CHAPTER 1

Basics

1.1 WEB PAGE

A web page is a document which supports hypertext. Hypertext is the text which can be clicked to open another web page to jump to another position within same web page.

1.2 WEB SITE

It is a collection of web pages which are interlinked with each other with the help of hyperlinks. By using these hyperlinks we can switch from one web page to another in a web site.

1.3 WORLD WIDE WEB

The **World Wide Web** is a network of information resources. The world wide web relies on three mechanisms to make websites and web resources available to the internet users:

- A uniform naming scheme for locating resources on the world wide Web called URL.
- Protocols for getting access to named web resources (e.g., HTTP, HTTPS, SMTP etc).
- Hypertext for easy navigation among web resources.

1.4 URL(UNIFORM RESOURCE LOCATOR)

URL is the complete path and name of a web resource. Every resource available on the Internet like HTML document, image, video clip, program, etc. has an address that may be encoded by a **URL**

URL consist of three elements:

1. The naming scheme to access the resource.
2. The name of the machine hosting the resource.
3. The name of the resource given as a path.

For example http://www.gndu.ac.in/Document

This URL specifies that there is a document available via the HTTP protocol residing on the machine named www.gndu.ac.in accessible via the path "/Document".

In HTML, URLs are used to:

- Link to another document or resource.
- Link to an external style sheet or script.
- Include an image, object, or applet in a page.
- Create an image map.
- Submit a form.
- Create a frame document.
- Cite an external reference.
- Refer to metadata conventions describing a document.

1.5 HTML

HTML stands for Hypertext Markup Language. It is used to create web pages and web sites. This is the basic language for creating any web page. By using HTML we can :

- Publish online documents with headings, text, tables, lists, photos, etc.
- Retrieve online information via hypertext links at the click of a button.
- Design forms for conducting transactions with remote services for use in searching information, making reservations, ordering products etc.
- Include spread-sheets, video clips, sound clips, and other applications directly in their documents.

1.5.1 Elements of HTML

There are basically four elements of HTML they are:

1.5.1.1 <html> Element

<html> is the element that begins and ends each and every web page. Its main purpose is to hold each web element nicely in position and serves the role of book cover. All other HTML elements are encapsulated within the <html> element. Each web page should start with <html> and end with </html>

1.5.1.2 <head> Element

The <head> is usually the first element contained inside the <html> element. While it is also an element container that encapsulates other HTML elements, these elements are not directly displayed by a web browser. Instead they function behind the scenes providing more descriptive information about the HTML document like its page title and other meta data. Other elements used for scripting like JavaScript and formatting (CSS) are also contained within the <head> element. It starts as:

<html>

<head>

</head>

</html>

1.5.1.3 <title> Element

The <title> element adds a title to a web page. Web page titles are displayed at the top of any browser window or at the top of browser tabs. It is written as:

```
<html>
<head>
<title>My Web Page!</title>
</head>
</html>
```

1.5.1.4 <body> Element

The element that encapsulates all the visual elements like paragraphs, pictures, tables etc. in a web page is the **<body>** element. **<body>** element is one of the four main web page elements.It contains all of the page's viewable content. It is written as:

```
<html>
<head>
<title>My Web Page!</title>
</head>
<body>
Web Page contents
</body>
</html>
```

1.5.2 Tags of HTML

Tag is a way to instruct the web browser to display some content in a specific way. Every tag has three properties, the opening tag, the content and the closing tag. The general format of using a tag is **<tag>Content</tag>.**

1.5.2.1 Commonly used tags of HTML

- tag: It is known as paragraph tag. It starts with and ends with . This tag is reserved specifically for blocks of text. After displaying the content, the next line will automatically start from next line.
- <h1> to <h6> : They are known as heading tags. We can start them like <h1> and end like <h1>. We can similarly use other heading tags. These tags display the text in bold and automatically insert newline character after displaying the content.
- : This tag is used to make text bold. It start as and ends as .
- : This tag is used to make text italic. It start as and ends as .
- <u>: This tag is used to make text underlined. It start as <u> and ends as </u>.
- <s>: This tag is used to make text striked. It start as <s> and ends as </s>.
- <sup>: This tag is used to make text superscripted. It is generally used to create power of any number. It start as ^{and ends as}.
- <sub>: This tag is used to make text subscripted. It start as _{and ends as}.
- ... : <ul> stands for unordered list. It is used to create a list of values with bullets.
- ... : <ol> stands for ordered list. It is used to create a numbered list of values.
- <a>: a stands for anchor. It is used to create hyperlinks.
- <iframe>: This tag is used to embed a web page into another web page.
- <img>: This tag is used to insert an image in the web page.
- <table>: This tag is used to insert a table in a web page.
- <form>: This tag is used to create a form which can further contain controls like Textboxes, Checkboxes, radio buttons, images, buttons etc.
-
: This is known as break. It is used to take the contents to next line.
- <hr>: It is known as horizontal row. It is used to display a horizontal line.
- <font>: This tag is used to give colour, size, font style to text.

