BON SECOURS COLLEGE FOR WOMEN

Nationally Accredited with 'A' Grade by NAAC
Affiliated to Bharathidasan University, Tiruchirappalli
UGC Regognized 2(f) and 12(B) Institution
Vilar Bypass, Thanjavur – 613 006

PG DEPARTMENT OF COMPUTER SCIENCE

TUTORIAL MATERIAL

DISTRIBUTED TECHNOLOGIES

&

QUESTION BANK

CONTENTS

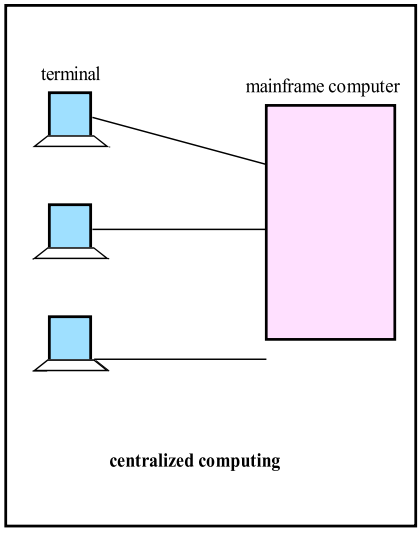
UNIT-I: INTRODUCTION TO DISTRIBUTED COMPUTING

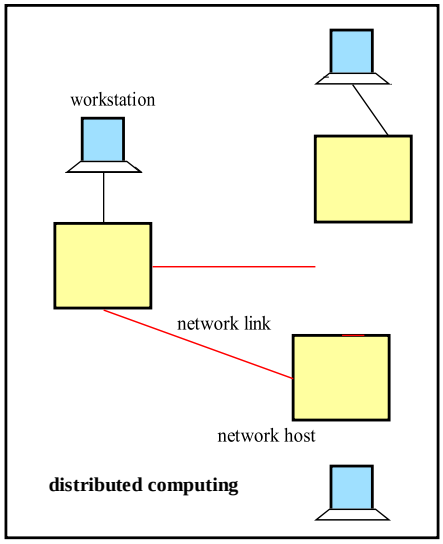
Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of multiple autonomous computers that communicate through a computer network.

A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs

In distributed computing, each processor has its own private memory (distributed memory). Information is exchanged by passing messages between the processors.
The processors in a typical distributed system run concurrently in parallel.

A distributed system is a collection of independent computers, interconnected via a network, capable of collaborating on a task.





Example Distributed systems:

- Internet
- ATM (bank) machines
- Intranets

Goals/Benefits:

- ✓ Resource sharing
- ✓ Scalability
- ✓ Fault tolerance and availability
- ✓ Performance
- ✓ Economics
- ✓ Speed
- ✓ Inherent distribution
- ✓ Reliability
- ✓ Incremental growth

Disadvantages:

- ✓ Software
- ✓ Network
- ✓ More components to fail
- ✓ Security

- ❖ Parallel computing can be considered a subset of distributed computing

Challenges involved in establishing remote connection

The challenges involved in remote connection are as mentioned below.

• Heterogeneity
• Openness
• Security
• Scalability
• Failure handling
• Concurrency
• Transparency

Heterogeneity

• Different networks, hardware, operating systems, programming languages, developers.
• We set up protocols to solve these heterogeneities.
• Middleware: a software layer that provides a programming abstraction as well as masking the heterogeneity.
• Mobile code: code that can be sent from one computer to another and run at the destination.

Openness

• Make it easier to build and change

• The openness of DS is determined primarily by the degree to which new resource-sharing services can be added and be made available for use by a variety of client programs.

• Open systems are characterized by the fact that their key interfaces are published.

• Open DS are based on the provision of a uniform communication mechanism and published interfaces for access to shared resources.

• Open DS can be constructed from heterogeneous hardware and software.

Security

• Security for information resources has three components:

- Confidentiality: protection against disclosure to unauthorized individuals.
- Integrity: protection against alteration or corruption.
- Availability: protection against interference with the means to access the resources.

• Two new security challenges:

- Denial of service attacks (DoS).
- Security of mobile code.

Scalability

• A system is described as scalable if it remains effective when there is a significant increase in

the number of resources and the number of users.

• Distributed system should be more reliable

than single system.

• Challenges:

- Controlling the cost of resources or money.
- Controlling the performance loss.
- Preventing software resources from running out
- Avoiding performance bottlenecks.

Failure handling

• When faults occur in hardware or software, programs may produce incorrect results or they may stop before they have completed the intended computation.

• Techniques for dealing with failures:

- Detecting failures
- Masking failures
- Tolerating failures
- Recovering from failures
- Redundancy

Concurrency

• There is a possibility that several clients will attempt to access a shared resource at the same time.

• Any object that represents a shared resource in a distributed system must be responsible for ensuring that operates correctly in a concurrent environment. Redundancy improves it.

Transparency

• Transparency is defined as the concealment from the user and the application programmer of the separation of components in a distributed system, so that the system is perceived as a whole rather than as a collection of independent components.

• Eight forms of transparency:

- Access transparency

- Location transparency
- Concurrency transparency
- Replication transparency
- Failure transparency
- Mobility transparency
- Performance transparency
- Scaling transparency

Strategies Involved In Remote Computation

The Remote Computation System (RCS)

- Today many high performance computers are reachable over some network.
- However, the access and use of these computers is often complicated.
- This prevents many users to work on such machines.
- The Goal of the Remote Computation System (CS) is to provide easy access to modern parallel algorithms on supercomputers for the inexperienced user.
- Wide area computer networks have become a basic part of today's computing infrastructure.
- These networks connect a variety of machines, from workstations to supercomputers, presenting an enormous computing resource.
- Furthermore, sufficient software for solving problems in numerical

linear algebra on high performance computers is around today.

- However, the access and the use of these computers and the software is often complicated.
- A major problem for the inexperienced user to exploit such high performance computers is that he has to deal with machine dependent low level details.
- The goal of this project is to make high performance computing accessible to scientists and engineers without the need for extensive training in parallel computing and allowing them to use resources best suited for a particular phase of the computation.
- Also, the emphasis is laid on algorithms for solving problems in numerical linear algebra, the concepts presented here are applicable to any high performance algorithms
- This goal shall be achieved with a remote computation system (RCS), which provides an easy-to-use mechanism for using computational resources remotely.
- The user's view of the RCS is that of an ordinary software library.
- The user calls RCS library routines (e.g. to solve a system of linear equations) within his program running on a workstation.
- In contrast to common libraries, the problem is not necessarily solved on the local workstation, but is dynamically allocated on an arbitrary machine in a given pool of computers, in order to minimize the response time.

- Because RCS is called asynchronously, it allows distributed applications with several solvers running concurrently on different computer platforms.
- The Remote Computation System consists of two components
    - A library of interface routines
    - The run time system.
- The underlying computational software can be any existing scientific package such as LAPACK
- Before running a RCS application, the user first has to start up the RCS run time system.
- RCS is a single user system but multiple RCS applications are allowed per user to run concurrently.
- The server is the core of the RCS. Its task is to accept requests from user's application and to start an appropriate solver on a host in the pool.
- If the remote host is not specified by the user, the server selects the solver-host pair such that the response time is minimized
- Such a selection process has not yet been done in the context of a numerical library.
- In order to make of an optimal choice, the server needs information about
    - The problems which RCS can solve

- The host computers in the pool and their characteristics such as number of processors etc.
- The available computational software (solvers) on each host and their characteristics. For instance, a theoretical model is required to assess its response time
- Dynamic parameters as the current workload on each host and the available communication bandwidth on the network.

- A daemon called monitor on each host is responsible for periodically measuring the dynamic parameters.
- All other information is static and is read from a configuration file at startup time.
- With Remote Desktop, you can have access to a Windows session that is running on your computer when you are at another computer.
- This means, for example, that you can connect to your work computer from home and have access to all of your programs, files, and network resources as though you were sitting at your computer at work.
- You can leave programs running at work and when you get home, you can see your work desktop displayed on your home computer, with the same programs running.
- You can keep your programs running and preserve the state of your Windows session while another user is logged on. When that user logs off, you can reconnect to your session in progress.
- To use Remote Desktop, you need:

- A computer ("host" computer) running Windows XP Professional with Service Pack 2 or Windows Server 2003 with Service Pack 2 ("remote" computer) with a connection to a local area network (LAN) or the Internet.
- A second computer ("client" computer) with access to the LAN via a network connection, modem, or virtual private network (VPN) connection. This computer must have Remote Desktop Connection installed.
- Appropriate user accounts and permissions.

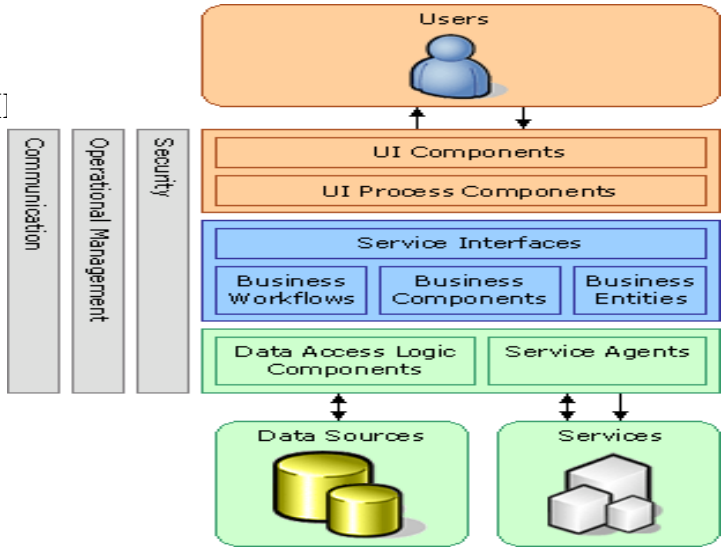
Distributed Computing Using .NET Remoting

Distributed computing has become the identity of present generation software applications. In past, developers used technologies like DCOM (Distributed Component Object Model) by Microsoft, CORBA (Common Object Request Broker Architecture) by OMG and Java RMI (Remote Method Invocation) by SUN for the same purpose.

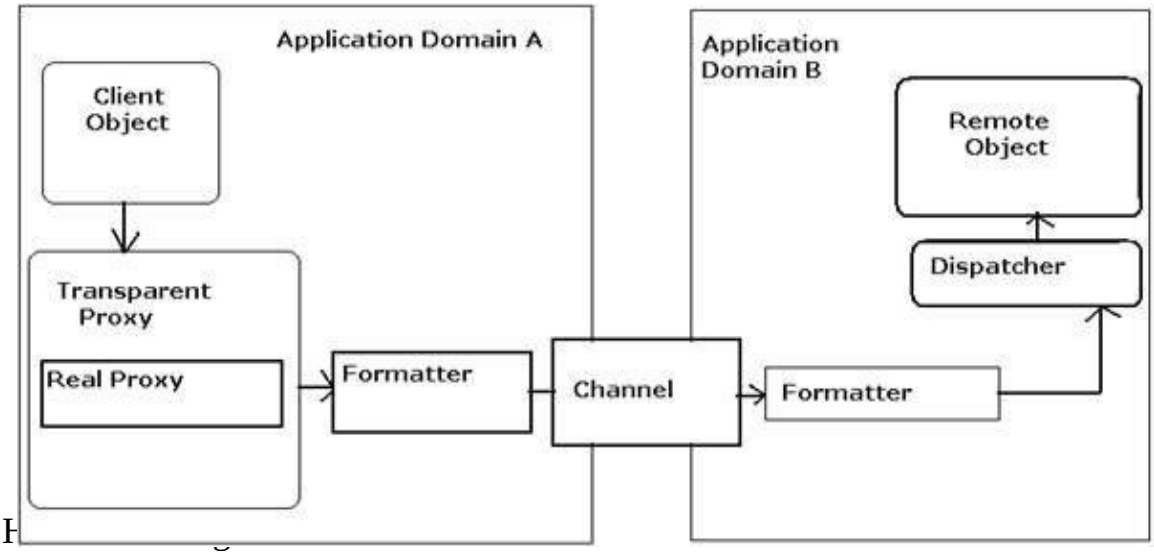
Microsoft .Net Remoting is an extensible framework provided by Microsoft .Net Framework, which enables communication across Application Domains (AppDomain).

AppDomain is an isolated environment for executing Managed code. Objects within same AppDomain are considered as local whereas object in a different AppDomain is called Remote object. Microsoft .Net Remoting comes into picture when an application requires communication between different AppDomains.

REMOTING I



H



Just like other distribute computing technologies, in .Net Remoting also, client object doesn't make a direct call to the remote object, rather it creates a proxy object of the remoting object and then uses the proxy object to invoke methods of remote object. When the client object calls a method of remote object via proxy, the call is formatted by a formatting object (SOAP,

Binary or any Custom formatter). After formatting the call is transferred to the remote object via proper channel (TCP Channel, HTTP Channel or any Custom channel) where the method is executed. Now the entire process is reversed to return appropriate result to the client object.

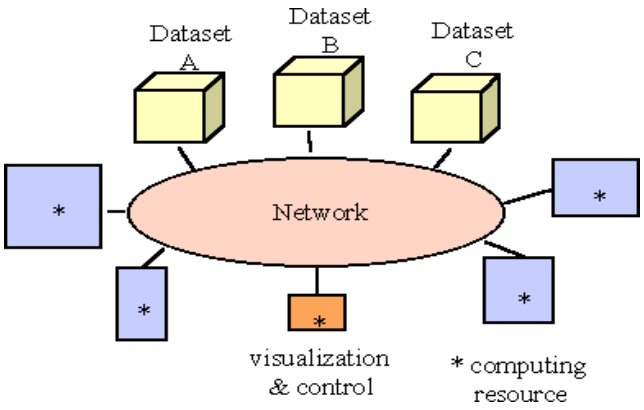


Figure 1: .Net remoting Architecture

Remote Object

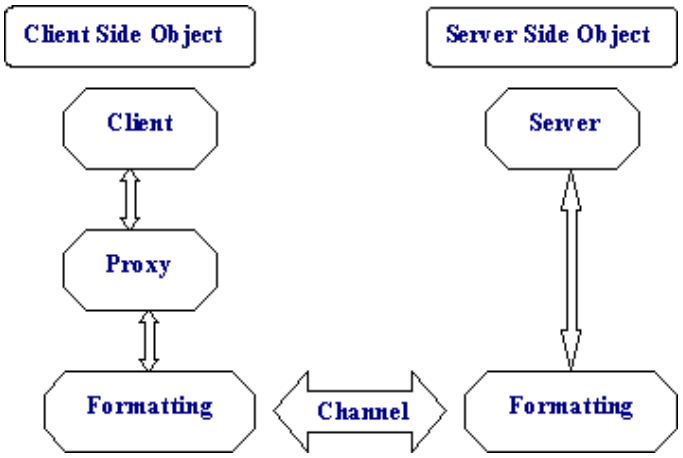
Remote object (Located at server side) is derived from System.MarshalByRefObject class which provides required functionality for communicating with an object in different AppDomain. Any object that needs to be transferred across appDomains has to be passed by value and should implement ISerializable interface. An object which doesn't implement ISerializable interface can't be transmitted across appDomains.

Example:

```
using System;
namespace Employee
{
```

```
public class Employee: MarshalByRefObject
{
//Constructor
public Employee
{
// Implementation details
}
//Other functions will come here
}
}
```

Proxy Object



This is created when a client object activates a remote object.

We can have two types of remoting objects i.e. Client Activated Object or Server Activated Object.

Client Activated Object: Client Activated Remote object is one whose life is controlled by client object. Here one single instance of remote object will exist per client object. Client Activated Object is created using new keyword. Client Activated object can store state information for a specific client.

Server Activated Object: Contrary to Client Activated Remote objects, life time for Server Activated Object is controlled by Server. These objects are created when client object calls a method on the proxy object. There are two types of Server Activated Objects i.e. SingleCall and Singleton.

SingleCall: They serve only one client request. Once the client request is over, they are subjected to garbage collection. They don't store any state information.

Singleton: They serve multiple clients, thereby allowing information sharing between requests. They are stateful objects unlike SingleCall, which is stateless.

```
//Creating a new instance of a remote object using new.
Employee Emp = new Employee();
//Creating a new instance of a remote object using CreateInstance.
Employee Emp = (Employee)Activator.CreateInstance(...);
```

Note: Above method creates an instance of the remote object based on the parameter passed. For a complete listing of the same, please refer MSDN.

//Retrieving an Existing Instance.

Employee Emp = (Employee)Activator.GetObject(typeof(Employee), HTTP://[Path]);

All the calls to the remote object (at server side) are routed through proxy object. To make things little complicated, there are two types of proxy objects involved in the process i.e. Transparent proxy and Real proxy. Transparent proxy provides the implementation of all public method to Client object which means that client object always talks to Transparent proxy which in turn makes call to Real proxy. Real proxy passes the message to channel object. Developers can customize the Real proxy to include additional functionalities if required.

Formatters

They encode and decode the message between client application and Remote object. .Net Framework provides SOAP and Binary formatter. It also supports custom formatters (IRemotingFormatter) developed by programmers.

Binary Formatter*:* System.Runtime.Serialization.Formatters.Binary SOAP Formatter*:* System.Runtime.Serialization.Formatters.Soap

Channels

They are responsible for transmitting the message over the network. .Net Framework provides HTTPChannel and TCPChannel. It also supports custom channels (IChannel) developed by programmers.

HTTPChannel: System.Runtime.remoting.Channels.Http

TCPChannel: System.Runtime.remoting.Channels.Tcp

By default HTTP Channel uses SOAP Formatter and TCP Channel uses Binary Formatter.

Channels need to be registered with the remoting service as shown below.

```
//Registering a channel
ChannelServices.RegisterChannel();
```

Hosting a Remoting Application

Remoting Host is a runtime environment for the remote object i.e. Microsoft IIS Server.

Step By Step: Let's see the entire step once again.

1. Client object registers a channel.
2. Creation of Proxy object (Client activated or Server Activated)
3. Calling the method of remote object via proxy.
4. Client side formatter formats the message and transmits via appropriate channel.
5. Server side formatter reformats the message.
6. The specified function on remote object is executed and the result is returned.
7. Above process of formatting and reformatting is reversed and the result is returned to client object.

Above article explains the basic terms and technology involved in .Net Remoting. It's possible to create complex distributed applications using .Net Remoting. Developers can create their own custom channels and formatters depending on business needs. There is no built in security provided by .Net Remoting framework. The security features need to be provided by the hosting environment.

Distributed Systems in Java:

Java Remote Method Invocation (RMI) allows you to write distributed objects using Java.

RMI provides a simple and direct model for distributed computation with Java objects. These objects can be new Java objects, or can be simple Java wrappers around an existing API. Java embraces the "Write Once, Run Anywhere model. RMI extends the Java model to be run everywhere."

Because RMI is centered on Java, it brings the power of Java safety and portability to distributed computing. You can move behavior, such as agents and business logic, to the part of your network where it makes the most sense. When you expand your use of Java in your systems, RMI allows you to take all the advantages with you.

RMI connects to existing and legacy systems using the standard Java native method interface JNI. RMI can also connect to existing relational database using the standard JDBC package. The RMI/JNI and RMI/JDBC combinations let you use RMI to communicate today with existing servers in non-Java languages, and to expand your use of Java to those servers when it

makes sense for you to do so. RMI lets you take full advantage of Java when you do expand your use.

Advantages

At the most basic level, RMI is Java's remote procedure call (RPC) mechanism. RMI has several advantages over traditional RPC systems because it is part of Java's object oriented approach. Traditional RPC systems are language-neutral, and therefore are essentially least-common-denominator systems-they cannot provide functionality that is not available on all possible target platforms.

RMI is focused on Java, with connectivity to existing systems using native methods. This means RMI can take a natural, direct, and fully-powered approach to provide you with a distributed computing technology that lets you add Java functionality throughout your system in an incremental, yet seamless way.

The primary advantages of RMI are:

- Object Oriented: RMI can pass full objects as arguments and return values, not just predefined data types. This means that you can pass complex types, such as a standard Java hashtable object, as a single argument. In existing RPC systems you would have to have the client decompose such an object into primitive data types, ship those data types, and the recreate a hashtable on the server. RMI lets you ship objects directly across the wire with no extra client code.
- Mobile Behavior: RMI can move behavior (class implementations) from client to server and server to client. For example, you can define

an interface for examining employee expense reports to see whether they conform to current company policy. When an expense report is created, an object that implements that interface can be fetched by the client from the server. When the policies change, the server will start returning a different implementation of that interface that uses the new policies. The constraints will therefore be checked on the client side-providing faster feedback to the user and less load on the server-without installing any new software on user's system. This gives you maximal flexibility, since changing policies requires you to write only one new Java class and install it once on the server host.

