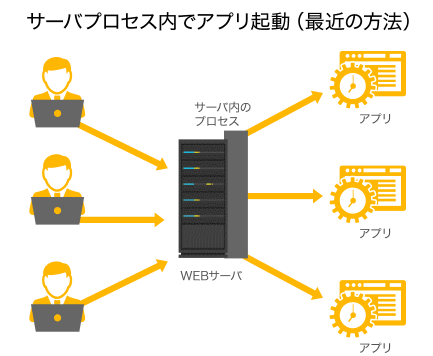
2024

㈱スカイオーシャン

[会社名]

2024年2月12日



VS　事はじめ基礎知識

目次

[**VSのドキュメントウインドウとは？** 3](#_Toc157187613)

[**基本の基本、３つのキーワード** 5](#_Toc157187614)

[**オブジェクトって何だろう？** 6](#_Toc157187615)

[**オブジェクトとプロパティの関係** 6](#_Toc157187616)

[**データ型とは** 7](#_Toc157187617)

[データ型の変換 8](#_Toc157187618)

[変数とは？を深く理解する 10](#_Toc157187619)

[変数を宣言してデータを格納（Ｄｉｍステートメント） 10](#_Toc157187620)

[新規にWebサイトを作成 11](#_Toc157187621)

[**最新のVisualStudio２０１9でのプロジェクト作成手順** 11](#_Toc157187622)

[**テストページで基礎を極める** 12](#_Toc157187623)

[**サイト内に新規にWebページを作成する** 12](#_Toc157187624)

[**変数のスコープ** 22](#_Toc157187625)

[**変数のページ内のスコープの実験** 23](#_Toc157187626)

[**ToString（）とは** 25](#_Toc157187627)

[**変数の使用法について** 25](#_Toc157187628)

[**1.** **配列：複数の値を表す変数** 26](#_Toc157187629)

[**TestHairetsu.aspxページを作成** 27](#_Toc157187630)

[**実行画面での、文字列の表示方法** 33](#_Toc157187631)

[プロシージャ（関数）とは？ 34](#_Toc157187632)

[クラス（Class）とは？ 40](#_Toc157187633)

[クラスを作って、使用してみる 40](#_Toc157187634)

[**.NET 名前空間** 40](#_Toc157187635)

[**htmlコードとプログラム言語の関係** 42](#_Toc157187636)

[**プログラマーなら“CSS”を理解しよう** 43](#_Toc157187637)

[**マスターページの構造** 46](#_Toc157187638)

[**メニュー表示部分** 46](#_Toc157187639)

[**ページの表示部分とフッター部分** 46](#_Toc157187640)

[**HTMLとCSSの関係** 48](#_Toc157187641)

[**スタイルの定義位置の検索の仕方** 49](#_Toc157187642)

[**HTMLの<a>~</a>要素を理解する** 51](#_Toc157187643)

[これだけは、ながめておこう！！ 52](#_Toc157187644)

[If文とは 52](#_Toc157187645)

[If文のサンプル 53](#_Toc157187646)

[**String型の値の比較** 53](#_Toc157187647)

[**ネスト構造 if文の中にif文** 54](#_Toc157187648)

[**値の一部で比較する(like演算子)** 55](#_Toc157187649)

[**null(nothing)を比較** 56](#_Toc157187650)

[**三項演算子** 57](#_Toc157187651)

[**比較演算子** 58](#_Toc157187652)

[**論理積(And) AかつB** 59](#_Toc157187653)

[**論理和(Or) AまたはB** 61](#_Toc157187654)

[**排他的論理和(Xor)** 62](#_Toc157187655)