1.5.3 HTML FORMS

We can create forms in HTML. Form are basically meant for entering data online. HTML Forms are used to select different kinds of user input. An HTML form can contain input elements like text fields, checkboxes, radio-buttons, submit buttons and many other controls. A form can also contain select lists, textarea, fieldset, legend, and label elements. The <form> tag is used to create an HTML form as:

```
<form>
:
<input elements>
:
</form>
```

The most important form element is the <input> element. The <input> element is used to select user information. An element can vary in many ways depending on the type attribute. An element can be of type text field, checkbox, password, radio button, submit button etc. The most commonly used inputs are:

1.5.3.1 Text Fields

`<input type="text">` defines a one-line input field in which a user can enter text.

1.5.3.2 Password Field

`<input type="password">` defines a password field. Whatever we type in it will appear as bullets.

1.5.3.3 Radio Buttons

`<input type="radio">` defines a radio button. Radio buttons let a user select only one radio button among multiple radio button

1.5.3.4 Checkboxes

`<input type="checkbox">` defines a checkbox. Checkboxes let a user select zero or more options among a limited number of choices.

1.5.3.5 Submit Button

`<input type="submit">` defines a submit button.

A submit button is used to send form data to a server. The data is sent to the page specified in the form's action attribute. The file defined in the action attribute usually performs some operation with the received input.

1.5.3.6 Reset Button

`<input type="reset">` defines a reset button. A reset button is used to clear the contents of various fields of the form.

1.5.3.7 Simple Button

`<input type="button">` defines a simple button. Simple button usually work with the help of scripting.

1.5.3.8 TextArea

The **<textarea>** tag defines a multi-line text input control. A text area can hold an unlimited number of characters. The size of a text area can be specified by the cols and rows attributes or through CSS' height and width properties. It starts with <texarea> and ends with </textarea> tag.

Example:

<textarea cols="30" rows="3"></textarea>

It will display a textarea with 3 rows and 30 characters in one line at a time.

1.5.3.9 Drop Down List

The <select> element is used to create a drop-down list. The <option> tags inside the <select> element define the available options in the list.

1.5.3.10 Label

The <label> tag defines a label for an <input> element. The <label> element does not render as anything special for the user. However, it provides a usability improvement for mouse users, because if the user clicks on the text within the <label> element, it toggles the control. The for attribute of the <label> tag should be equal to the id attribute of the related element to bind them together.

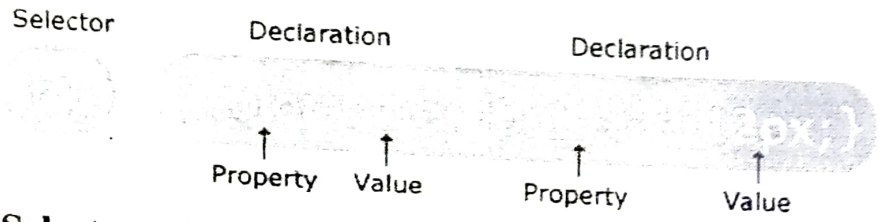
We can design a form as follows using the above tags as:

1.6 STYLE SHEETS

Style sheets simplify HTML markup and relieve HTML of the responsibilities of presentation. They give users control over the presentation of documents like font information, alignment, colors etc.

Style information can be specified for individual elements or groups of elements. Style information may be specified in an HTML document or in external style sheets. The mechanisms for associating a style sheet with a document is independent of the style sheet language. Most commonly used stylesheet language is CSS(Cascading StyleSheet).

A CSS rule set consists of a selector and a declaration block:



Selector refers to the HTML tag which we want to style.

Declaration block contains one or more declarations separated by semicolons. Each declaration includes a property name and a value separated by a colon. A CSS declaration always ends with a semicolon and declaration groups are surrounded by curly braces:

Example1

```
p {color:red; }
```

In this example, the HTML tag has been defined with property **color:red**. It means that whatever we put inside tag combination, will appear in red colour.