- Design Patterns: Passing objects lets you use the full power of object oriented technology in distributed computing, such as two- and three-tier systems. When you can pass behavior, you can use object oriented design patterns in your solutions. All object oriented design patterns rely upon different behaviors for their power; without passing complete objects-both implementations and type-the benefits provided by the design patterns movement are lost.
- Safe and Secure: RMI uses built-in Java security mechanisms that allow your system to be safe when users downloading implementations. RMI uses the security manager defined to protect systems from hostile applets to protect your systems and network from potentially hostile downloaded code. In severe cases, a server can refuse to download any implementations at all.
- Easy to Write/Easy to Use: RMI makes it simple to write remote Java servers and Java clients that access those servers. A remote interface is an actual Java interface. A server has roughly three lines of code to declare itself a server, and otherwise is like any other Java object. This

simplicity makes it easy to write servers for full-scale distributed object systems quickly, and to rapidly bring up prototypes and early versions of software for testing and evaluation. And because RMI programs are easy to write they are also easy to maintain.

- Connects to Existing/Legacy Systems: RMI interacts with existing systems through Java's native method interface JNI. Using RMI and JNI you can write your client in Java and use your existing server implementation. When you use RMI/JNI to connect to existing servers you can rewrite any parts of you server in Java when you choose to, and get the full benefits of Java in the new code. Similarly, RMI interacts with existing relational databases using JDBC without modifying existing non-Java source that uses the databases.
- Write Once, Run Anywhere: RMI is part of Java's "Write Once, Run Anywhere" approach. Any RMI based system is 100% portable to any Java Virtual Machine *, as is an RMI/JDBC system. If you use RMI/JNI to interact with an existing system, the code written using JNI will compile and run with any Java virtual machine.
- Distributed Garbage Collection: RMI uses its distributed garbage collection feature to collect remote server objects that are no longer referenced by any clients in the network. Analogous to garbage collection inside a Java Virtual Machine, distributed garbage collection lets you define server objects as needed, knowing that they will be removed when they no longer need to be accessible by clients.
- Parallel Computing: RMI is multi-threaded, allowing your servers to exploit Java threads for better concurrent processing of client requests.
- The Java Distributed Computing Solution: RMI is part of the core Java platform starting with JDK?? 1.1, so it exists on every 1.1 Java

Virtual Machine. All RMI systems talk the same public protocol, so all Java systems can talk to each other directly, without any protocol translation overhead.

Passing Behavior

When we described how RMI can move behavior above, we briefly outlined an expense report program. Here is a deeper description of how you could design such a system. We present this to show how you can use RMI's ability to move behavior from one system to another to move computing to where you want it today, and change it easily tomorrow. The examples below do not handle all cases that would arise in the real world, but instead give a flavor for how the problem can be approached.

Server-Defined Policy

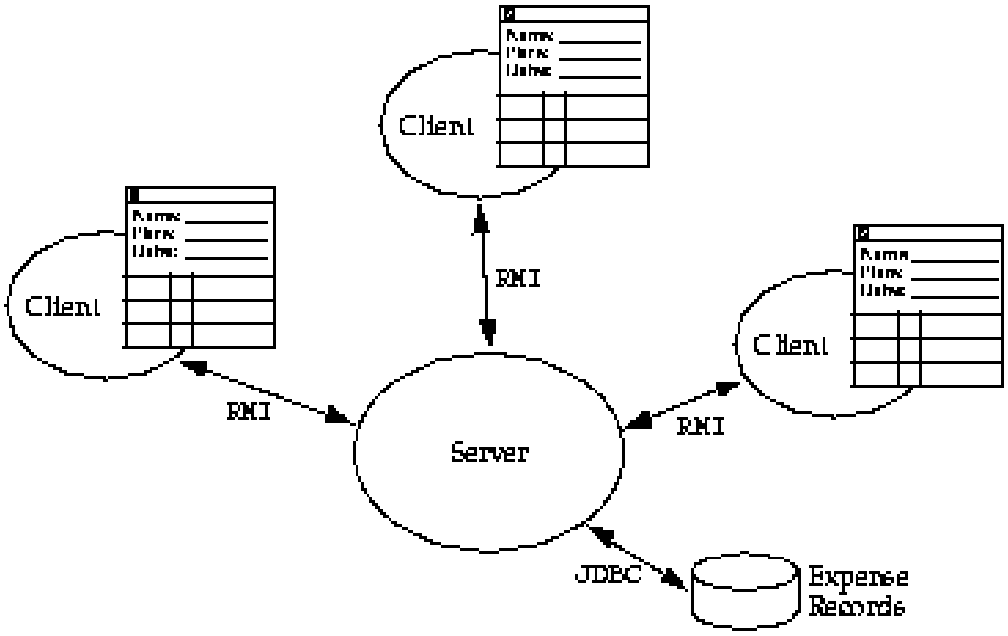


FIGURE 1 An Expense Reporting Architecture

Figure 1 shows the general picture of such a dynamically configurable expense reporting system. A client displays a GUI (graphical user interface) to a user, who fills in the fields of the expense report. Clients communicate with the server using RMI. The server stores the expense reports in a database using JDBC, the Java relational database package. So far this may look like any multi-tier system, but there is an important difference-RMI can download behavior.

Suppose that the company's policies about expense reports change. For example, today the company requires receipts only for expenses over $20. Tomorrow the company decides this is too lenient-it wants receipts for everything, except for meals that cost less than $20. Without the ability to download behavior, you have the following alternatives when designing your system for change:

- Install the policy with the client. When the policy changes, this requires updating all clients that contain the policy. You could reduce the problem by installing the client on a handful of server machines and requiring all users to run the client from one of those servers. This still would not completely solve the problem-anyone who leaves the program up and runn ing for days would not be updated, and there are always some people who copy the software to a local disk for efficiency.
- You could have the policy checked by the server when each entry is added to the expense report. This would result in a lot of traffic between client and server, clogging the network and burdening the server. It would also make the system more fragile-a network failure

would halt people in their tracks instead of only affecting them when they actually submit an expense report or start a new one. It would also mean that adding an entry would be slow, since it would require a round trip across the network to the (burdened) server.

- You could have the policy checked by the server when the report is submitted. This lets the user create a lot of bad entries which must then be reported in a batch instead of catching the first error immediately, giving the user a chance to stop making the error. Users need immediate feedback on errors to avoid wasted time.

With RMI you can have the client upload behavior from the server with a simple method invocation, providing a flexible way to offload computation from the server to the clients while providing users with faster feedback. When a user is ready to write up a new expense report, the client asks the server for an object that embodies the current policies for expense reports as expressed via a Policy interface written in Java. The object can implement the policy in any way. If this is the first time that the client's RMI runtime has seen this particular implementation of the policy, RMI will ask the server for a copy of the implementation. Should the implementation change tomorrow, a new kind of policy object will be returned to the client, and the RMI runtime will then ask for that new implementation.

This means that policy is always dynamic. You can change the policy by simply writing a new implementation of the general Policy interface, installing it on the server, and configuring the server to return objects of this new type. From that point on, any new expense reports will be checked against the new policy by every client.

This is a better approach than any static approach because:

- All clients don't need to be halted and updated with new software- software is updated on the fly as needed.
- The server is not burdened with entry checking that can be done locally.
- Allows dynamic constraints because object implementations, not just data, are passed between client and server.
- Lets users know immediately about errors.

Here is the remote interface that defines the methods the client can invoke on the server:

```
import java.rmi.*;
public interface ExpenseServer extends Remote {
   Policy getPolicy() throws RemoteException;
   void submitReport(ExpenseReport report)
      throws RemoteException, InvalidReportException;
}
```

The import statement imports the Java RMI package. All the RMI types are defined in the package java.rmi or one of its subpackages. The interface ExpenseServer is a normal Java interface with two interesting characteristics

- It extends the RMI interface named Remote, which marks the interface as one available for remote invocation.
- All its methods throw RemoteException, which is used to signal network and messaging failures. Remote methods can throw any other exception you like, but they must throw at least RemoteException so

that you can handle error conditions that only arise in distributed systems. The interface itself supports two methods: getPolicy which returns an object that implements the Policy interface, and submitReport which submits a completed expense request, throwing an exception if the report is malformed for any reason.

The Policy interface itself declares a method that lets the client know if it is acceptable to add an entry to the expense report:

```
public interface Policy {
   void checkValid(ExpenseEntry entry)
      throws PolicyViolationException;
}
```

If the entry is a valid one-one that matches current policy-the method returns normally. Otherwise it throws an exception that describes the error. The Policy interface is local (not remote), and so will be implemented by an object local to the client-one that runs in the client's virtual machine, not across the network. A client would operate something like this:

```
Policy curPolicy = server.getPolicy();
start a new expense report
show the GUI to the user
while (user keeps adding entries) {
   try {
      curPolicy.checkValid(entry); // throws exception if not OK
      add the entry to the expense report
   } catch (PolicyViolationException e) {
```

```
        show the error to the user
    }
}
server.submitReport(report);
```

When the user asks the client software to start up a new expense report, the client invokes server.getPolicy to ask the server to return an object that embodies the current expense policy. Each entry that is added is first submitted to that policy object for approval. If the policy object reports no error, the entry is added to the report; otherwise the error will be displayed to the user who can take corrective action. When the user is finished adding entries to the report, the entire report is submitted.

UNIT-II: ADVANCED ADO.NET

Introduction

ADO.NET provides consistent access to data sources such as Microsoft SQL Server and XML, as well as to data sources exposed through OLE DB and ODBC. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update the data that they contain.

ADO.NET includes .NET Framework data providers for connecting to a database, executing commands, and retrieving results.

ADO.NET is a data-access technology that enables applications to connect to data stores and manipulate data contained in them in various ways. It is based on the .NET Framework and it is highly integrated with the rest of the Framework class library. The ADO.NET API is designed so it can be used from all programming languages that target the .NET Framework, such as Visual Basic, C#, J# and Visual C++.

ADO uses a small set of Automation objects to provide a simple and efficient interface to OLE DB. This interface makes ADO a good choice for developers in higher level languages, such as Visual Basic and VBScript, who want to access data without having to learn the DETAILS of COM and OLE DB.

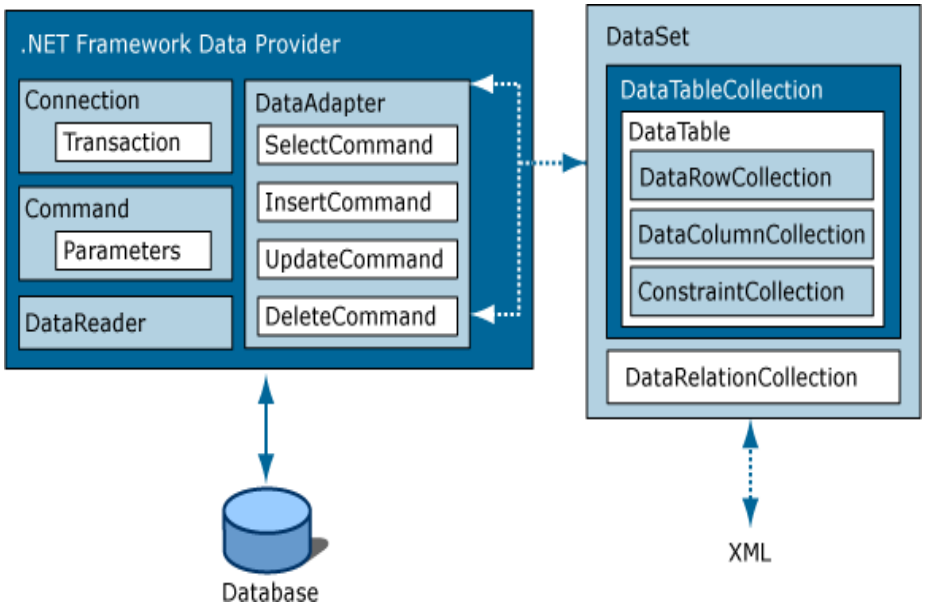
ADO.NET provides functionality to developers writing managed code similar to the functionality provided to native component object model (COM) developers by ActiveX Data Objects (ADO)

Disconnected Data access

ADO.NET Components

There are two components of ADO.NET that you can use to access and manipulate data:

- .NET Framework data providers
- The DataSet



1. NET Framework Data Providers

The NET Framework Data Providers are components that have been explicitly designed for data manipulation and fast, forward-only, read-only access to data.

The Connection object provides connectivity to a data source.

The Command object enables access to database commands to return data, modify data, run stored procedures, and send or retrieve parameter information.

The DataReader provides a high-performance stream of data from the data source. Finally, the DataAdapter provides the bridge between the DataSet object and the data source.

The DataAdapter uses Command objects to execute SQL commands at the data source to both load the DataSet with data, and reconcile changes made to the data in the DataSet back to the data source.

i) The Connection object

Listed below are the common connection object methods we could work with:

- Open - Opens the connection to our database
- Close - Closes the database connection
- Dispose - Releases the resources on the connection object. Used to force garbage collecting, ensuring no resources are being held after

our connection is used. Incidentally, by using the Dispose method you automatically call the Close method as well.

- State - Tells you what type of connection state your object is in, often used to check whether your connection is still using any resources. Ex. if (ConnectionObject.State == ConnectionState.Open)

ii) The Command Object

- ExecuteReader - Simply executes the SQL query against the database, using the Read() method to traverse through data.
- ExecuteNonQuery – Used whenever you work with SQL stored procedures with parameters.
- ExecuteScalar - Returns a lightning fast single value as an object from your database Ex. object val = Command.ExecuteScalar(); Then check if != null.
- ExecuteXmlReader - Executes the SQL query against SQL Server only, while returning an XmlReader object.
- Prepare – Equivalent to ADO's Command.Prepared = True property. Useful in caching the SQL command so it runs faster when called more than once. Ex. Command.Prepare();
- Dispose – Releases the resources on the Command object. Used to force garbage collecting, ensuring no resources are being held after our connection is used. Incidentally, by using the Dispose method you automatically call the Connection object's Close method as well.

iii) The DataReader Object

- Read – Moves the record pointer to the first row, which allows the data to be read by column name or index position.
- HasRows - HasRows checks if any data exists, and is used instead of the Read method. Ex. if (DataReader.HasRows).
- IsClosed - A method that can determine if the DataReader is closed.
- Next Result - Equivalent to ADO's NextRecordset Method, where a batch of SQL statements are executed with this method before advancing to the next set of data results.
- Close – Closes the DataReader

iv) The DataAdapter

Using an adapter, you can read, add, update, and delete records in a data source. To allow you to specify how each of these operations should occur, an adapter supports the following four properties:

- SelectCommand – reference to a command (SQL statement or stored procedure name) that retrieves rows from the data store.
- InsertCommand – reference to a command for inserting rows into the data store.
- UpdateCommand – reference to a command for modifying rows in the data store.
- DeleteCommand – reference to a command for deleting rows from the data store.

2. The DataSet

The ADO.NET DataSet is explicitly designed for data access independent of any data source. As a result, it can be used with multiple and differing data

sources, used with XML data, or used to manage data local to the application.

The ADO.NET DataSet contains DataTableCollection and their DataRelationCollection . It represents a collection of data retrieved from the Data Source.

The DataSet contains a collection of one or more DataTable objects made up of rows and columns of data, as well as primary key, foreign key, constraint, and relation information about the data in the DataTable objects.

We can use Dataset in combination with DataAdapter class. The DataSet object offers a disconnected data source architecture. The Dataset can work with the data it contain, without knowing the source of the data coming from. That is, the Dataset can work with a disconnected mode from its Data Source . It gives a better advantage over DataReader , because the DataReader is working only with the connection oriented Data Sources.

In any .NET data access page, before you connect to a database, you first have to import all the necessary namespaces that will allow you to work with the objects required. As we're going to work with SQL Server, we'll first import the namespaces we need. Namespaces in .NET are simply a neat and orderly way of organizing objects, so that nothing becomes ambiguous.

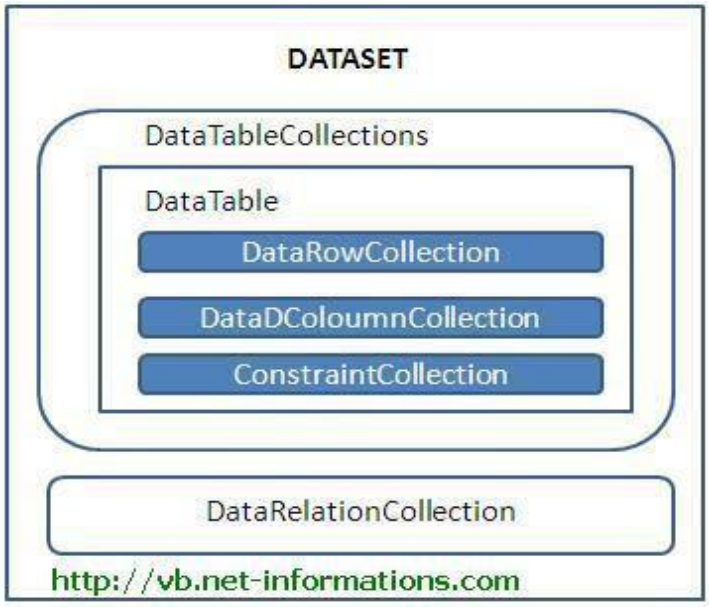
Note

(Namespaces: All the classes are defined in single name called namespaces in ASP.NET.)

Example:

1. <%@ Import Namespace="System" %>
2. <%@ Import Namespace="System.Data" %>
3. <%@ Import Namespace="System.Data.SqlClient" %>)

The Dataset contains the copy of the data we requested. The Dataset contains more than one Table at a time. We can set up Data Relations between these tables within the DataSet. The data set may comprise data for one or more members, corresponding to the number of rows.



GridView Control

GridView control is a successor to the ASP.NET 1.X DataGrid control. It provides more flexibility in displaying and working with data from your database in comparison with any other controls. The GridView control enables you to connect to a datasource and display data is tabular format

Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc.

ProductID	ProductName	UnitPrice	UnitsInStock
1	Dell Studio XPS 8100	999	6
2	Dell Studio XPS 9100	1399	7
3	Dell Inspiron 1545/9176 Laptop	399	6
4	Dell Inspiron M301Z Laptop	599	8
5	HP Envy 13-1100EA Laptop	799	9
6	Motorola A1010	399	4
7	Samsung i710	699	10
8	Nokia N95	499	4
9	HP - Deskjet Printer	599	2
10	Epson - WorkForce 30 Printer	499	9
11	InFocus IN1100	799	9
12	InFocus IN102 Projector	599	9
		8288	83

Average Price: 690.67

GridViews support:

- Automatic sorting (click on a column heading to sort by that column)
- Automatic paging (sort of - true paging is only possible if you use more complicated data sources)
- Editing and deleting of data
- Selection of rows

Important properties

Behavior Properties of the GridView Control	
AllowPaging	true/false. Indicate whether the control should support paging.
AllowSorting	true/false. Indicate whether the control should support sorting.
SortExpression	Gets the current sort expression (field name) that determines the order of the row.
SortDirection	Gets the sorting direction of the column sorted currently (Ascending/Descending).
DataSource	Gets or sets the data source object that contains the data to populate the control.
DataSourceID	Indicate the bound data source control to use (Generally used when we are using SqlDataSource or AccessDataSource to bind the data, See 1st Grid example).
AutoGenerateEditButton	true/false. Indicates whether a separate column should be added to edit the record.
AutoGenerateDeleteButton	true/false. Indicates whether a separate column should be added to delete the record.
AutoGenerateSelectButton	true/false. Indicate whether a separate column should be added to select a particular record.
AutoGenerateColumns	true/false. Indicate whether columns are automatically created for each field of the data source. The default is true.

Style Properties of the GridView Control	
AlternatingRowStyle	Defines the style properties for every alternate row in the GridView.
EditRowStyle	Defines the style properties for the row in EditView (When you click Edit button for a row, the row will appear in this style).
RowStyle	Defines the style properties of the rows of the GridView.
PagerStyle	Defines the style properties of Pager of the GridView. (If AllowPaging=true, the page number row appears in this style)
EmptyDataRowStyle	Defines the style properties of the empty row, which appears if there is no records in the data source.
HeaderStyle	Defines the style properties of the header of the GridView. (The column header appears in this style.)
FooterStyle	Defines the style properties of the footer of GridView.
Appearance Properties of the GridView Control	
CellPadding	Indicates the space in pixel between the cells and the border of the GridView.

