  
    using System;  
    using System.Web.Services.Protocols;  
    using System.Web.Services;  
  
    [System.Web.Services.WebServiceBindingAttribute(Name="Service1Soap", Namespace="http://tr-0080-  
2.dorm.temple.edu/UniversityPortal/")]  
  
    public class Service1 : System.Web.Services.Protocols.SoapHttpClientProtocol {  
  
        [System.Diagnostics.DebuggerStepThroughAttribute()]  
        public Service1() {  
            this.Url = "http://tr-0080-2.dorm.temple.edu/UniversityPortal/Service1.asmx";  
        }  
  
        [System.Diagnostics.DebuggerStepThroughAttribute()]  
        [System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://tr-0080-  
2.dorm.temple.edu/UniversityPortal/Authentincate", RequestNamespace="http://tr-0080-  
2.dorm.temple.edu/UniversityPortal/", ResponseNamespace="http://tr-0080-  
2.dorm.temple.edu/UniversityPortal/",  
        Use=System.Web.Services.Description.SoapBindingUse.Literal,  
        ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]  
  
        public bool Authentincate(string Student_ID, string Password) {  
            object[] results = this.Invoke("Authentincate", new object[] {  
                Student_ID,  
                Password});  
            return ((bool)(results[0]));  
        }  
  
        [System.Diagnostics.DebuggerStepThroughAttribute()]
```

```
public System.IAsyncResult BeginAuthenticate(string Student_ID, string Password,
System.AsyncCallback callback, object asyncState) {
    return this.BeginInvoke("Authenticate", new object[] {
        Student_ID,
        Password}, callback, asyncState);
}

[System.Diagnostics.DebuggerStepThroughAttribute()]
public bool EndAuthenticate(System.IAsyncResult asyncResult) {
    object[] results = this.EndInvoke(asyncResult);
    return ((bool)(results[0]));
}

[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.Web.Services.Protocols.SoapDocumentMethodAttribute("http://tr-0080-
2.dorm.temple.edu/UniversityPortal/Schools_Attended", RequestNamespace="http://tr-0080-
2.dorm.temple.edu/UniversityPortal/", ResponseNamespace="http://tr-0080-
2.dorm.temple.edu/UniversityPortal/",
Use=System.Web.Services.Description.SoapBindingUse.Literal,
ParameterStyle=System.Web.Services.Protocols.SoapParameterStyle.Wrapped)]
public System.Data.DataSet Schools_Attended(string Student_ID, string Password) {
    object[] results = this.Invoke("Schools_Attended", new object[] {
        Student_ID,
        Password});
    return ((System.Data.DataSet)(results[0]));
}