Example2

```
h1 {color:red;text-align:center;}
```

In this example, the HTML tag <h1> has been defined with property **color:red** and **text-align:center.** It means that whatever we put inside<h1> </h1> tag combination will appear in red colour and will be center aligned in the web page.

1.6.1 Using CSS in a web page

CSS can be used in a web page in three ways as follows:

1.6.1.1 Internal Style Sheet

An internal style sheet should be used when a single document has a unique style. We define internal styles in the head section of an HTML page inside the <style> tag like this:

```
<head>
<style>
hr {color: sienna;}
p {margin-left: 20px;}
body {background-image: url("images/background.gif");}
</style>
</head>
```

1.6.1.2 Inline Styles

An inline style can be used anywhere in any tag. To use inline styles, we need to add the style attribute to the tag. The style attribute can contain any CSS property. The example shows how to change the color and the left margin of a paragraph:

```
<p style="color:sienna;margin-left:20px;">This is a
paragraph.</p>
```

1.6.1.3 External Style Sheet

An external style sheet is basically useful when the style is applied to many pages. With an external style sheet, we can change the look of an entire web site by changing just one file. Each page must include a link to the style sheet with the <link> tag. The <link> tag goes inside the head section as:

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

An external style sheet can be written in any text editor. The file should not contain any html tags. The style sheet file must be saved with a .css extension. An example of a style sheet file is shown below:

myStyle.css

```
hr {color: sienna;}
p {margin-left: 20px;}
body {background-image: url("images/background.gif");}
```

1.6.1.4 CSS Class

The class selector allows us to style different tags with same properties. We need to use dot operator with the user defined name while defining a class.

Example

```
.greenboldtext{
  font-size: small;
  color: #008080;
  font-weight: bold;
}
```

We can use this class in the body as

```
<p class="greenboldtext">Hello</p>
```

Following program demonstrates the use of CSS

```
<html>
<head>
<style type="text/css">
      .BoldStyle {
      font-family: Verdana, Arial, Helvetica, sans-serif;
      font-weight: bold; color: #FF0000;
}
```

```
        .ItalicStyle {
        font-family: Verdana, Arial, Helvetica, sans-serif;
        font-style: italic;
        color: #FF0000;
        }
</style>
</head>
<body>
<div class="BoldStyle">
 <p>With CSS you define the colors and sizes in "styles". Then
as you write your documents you
refer to the styles.</p>
</div>
<div class="ItalicStyle">
<p>
Therefore: if we change a certain style it will change the look of your entire site.
</p></div>
```

1.6.1.5 Most commonly used CSS Properties

Text and Fonts	Colours and Backgrounds	The Box Model	Positioning and Display
• font • font-family	• color • background-color	• padding, padding-top, padding-right, padding-bottom, padding-left	• position • top
• font-size • font-weight • font-style • font-variant • line-height • letter-spacing • word-spacing • text-align • text-decoration • text-indent • text-transform • vertical-align • white-space	• background • background-image • background-repeat • background-position • background-attachment	• border, border-top, border-right, border-bottom, border-left • border-style, border-top-style, border-right-style, border-bottom-style, border-left-style • border-color, border-top-color, border-right-color, border-bottom-color, border-left-color • border-width, border-top-width, border-right-width, border-bottom-width, border-left-width • outline • outline-style • outline-color • outline-width • margin, margin-top, margin-right, margin-bottom, margin-left • width • height • min-width • max-width • min-height • max-height	• right • bottom • left • clip • overflow • z-index • float • clear • display • visibility

Lists	Tables	Others	
• list-style • list-style-type • list-style-image • list-style-position	• table-layout • border-collapse • border-spacing • empty-cells • caption-side	• backgroundbackground-attachment • background-color • background-image • background-position • background-repeat • border • border-collapse • border-color • border-spacing • border-style • border-width • bottom • caption-side • clear • clip • color • content • counter-increment • counter-reset • cursor • direction • display	• outline • outline-color • outline-style • outline-width • overflow • padding • page-break-after • page-break-before • page-break-inside • position • quotes • right • table-layout • text-align • text-decoration • text-indent • text-transform

1.7 SCRIPTING

Scripting is the way through which we may create dynamic web pages For example, smart forms that react as users fill them. The mechanisms provided to include scripts in an HTML document are independent of the scripting language. Scripting is broadly categorized into two types:

- Client Side Scripting : In client side scripting, all operations are performed inside client browser. Most commonly used client side scripting language is Javascript.
- Server Side Scripting: In server side scripting, all operations are performed at the server end and result is sent back to the web browser in the form of HTML. Most commonly used server side scripting languages used today are PHP, JSP, ASP, C#, VB etc.