CellSpacing	Indicates the space in pixel between cells.
GridLines	Both/Horizontal/Vertical/None. Indicates whether GrdiLines should appear or not, if yes Horizontal, Vertical or Both.
HorizontalAlign	Indicates the horizontal align of the GridView.
EmptyDataText	Indicates the text to appear when there is no record in the data source.
ShowFooter	Indicates whether the footer should appear or not.
ShowHeader	Indicates whether the header should appear or not. (The column name of the GridView)
BackImageUrl	Indicates the location of the image that should display as a background of the GridView.
Caption	Gets or sets the caption of the GridView.
CaptionAlign	left/center/right. Gets or sets the horizontal position of the GridView caption.
State Properties of GridView Control	
Columns	Gets the collection of objects that represent the columns in the GridView.
EditIndex	Gets or sets the 0-based index that identifies the row currently to be edited.
FooterRow	Returns a GridViewRow object that represents

	the footer of the GridView.
HeaderRow	Returns a GridViewRow object that represents the header of the GridView.
PageCount	Gets the number of the pages required to display the reocrds of the data source.
PageIndex	Gets or sets the 0-based page index.
PageIndex	Gets or sets the number of records to display in one page of GridView.
Rows	Gets a collection of GridViewRow objects that represents the currently displayed rows in the GridView.
DataKeyNames	Gets an array that contains the names of the primary key field of the currently displayed rows in the GridView.
DataKeys	Gets a collection of DataKey objects that represent the value of the primary key fields set in DataKeyNames property of the GridView.
Events associated with GridView Control	
PageIndexChanging, PageIndexChanged	Both events occur when the page link is clicked. They fire before and after GridView handles the paging operation respectively.

RowCancelingEdit	Fires when Cancel button is clicked in Edit mode of GridView.
RowCommand	Fires when a button is clicked on any row of GridView.
RowCreated	Fires when a new row is created in GridView.
RowDataBound	Fires when row is bound to the data in GridView.
RowDeleting,RowDeleted	Both events fires when Delete button of a row is clicked. They fire before and after GridView handles deleting operaton of the row respectively.
RowEditing	Fires when a Edit button of a row is clicked but before the GridView hanldes the Edit operation.
RowUpdating, RowUpdated	Both events fire when a update button of a row is clicked. They fire before and after GridView control update operation respectively.
Sorting, Sorted	Both events fire when column header link is clicked. They fire before and after the GridView handler the Sort operation respectively.

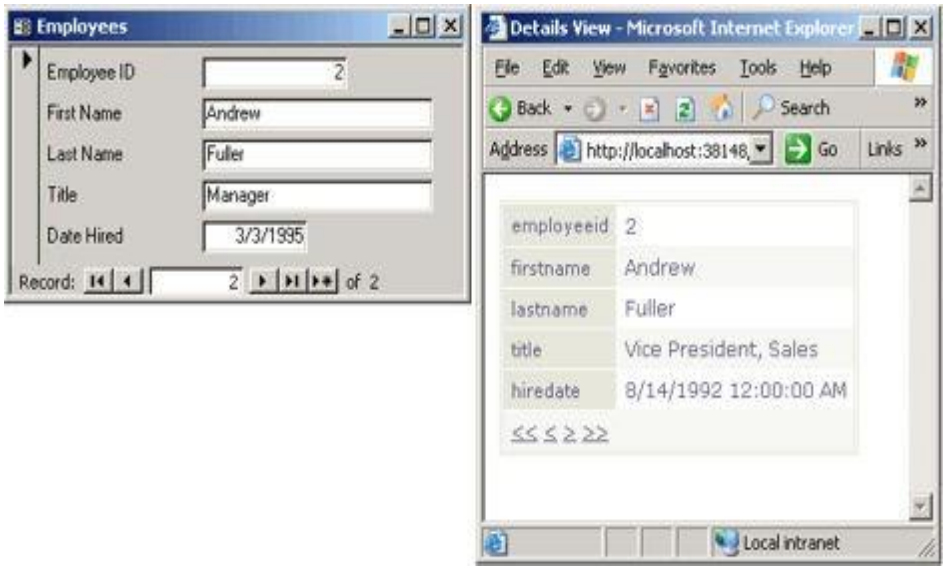
A DetailsView Control

In ASP.NET 2.0, DetailsView is a data-bound control that renders a single record at a time from its associated data source. It can optionally provide

paging buttons to navigate between records, and a command bar to execute basic operations on the current record (Insert, Update, Delete). DetailsView generates a user interface similar to the Form View of a Microsoft Access database, and is typically used for updating/deleting any currently displayed record or for inserting new records.

The key aspects of a DetailsView control:

- Be a composite control and act as a naming container.
- Be data-bindable to enumerable data sources.
- Support some style properties.
- Provide a navigation bar (pager).
- Support replaceable views of the record fields.
- Provide a command bar for common operations.



Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc.

Important properties

Behavior Properties of the DetailsView Control	
AllowPaging	true/false. Indicate whether the control should support navigation.
DataSource	Gets or sets the data source object that contains the data to populate the control.
DataSourceID	Indicate the bound data source control to use (Generally used when we are using SqlDataSource or AccessDataSource to bind the data, See 1st Grid example).
AutoGenerateEditButton	true/false. Indicates whether a separate column with edit link/button should be added to edit the record.
AutoGenerateDeleteButton	true/false. Indicates whether a separate column with delete link/button should be added to delete the record.
AutoGenerateRows	true/false. Indicate whether rows are automatically created for each field of the data source. The default is true.
DefaultMode	read-only/insert/edit. Indicate the default display mode.
Style Properties of the DetailsView Control	
AlternatingRowStyle	Defines the style properties for every alternate row in the DetailsView.
EditRowStyle	Defines the style properties for the row in

	EditView (When you click Edit button for a row, the row will appear in this style).
RowStyle	Defines the style properties of the rows of the DetailsView.
PagerStyle	Defines the style properties of Pager of the DetailsView. (If AllowPaging=true, the page number row appears in this style)
EmptyDataRowStyle	Defines the style properties of the empty row, which appears if there is no records in the data source.
HeaderStyle	Defines the style properties of the header of the DetailsView. (The column header appears in this style.)
FooterStyle	Defines the style properties of the footer of DetailsView.
Appearance Properties of the DetailsView Control	
CellPadding	Indicates the amount of space in pixel between the cells and the border of the DetailsView.
CellSpacing	Indicates the amount of space in pixel between cells.
GridLines	Both/Horizontal/Vertical/None. Indicates whether GrdiLines should appear or not, if yes Horizontal, Vertical or Both.
HorizontalAlign	Indicates the horizontal alignment of the

	DetailsView.
EmptyDataText	Indicates the text to appear when there is no record in the data source.
BackImageUrl	Indicates the location of the image that should display as a background of the DetailsView.
Caption	Gets or sets the caption of the DetailsView.
CaptionAlign	left/center/right. Gets or sets the horizontal position of the DetailsView caption.
State Properties of DetailsView Control	
Rows	Gets the collection of objects that represent the rows in the DetailsView.
FooterRow	Returns a DetailsViewRow object that represents the footer of the DetailsView.
HeaderRow	Returns a DetailsViewRow object that represents the header of the DetailsView.
PageCount	Gets the number of the pages required to display the records of the data source.
PageIndex	Gets or sets the 0-based page index.
DataKeyNames	Gets an array that contains the names of the primary key field of the currently displayed rows in the DetailsViewRow.
DataKeys	Gets a collection of DataKey objects that

	represent the value of the primary key fields set in DataKeyNames property of the DetailsViewRow.
Events of the DetailsView Control	
ItemCommand	Fires when any clickable element on the control is clicked.
ItemCreated	Fires after DetailsView fully creates all rows of the record.
ItemDeleting, ItemDeleted	Both event fires when current record is deleted. The first one fires before and other fires after record is deleted.
ItemInserting, ItemInserted	Both event fires when an item is inserted. The first one fires before and second after the item is created.
ItemUpdating, ItemUpdated	Both event fires when an item is updated. The first one fires before and second fires after the record is updated.
ModeChanging, ModeChanged	Both event fires when DetailsView change its display mode. The first one fires before and second fires after display mode is changed.
PageIndexChanging, PageIndexChanged	Both event fires when the DetailsView move to another record. The first one fires before and second fires after page is changed.

FormView Control

The FormView control is used to display a single record from database. It's greater flexiblity is, it displays user-defined templates instead of row fields.

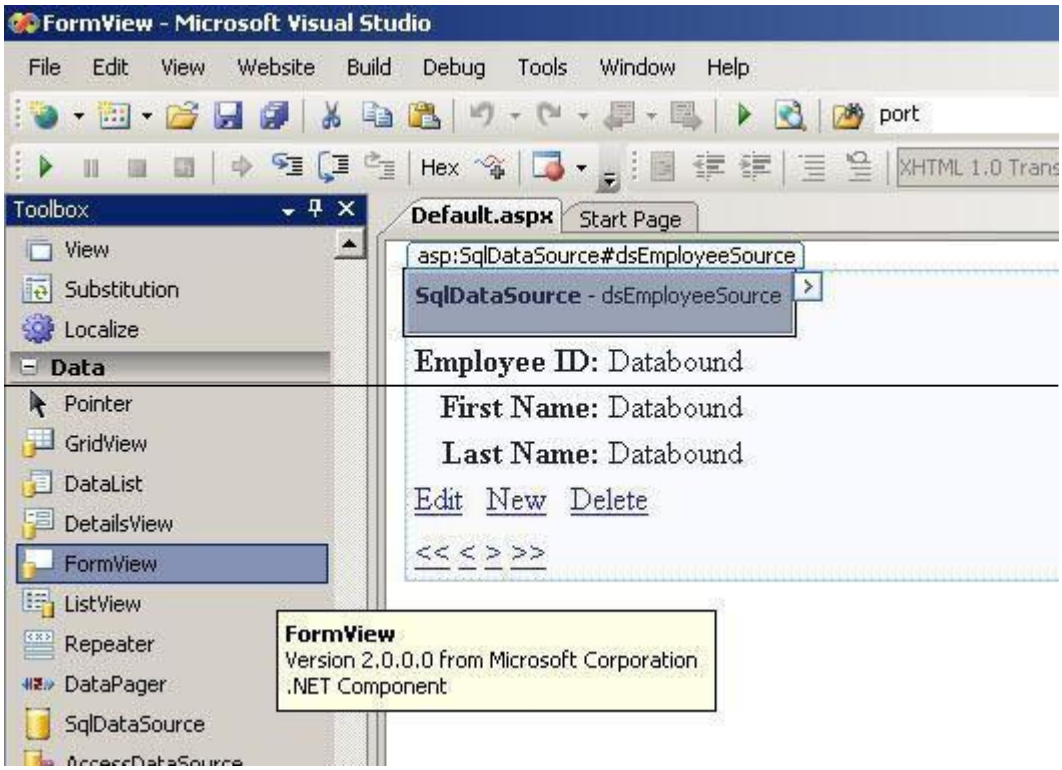
The difference between the FormView and the DetailsView controls is that the DetailsView control uses a tabular layout where each field of the record is displayed as a row of its own. In contrast, the FormView control does not specify a predefined layout for displaying the record. Instead, you create a template containing controls to display individual fields from the record. The template contains the formatting, controls, and binding expressions used to create the form.

The FormView control is typically used for updating and inserting new records, and is often used in master/detail scenarios where the selected record of the master control determines the record to display in the FormView control.

It has the following features:

- Binding to data source controls, such as SqlDataSource and ObjectDataSource.
- Built-in inserting capabilities.
- Built-in updating and deleting capabilities.
- Built-in paging capabilities.
- It's properties, handle events can be set dynamically with FormView object model.
- It can be customized through templates, themes and styles.

- Template are used to display/edit the FormView control.



Its properties like BackColor, ForeColor, BorderColor, BorderStyle, BorderWidth, Height etc.

Important properties

HeaderTemplate	Displays the content at header row of the template. The header row is displayed at the top of the FormView control when the HeaderText or HeaderTemplate property is set
EmptyDataTemplate	This is used display the content when datasource control does not contain any records. It alerts the ser that datasource has no records.

ItemTemplate	It is used to display the content when Formview is in read-only mode. The item template usually contains controls to display the field values of a record, as well as command buttons to edit, insert, and delete a record.
EditItemTemplate	Displays the content for the data row when the FormView control is in edit mode. This template usually contains input controls and command buttons with which the user can edit an existing record.
InsertItemTemplate	Displays the content for the data row when the FormView control is in insert mode. This template usually contains input controls and command buttons with which the user can add a new record.
PagerTemplate	Displays the content for the pager row displayed when the paging feature is enabled (when the AllowPaging property is set to true). This template usually contains controls with which the user can navigate to another record.
FooterTemplate	The footer row is displayed at the bottom of the FormView control when the FooterText or FooterTemplate property is set. If both the FooterText and FooterTemplate properties are set, the FooterTemplate property takes precedence.

Methods of the FormView Control	
ChangeMode	ReadOnly/Insert/Edit. Change the working mode of the control from the current to the defined FormViewMode type.
InsertItem	Used to insert the record into database. This method must be called when the DetailsView control is in insert mode.
UpdateItem	Used to update the current record into database. This method must be called when DetailsView control is in edit mode.
DeleteItem	Used to delete the current record from database.

ItemCommand	Occurs when a button within a FormView control is clicked. This event is often used to perform a task when a button is clicked in the control.
ItemCreated	Occurs after all FormViewRow objects are created in the FormView control. This event is often used to modify the values of a record before it is displayed.
ItemDeleted	Occurs when a Delete button (a button with its CommandName property set to "Delete") is clicked, but after the FormView control deletes the record from the data source. This event is often used to check the results of the delete operation.
ItemDeleting	Occurs when a Delete button is clicked, but before the FormView control deletes the record from the data source. This event is often used to cancel the delete operation.

ItemInserted	Occurs when an Insert button (a button with its CommandName property set to "Insert") is clicked, but after the FormView control inserts the record. This event is often used to check the results of the insert operation.
ItemInserting	Occurs when an Insert button is clicked, but before the FormView control inserts the record. This event is often used to cancel the insert operation.
ItemUpdated	Occurs when an Update button (a button with its CommandName property set to "Update") is clicked, but after the FormView control updates the row. This event is often used to check the results of the update operation.
ItemUpdating	Occurs when an Update button is clicked, but before the FormView control updates the record. This event is often used to cancel the update operation.
ModeChanged	Occurs after the FormView control changes modes (to edit, insert, or read-only mode). This event is often used to perform a task when the FormView control changes modes.
ModeChanging	Occurs before the FormView control changes modes (to edit, insert, or read-only mode). This event is often used to cancel a mode change.

Crystal Reports in ASP.NET

Crystal Reports is the standard reporting tool for Visual Studio .NET used to display data of presentation quality. You can display multiple-level totals, charts to analyze data, and much more in Crystal Reports. Creating a Crystal Report requires minimal coding since it is created in Designer interface. It is available as an integrated feature of Microsoft Visual Studio .NET, Borland Delphi, and C#Builder.

Advantages of Crystal Reports

Some of the major advantages of using Crystal Reports are:

1. Rapid report development since the designer interface would ease the coding work for the programmer.

2. Can extend it to complicated reports with interactive charts and enhance the understanding of the business model

3. Exposes a report object model, can interact with other controls on the ASP.NET Web form

4. Can programmatically export the reports into widely used formats like .pdf, .doc, .xls, .html and .rtf

Implementation Models

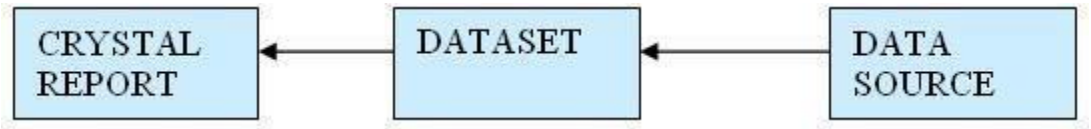
Crystal Reports need database drivers to connect to the data source for accessing data. Crystal Reports in .net support two methods to access data from a data source:

The Pull Method

When this model is used to access data from the data source, the database driver directly retrieves the data from the data source. This model does not require the developer to write code for creating a connection and retrieving data from the data source. It is the Crystal report that manages the SQL commands for connecting by using the specified driver.

The Push Method

When this model is used to access data from data source, the developer writes the code to connect to the data source and retrieve data. The data from the data source is cached in dataset and multiple crystal reports accesses data from the dataset. The performance can be optimized in this manner by using connection sharing and manually limiting the number of records that are passed on to the report.



Crystal Reports Types

Crystal Report Designer can load reports that are included into the project as well as those that are independent of the project.

Strongly-typed Report

When you add a report file into the project, it becomes a "strongly-typed" report. In this case, you will have the advantage of directly creating an instance of the report object, which could reduce a few lines of code, and cache it to improve performance. The related .vb file, which is hidden, can be viewed using the editor's "show all files" icon in the Solution Explorer.

Un-Typed Report

Those reports that are not included into the project are "un-typed" reports. In this case, you will have to create an instance of the Crystal Report Engine's "ReportDocument" object and manually load the report into it.

Creating Crystal Reports

You can create a Crystal Report by using three methods:

1. Manually i.e. from a blank document
2. Using Standard Report Expert
3. From an existing report

Using Pull Method

Creating Crystal Reports Manually.

We would use the following steps to implement Crystal Reports using the Pull Model:

1. Create the .rpt file (from scratch) and set the necessary database connections using the Crystal Report Designer interface.

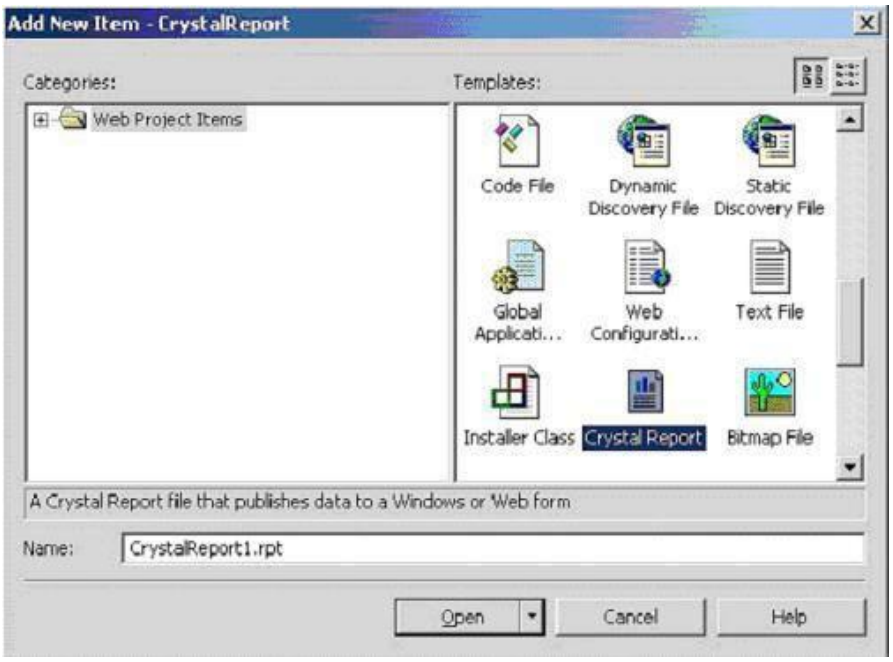
2. Place a CrystalReportViewer control from the toolbox on the .aspx page and set its properties to point to the .rpt file that we created in the previous step.