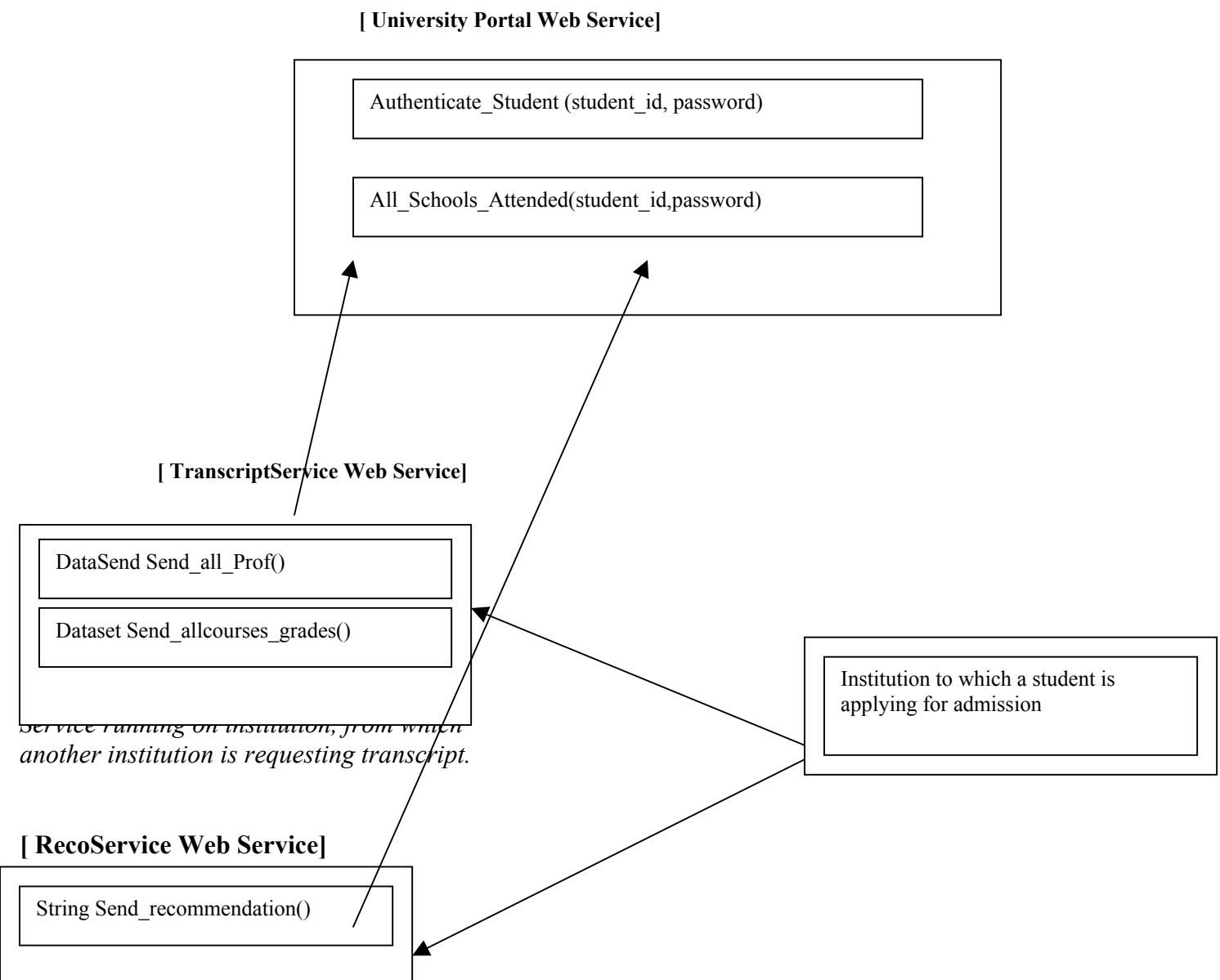
[System.Diagnostics.DebuggerStepThroughAttribute()]
public System.IAsyncResult BeginSchools_Attended(string Student_ID, string Password,
System.AsyncCallback callback, object asyncState) {
    return this.BeginInvoke("Schools_Attended", new object[] {
        Student_ID,
        Password}, callback, asyncState);
}