1.7.1 Steps to insert Javscript in a web page

In HTML, JavaScripts can be inserted between <script> and </script> tags. JavaScripts can be put inside <body> as well as <head> section of an HTML page. To insert JavaScript into an HTML page, we need to use the <script> tag. The <script> and </script> tells where the JavaScript starts and ends. The lines between <script> and </script> contain the JavaScript code.

Example

```
<html>
<head>
<script>
function myFunction()
{
    document.write( "Hello");
}
</script>
</head>
<body>
<script language="javascript">myFunction();</script>
</body>
</html>
```

When the above program runs, browser will display Hello as the fuction named myFunction has been called inside body section of the web page.

1.8 THE MICROSOFT .NET FRAMEWORK

The .NET Framework is the infrastructure for the Microsoft .NET platform. The .NET framework is a software development framework. It provides a controlled programming environment in which softwares can be developed, installed and executed on Windows-based operating systems.

It enables developers to rapidly build wide variety of applications that run on windows operating system like web applications, windows applications, console applications and web services etc.

The main features of .NET Framework are:

- **Interoperability:** This allows for .NET-developed programs to access functionalities in programs developed outside .NET.
- **Common Runtime Engine:** It is also known as common language runtime. It allows programs developed in .NET to have common behaviors in memory usage, exception handling and security.
- **Language Independence:** Common language infrastructure specifications (CLI) allow for the exchange of data types between two programs developed in different languages.
- **Base Class Library:** It has a rich library of code for most common functions used by programmers to avoid repetitive rewriting of code.

- **Ease of Deployment:** There are tools to ensure the ease of installing programs without interfering with previously installed applications.
- **Security:** Programs developed in .NET are based on a common security model.

1.8.1 .NET Framework Components

NET Framework consists of various components which are as follow:

1.8.2 Common Language Runtime (CLR)

Common Language Runtime (CLR) is the environment that manages the execution of programs written in any of several supported languages allowing them to share common object oriented classes written in any of the languages. Microsoft refers to its Common Language Runtime as a "managed execution environment." A program compiled for the CLR does not need a language-specific execution environment and can easily be moved to and run on any Windows based system.

CLR transforms source code into a form of bytecode known as Common Intermediate Language (CIL). At run time, CLR handles the execution of the CIL code.

Developers write code in a supported .NET language such as C# or VB.Net. The .NET compiler then converts this code into CIL code. During run time, the CLR converts the CIL code into something that can be understood by the operating system. Alternately, the CIL code can be transformed into native code by using the native image generator (NGEN).

The language compilers store metadata that describes the members, types and references in the compiled code. The CLR uses the metadata to lay out instances in memory, locate and load classes, enforce security, set runtime context boundaries, and generate native code.

CLR allows for the easy use of different supported languages to achieve a common goal. This makes it flexible for developers to choose their own programming language, provided it is supported by the .NET framework. With CLR, .NET can manage the execution of all supported languages by transforming them to bytecode and then into the native code for the chosen platform.

Using NGEN makes the code run faster because CLR will not have to transform the bytecode into native code each time. Bytecode is object-oriented programming (OOP) code compiled to run on a virtual machine (VM) instead of a central processing unit (CPU).

CLR abstracts code execution and provides different services like memory management, security, exception handling etc.

The most important feature of CLR is that it automatically handles allocation and deallocation of memory resources. Because CLR provides automatic memory management, developer needs not to worry about memory management there by increases developer productivity and the code quality.

Automatic Garbage collector (GC) is responsible for collecting the unused objects. CLR invokes GC automatically and an application can also invoke GC explicitly by making use of GC.Collect method.

Code which makes use of CLR is called as managed code. Managed code takes advantage of the memory management services provided by the CLR. Code which does not make use of CLR such code written to call COM components is called as unmanaged code. .

1.8.3 Base Class Library

Base Class library is a set of pre-defined code for handling common programming tasks like Request processing, output, file operations, database operations, XML operations, graphical operations etc. It has a rich library of code for most common functions used by programmers to avoid repetitive rewriting of code.