3. Call the databind method from your code behind page.

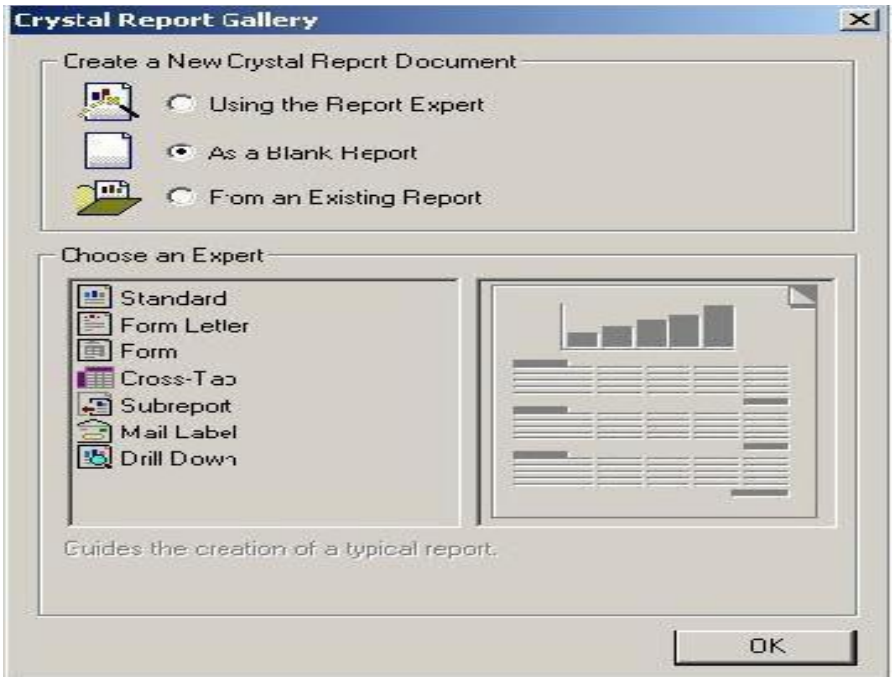
Creating Crystal Reports

I. Steps to create the report i.e. the .rpt file

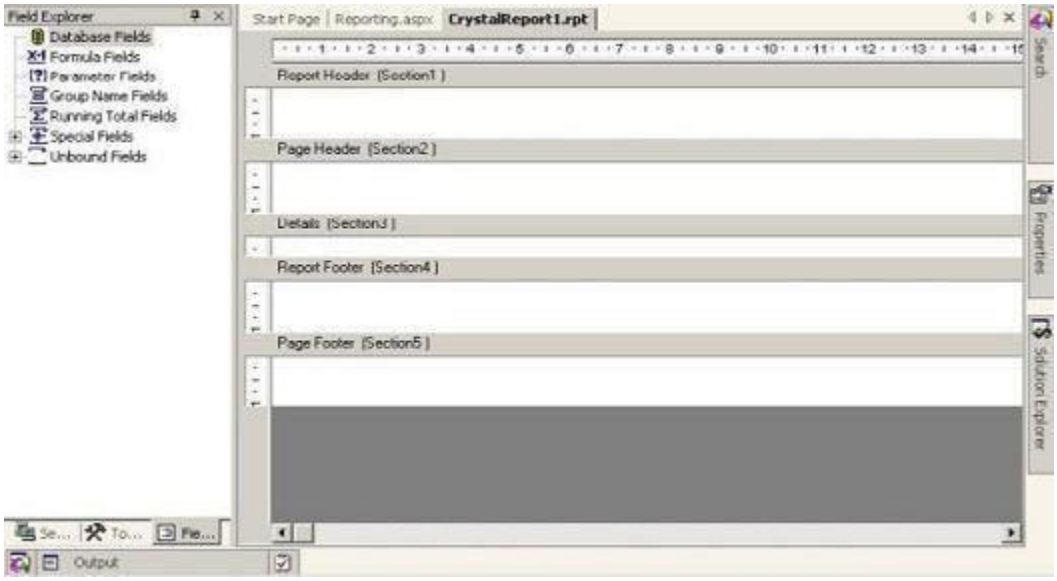
1) Add a new Crystal Report to the web form by right clicking on the "Solution Explorer", selecting "Add" --> "Add New Item" --> "CrystalReport".



2) On the "Crystal Report Gallery" pop up, select the "As a Blank Report"radio button and click "ok".



3) This should open up the Report File in the Crystal Report Designer.



4) Right click on the "Details Section" of the report, and select "Database" - "Add/Remove Database".

5) In the "Database Expert" pop up window, expand the "OLE DB (ADO)"option by clicking the "+" sign, which should bring up another "OLE DB (ADO)" pop up.

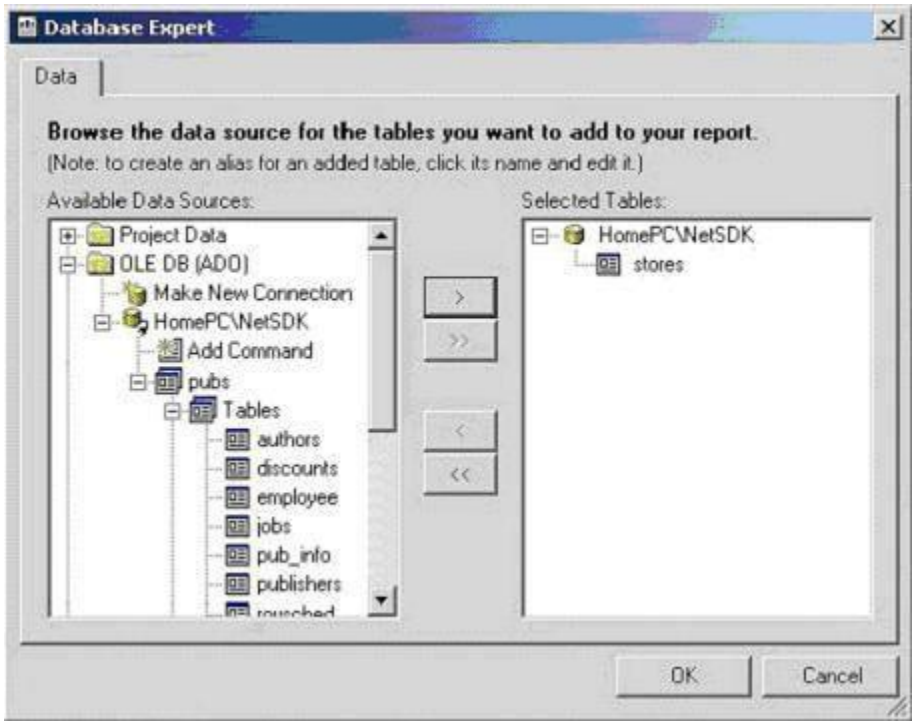
6) In the "OLE DB (ADO)" pop up, Select "Microsoft OLE DB Provider for SQL Server" and click Next.

7) Specify the connection information.

8) Click "Next" and then click "Finish".

9) Now you should be able to see the Database Expert showing the table that have been selected.

10) Expand the "Pubs" database, expand the "Tables", select the "Stores" table and click on ">" to include it into the "Selected Tables" section.

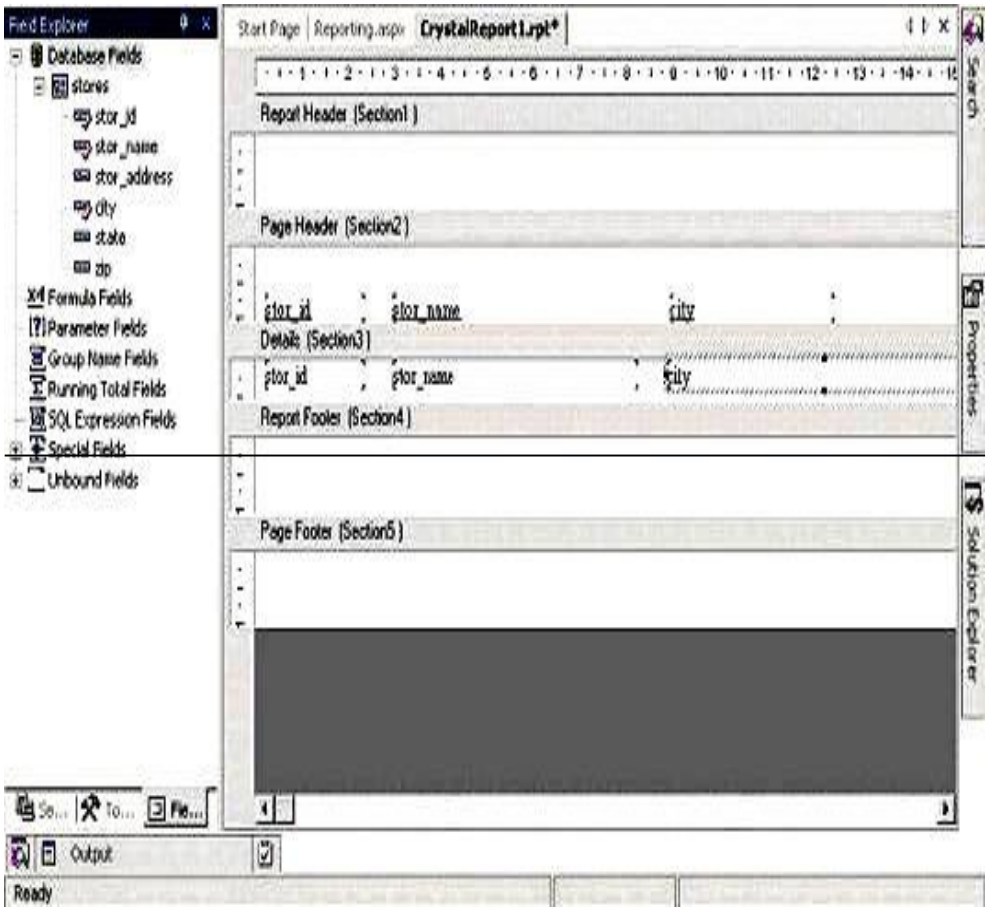


Note: If you add more than one table in the database Expert and the added tables have matching fields, when you click the OK button after adding the tables, the links between the added tables is displayed under the Links tab. You can remove the link by clicking the Clear Links button.

11) Now the Field Explorer should show you the selected table and its fields under the "Database Fields" section, in the left window.

12) Drag and drop the required fields into the "Details" section of the report. The field names would automatically appear in the "Page Header" section of the report. If you want to modify the header text then right click on the text of the "Page Header" section, select "Edit Text Object" option and edit it.

13) Save it.



Role of ADO.NET in Distributed Applications

The rapid development of web applications makes software development companies review the existing methods of working with data sources and adapt them to the web application specifications. The unpredicted growth of the number of clients makes web developers move from client-server to three-tier architecture, which sometimes brings out problems. Databases are unable to support the unlimited number of active connections limiting the availability of the site and causing losses. The ADO.NET (ActiveX Data Objects) technology can solve these problems and at the same time keep convenience and simplicity of programming.

Advantages and innovations in ADO.NET technology

- Using the disconnected model for accessing the data. ADO.NET application development technology offers an alternative to a traditional data access model. Normally, client-server applications use the technology of access to the data source where the connection with the base is maintained all the time. However, after the wide spread of the Internet based applications some vulnerabilities of this approach have been discovered. The experience of web developers has shown that the applications with the constant connection with the data source are difficult in scaling. All these problems are produced by the constant connection with database and are solved in ADO.NET. ADO.NET technology makes use of another data access model. ADO.NET access model establishes the connection only for some limited time when it's necessary to take some actions with the database. Thus, ADO.NET allows sidestepping these limitations of web application development process.
- Data string in the DataSet objects. In general, DataSet is a simplified relational database and can perform the most typical for these bases operations. Owing to ADO.NET application development technology, in contrast to Recordset, we can store several tables in one DataSet as well as the relations between them, perform the operations of selecting, deleting and updating the data. ADO.NET gives an opportunity any minute to get the latest information from the database using the call function FillDataSet. Thus, ADO.NET application development technology makes DataSet extremely convenient for most web applications. ADO.NET application development

technology allows us to extract the data from the base and somehow handle it whenever it is necessary.

- Deep integration with XML. XML, a widely spread language, plays an important role in ADO.NET and brings some more benefits to ADO.NET application development technology in comparison with the traditional approach. It isn't necessary for a programmer working with ADO.NET to have the experience of working with XML or the knowledge about this language. ADO.NET makes all the operations transparent for web developers. XML (eXtensible Markup Language) represents an industrial standard supported by practically any modern platform, which allows transmitting data to any component that can work with XML and can be executed under any operating system. Thus, deep integration of ADO.NET with XML provides .NET application developers with ample opportunities.

Many application developers have already noticed the simplicity and convenience of the ADO.NET technology. ADO.NET application development technology provides an intuitive interface and logical set of objects. All these features make ADO.NET more appealing to.NET web developers.

UNIT-III ADVANCED ASP.NET

ASP.NET is a Web application framework developed and marketed by Microsoft to allow programmers to build dynamic Web sites, Web applications and Web services. ASP.NET is built on the Common Language Runtime (CLR), allowing programmers to write ASP.NET code using any supported .NET language.

ASP.NET Web pages, known officially as Web Forms are the main building block for application development. Web forms are contained in files with an ".aspx" extension; these files typically contain HTML markup, as well as markup defining server-side Web Controls and User Controls. With ASP.NET Framework 2.0, Microsoft introduced a new *code-behind* model which allows static text to remain on the .aspx page, while dynamic code remains in an .aspx.vb or .aspx.cs or .aspx.fs file (depending on the programming language used).

ASP.NET and the .NET Framework

ASP.NET is part of the Microsoft .NET Framework. To build ASP.NET pages, you need to take advantage of the features of .NET Framework, which consists of two parts:

1. The Framework Class Library

2. The Common Language Runtime.

Understanding the Framework Class Library

The .NET Framework contains more than 13,000 classes you can use when building applications.

Framework Class Library was designed to make it easier to perform the most common programming tasks. Following are just a few examples of the classes in the framework:

- File class-Enables you to represent a file on your hard drive. You can use the File class to check whether a file exists, create a new file, delete a file, and perform many other file-related tasks.
- Graphics class-Enables you to work with different types of images such as GIF, PNG, BMP, and JPEG. You can use the Graphics class to draw rectangles, arcs, ellipses, and other elements on an image
- Random class-Enables you to generate a random number.
- SmtpClient class-Enables you to send email. You can use the SmtpClient class to send emails that contain attachments and HTML content.

Each class in the Framework can include properties, methods, and events. The properties, methods, and events exposed by a class are the members of a class.

For example, following is a partial list of the members of the SmtpClient class:

- Properties
- Host-The name or IP address of your email server
- Port-The number of the port to use when sending an email message
- Methods

- Send-Enables you to send an email message synchronously
- SendAsync-Enables you to send an email message asynchronously
- Events
- SendCompleted-Raised when an asynchronous send operation completes

For example, the SmtpClient class includes two properties named Host and Port, which enable you to specify the email server and port to use when sending an email message.

The SmtpClient class also includes two methods you can use to send an email:

i) Send()

ii) SendAsync().

The Send method blocks further program execution until the send operation is completed. The SendAsync() method, on the other hand, sends the email asynchronously. Unlike the Send() method, the SendAsync() method does not wait to check whether the send operation was successful.

Finally, the SmtpClient class includes an event named SendCompleted, which is raised when an asynchronous send operation completes. You can create an event handler for the SendCompleted event that displays a message when the email has been successfully sent.

Understanding Namespaces

All the classes are defined in single name called namespaces in ASP.NET.

For example

1. All the classes related to working with the file system are located in the System.IO namespace.

2. All the classes for working a Microsoft SQL Server database are located in the System.Data.SqlClient namespace.

Before you can use a class in a page, you must indicate the namespace associated with the class. There are multiple ways of doing this.

1. First you can fully qualify a class name with its namespace. For example, because the File class is contained in the System.IO namespace, you can use the following statement to check whether a file exists:

System.IO.File.Exists("SomeFile.txt")

2. A second option is to import a namespace.

You can add an <$I<%@ Import % directive><%@ Import %> directive to a page to import a particular namespace.

<%@ Import Namespace="System.Net.Mail" %>

ASP.NET gives you the most commonly used namespaces for free:

- System
- System.Collections

- System.Collections.Generic
- System.Collections.Specialized
- System.ComponentModel.DataAnnotations
- System.Configuration
- System.Data.Entity.Linq
- System.Data.Linq
- System.Text
- System.Text.RegularExpressions
- System.Web
- System.Web.Caching
- System.Web.DynamicData
- System.Web.SessionState
- System.Web.Security
- System.Web.Profile
- System.Web.UI
- System.Web.UI.WebControls
- System.Web.UI.WebControls.WebParts
- System.Web.UI.HtmlControls
- System.Xml.Linq

Understanding the Common Language Runtime

The second part of the .NET Framework is the Common Language Runtime (CLR). The Common Language Runtime is responsible for executing your application code.

When you write an application for the .NET Framework with a language such as C# or Visual Basic .NET, your source code is never compiled

directly into machine code. Instead, the C# or Visual Basic compiler converts your code into a special language named MSIL (Microsoft Intermediate Language).

MSIL looks very much like an object-oriented assembly language. However, unlike a typical assembly language, it is not CPU specific. MSIL is a low-level and platform-independent language.

When your application actually executes, the MSIL code is "just-in-time" compiled into machine code by the JITTER (the Just-In-Time compiler). Normally, your entire application is not compiled from MSIL into machine code. Instead, only the methods that are actually called during execution are compiled.

In reality, the .NET Framework understands only one language: MSIL. However, you can write applications using languages such as Visual Basic .NET and C# for the .NET Framework because the .NET Framework includes compilers for these languages that enable you to compile your code into MSIL.

You can write code for the .NET Framework using any one of dozens of different languages, including the following:

- Ada
- Apl
- Caml
- COBOL
- Eiffel
- Forth

- Fortran
- JavaScript
- Oberon
- PERL
- Pascal
- PHP
- Python
- RPG
- Scheme
- Small Talk

The vast majority of developers building ASP.NET applications write the applications in either C# or Visual Basic .NET.

Understanding ASP.NET Controls

ASP.NET controls are the heart of the ASP.NET Framework. An ASP.NET control is a .NET class that executes on the server and renders certain content to the browser.

The ASP.NET Framework contains over 70 controls. These controls can be divided into eight groups:

- Standard Controls—The standard controls enable you to render standard form elements such as buttons, input fields, and labels.
- Validation Controls—The validation controls enable you to validate form data before you submit the data to the server. For example, you can use a RequiredFieldValidator control to check whether a user entered a value for a required input field.

- Rich Controls—The rich controls enable you to render things such as calendars, file upload buttons, rotating banner advertisements, and multi-step wizards.
- Data Controls—The data controls enable you to work with data such as database data. For example, you can use these controls to submit new records to a database table or display a list of database records.
- Navigation Controls—The navigation controls enable you to display standard navigation elements such as menus, tree views, and bread crumb trails.
- Login Controls—The login controls enable you to display login, change password, and registration forms.
- HTML Controls—The HTML controls enable you to convert any HTML tag into a server-side control.

HTML Server Controls

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a runat="server" attribute to the HTML element. This attribute indicates that the element should be treated as a server control.

Note: All HTML server controls must be within a <form> tag with the runat="server" attribute!

Note: ASP.NET requires that all HTML elements must be properly closed and properly nested.

HTML Server Control Description

HtmlAnchor	Controls an <a> HTML element
HtmlButton	Controls a <button> HTML element
HtmlForm	Controls a <form> HTML element
HtmlGeneric	Controls other HTML element not specified by a specific HTML server control, like <body>, , , etc.
HtmlImage	Controls an <image> HTML element
HtmlInputButton	Controls <input type="button">, <input type="submit">, and <input type="reset"> HTML elements.)

ASP.NET Pages

The ASP.NET Framework enables you to create two different types of ASP.NET pages.

1. Single-file
2. Code Behind(two-file ASP.NET pages)

Single-file

In a single-file ASP.NET page, a single file contains both the page code and page controls. The page code is contained in a <script runat="server"> tag.

Code Behind(two-file ASP.NET pages)

you can create a two-file ASP.NET page. A two-file ASP.NET page is normally referred to as a "code-behind" page. In a code-behind page, the page code is contained in a separate file.

Code declaration blocks are lines of code enclosed in <script> tags. They contain the runat=server attribute, which tells ASP.NET that these controls can be accessed on the server and on the client. Optionally you can specify the language for the block. The code block itself consists of the definition of member variables and methods.

Example:

```
<asp:Label
id="Label1"
Runat="server" />
```

Note - When using Visual Web Developer, you create a code-behind page by selecting Web Site, Add New Item, selecting the Web Form Item, and checking the Place Code in Separate File check box before adding the page.

Page Events

Whenever you request an ASP.NET page, a particular set of events is raised in a particular sequence. This sequence of events is called the "page execution lifecycle."

Here is the sequence of events that are raised whenever you request a page:

1. PreInit

2. Init
3. InitComplete
4. PreLoad
5. Load
6. LoadComplete
7. PreRender
8. PreRenderComplete
9. SaveStateComplete
10. Unload

Advantages of ASP.NET (Features)

1. Separation of Code from HTML
   To make a clean sweep, with ASP.NET you have the ability to completely separate layout and business logic. This makes it much easier for teams of programmers and designers to collaborate efficiently. This makes it much easier for teams of programmers and designers to collaborate efficiently.

2. Support for compiled languages
   developer can use VB.NET and access features such as strong typing and object-oriented programming. Using compiled languages also means that ASP.NET pages do not suffer the performance penalties associated with interpreted code. ASP.NET pages are precompiled to byte-code and Just In Time (JIT) compiled when first requested. Subsequent requests are directed to the fully compiled code, which is cached until the source changes.

3. Use services provided by the .NET Framework The .NET Framework provides class libraries that can be used by your application. Some of the key classes help you with input/output, access to operating system services, data access, or even debugging. We will go into more detail on some of them in this module.