[System.Diagnostics.DebuggerStepThroughAttribute()]
public System.Data.DataSet EndSchools_Attended(System.IAsyncResult asyncResult) {
    object[] results = this.EndInvoke(asyncResult);
    return ((System.Data.DataSet)(results[0]));
}
}
}
```

5.5 A SAMPLE WEB SERVICE

Application to undergraduate and graduate studies requires sending transcripts and recommendation by post. It would be a nice idea if Transcripts and Recommendation were implemented as web services. When a student as to apply to a university then the university's server program could automatically contacts the

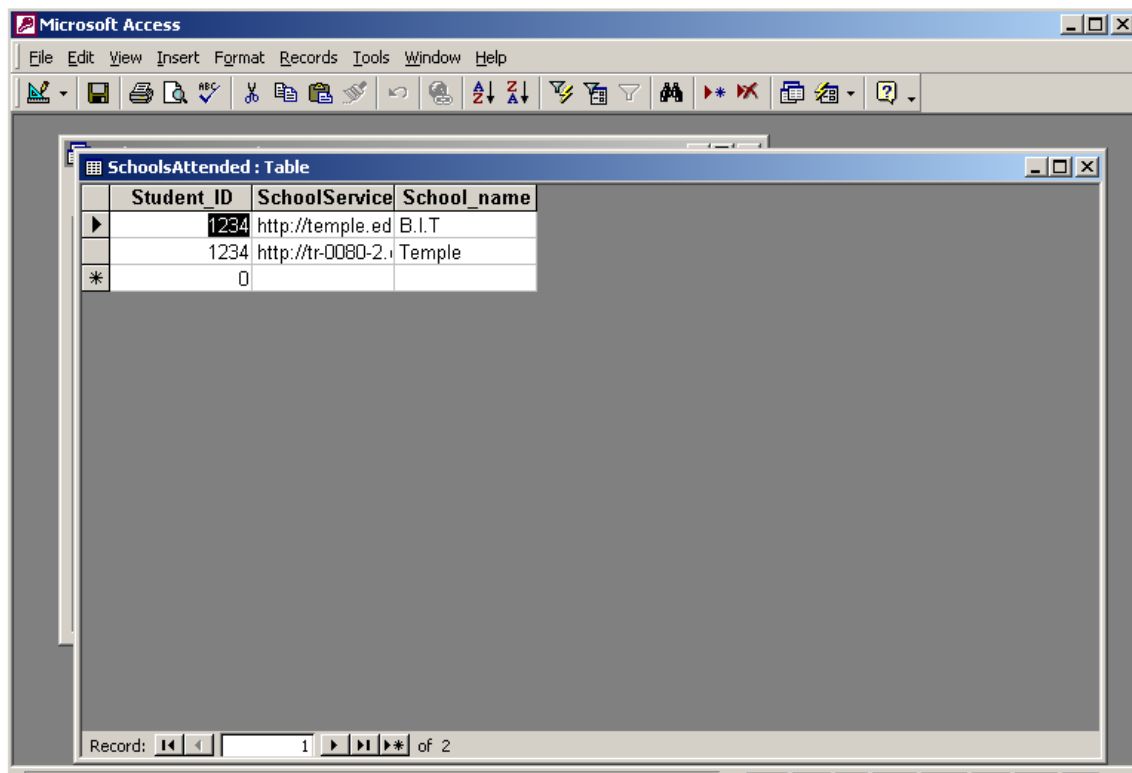
institution applicant has attended and collect transcripts from them. In addition to this transcript service also provides a list of Prof who have taught the applicant. The applicant can choose which Prof recommendation to send. The recommendations in turn are implemented as a web service on the Prof computer.

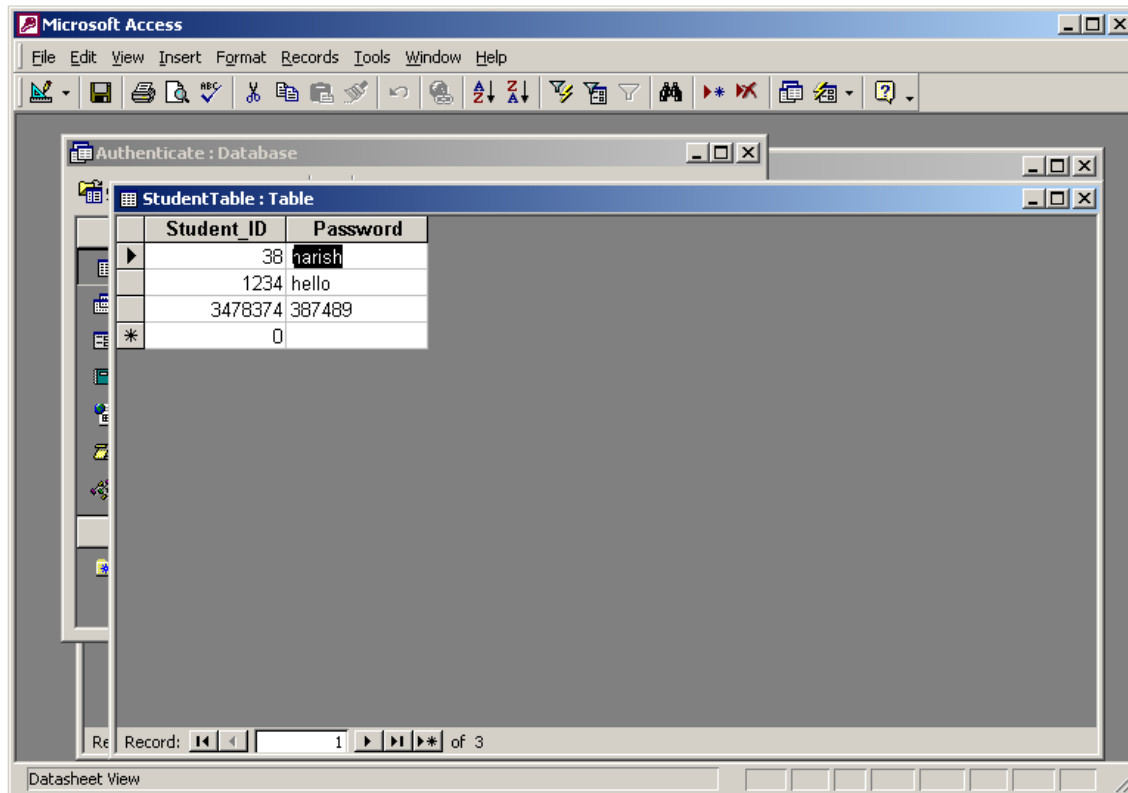


Service running on Professors system from whom student is requesting recommendation.

5.5.1 University Portal

The server running this service maintains a database (Authenticate.mdb), which contains SchoolsTable and SchoolsAttended Table. The SchoolsTable contains Student_ID and Password used for authenticating the student. The SchoolsAttended table contains the list of all institution with their URL's students have attended.





Code for UniversityPortal

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.Data;
using System.Data.OleDb;

namespace UniversityPortal
{
    /// <summary>
    /// Summary description for Servicel.
    /// </summary>

    [WebService(Namespace="http://tr-0080-
        2.dorm.temple.edu/UniversityPortal/")]

    public class Servicel : System.Web.Services.WebService
    {
```

```
private OleDbConnection con;

public Service1()
{
    //CODEGEN: This call is required by the ASP.NET Web
    Services Designer
    InitializeComponent();
    con = new OleDbConnection();
    con.ConnectionString =
        "Provider=Microsoft.JET.OLEDB.4.0;"+@"data source =
        c:\My Documents\Authenticate.mdb";
    con.Open();
}

#region Component Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
}
#endregion

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
}

[WebMethod]
public bool Authentincate(String Student_ID,String Password)
{
    DataSet ds = new DataSet("StudentTable");
    OleDbDataAdapter sda = new OleDbDataAdapter("Select * from
    StudentTable where Student_ID = "+Student_ID,con);

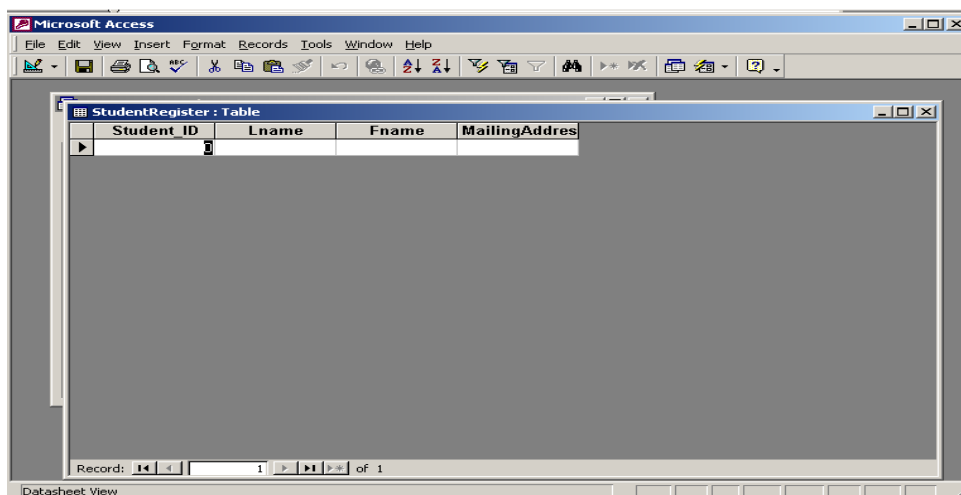
    sda.Fill(ds);
    if (ds.Tables[0].Rows.Count == 0 )
        return(false);
    if (
        ds.Tables[0].Rows[0]["Password"].ToString(
        ).Equals(Password))
        return(true);
    else
        return(false);
}
```