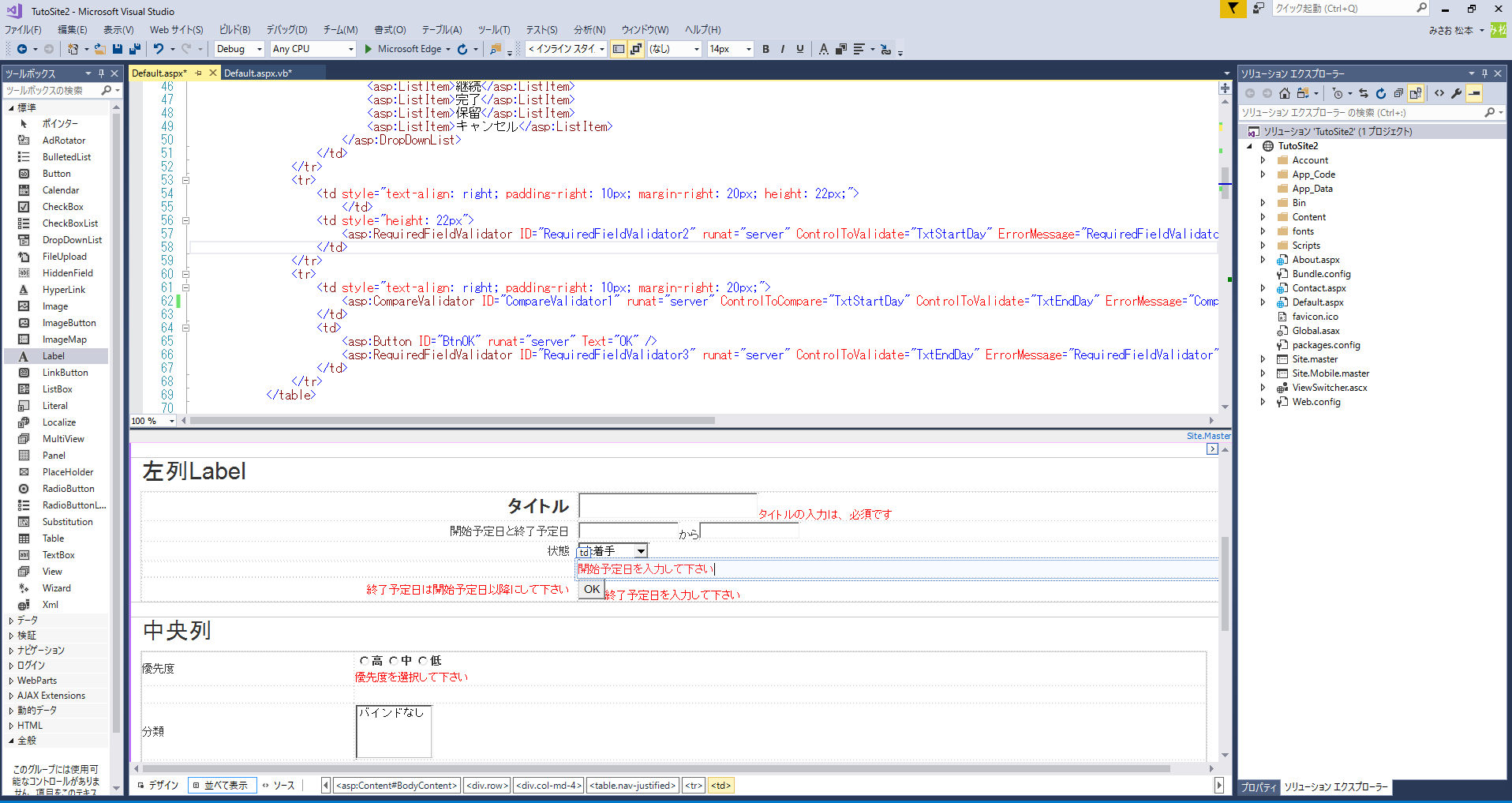
[**論理否定(Not) Aではない** 64](#_Toc157187656)