4. Graphical Development Environment
   Visual Studio .NET provides a very rich development environment for Web
   developers. You can drag and drop controls and set properties the way you do in Visual Basic 6. And you have full IntelliSense support, not only for your code, but also for HTML and XML.

5. State management
   To refer to the problems mentioned before, ASP.NET provides solutions for session and application state management. State information can, for example, be kept in memory or stored in a database. It can be shared across Web farms, and state information can be recovered, even if the server fails or the connection breaks down.

6. Update files while the server is running Components of your application can be updated while the server is online and clients are connected. The Framework will use the new files as soon as they are copied to the application. Removed or old files that are still in use are kept in memory until the clients have finished.

7. XML-Based Configuration Files

   Configuration settings in ASP.NET are stored in XML files that you can easily read and edit. You can also easily copy these to another server, along with the other files that comprise your application.

ASP.NET Overview

Here some point that gives the quick overview of ASP.NET.

- ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services
- Like ASP, ASP.NET is a server-side technology
- Web Applications are built using Web Forms. ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.
- Web Forms are designed to make building web-based applications as easy as building Visual Basic applications

ASP.NET Architecture

ASP.NET is based on the fundamental architecture of .NET Framework. Visual studio provide a uniform way to combine the various features of this Architecture.

Architecture is explained form bottom to top in the following discussion.

1. At the bottom of the Architecture is Common Language Runtime. NET Framework common language runtime resides on top of the operating system services. The common language runtime loads and executes code that targets the runtime. This code is therefore called managed code. The runtime gives you, for example, the ability for cross-language integration.
2. .NET Framework provides a rich set of class libraries. These include base classes, like networking and input/output classes, a data class library for data access, and classes for use by programming tools, such as debugging services. All of them are brought together by the Services Framework, which sits on top of the common language runtime.
3. ADO.NET is Microsoft's ActiveX Data Object (ADO) model for the .NET Framework. ADO.NET is not simply the migration of the popular ADO model to the managed environment but a completely new paradigm for data access and manipulation.

ADO.NET is intended specifically for developing web applications. This is evident from its two major design principles:

1. Disconnected Datasets—In ADO.NET, almost all data manipulation is done outside the context of an open database connection.
2. Effortless Data Exchange with XML—Datasets can converse in the universal data format of the Web, namely XML.

4. The 4th layer of the framework consists of the Windows application model and, in parallel, the Web application model. The Web application model-in the slide presented as ASP.NET- includes Web Forms and Web Services. ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.

   Web Services brings you a model to bind different applications over the Internet. This model is based on existing infrastructure and applications and is therefore standard-based, simple, and adaptable.

   Web Services are software solutions delivered via Internet to any device. Today, that means Web browsers on computers, for the most part, but the device-agnostic design of .NET will eliminate this limitation.
5. One of the obvious themes of .NET is unification and interoperability between various programming languages. In order to achieve this;

certain rules must be laid and all the languages must follow these rules. In other words we cannot have languages running around creating their own extensions and their own fancy new data types. CLS is the collection of the rules and constraints that every language (that seeks to achieve .NET compatibility) must follow.

6. The CLR and the .NET Frameworks in general, however, are designed in such a way that code written in one language can not only seamlessly be used by another language. Hence ASP.NET can be programmed in any of the .NET compatible.

AdRotator Control

The AdRotator control is used to display a sequence of ad images.The AdRotator is a special purpose control in ASP.NET that is used to display flashing Banner ads. The control is capable of displaying ads randomly or sequentially as set by the users. Each time the page is refreshed or reloaded a new ad can be displayed to the user. We can also assign priorities to the ads in such a way that certain ads are displayed frequently than others.

AdRotator control is available in ASP.Net to make the task of rotating the advertisement images in a web form quickly and easily.

AdRotator control are used to create a dynamic ads. The AdRotator Control presents ad images each time a user enters or refreshes a webpage. When the ads are clicked, it will navigate to a new Web location. The AdRotator control is used to display a sequence of ad images.The AdRotator control to work we need an Advertisement file (XML file) and some sample images.

Adding the AdRotator web server control to your web application. First, select the AdRotator and drag and drop the control to your web form. Map the XML file which contains the details about each and every ad.

This control uses an XML file to store the ad information. The XML file must begin and end with an <Advertisements> tag. Inside the <Advertisements> tag there may be several <Ad> tags which defines each ad.

The predefined elements inside the <Ad> tag are listed below:

Element	Description
<ImageUrl>	Optional. The path to the image file
<NavigateUrl>	Optional. The URL to link to if the user clicks the ad
<AlternateText>	Optional. An alternate text for the image
<Keyword>	Optional. A category for the ad
<Impressions>	Optional. The display rates in percent of the hits

Properties

Property	Description
AdvertisementFile	Specifies the path to the XML file that contains ad information
AlternateTextField	Specifies a data field to be used instead of the Alt text for an ad
ImageUrlField	Specifies a data field to be used instead of the

	ImageURL attribute for an ad
KeywordFilter	Specifies a filter to limit ads after categories
NavigateUrlField	Specifies a data field to be used instead of the NavigateUrl attribute for an ad
runat	Specifies that the control is a server control. Must be set to "server"
Target	Specifies where to open the URL
Height	The height of the ad in pixels. This value overrides the default height setting for the AdRotator control.
Width	The width of the ad in pixels. This value overrides the default width setting for the AdRotator control.

The advertisement file is an XML file. The following are some of the elements of this XML file.

XML code that has the details about the ads. The file Ads.xml looks like the code below:

```
<Advertisements>
  <Ad>
   <ImageUrl>adimages/2.jpg</ImageUrl>
   <NavigateUrl>http://cat2.com</NavigateUrl>
   <AlternateText>Cat 2</AlternateText>
   <Impressions>30</Impressions>
```

```
</Ad>
<Ad>
  <ImageUrl>adimages/3.jpg</ImageUrl>
  <NavigateUrl>http://cat3.com</NavigateUrl>
  <AlternateText>Cat 3</AlternateText>
  <Impressions>20</Impressions>
</Ad>
```

Following are the important events of the AdRotator Class:

Events	Description
AdCreated	It is raised once per round trip to the server after creation of the control, but before the page is rendered
DataBinding	Occurs when the server control binds to a data source.
DataBound	Occurs after the server control binds to a data source.
Disposed	Occurs when a server control is released from memory, which is the last stage of the server control lifecycle when an ASP.NET page is requested
Init	Occurs when the server control is initialized, which is the first step in its lifecycle.
Load	Occurs when the server control is loaded into the Page object.

Unload	Occurs when the server control is unloaded from memory.

MultiView Control:

The MultiView control represents a control that acts as a container for groups of View controls.. It creates a set of views and one view is visible at a time. Use View control to create views inside the MutliView control. Add the View controls into a MultiView control.

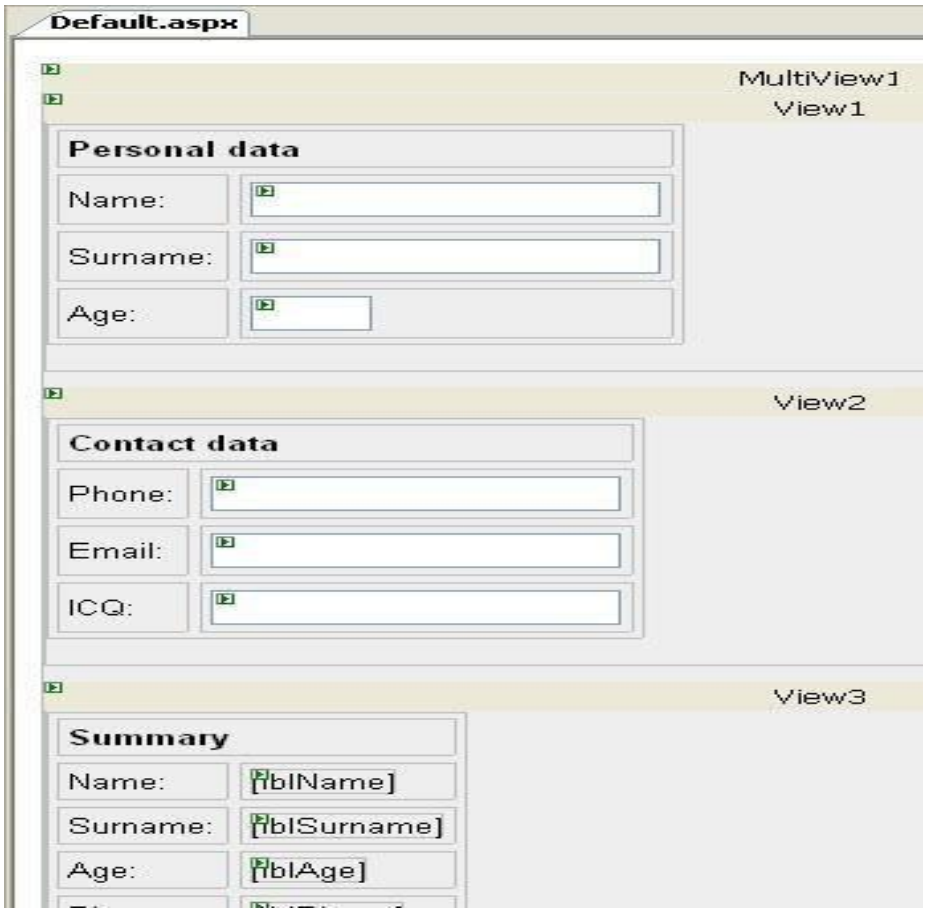
The main advantage of the multiview control is that we can specify the required view only (i.e display the required view only) on a single page.Multiview control helps us to create different views in the same page and display the view as the user clicks the links.
The MultiView control acts as a container for groups of View controls. Each View control in turn contains child controls such as buttons and text boxes.

The MultiView and Wizard controls both allow you to create multiple sections of controls on the same page. The Wizard control has features built in to facilitate its operation such as built-in Next and Prev buttons. With the Multiview control the responsibility to add and code the navigation falls to you.Figure: MultiView controls contain individual View controls

Me.MultiView1.ActiveViewIndex = 1
Me.MultiView1.SetActiveView( )

Me.MultiView1.Views(1))

Figure : The NextView CommandName property value advances to the next view.



Properties of the View and MultiView controls

Both the View and MultiView controls are derived from Control class and inherits all its properties, methods and events. The most important property of the View control is Visible property of type Boolean, which sets the visibility of a view.

The MultiView control has the following important properties:

Properties	Description
Views	Collection of View controls within the MultiView
ActiveViewIndex	A zero based index that denotes the active view; if no view is active then the index is -1.

The CommandName attribute of the Button controls associated with the navigation of the MultiView control are associated with some related field of the MultiView control.

For example, if a Button control with CommandName value as NextView is associated with the navigation of the multiview, it automatically navigates to the next view when the button is clicked.

The following table shows the default command names for the above properties:

Properties	Description
NextViewCommandName	NextView

PreviousViewCommandName	PrevView
SwitchViewByIDCommandName	SwitchViewByID
SwitchViewByIndexCommandName	SwitchViewByIndex

The following are the important methods of the MultiView control:

Methods	Description
SetActiveview	Sets the active view
GetActiveview	Retrieves the active view

Every time a view is changed, the page is posted back to the server and a number of events are raised. Some important events are:

Events	Description
ActiveViewChanged	Raised when a view is changed
Activate	Raised by the active view
Deactivate	Raised by the inactive view

Apart from the above mentioned properties, methods and events, multi view control inherits the members of the control and object class.

Wizard Control

Wizard control eliminates the need to design forms to execute a step by step process in the actual business flow. This simplifies the work of developers to design and write the code. The ASP.NET Wizard control simplifies many of the tasks associated with building a series of forms to collect user data. The control provides a mechanism that allows you to easily build the desired wizard as a collection of steps

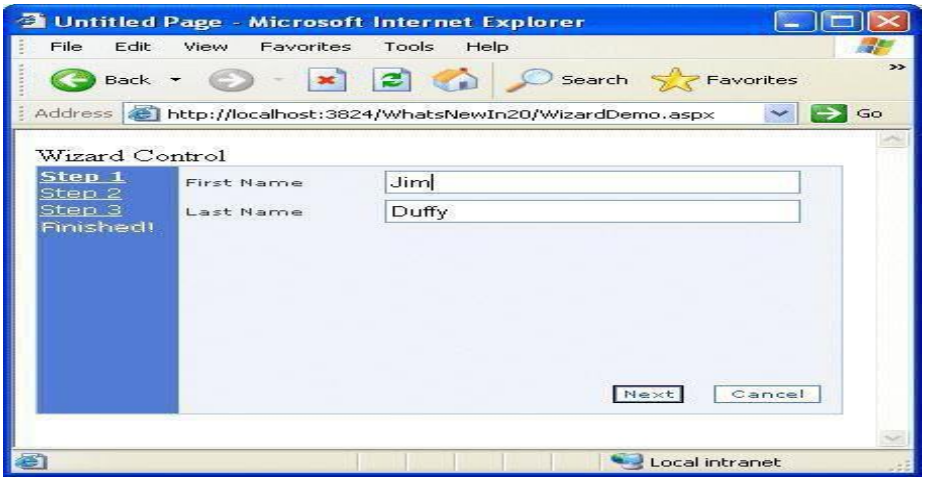
The control provides a mechanism that allows you to easily build the desired wizard as a collection of steps, add a new step, or reorder the steps. You don't have to write any infrastructure whatsoever for navigation or to persist user data between steps.

The Wizard control works much like the MultiView control in that they both contain sections to place controls in. While the sections in a MultiView control are views, the sections in a Wizard control are called steps. The Wizard control has features built in to facilitate its operation such as built-in Next and Prev buttons. With the Multiview control the responsibility to add and code the navigation falls to you.

These steps are stored in the WizardSteps collection. The primary difference between the two controls is that the Wizard control can display links to all of the steps in a sidebar on the left-hand side of the control.

You can add or remove steps from a wizard control by selecting the Add/Remove WizardSteps option from the smart tag Wizard Tasks menu.

Figure : Use the Wizard control to implement step-by-step processes.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:3824/WhatsNewIn20/WizardDemo.aspx Go
Wizard Control
Step 1
Step 2
Step 3
Finished!
First Name Jim
Last Name Duffy
Next Cancel
Local intranet


Add a New Employee
Header
First Name Dino
Last Name
Sidebar
Enter Employee Name
Personal Information
Optional Information
Finalizing...
View
Next
Navigation Bar

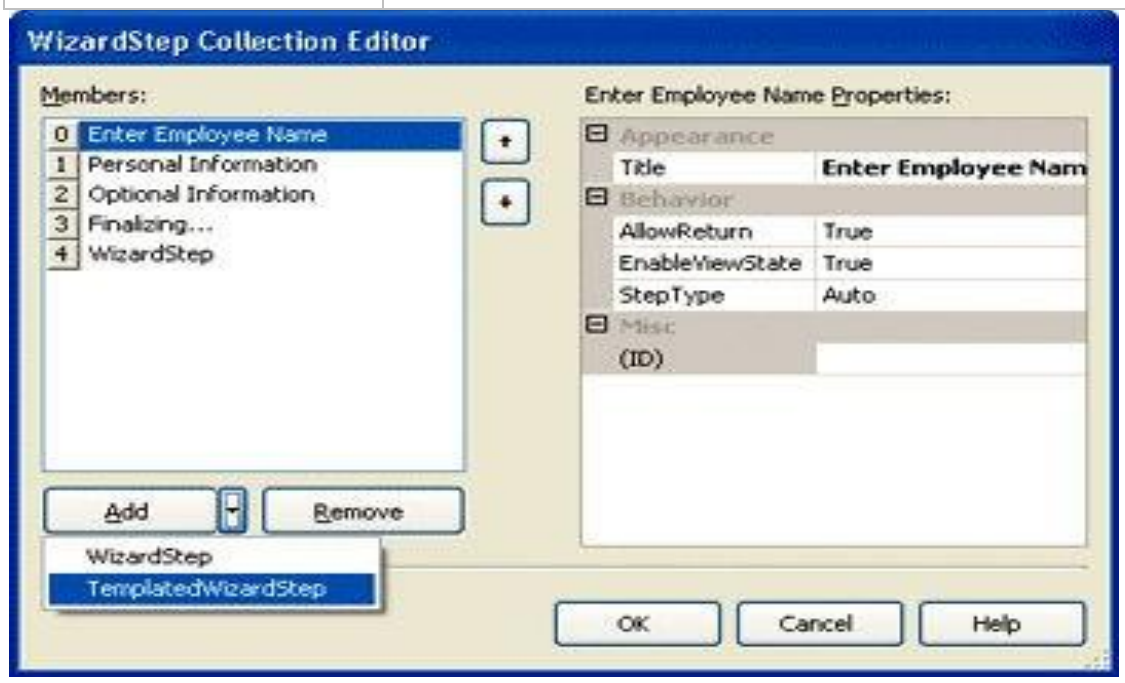
Wizard control properties:

Template	Description
FinishNavigationTemplate	Specifies the navigation bar shown before the last page of the wizard; by default, navigation bar contains the Previous and Finish buttons
HeaderTemplate	Specifies the title bar of the wizard
SideBarTemplate	Used to display content in the left side of the wizard control
StartNavigationTemplate	Specifies the navigation bar for the first view in the wizard; by default, contains only the Next button
StepNavigationTemplate	Specifies the navigation bar for steps other than first, finish, or complete; by default, contains Previous and Next buttons

CancelButtonStyle	Wizard Cancel button
FinishStepButtonStyle	Wizard Finish button
FinishStepPreviousButtonStyle	Wizard Previous button at the finish step
HeaderStyle	Wizard header
NavigationButtonStyle	Navigation buttons
NavigationStyle	Navigation area
NextStepButtonStyle	Wizard Next button
PreviousStepButtonStyle	Wizard Previous button

SideBarButtonStyle	Buttons on the sidebar
StartStepNextButtonStyle	Wizard Next button at the start step
StepStyle	Area where steps are displayed

Property	Description
ActiveStep	Returns the current wizard step object; the object is an instance of t WizardStep class
ActiveStepIndex	Gets and sets the zero-based index of current wizard step
DisplaySideBar	Toggles the visibility of the sidebar; the default value is True
FinishStepButtonText	Gets and sets the text for the Finish button
HeaderText	Gets and sets the title of the wizard
NextStepButtonText	Gets and sets the text for the Next button
PreviousStepButtonText	Gets and sets the text for the Previous button



The wizard ActiveStep property can be set by the ActiveStepIndex="0" of the wizard control. The first form will have the next navigation control.

Setting	Description
Auto	Default setting; forces the wizard to determine how each contained step should be treated.
Complete	The last page that the wizard displays, usually after the wizard has been completed. The navigation bar and the sidebar aren't displayed.
Finish	The last page used for collecting user data. It lacks the Next button, and it shows the Previous and Finish buttons.
Start	The first screen displayed, with no Previous button.
Step	All other intermediate pages, in which the Previous and Next

buttons are displayed.

Wizard Events -

- ActiveStepChanged - This event fires when the control go to a new step.
- CancelButtonClick - The cancel button is not shown by default, you can add to every step by Wizard.DisplayCancelButtop property. This event fires when Cancel button clicked.
- FinishButtonClick - This event fires when FinishButton clicked.
- NextButtonClick - This event fires when Next button clicked on any step.
- PreviousButtonClick - This event fires when Previous button clicked on any step.
- SideBarButtonClick - This event fires when a button in the sidebar area clicked.

ImageMap control

The ASP.NET ImageMap control allows you to create an image that has individual regions that users can click, which are called hot spots. Each of these hot spots can be a separate hyperlink or postback event.

ImageMap Elements:

- The ImageMap control consists primarily of two pieces. The first is an image, which can be a graphic in any standard web graphic format, such as a .gif, .jpg, or .png file.
- The second element is a collection of hotspot controls. Each hotspot control is a different element. For each hotspot control, you define its shape — a circle, rectangle, or polygon — and the coordinates that specify the location and size of the hot spot.