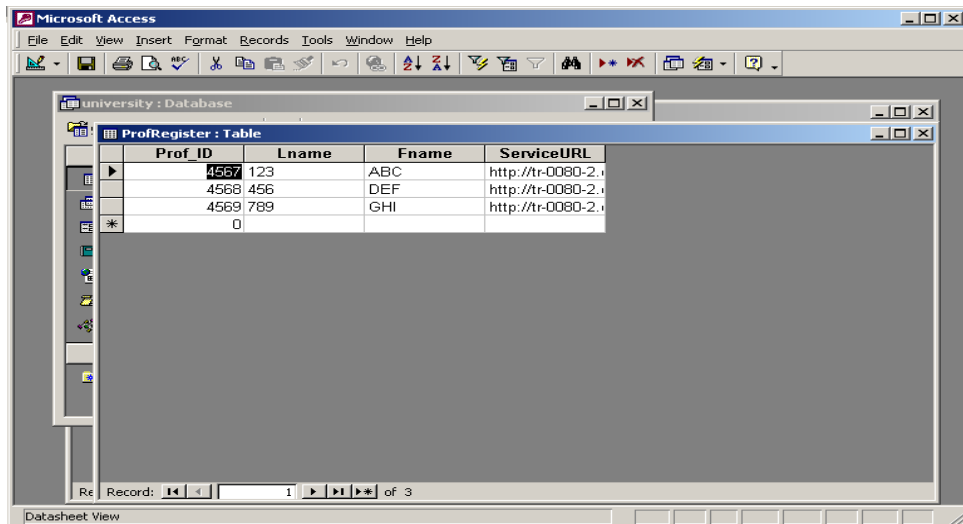
```
[WebMethod]
public DataSet Schools_Attended(String Student_ID,String
    Password)
{
    DataSet ds = new DataSet("StudentTable");
    if(this.Authentincate(Student_ID,Password))
    {
        OleDbDataAdapter sda = new OleDbDataAdapter("Select *
            from SchoolsAttended where Student_ID =
            "+Student_ID,con);
        sda.Fill(ds);
    }
    return(ds);
}

// WEB SERVICE EXAMPLE
// The HelloWorld() example service returns the string Hello
World
// To build, uncomment the following lines then save and build
the project
// To test this web service, press F5
}
}
```