**VSのドキュメントウインドウとは？**

編集中の、Webページや、ソースコード（htmlコード）を表示します。下図のような

各ウインドウの配置で、本チュートリアルを進めて行きます。（今の時点では、コードビューとデザインビューの位置を確認する程度にして下さい。）

解説を実践中に、その都度参照して下さい。



表示

**コードビュー**

**デザインビュー**

**３**

**２**

**１**

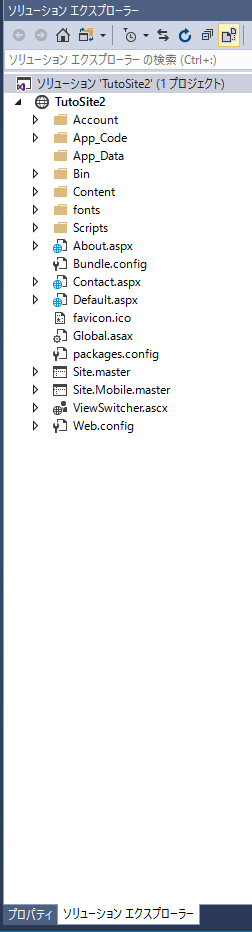
**ソリューションエクスプローラと**

**プロパティウインドウ**

**ツールボックス**r

1. “ソース、デザイン、並べて表示”等のタブが容易されています。
2. 左側に“ツールボックス”
3. 右側に“ソリューションエクスプローラ”と“プロパティウインドウ”
4. 中央の下が、“デザインビュー”、上が“コードビュー”です。

上図のドキュメントウインドウ内の“ツールボックス”、“ソリューションエクスプローラ”、“プロパティウインドウ”等は、ドキュメントウインドウの最上部のメニュー“表示”をクリックすると、各ウインドウの項目が表示されて、表示したいウインドウが選択できます。各ウインドウは、フローティングウインドウになっていて、マウスで自由に動かせますので、前図のように、配置して下さい。ソリューションエクスプローラとプロパティウインドウを配置した状態を次図に表示しておきます。



上の図の下部の赤い四角の部分は、次図のようにタブで選択できるようになっています。



両者共、フローティングウインドウですので、数回練習が必要です。私の場合、何回もメニューの“表示”からこのフローティングの表示を繰り返して、フローティングウインドウを使用する練習をした経験を思い出します。

**基本の基本、３つのキーワード**

この章では、いよいよ、コンピュータ―プログラマーとして最初に身に着けなればならない知識を少しづつ掲載していきます。

まず、次の３つのキーワードに慣れましょう。

1. オブジェクト
2. データ型
3. 変数

**オブジェクトって何だろう？**

**オブジェクトとプロパティの関係**

オブジェクトとは、物、目標物、対象という意味の英単語です。

例えば、“靴”という“物”を例にします、その靴をコンピュータープログラムの世界では「“靴”というオブジェクトとして取り扱います」、靴全体に、赤い色を付けて表示するためには、「“靴の全体のColor“を“赤”にします」、靴底の色を青色にして表示する場合は「“靴の底のColor”を“青”にします」、また靴ひもを付ける場合は「“靴ひも“を”あり“にします」。

上記のように、“靴の全体のColor”、“靴の底のColor”、“靴ひも”等のことを「プロパティ名」といいます、また、“赤”、“青”、”あり“等のことをプロパティの値と呼びます。

プログラマーの初心者を抜け出した時には、下記のような正しい解説が理解出来ます。

**オブジェクトとは**

ソフトウェアのユーザインターフェースなどでは、ユーザーの操作対象として列挙されたアイテム(指定されたフォルダに含まれるファイルなど)のことをオブジェクトとして、指すことが多い。

オブジェクト指向プログラミングにおいては、ソフトウェアが扱おうとしている現実世界に存在する物理的あるいは抽象的な実体を、属性(データ)と操作(メソッド)の集合としてモデル化し、コンピュータ上に再現したもの。オブジェクトを定義するモデルはクラスと呼ばれる。クラスに基づいて実際にコンピュータのメモリ上に展開されたオブジェクトのことをインスタンス(実体)と言うが、このインスタンスの意味で、オブジェクトと呼ぶ場合も多い。