1.8.4 Common Type System (CTS)

Common Type System is a specification that defines the data types supported by CLR so that program written in one .NET Language can make use of the program written using another .NET Language. .NET Framework supports multiple languages and this feature is called language independence.

.NET programs can be written in C#, VB.NET, J# many other languages. This gives the developer to pick and choose a language in which he/she is comfortable.

It is one of the most attractive features of the .NET. It provides a very rich and standard set of data types.

The CTS is object oriented but also supports procedural and functional languages as well.

The CTS allows .NET to provide a unified programming model and to support multiple languages. The CTS supports two general categories of types each of which actually have a number of subcategories.

1.8.5 Intermediate Language

Code that is written in C#, VB.NET etc. are converted into Intermediate code (IL) by the respective .NET language compliers. CLR then converts IL into machine language during the code execution.

1.8.6 Assembly

Assembly is a physical unit of deployment. Assembly consists of one or more files or namespaces. It is used for versioning and code security. Assemblies are DLL or EXE files which get generated when code written in .NET Language like C#, VB.NET is compiled and hence contains IL code.

It is self-describing. It is typically one physical DLL or EXE in the Windows PE file format but can be made up of multiple files.

An assembly must have a manifest that describes its contents and it usually contains MSIL, resources, and metadata describing the types contained within the assembly.

An assembly is made up of four elements:

- Manifest
- Metadata describing the types
- Modules
- Resources

1.8.7 Metadata

Metadata means data that describes data. In the .NET, metadata is the information that is stored in the Assembly to make it self-describing. Metadata describes the elements of the Common Type System that we use in our application as well as the information that the runtime needs for type safety and security.

1.8.8 Manifest

The Manifest is what describes the assembly itself. The Manifest contains:

- A simple name
- A four-part version number of the form:

  Major.Minor.Build.Revision
- Publisher's private key
- Culture (Locale)
- List of files that make up the assembly
- List of dependent assemblies
- Permission requests
- Exported types
- Resources

1.8.9 Modules

These are various methods which can be used in various .NET based applications as per their requirements.

1.8.10 Resources

These may be the namespaces or files which may be used in a .NET application.

1.8.11 Namespace

Namespace is defined as the collections of classes. There are large number of pre defined classes in .NET framework. These classes are very much used in programming. When we make use of methods that are in these built in classes, we need to provide a reference to these classes.

To make application maintenance easier due to increase in code, .NET provides this feature to arrange the code according to functionality. One Assembly can contain multiple namespaces. Similarly one namespace can be included in multiple assemblies.

1.8.12 Framework Class Library (FCL)

The .NET FCL is the key component of .NET framework. It provides core functionalities of .NET architecture, which include:

- Base data types
- Object type
- Implementation of data structures
- Garbage collection
- Security, data access and database connectivity
- Network communications
- Support for implementing rich client GUI for both Windows and Web-based applications

The Framework class library (FCL) is a comprehensive collection of reusable types including classes, interfaces and data types included in the .NET Framework to provide access to system functionality.

The .NET FCL forms the base on which applications, controls and components are built in .NET. It can be used for developing applications such as console applications, Windows GUI applications, ASP.NET applications, Windows and Web services, workflow-enabled applications, service oriented applications using Windows Communication, XML Web services, etc.

The reusable types of FCL provide a simple interface to developers due to:

- Their self-documenting nature
- Easier to understand the framework which optimizes the development process.
- Integration of third-party components with classes in FCL.

FCL acts as a standard library, which can be used in a consistent manner by all the .NET languages and common language compliant (CLC-compliant) compilers.

FCL is designed to provide services similar to the Windows application programming interface (API), which was used before .NET was created. FCL has its code base as managed, object-oriented and easy to use, while Windows API is unmanaged, modular and cumbersome to use.

The .NET FCL is integrated with the Common Language Runtime (CLR) of the Framework, which manages the code execution. Its classes follow the object model as used by the Intermediate Language (IL) and are based on single inheritance. The classes and interfaces are grouped into namespaces so that they can be accessed easily.

1.8.13 .NET Framework Configuration Tool

The .NET framework configuration tool (Mscorcfg.msc) is an administrative tool used to manage and configure .NET assemblies that are placed in the global assembly cache. It also enables us to modify code access security policy and adjust remote services.

This Microsoft Management Console snap-in provides a graphical interface for users and administrators, allowing them to configure several aspects of common language runtime, including the security policy at enterprise, machine and user levels.