5.5.2 TranscriptService Code

The server running this service maintains a database (University.mdb) containing ProfRegister and StudentRegister tables. The StudentRegister table contains the courses that a student has take, grades he has got and Professor who had taught him. The ProfRegister contains the URL of the computers Profs have.





TranscriptService Code

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace TranscriptService
{
    /// <summary>
    /// Summary description for Service1.
    /// </summary>
    public class TranscriptProvider : System.Web.Services.WebService
    {
        private OleDbConnection con;

        public TranscriptProvider()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services
            Designer
            InitializeComponent();
            con = new OleDbConnection();
            con.ConnectionString =
                "Provider=Microsoft.JET.OLEDB.4.0;" + @"data source =
                c:\My Documents\University.mdb";
            con.Open();
        }
    }
}
```

```
}

#region Component Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
    }
#endregion

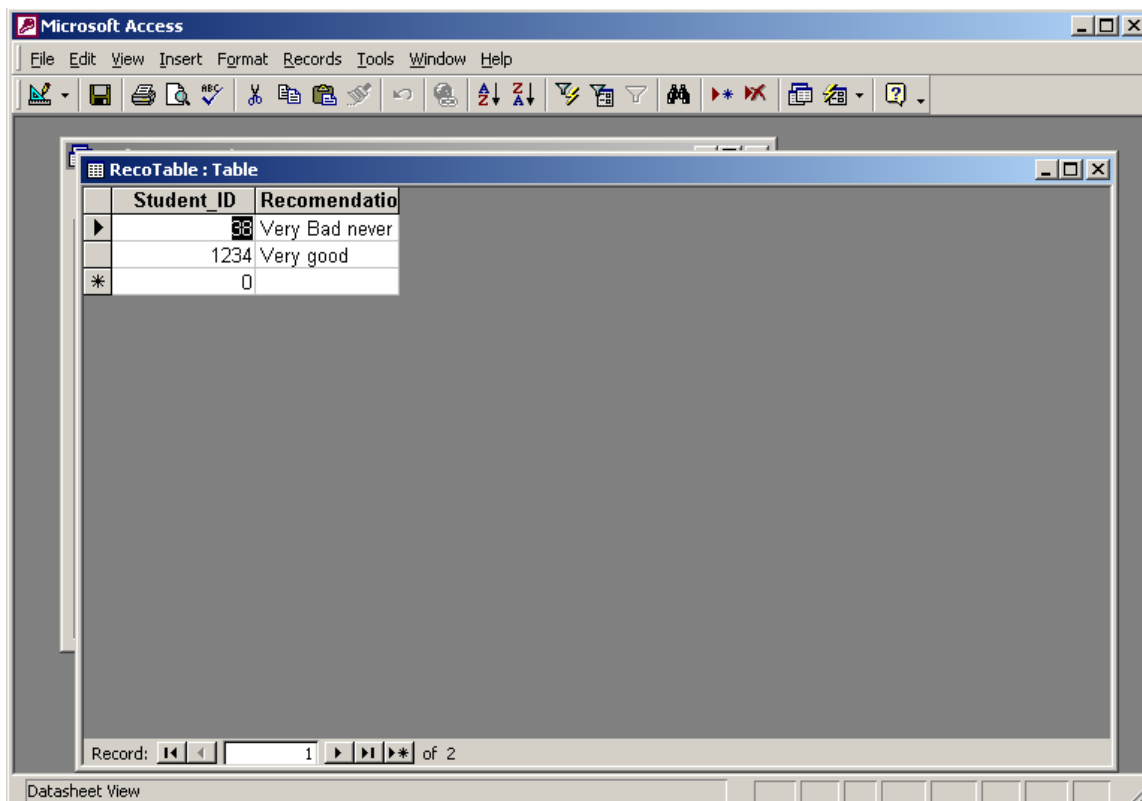
    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
    }