Different types of hot spots

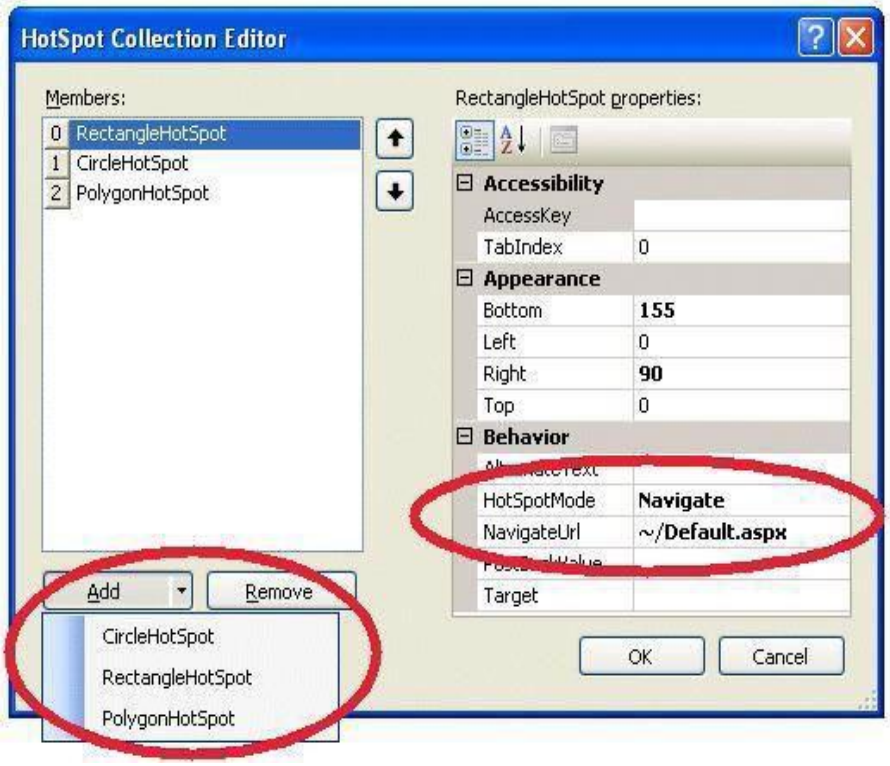
There are three different types of hot spots offered by ImageMap control. They are:

- CircleHotspot
- RectangleHotspot
- PolygonHotspot

CircleHotspot: CircleHotspot defines circle shaped hot spot region in an ImageMap control. To define the region for a circle hot spot, we should define X and Y coordinates for circle as well as radius property which usually is the distance from the center of circle to the edge.

RectangleHotspot: RectangleHotspot defines rectangle shaped hot spot region in an ImageMap control. To define the region for a Rectangle hot

spot, we define Top, Bottom, Left and Right coordinates. Similar is the case for the Polygon hot spot.



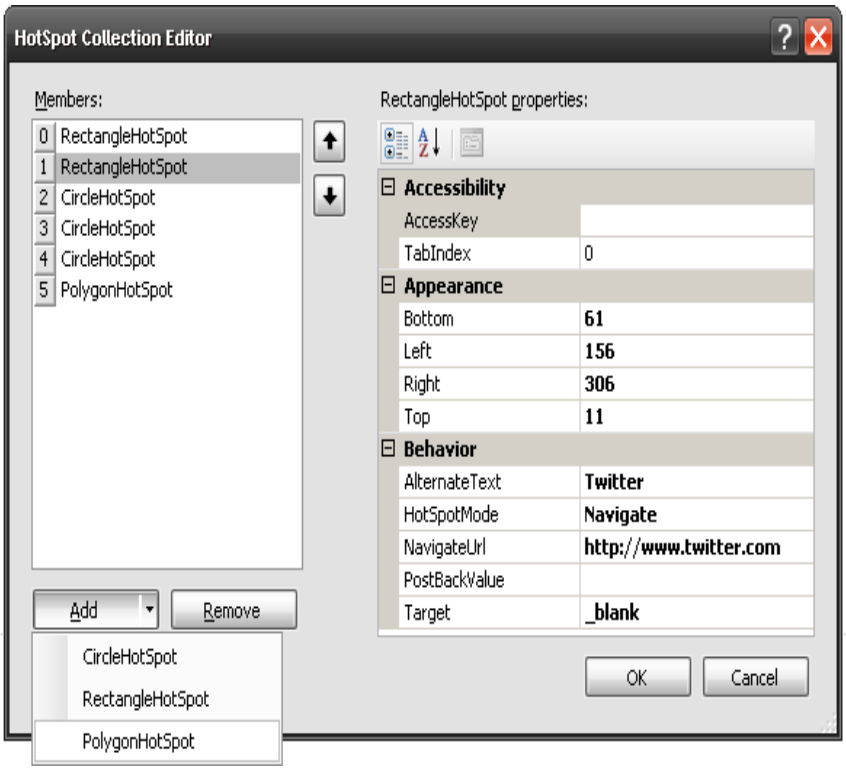
Hot spots features

- There is no limit on number of hotspots each image may contain.
- Each hotspot is characterized by various attributes like shape, location and size.

- Overlapping hotspots are perfectly valid.
- Hot spots are defined using x and y coordinates.
- Hot spots can be assigned Image URL's and are capable of postback.

All these hotspots types have some properties to customize the hotspot region and behavior. We can specify Left, Top, Right and Bottom for Rectangle hotspot and X, Y and Radius properties for Circle hotspot.

Each hot spots can be configured as a hyperlink that goes to a URL that you provide for that hot spot. Alternatively, we can configure the control to perform a postback when a user clicks a hot spot, providing a unique value for each hot spot. The postback raises the ImageMap control's Click event. In the event handler, you can read the unique value that you assign to each hot spot.



Master pages:

ASP.NET master pages allow you to create a consistent layout for the pages in your application. A single master page defines the look and feel and standard behavior that you want for all of the pages (or a group of pages) in your application. You can then create individual content pages that contain the content you want to display. When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page.

Replaceable Content Placeholders

In addition to static text and controls that will appear on all pages, the master page also includes one or more ContentPlaceHolder controls. These placeholder controls define regions where replaceable content will appear. In turn, the replaceable content is defined in content pages.

Content Pages

You define the content for the master page's placeholder controls by creating individual content pages, which are ASP.NET pages (.aspx files and, optionally, code-behind files) that are bound to a specific master page. The binding is established in the content page's @ Page directive by including a MasterPageFile attribute that points to the master page to be used.

From the user's perspective, the combined master and content pages are a single, discrete page. The URL of the page is that of the content page.

From a programming perspective, the two pages act as separate containers for their respective controls. The content page acts as a container for the master page. However, you can reference public master-page members from code in the content page.

Note that the master page becomes a part of the content page. In effect, the master page acts in much the same way a user control acts — as a child of the content page and as a container within that page. In this case, however, the master page is the container for all of the server controls that are rendered to the browser.

Advantages of Master pages

Master pages provide functionality that developers have traditionally created by copying existing code, text, and control elements repeatedly; using framesets; using include files for common elements; using ASP.NET user controls; and so on.

Advantages of master pages include the following:

- They allow you to centralize the common functionality of your pages so that you can make updates in just one place.
- They make it easy to create one set of controls and code and apply the results to a set of pages. For example, you can use controls on the master page to create a menu that applies to all pages.
- They give you fine-grained control over the layout of the final page by allowing you to control how the placeholder controls are rendered.
- They provide an object model that allows you to customize the master page from individual content pages.

- Master pages allow you to create a consistent look and behavior for all the pages (or group of pages) in your web application.
- A master page provides a template for other pages, with shared layout and functionality. The master page defines placeholders for the content, which can be overridden by content pages. The output result is a combination of the master page and the content page.
- The content page contains the content you want to display.

- When users request the content page, ASP.NET merges the pages to produce output that combines the layout of the master page with the content of the content page.

Master Page Example

```
<%@ Master %>
<html>
<body>
<h1>Standard Header For All Pages</h1>
<asp:ContentPlaceHolder id="CPH1" runat="server">
</asp:ContentPlaceHolder>
</body>
</html>
```

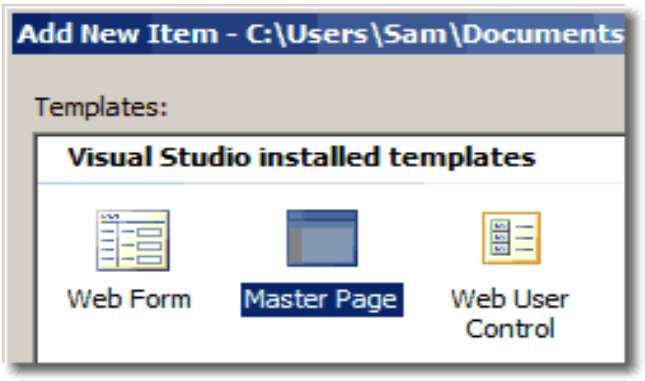
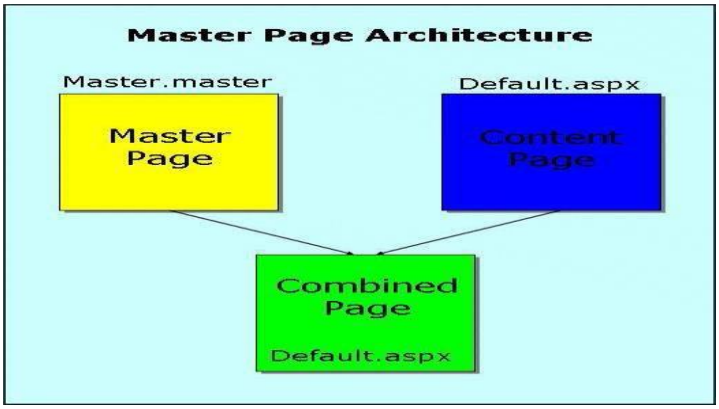


Figure: Master pages combine with content pages to form the rendered page.



A complete master page contains HTML, optional ASP.NET web controls, optional user controls, and one or more required ContentPlaceHolder controls. A ContentPlaceHolder control is a special ASP.NET web container control (<asp:contentplaceholder>) responsible for containing the controls placed on a content web page. You will find the ContentPlaceHolder control in the Standard section of the Toolbox. You can place one or more ContentPlaceHolder controls on a master page. All content on a content form is restricted to one of the ContentPlaceHolder controls defined on the content page's master page. The master page content on a content page is grayed out and not editable from within the content page. The only live areas available in the designer are the ContentPlaceHolder controls defined in the master page.

To create a master page you simply create a new page of type master page (see Figure).

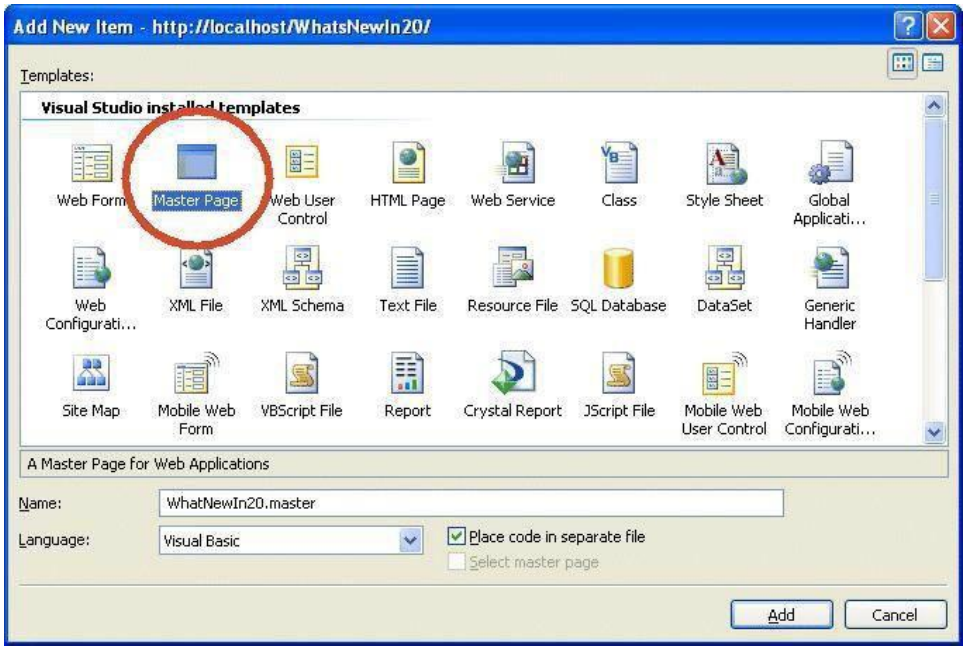


Figure: A master page is comprised of HTML, web controls, and one or more ContentPlaceHolders

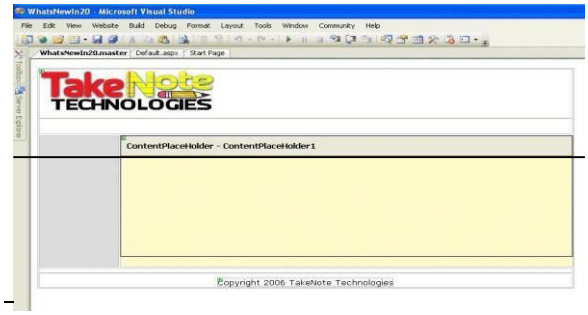
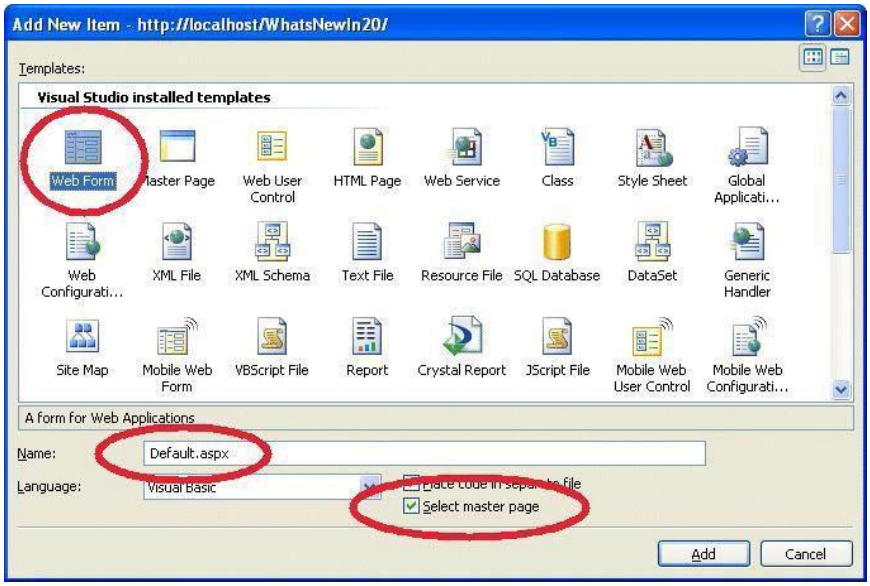


Figure: Check the "Select master page" checkbox to create a content page.



Site Navigation

You can easily build navigation into your pages by using the following ASP.NET site-navigation controls:

There are three Site Navigation control in asp.net

- SiteMapPath Control
- Menu Control
- Treeview Control

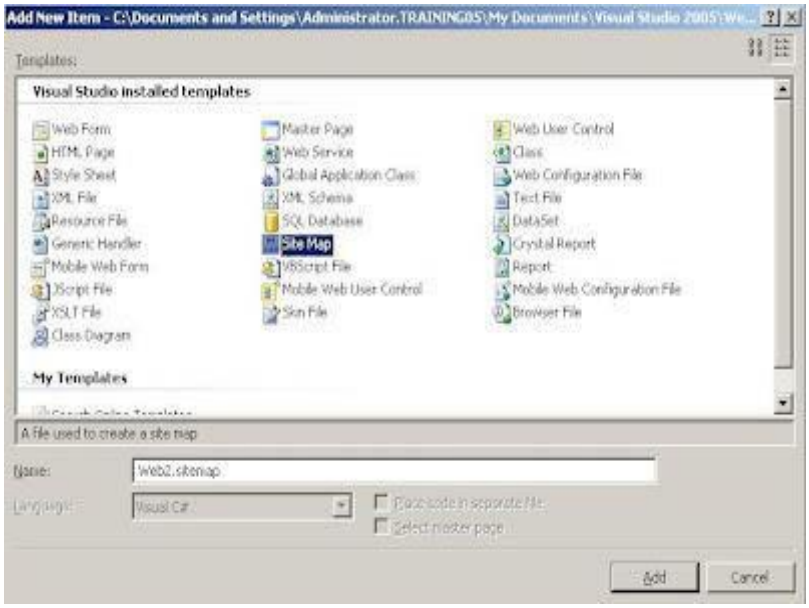
Before using Site Navigation control let go through overview of sitemap, which is used as datasource to assign this control.

SiteMap

SiteMap file is an XML File, which contains details of navigation that is followed by navigation control.

For Creating Web.SiteMap file

- Right click the project in solution explorer and add new item.
- Select Site Map from the add new item dialog.
- It will create "web.sitemap" file in your root directory



A Sample format of Site Map

<siteMapNode url="" title="" description="">
<siteMapNode url="" title="" description="" />
<siteMapNode url="" title="" description="" />
</siteMapNode>

Here, siteMapNode contains following properties. Url - describes URL to be redirect on node click. Title - describes Text to be displayed on node.

Description - describes Text to be displayed as Tooltip.

SiteMapPath

This control displays a navigation path — which is also known as a breadcrumb or eyebrow — that shows the user the current page location and displays links as a path back to the home page.

The control provides many options for customizing the appearance of the links.

The SiteMapPath control creates navigation mechanism which is generally referred to as *breadcrumb navigation.* This is a linear path defining where the user is currently located in navigation structure.It usually helps end user to know his location in relation to the rest of the site

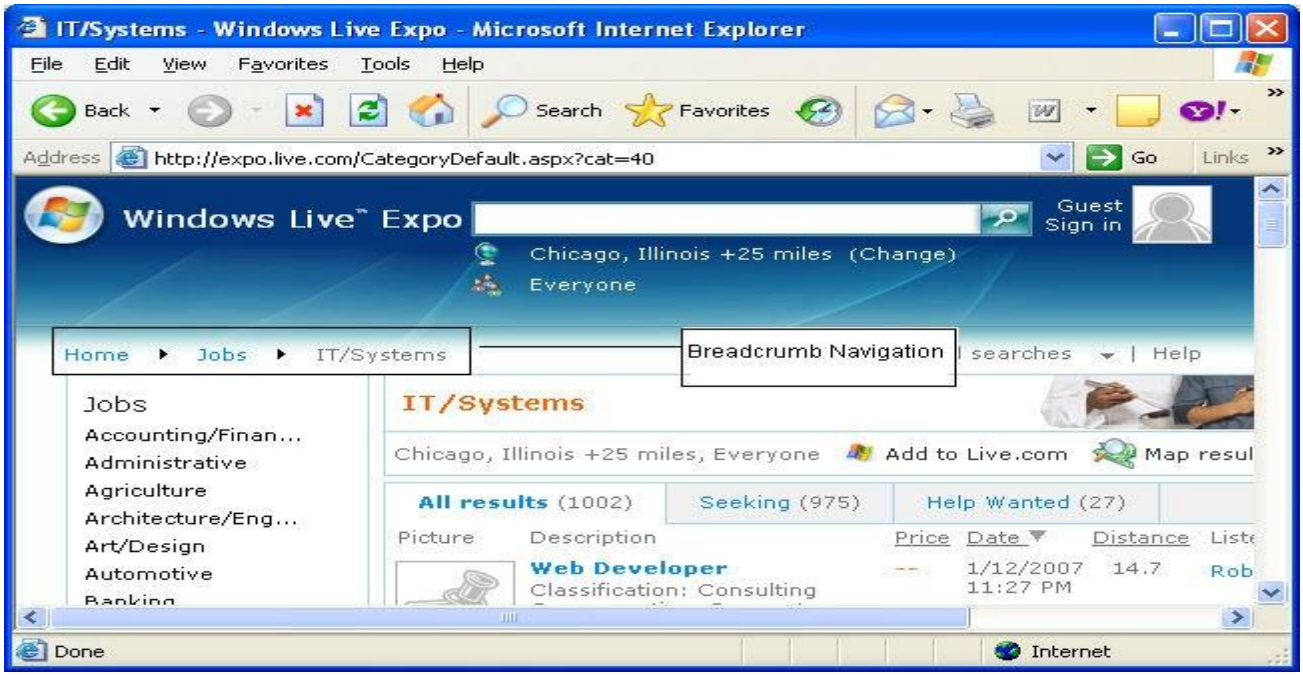


Fig.Breadcrumb navigation in msn website. Surrounded by Black box

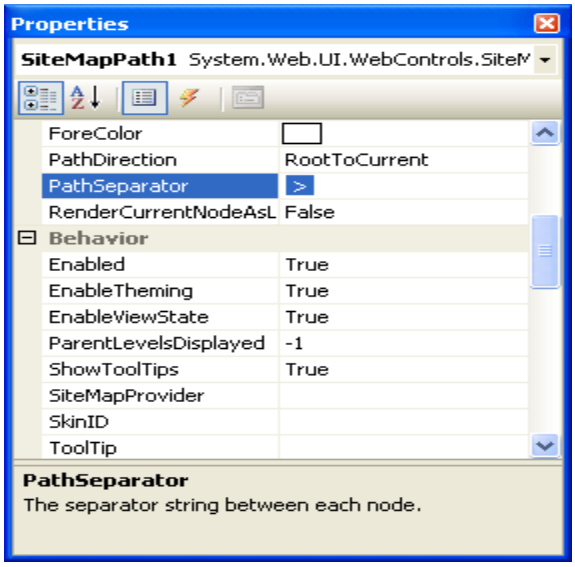
The SiteMapPath control creates *breadcrumb navigation* with very little effort on your part.