**データ型とは**

　ここで、次に進む前に抑えておきたい大事な事を眺めておきましょう。

データ型とは、データの種類とサイズのことです。たとえば、商品名などの文字列を扱うときはString型、小数点以下の数値で計算するときはSingle型など、そのデータの種類やサイズによって使用する変数（データを格納するための容量を持つ入れ物）のデータ型を宣言します。プログラムでは、様々な種類のデータを使用します。それらのデータを変数に記憶させて効率的にメモリを管理するために、VisualBasic .NETでは、データを変数に格納するときにあらかじめ用意されたデータ型も宣言します。

次表に一覧表を作成しました。

データ型の種類（参照用として、利用して下さい）

|  |  |  |
| --- | --- | --- |
| **データ型** | **サイズ** | **値の範囲** |
| 短整数型（Short） | ２バイト | -32,768～32,767 |
| 整数型（Integer） | 4バイト | -2,147,483,648～2,147,483,647 |
| 長整数型（Long） | 8バイト | -9,223,372,036,854,775,808  ～9,223,372,036,854,775,807 |
| 単精度浮動小数点数型（Single） | 4バイト | 負の値:  -3.4028235E+38～-1.401298E-45  正の値:  1.401298E-45～3.4028235E+38 |
| バイ精度浮動小数点数型（Double） | 8バイト | 負の値:  ･４.797693.134862315=7=OE÷308  ～-4.94065645841246544E-324  正の値:  ４,94065646841246544=Ｅ-324  ～1.79769313486231570E+308 |
| １０進数型（Decimal） | 16バイト | 整数の場合:  ･79､228,162,514,264,337,593,543,950,335  ~79,228,162,514,264,337,593､543,950,335  小数点以下28桁の場合:  -7.9228162514264337593543950335  ～7.9228162514264337593543950335  0を除いた場合の絶対値の最小値:  0.0000000000000000000000000001(IE-28) |
| バイト型（Byte） | 1バイト | O～255 (符号なし) 1byteは8bit。つまり「2の8乗」です。これを計算すると、1byteで表せる情報は256種類ということになります。 |
| Char型（Char） | 2バイト | O～65535(符号なし) |
| 文字列型（String） | 格納サイズにより可変 | O～約20億個のUnicode文字 |
| ブール型（Bool） | 2バイト | True(真)またはFalse(偽) |
| 日付型（Date） | 8バイト | 0001年1月1日0：00：00～9999年12月31日 |
| オブジェクト型（Object） | 4バイト | 任意の型を格納出来る |
| ユーザー定義型（構造体） | 格納サイズにより可変 | 実装するプラットフォームに依存 |

難しいようなことが書いてありますが、眺めて置く程度で充分です。

私の場合上表を印刷して、参照用に壁に貼っていました。

## データ型の変換

.NETにおけるデータ型の変換には、\*\*暗黙的な変換\*\*と\*\*明示的な変換\*\*の2つの方法があります。暗黙的な変換は、コンパイラが自動的に行う変換で、明示的な変換は、プログラマが明示的に指定する変換です。

暗黙的な変換は、コンパイラが自動的に行うため、プログラマが特別な処理を書く必要はありません。ただし、暗黙的な変換が行われる場合、データの精度が失われる可能性があるため、注意が必要です。

明示的な変換は、プログラマが明示的に指定するため、コンパイラが変換を行う前に、プログラマが指定した処理が実行されます。明示的な変換は、データの精度を維持するために使用されます。

例えば、整数型の変数 `i` を浮動小数点型の変数 `d` に変換する場合、以下のように記述します。

```csharp

int i = 10;

double d = (double)i;

```

このように、変換先のデータ型を変換元のデータ型でキャストすることで、明示的な変換が行われます。

詳細については、Microsoftの公式ドキュメント¹を参照してください。

(1) 【初心者向け】Pythonの変数の型変換をする方法を徹底解説 .... https://aizine.ai/python-type-conversion0922/.