This tool can perform the following functions:

- It can display the properties and dependencies of an application.
- It can be used to configure an assembly for an application.
- It can perform complete security-related tasks such as displaying the security configuration of the .NET framework, adjusting the level of trust of an assembly, adjusting zone security and resetting all policy levels.
- It can be used to evaluate an assembly.
- It can be used to create a deployment package.
- Application users can use the Microsoft .NET Framework Wizard for executing limited configuration tasks.
- The Code Access Security Policy Tool (Caspol.exe) is as alternate, command-line tool for configuring a security policy, with the additional facility to write a batch script.
- The .NET framework configuration tool can also be used to administer code access security to manage Fully trusted assemblies, Named permission sets, Code groups and Security

1.8.14 ActiveX Data Object.NET (ADO.NET)

ActiveX Data Object.NET (ADO.NET) is a software library in the .NET framework consisting of software components providing data access services. ADO.NET is designed to enable developers to write managed code for obtaining disconnected access to data sources which can be relational or non-relational This feature of ADO.NET helps to create data-sharing as well as distributed applications.

ADO.NET provides connected access to a database connection using the .NET managed providers and disconnected access using datasets which are applications using the database connection only during retrieval of data or for data update. Dataset is the component helping to store the persistent data in memory to provide disconnected access for using the database resource efficiently and with better scalability.

ADO.NET evolved from ADO, which is also a technology similar to ADO.NET with a few basic structural changes. Although there is a provision to work in disconnected mode using ADO, data is transmitted to the database in ADO.NET more efficiently using data adapters. The in-memory representation of data differs between ADO and ADO.NET. ADO.NET can hold the data in a single result table, but ADO holds multiple tables along with their relationship details. Unlike ADO, data transmission between applications using ADO.NET does not use COM (component object model) marshalling but uses dataset, which transmits data as an XML stream.

The architecture of ADO.NET is based on two primary elements: DataSet and .NET framework data provider. Dataset provides the following components:

- A complete set of data including related tables, constraints and their relationships
- Functionality-like access to remote data from XML Web service
- Manipulation of data dynamically
- Data processing in a connectionless manner
- Provision for hierarchical XML view of relational data
- Usage of tools like XSLT and XPath Query to operate on the data
- .NET framework data provider includes the following components for data manipulation:
- Connection object to provides connectivity to the data source
- Command object to execute the database statements needed to retrieve data, modify data or execute stored procedures.
- DataReader object to retrieve data in forward only and read-only form.

- DataAdapter object to act as a bridge between dataset and data source to load the dataset and reconcile changes made in dataset back to the source.
- ADO.NET entity framework abstracts the level of data programming so as to eliminate the mismatch between data models and languages which application developers may have to deal .

1.8.15 The Common Language Specification (CLS)

The CLS is a specification that defines the rules to support language integration in such a way that programs written in any languagecan interoperate with one another taking full advantage of inheritance, polymorphism, exceptions and other features.

1.8.16 .NET Compact Framework (.NET CF)

.NET compact framework (.NET CF) is a subset of .NET framework that provides a hardware independent environment for executing .NET applications on resource-constrained devices like personal digital assistants (PDA), mobile phones and set-top boxes. .NET CF supports embedded and mobile devices that are built with Microsoft Windows CE operating system. .NET CF includes the following:

- Optimized common language runtime (CLR) with architecture inherited from .NET framework
- A subset of .NET Framework Class Library
- .A set of classes exclusively designed for optimal performance.

.NET CF makes up the platform for accessing the features of a smart device and makes it possible for applications and components to interact on the device and over the Internet. It provides the interoperability to access native functions of the Windows CE operating system and to integrate native components in managed code allowing both native and desktop device application developers to build applications for Windows Mobile and Windows Embedded CE devices.

.NET CF's programming model is similar to .NET and thus offers the benefits of using managed code with the .NET framework such as type safety, garbage collection and exception handling as well as providing XML Web services to hand held devices. Some of the features of .NET CF that differ from .NET framework and need to be considered while developing applications are the minimized CLR, optimized memory, specialized controls, and the lack of features like remoting and reflection.

.NET CF presents a rich development and execution environment for smart devices running Pocket PC (versions, 2002, 2003 and Phone Edition) or Windows CE.NET 4.1 or above. It provides a class library that is suitable for developers of both these platforms, irrespective of differences in their behavior and usage.