Fig.Output Listing of AsiaNews page displaying breadcrumb navigation

Properties of SiteMapPath Control:

PathSeparator Property:

PathSeparator property defines the element to separate the link elements. By default, (>) greater than symbol is used to separate link elements as shown in above listing



PathDirection Property:

This property changes the direction of the links generated in the output. Possible settings for this property are RootToCurrent and CurrentToRoot. In the above example, I have used default RootToCurrent setting. If we change

the setting to CurrentToRoot you will the output as shown below. I think you got the difference.

Output listing with PathDirection set to CurrentToRoot

Public Properties of SiteMapPath class

ParentLevelsDisplayed : It specifies the number of levels of parent nodes and then displays the control accordingly related to the currently displayed node.

RenderCurrentNodeAsLink : It specifies whether or not the site navigation node that represents the currently displayed page is rendered as a hyperlink.

PathSeperator : It specifies the string that displays the SiteMapPath nodes in the rendered navigation path.

Style properties of the SiteMapPath class

CurrentNodeStyle : It specifies the style used for the display text for the current node.

RootNodeStyle : It specifies the style for the root node style text.

NodeStyle : It specifies the style used for the display text for all nodes in the site navigation path.

a. NodeStyle property: Applies styles to all links in the sitemap uniformly
b. RootNodeStyle property : Applies styles to the root link in the SiteMapPath navigation
c. CurrentNodeStyle property: Gets the style used for the display text for the current node.

Menu Control:

Another important navigation control in ASP.NET 3.5 which allows the end user to navigate through a large collection of options(links) with very less effort. This is used to display menu in a web page and used in combination with SiteMapDataSource control for navigating a web site. You can customize the appearance of menu control through styles, user-defined templates and themes.

The Menu control is used to display menus. The menu can be displayed vertically or horizontally. Below are two screenshots showing vertical and horizontal menus:

Horizontal Menu:

Vertical Menu:

For making a menu change its orientation you just need to change the Orientation property of the Menu control to Horizontal or Vertical.

Eg.,

Menu.Orientation = Orientation.Vertical;

Setting the property to Orientation.Horizontal changes the orientation back to horizontal.

You can set individual properties of the Menu control to specify the size, color, font, and other characteristics of its appearance. In addition, you can apply skins and themes to the Menu control.

Menu control displays two types of menus: a Static menu and Dynamic menu. The static menu is always displayed in menu control. By default, only menu items at the root levels are displayed. You can also display additional menu levels by setting StaticDisplayLevels property.

Menu items with a higher level than the value specified by StaticDisplayLevels property are displayed in dynamic menu. A Dynamic menu appears only when the user positions the mouse pointer over the parent menu item that contains a Dynamic submenu.

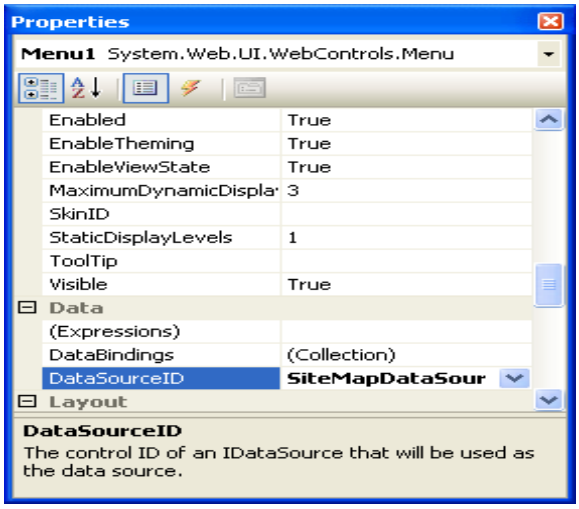
Menu control exposes events such as MenuItemClick, Load,UnLoad,Prerender,DataBound and certain other events. MenuItemClick event enables you to take some action when end user clicks one of the available menu items.

When the user clicks a menu item, the Menu control can either navigate to a linked Web page or simply post back to the server. If the NavigateUrl of a menu item is set, the Menu control navigates to the linked page; otherwise, it posts the page back to the server for processing. By default, a linked page is displayed in the same window as menu control.

Features

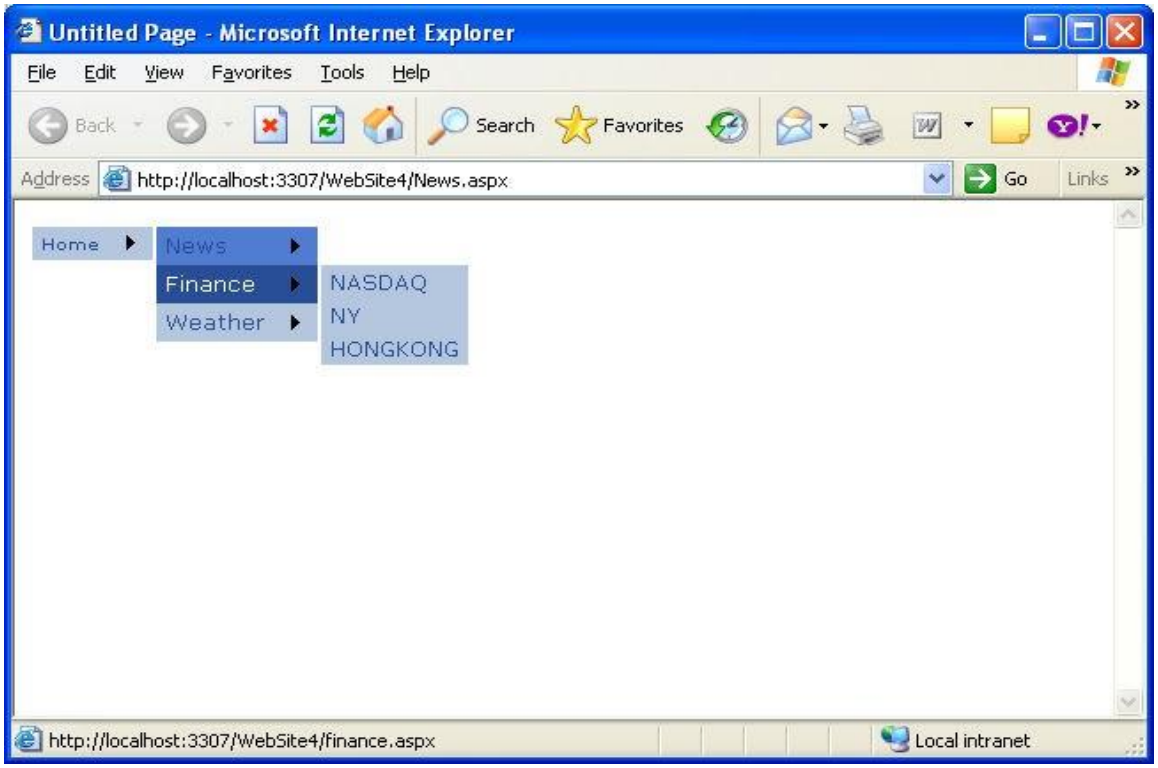
- Multiple menus on a single page are allowed.
- Generated code is dramatically small.
- Support for scrollable menus.
- Support for tooltips.
- Support for client-side and server-side events.
- Colors, sizes, borders and fonts can all be customized to visually integrate with any project.
- Support for icons, background images, separators, user-defined images for submenu arrows.
- Each menu item, submenu, separator can be customized separately.
- Each submenu can be positioned separately.

Drag and drop the menu server control from Navigation Section of Toolbox and similarly drag and drop the SiteMapDataSource control from Data Section of Toolbox and connect the two by using Menu control's DataSourceId property. From this example, you can see that I'm using a SiteMapDataSource control that automatically works with the application's web.sitemap file. DataSourceID property will connect the menu control with SiteMapDataSource control



Menu Control Properties dialog box

Fig.Output listing for Menu Control using SiteMapDataSource control using sitemap xml



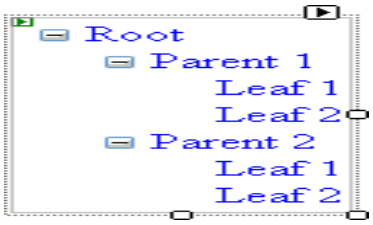
TreeView Control

TreeView control display data in hierarchical order.

The TreeView control consists of nodes and there are three types of nodes that you can add to a TreeView control.

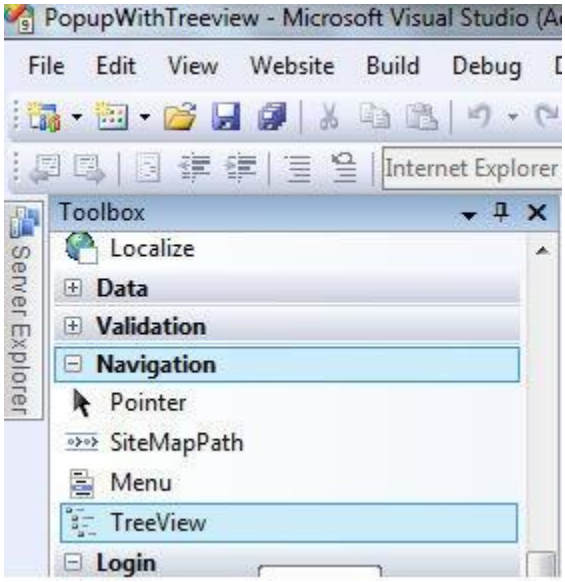
- Root - A root node is a node that has no parent node. It has one or more child nodes.
- Parent - A node that has a parent node and one or more child nodes
- Leaf - A node that has no child nodes

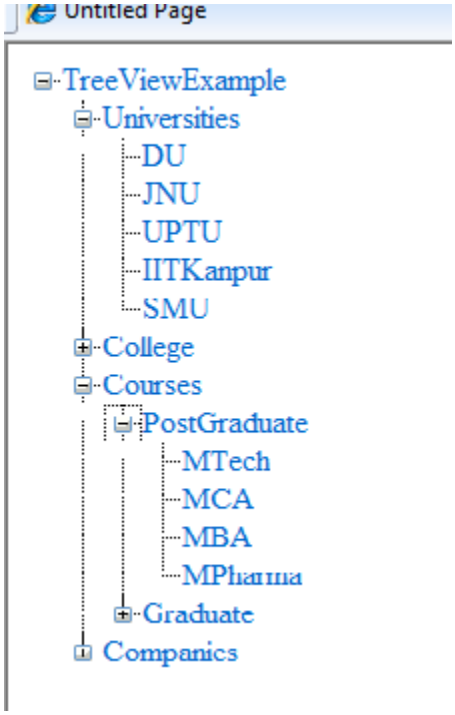
Figure 1: TreeView control structure

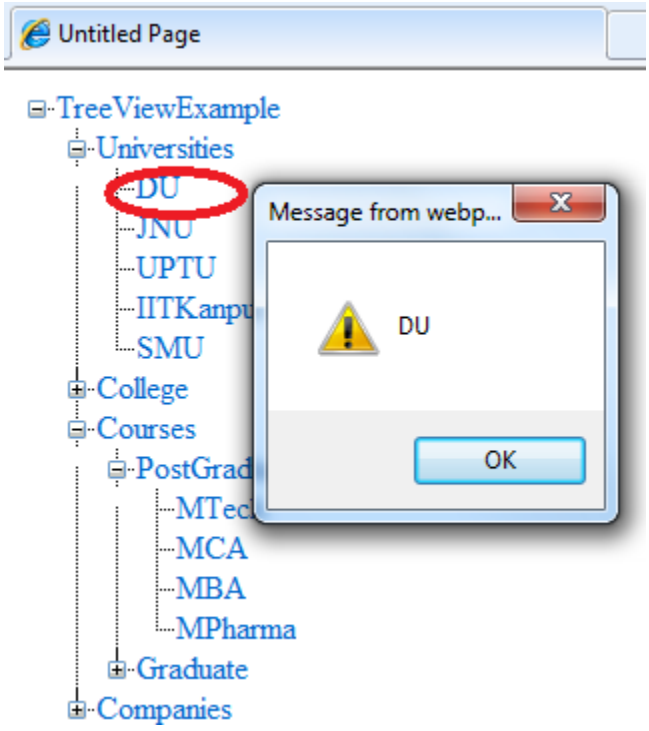


Follow these steps:

Step 1: Open Visual Studio 2008 and drag a TreeView control from the toolbar and drop on page as follows:


PopupWithTreeview - Microsoft Visual Studio
File Edit View Website Build Debug
Internet Explorer
Toolbox
Server Explorer
Localize
Data
Validation
Navigation
Pointer
SiteMapPath
Menu
TreeView
Login


Untitled Page
TreeViewExample
Universities
DU
JNU
UPTU
IITKanpur
SMU
College
Courses
PostGraduate
MTech
MCA
MBA
MPharma
Graduate
Companies

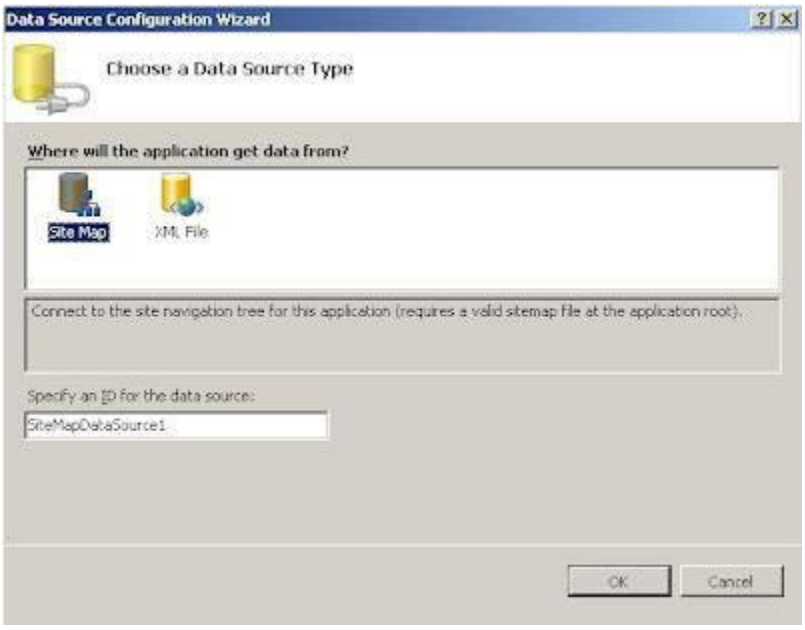

Untitled Page
TreeViewExample
Universities
DU
JNU
UPTU
IITKanpu
SMU
College
Courses
PostGrad
MTec
MCA
MBA
MPharma
Graduate
Companies
Message from webp...
DU
OK


NavigationControl - Microsoft Visual Studio
File Edit View Website Build Debug Format Layout Tools Test
usa.aspx stcharles.aspx mumbai.aspx la.aspx india.aspx edison.asp
Countries
Welcome to Sales Site
Sales Contries
Root
Parent 1
Leaf 1
Leaf 2
Parent 2
Leaf 1
Leaf 2
TreeView Tasks
Auto Format...
Choose Data Source: (None)
(None)
<New data source...>
Edit Nodes...
Customize Line Images...
Show Lines
http://dotnetguts.blogspot.com

Select "New data source" and select sitemap and press ok.

ASP.NET provides the following hierarchical data source controls.

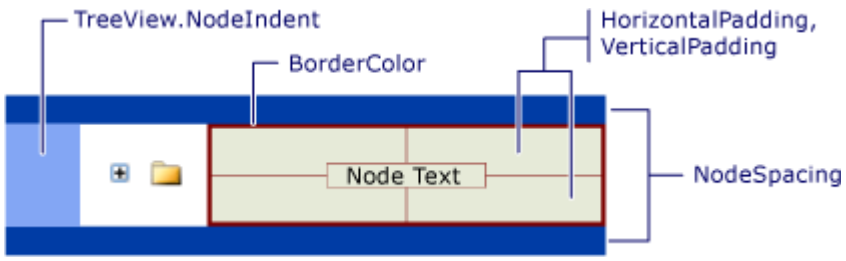
- XmlDataSource - This control allows you to bind to XML data, which can come from a variety of sources such as an external XML file, a DataSet object and so on. Once the XML data is bound to the XmlDataSource control, this control can then act as a source of data for other data-bound controls such as TreeView and Menu. For example, you can use the <asp:XmlDataSource> control to represent a hierarchical XML data source.
- SiteMapDataSource - This control basically retrieves the site map information from the web.sitemap file.



Displaying checkbox with treeview control, you can assign "ShowCheckBox" property of TreeView control to either

- All - Checkbox to all node.
- Root - Checkbox to root node.
- Parent - Checkbox to parent node.
- Leaf - Checkbox to leaf node.
- None - No Checkbox.

TreeNodeStyle Properties



Node Property	Description
NodeSpacing	Specifies the amount of vertical spacing between the entire current node and adjacent nodes above and below.

VerticalPadding	Specifies the amount of space rendered on the top and bottom of the TreeNode text.
HorizontalPadding	Specifies the amount space rendered on the left and right of the TreeNode text.
ChildNodesPadding	Specifies the amount of space above and below the child nodes of the TreeNode.
ImageUrl	Specifies the path to the image that displays next to the TreeNode.

Web Parts

A Web Part, also called a Web Widget, is an ASP.NET server control which is Web Part Pages by users at run time. It can be put into certain places in a web page by end users, after developing by programmer. End users can customize Web Parts pages by changing the page layout, adding and removing Web Parts, editing Web Parts properties, establishing connections between Web Parts, and more.

ASP.NET Web Parts is an integrated set of controls for creating sites that enable end users to modify the content, appearance, and behavior of web pages directly from a browser. The modifications can be applied to all users on the site or to individual users. When users modify pages and controls, the settings can be saved to retain a user's personal preferences across future browser sessions, a feature called personalization. These Web Parts capabilities mean that developers can empower end users to

personalize a web application dynamically, without developer or administrator intervention.

Advantages of Web Parts

- Web Parts allows for personalization of page content. They allow users to move or hide the Web Parts and add new Web Parts changing the page layout.
- Web Parts allows user to export or import Web Parts settings for use in other pages. Web Parts retain the properties, appearance and the data across the pages when imported or exported.
- Web Parts can be assigned role-based access. So you can determine which Web Parts can share by all or which should be hidden for certain roles. This helps us to provide customized content based on security.
- Web Parts can talk to each other. You can utilize the data in one Web Part in another Web Part for different purposes.

Web Parts Modes

The modular and customizable sites that you can build with the new Portal Framework enable you to put the web page that is in view into several modes for the end user. Modes are very powerful in that they enable user to edit Web Parts, delete the Web Parts or customize Web Parts.

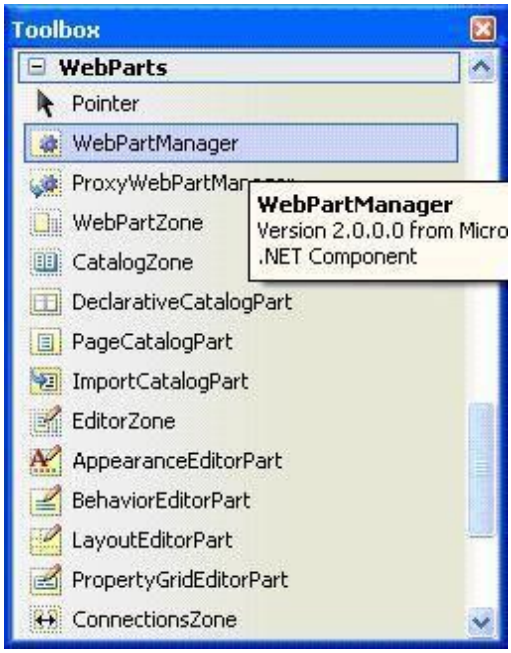
- a) Normal mode: End user cannot edit or move sections of page.
- b) Edit Mode: End user can edit Web Parts on the page including Web Parts title, color or even setting custom properties.

- c) Design Mode: End user can rearrange the order of the pages Web Parts in a WebPartZone.
- d) Catalog Mode: End user enjoys the choice to add new Web Parts or add deleted Web Parts in any WebPartZone on the page.