    [WebMethod]
    public DataSet getTranscript(String Student_ID,String Password)
    {
        edu.temple.dorm.tr_0080_2.Service1 auth = new
            edu.temple.dorm.tr_0080_2.Service1();
        DataSet ds = new DataSet("Transcript");
        if (auth.Authenticate(Student_ID,Password))
        {
            OleDbDataAdapter sda = new OleDbDataAdapter("Select
                * from GradeSheet where Student_ID =
                "+Student_ID,con);
            sda.Fill(ds);
            return(ds);
        }
        return (ds);
    }

    [WebMethod]
    public DataSet getProfessor(String Student_ID,String Password)
    {
        edu.temple.dorm.tr_0080_2.Service1 auth = new
            edu.temple.dorm.tr_0080_2.Service1();
        DataSet ds = new DataSet("Transcript");
        if (auth.Authenticate(Student_ID,Password))
        {
            OleDbDataAdapter sda = new OleDbDataAdapter("Select
                Lname,Fname,ServiceURL from ProfRegister where
                Prof_ID in ( Select distinct(Prof_ID) from
                GradeSheet where Student_ID =
                "+Student_ID+" )", con);
            sda.Fill(ds);
            return(ds);
        }
        return (ds);
    }
}
}
```

5.5.3 RecoService

The servers running this service maintain a database (Professor.mdb), which contains RecoTable. The RecoTable contains recommendation for every student the Professor has taught.



RecoService Code

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace RecoService
{
```


```
/// <summary>
/// Summary description for Service1.
/// </summary>
public class Reco : System.Web.Services.WebService
{
    private OleDbConnection con;
    public Reco()
    {
        //CODEGEN: This call is required by the ASP.NET Web Services
        Designer
        InitializeComponent();
        con = new OleDbConnection();
        con.ConnectionString = "Provider =
        Microsoft.JET.OLEDB.4.0;"+@"data source = C:\My
        Documents\Professor.mdb";
        con.Open();
    }

    #region Component Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
    }
    #endregion

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
    }

    [WebMethod]
    public DataSet getReco(String Student_ID,String Password)
    {
        edu.temple.dorm.tr_0080_2.Service1 auth = new
        edu.temple.dorm.tr_0080_2.Service1();
        DataSet ds = new DataSet("RecoTable");
        if (auth.Authenticate(Student_ID,Password))
        {
            OleDbDataAdapter sda = new OleDbDataAdapter("select
            * from RecoTable where Student_ID =
            "+Student_ID,con);
            sda.Fill(ds);
            return(ds);
        }
        return(ds);
    }
}
}
```

6 POWER POINT PRESENTATION OF .NET



.NET

- Understanding current state of affairs.
- What is .NET ?
- Developing Windows Application in .NET using C#.
- Developing Networking Applications in .NET using C#.
- What are web services?
- Developing Web application in .NET using C#.
- A sample web service.

Life as a Win32/C developer

- Windows applications were and still are written in C using raw Win32 API's
- Developer has to contend with memory management and ugly pointer arithmetic.
- C is a structured language and cannot leverage the benefits of OO application development.
- When developer code is mixed with global functions of raw windows API it results in spaghetti code.
- Though this approach does produce optimized code.
- Application gets compiled to x86 code and will not yield any information about itself.

Life as a MFC/C++ programmer

- Provided a object oriented wrapper over a "sane" subset of Windows API.
- Development time is still long.
- Learning curve is steep even to develop trivial applications.
- Developer still had to contend with memory management and pointer manipulation which led to buggy application.
- Like C , C++ code gets compiled directly into executable code for the underlying machine. This makes code reuse without documentation impossible (no reflection).
- The MFC could only be targeted by C++. If you wanted to use MFC you had to learn C++.

Life as a Java programmer

- Java offered a simple programming language with roots in C++ without a lot of complex syntax and memory management.
- The trouble is you use Java front to back during development cycle.
- [As a platform] Limited support for language independence and cross language inheritance.
- Java may not be appropriate for all application (such as graphics).

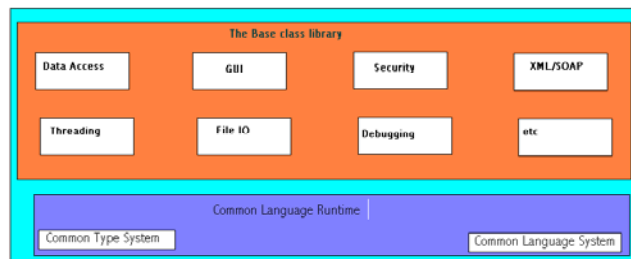
Enter .NET

- ***Full interoperability with existing code.***
- ***Complete and total language integration.*** Supports cross language inheritance, cross-language exception handling and cross language debugging.
- ***A common runtime engine shared by all .NET aware languages.***
- ***A common base class library.*** All .NET aware languages can use.
- ***Support for Web Services.***

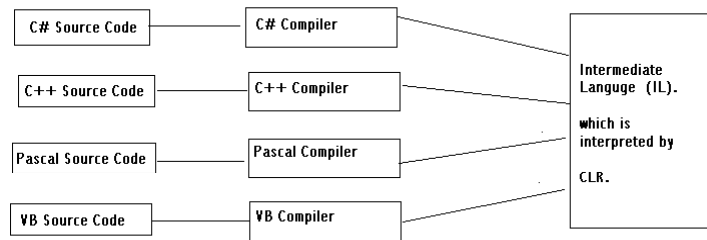
Building Blocks of .NET

- **Common Language Runtime (CLR)** : Responsible for locating, loading, managing .NET types.It is also responsible for automatic memory management, security, versioning etc.
- **Common Type System (CTS)** : Describes all possible data types supported b CLR. How data types can interact with each other. How they are described in “metadata”.
- **Common Language Specification (CLS)**: Since all .NET aware languages may not support all data types in CTS, CLS describes a common subset which all .NET aware languages have to support.
- **The Base class Library** : It contains classes for common tasks like File I/O, GUI design, Threading etc.These classes are available to all .NET aware languages (the class names remain the same). Classes performing related tasks are grouped together in Namespaces.

Basic Building Blocks



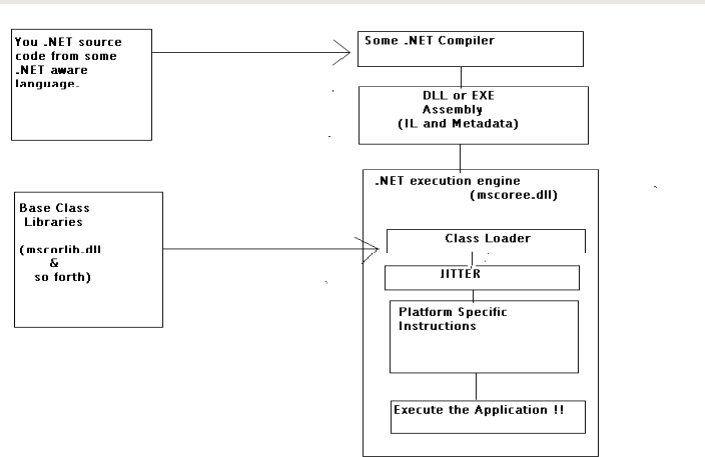
Compilation process



Binary executable contains.....