Building Web Parts

There are two basic ways to create a Web Part. You can treat any standard Microsoft ASP.NET control as a Web Part or you can build a custom control that derives from the base WebPart class.

You are not required to modify a control in any way to use it as a Web Part. Standard ASP.NET controls (such as the Calendar and GridView controls), Web User Controls, and even custom controls can all be used as Web Parts



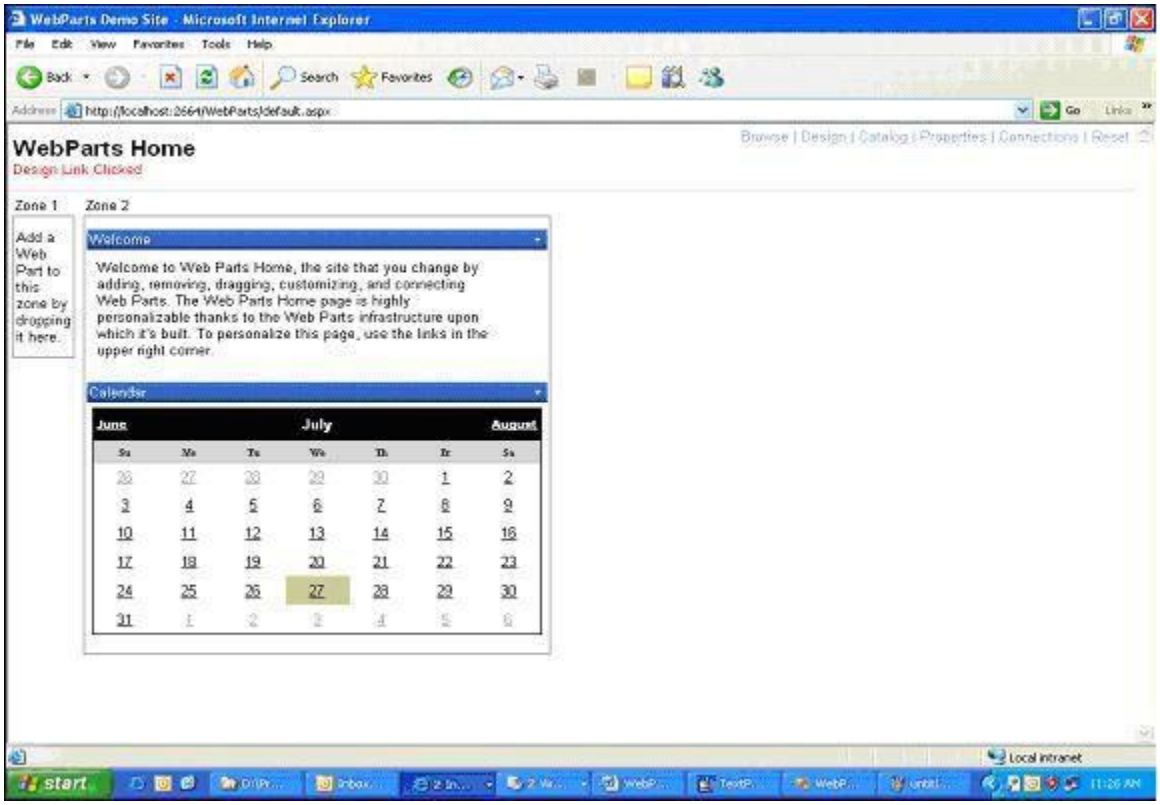
WebPartManager

WebPartManager is the most important of all the Web Part controls, responsible for managing and coordinating all controls inside WebPartZones. The Web Parts framework doesn't work without it, so every page that uses Web Parts must have an instance of WebPartManager declared, and it must be declared before other Web Parts controls. WebPartManager has no UI, so it's not visible on the page. It also exposes a very rich API for adding Web Parts to the page, closing Web Parts, connecting Web Parts, and more.

WebPartZone

WebPartZone is arguably the second most important Web Part control. It's used to define zones, which serve as containers for Web Parts. There is no practical limit to the number of WebPartZones a page can contain, and no limit to the number of Web Parts that a zone can contain. A WebPartZone control can host controls that do not derive from the WebPart class, by wrapping them with a GenericWebPart control at run time.

WebParts 'Welcome' and 'Calendar' are in WebPartZone 1 and WebPartZone 2 respectively in figure below:



User can drag and drop web parts from one WebPartZone to another WebPartZone in Design mode.

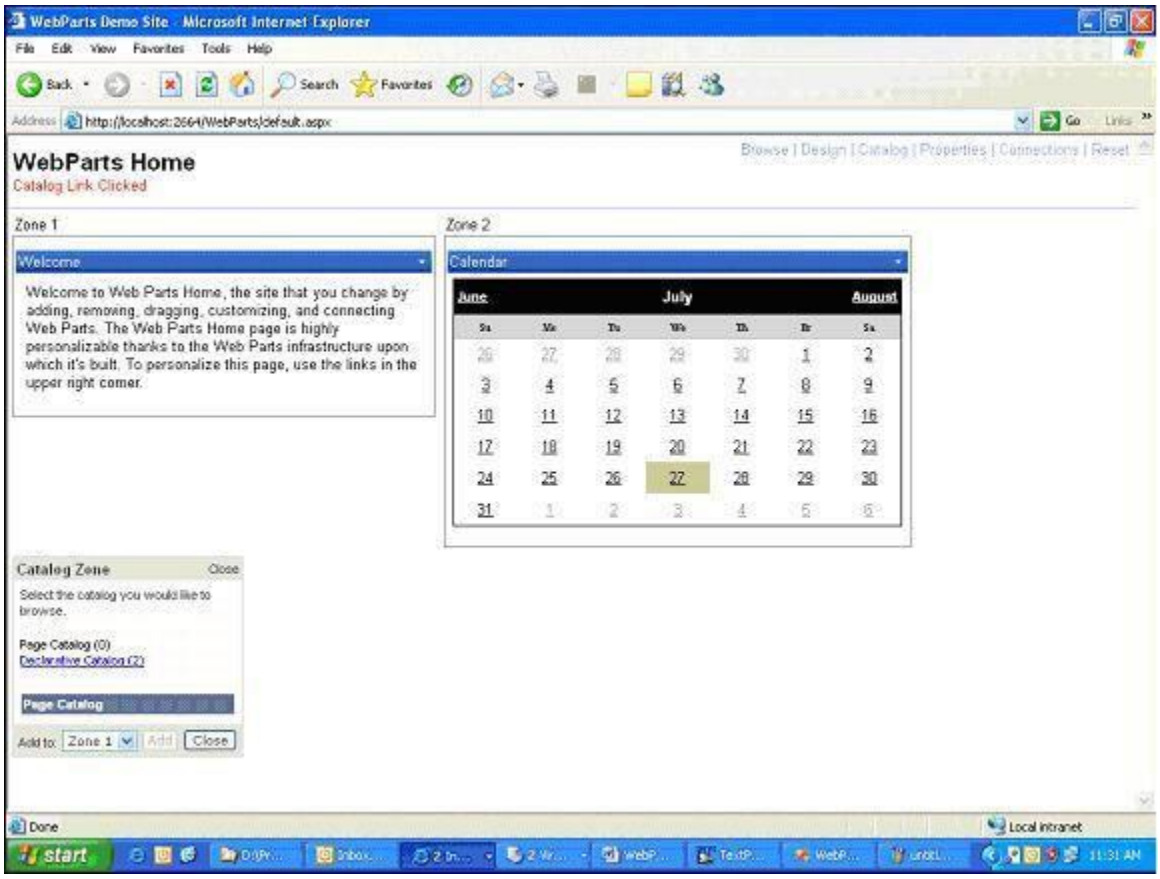
Catalog Zone

One of the chief benefits to building pages from Web Parts is that the content of these pages can be interactively configured by the user. It's a simple matter, for example, to enable users to restore closed Web Parts to the page by including a CatalogZone in the page.

The purpose of the CatalogZone control is to allow end users to customize Web Parts pages by adding Web Parts to them. Web Parts can come from three sources: Web Parts that were previously present in the page but were

closed, Web Parts that don't appear on the page by default but that can be added, and Web Parts imported from .WebPart files.

A CatalogZone control becomes visible only when a user switches a Web page to catalog display mode (CatalogDisplayMode) as shown below:



CatalogPart:

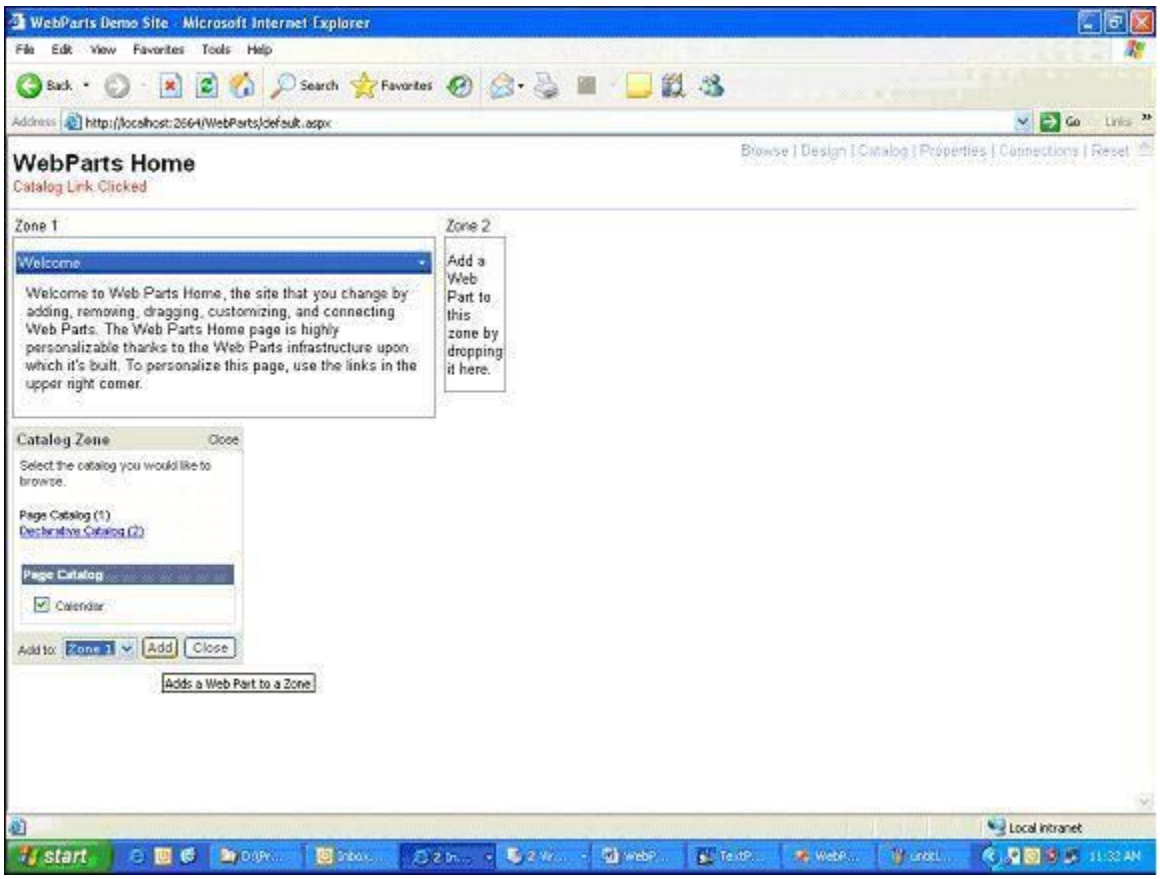
CatalogPart controls provide the UIs for adding Web Parts to the page. A CatalogZone can contain any combination of CatalogParts.

A CatalogZone can contain several types of CatalogPart controls. The following list summarizes the CatalogPart controls provided with the Web Parts control set:

- PageCatalogPart
- DeclarativeCatalogPart
- ImportCatalogPart

PageCatalogPart

The PageCatalogPart class serves one very specific purpose on a Web Parts page: it acts as a page catalog to contain any controls previously added to the page that a user has closed, and that the user can add back to the page. Add a PageCatalogPart control to your page if you want to provide users with the flexibility of closing and reopening controls. If your page does not allow users to close controls at all, there is no need to add a PageCatalogPart control to your page.



DeclarativeCatalogPart

The DeclarativeCatalogPart control provides a way for developers to add a set of server controls declaratively to a catalog on a Web page. A catalog, in the Web Parts control set, is simply a list of WebPart or other server controls that is visible when a page is in catalog display mode. A user can select controls from the list and add them to the Web page, which effectively gives users the ability to change the set of controls and the functionality on a page, as shown below:

ImportCatalogPart

The ImportCatalogPart control enables users to import a description file that describes settings on a WebPart control or server control that a user wants to add to a Web page. After the user has imported the description file, the WebPart control referenced in the file appears within the ImportCatalogPart control when the page is in catalog mode, and a user can then add the control to the page. User can view the ImportCatalogPart in Catalog Display mode, as shown below. User can browse the web part file and then upload by clicking Upload button.

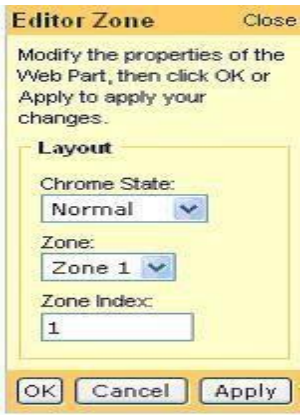
Editor Zone

The purpose of the EditorZone control is to allow end users to customize Web Parts pages by editing the properties of the page's Web Parts. Editing UIs are provided by EditorPart controls, which divide Web Part properties into four categories:

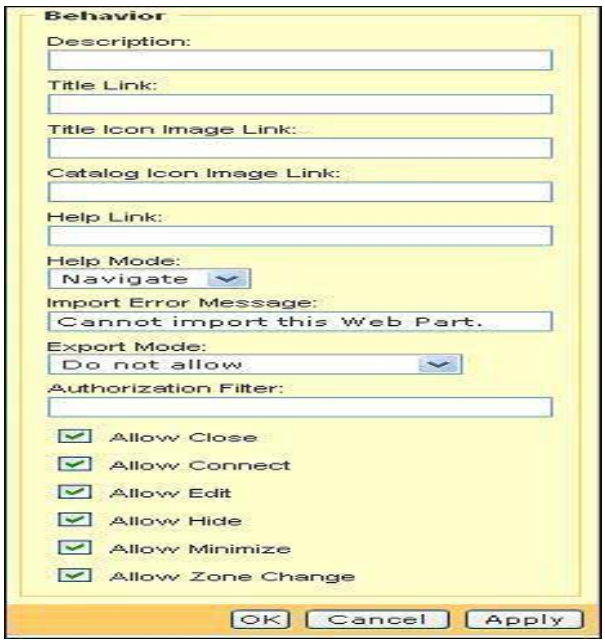
Properties that affect appearance, Properties that affect behavior, Properties that affect layout and Custom properties added by Web Part developers.

An EditorZone can contain any combination of EditorParts. EditorZones are only visible when the display mode is EditDisplayMode. User can click the Edit verb from webpart to open the Editor Zone.

Editor Zone with AppearanceEditorPart



Editor Zone with LayoutEditorPart



Editor Zone with BehaviorEditorPart

Web Parts can and often do implement custom properties to complement the built-in properties provided by the Web Parts framework. A Web Part that shows stock prices, for example, might implement a public property named "Stocks" to enable end users to specify which stock prices are shown. PropertyGridEditorParts provide UIs for editing custom properties. Attributing a property [WebBrowsable] enables that property to appear in a PropertyGridEditorPart. Of course, the PropertyGridEditorPart must be declared in an EditorZone if it's to appear on the page.

WebPart Connection

Connections enable Web Parts to share data. A classic example is a Web Part control that shows the current weather and allows the user to enter a zip code so the weather can be localized. Another Web Part on that page-- perhaps one that shows news headlines--might want that zip code so it, too, can localize content. Rather than require the user to enter the zip code twice, you can connect the two Web Parts so that one can get the zip code from the other. Connections can be defined statically by page developers, or they can be created dynamically by end users. The ConnectionsZone control provides a UI for creating connections dynamically.

Connection Provider

Writing a connection provider is no more difficult than writing a method that returns an interface reference and attributing that method [ConnectionProvider]. The first parameter to [ConnectionProvider] assigns a friendly name to the provider connection point and is displayed by the ConnectionsZone UI. The second parameter assigns a unique ID to the provider connection point. A provider can implement multiple provider connection points if desired. Each provider connection point must be assigned a unique ID.

ConnectionConsumer

Writing a connection consumer is a matter of writing a method that receives an interface reference and attributing that method [ConnectionConsumer].

The first parameter to [ConnectionConsumer] assigns a friendly name to the consumer connection point and is displayed by the ConnectionsZone UI. The second parameter assigns a unique ID to the consumer connection point.

QUESTION BANK

PG DEPARTMENT OF COMPUTER SCIENCE
Distributed Technology
P16CS22

Part – A (2 Mark Questions)

1. What is distributed system?
2. Mention the merits of distributed system?
3. What is remote object & remote access?
4. What is IIS?
5. What is DCOM?
6. Write Java technologies in distributed computing.

Part – B (5 Mark Questions)

1. Write short notes on distributing computing.
2. Mention the steps in hosting remote applications.
3. Discuss about various components of distributed computing
4. Write the challenges involved in establishing remote connection.

Part – C (10 Mark Questions)

1. Explain the current distributed computing practices through Dot NET.
2. Discuss about Strategies involved in remote computation

Unit - 2

Part – A (2 Mark Questions)

1. What do you meant by ADO.NET.
2. What are views?
3. What is a namespace?
4. What is CLR?
5. Define Crystal report.
6. What is Grid view?
7. What is Details View?
8. What is Form View?

Part – B (5 Mark Questions)

1. Justify "ASP.NET is a server-side technology".
2. Write a note on inline code page and code behind page.
3. What do you understand by view state?
4. Explain how ASP.NET maintains view state.
5. Discuss Form view controls.
6. Write note on role of ADO.
7. Explain .net framework.

Part – C (10 Mark Questions)

1. What is NET Framework Data Providers.
2. What is ASP.Net? Explain feature of ASP.Net over ASP.
3. Explain the Crystal report in distributed application.
4. Explain HTML Server Control
5. Explain AD Rotator asp.net control.
6. Explain common properties of list control of ASP.Net.

Unit - 3

Part – A (2 Mark Questions)

1. What is Asp.Net?
2. What are web parts?
3. What are the events of page and when it is executes?
4. What is Master Pages?
5. List out any two properties in site navigation.

Part – B (5 Mark Questions)

1. Write the feature in Website development.
2. Explain Master Pages.
3. How many modes involved in Image map control
4. What is the purpose of Web parts.
5. What is HTTP? Is it HTTP required for web application?
6. Explain ASP.NET Pages detail with Examples.
7. Discuss Site map Path & Site Navigation

.

Part – C (10 Mark Questions)

1. What is ASP.NET Programming? What is the advantages ofASP.NET?
2. Narrate the steps involved in Master Pages.
3. Summarize the controls involved in asp.net.
4. Write the steps to perform in wizard map control.
5. Write a Detailed note on ASP Web Controls.
6. Explain AdRotator ASP.NET control.
7. Explain Treeview Control.
8. Explain ImageMap control
9. Explain Menu control with example.

Unit- 4

. **Part – A (2 Mark Questions)**

1. Give any two mobile applications in ASP.net.
2. List any two securities in dotnet.
3. Explain in brief the two step compilation process.
4. What are attributes of login control?
5. What is authentication ?
6. Define response object.
7. Define QueryString.
8. What is Windows-based authentication ?

Part – B (5 Mark Questions)

1. How many types of authentication in security?
2. Write short note on state management in ASP?
3. What is the critical usage of web site development?

Part – C (10 Mark Questions)

1. How to configure an application using mobile dot-net?
2. Discuss steps followed in security in asp.net
3. Compare server-side and client-side management.
4. Explain mobile application development in ASP.Net.

Unit- 5

Part – A (2 Mark Questions)

1 What are the advantages of XML 3
2 What is the role of XML in web application
3 What is SOAP ?
4 What are Web Services?
5 What are web services platform elements?
6 What Are Environment Variables?
7 What is WSDL ?
8 What is UDDI?
9 What is REST?

Part – B (5 Mark Questions)

1. List out all Web services Standards.
2. Explain WSDL with all its elements.
3. Explain SOAP in details
4. Web services Data Type.
5. Why do we need web services?

Part – C (10 Mark Questions)

1 Discuss the role of web services in Distributed computing.
2 Explain the WSDL,UDDI and SOAP concepts involved in web services.
3 Write note on connecting a web service to a database.
4 How to accessing a web service through an ASP.Net Applications.
5 Briefly explain the message structure of SOAP.