- IL code which is conceptually similar to java byte codes.
- **Metadata** :Describes in detail the characteristic of every “type” within the binary (e.g constructor’s of a class, methods and their signatures etc).
- **Manifest** : Describes in detail the binary itself. Information like list of external references needed for execution, security level, origin of binary, version etc.
- IL is just in time compiled
- With IL there is a “potential” for .NET to become platform independent in addition to being language independent.

Understanding the .NET development process.



Developing Windows Application in .NET using C#

- Needs the System.windows.Forms namespace which contains all the classes for creating GUI.
- Class has to derive from Form class in the System.windows.Forms namespace.
- Visual components are added by calling the controls.Add method.
- A sample application which simulates the towers of Hanoi is provided to illustrate windows forms.

Event handling & Threading

- In C# events are handled by passing a “delegate function” to the object firing the events.
- “Delegate” is a type safe function pointer. A delegate is a data type.
- The function to which a delegate variable points must have the same signature as the delegate.
- Delegate allow the name of the function to be resolved at runtime.
- In java where an object listening to a particular event which can be generated by an object contained within it has to inherit a particular interface and register itself with the object issuing the event.
- Different threads of execution can be created in C# by creating an object of type Thread and passing it a delegate pointing to the function which a thread should start from when the thread starts.
- These concepts are illustrated in the simple web server project I have created

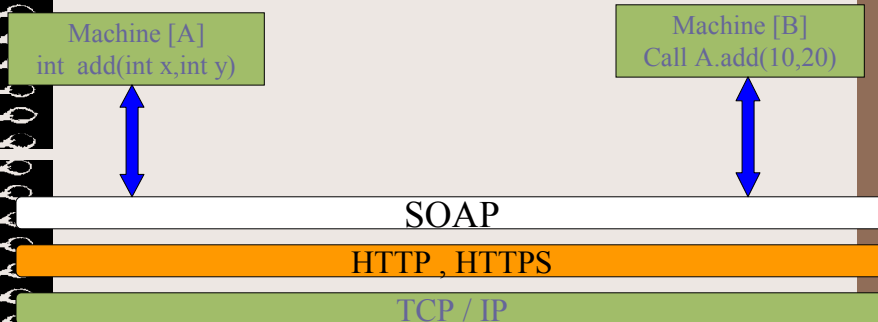
Web Services

- Why web services ?
- What are .NET web services ?
- How can I use a web service in my application.
- How can I know about web services?
- A sample web service.

Why web service ?

- No need to reinvent the wheel, makes software reuse feasible. Allows for structured programming in web applications.
- Privileged tasks like credit card validation must be done by a trusted party but this functionality is used by many applications. In these cases web services is the way to go
- Earlier attempts of achieving distributed style computing (COM,DCOM,CORBA,RMI etc) tend to be rigid and troubled developers with plumbing.
- Easier to scale up to internet size.
- Last but not the least will allow people to make money on the Internet by selling services.

How does this actually work !!!!!



A Sample Web Service..

- Application to undergraduate and graduate programs require sending transcript and recommendation in post.
- This causes delay and inconvenience.
- Office of academic records implements a Transcript service.
- All Prof.s have recommendation service running on their system.
- A central authenticating service authenticates student and institutions implementing services.
- University website makes use of all these services to gather transcript and recommendations of a student who is applying.

How can I use web service in my application ?

- Web Services can be discovered by universal discovery, description and integration (UDDI).
- Publicly accessible UDDI registries can be used by businesses to register information such as the Web services they expose. In turn, this information can be consumed by business partners in order to interact with the exposed Web services.
- Download web service contract which can automatically converted to a proxy class of the language that you are using.
- Used the web service as if it were a method of a local object.

What are .NET web services

- Web service is a method of a object which can be remotely invoked by other applications on the internet.
- A web service can be targeted by knowing its URL.
- A web service describes the service it provides in wsdl (method signature etc)
- All the parameter that are passed to and fro are wrapped in XML so that they can describe themselves.
- If objects have to be passed to and fro then they are XML serialized.
- An object's method can be made a web service by preceding method implementation by [web method] key word.
- That's all to web service developer need not even know about XML.