(2) 型変換とは - 意味をわかりやすく - IT用語辞典 e-Words. https://e-words.jp/w/%E5%9E%8B%E5%A4%89%E6%8F%9B.html.

(3) 【C言語入門】型のキャストまとめ(intからdouble、charへの型変換). https://www.sejuku.net/blog/25737.

(4) Java 基本データ型、参照型：基本データ型と参照型の型変換 .... <https://qiita.com/mrshouuge/items/72ab1937b3cef6c743af>.

URL　<https://learn.microsoft.com/ja-jp/dotnet/standard/base-types/type-conversion?redirectedfrom=MSDN>

URL　<https://learn.microsoft.com/ja-jp/dotnet/fundamentals/runtime-libraries/system-convert?source=recommendations>

変換の例 VB

Dim Str　As String

Dim　intX As Integer

Str = “32”

intX = Cint(Str)

intX = 100

Str = intX.ToString()

## 変数とは？を深く理解する

### 変数を宣言してデータを格納（Ｄｉｍステートメント）

　変数とは、データを一時的に格納しておくための入れ物で、プログラム実行中に変数に値を代入したり、値を変更したりすることができます。　変数を利用するには、変数の名前を宣言する必要があります。変数の宣言は、下記のように記述します。

　Dim　変数名　Ａｓ　データ型

　１つのDimステートメントで、複数の変数を宣言できます。その場合、カンマ「､」で区切って入力します。変数を宣言したら、その変数に値を代入しますが、変数の宣言と同時に値の代入を行うこともできます。

データ型一覧表

プログラムコード例

**プログラムコード例 1**

|  |
| --- |
| ‘文字列型変数ａを宣言する  　　　　　Dim ａ As String  　　　　　　･整数型変数ｂ．ｃを宣言する  　　　　　　　　Dim b, c As Integer  　　　　　　　　‘長整数型変数ｆと、単精度浮動小数点型変数ｈを宣言する  　　　　　　　　Dim f As Long, h As Single  ・値の代入  a = "test"  b = 365  '文字列型変数ｇの宣言と値の代入を同時に行う  Dim g As String = “test2'” |

それでは実践を通して理解度を深めていきます。

## 新規にWebサイトを作成

**最新のVisualStudio２０１9でのプロジェクト作成手順**

VisualStudio２０１７のメニュー「新規作成」－＞「プロジェクト」と進みます、下図

|  |
| --- |
|  |

上図の「プロジェクト」を選択すると、下図が表示されます、

|  |  |
| --- | --- |
|  | 旧バージョンのVisualStudioと少し違うところは、赤い四角で囲んだ部分の“以前のバージョン」という項目が追加されています。  本解説書を実践する場合は、この“以前のバージョン”を選択して、次に右の欄の「ASP.NET　Webフォームサイト」を選択します。  その後の、「名前」、「ソリューション名」等の設定の仕方は、旧バージョンと同じ要領で設定して下さい。  尚、旧、新バージョン共、開発に関する機能は特別変わりはないようです。 |

**テストページで基礎を極める**

**サイト内に新規にWebページを作成する**

名前は「TestHensu」にして下さい。

|  |
| --- |
| <asp:Content ID="Content1" ContentPlaceHolderID="MainContent" Runat="Server">  　　　<div class=”row”>  <div class=”col-md-6”>  <h2>左列</h2>  </div>  <div class=”col-md-6”>  <h2>右列</h2>  </div>  </div>  </asp:Content> |

Htmlコードビューの任意の位置で、マウスの右クリックして、メニューの「ブラウザーで表示」をクリックして、実行して下さい。下図のようになっています。



実行画面を閉じてください。

次に、コードビュー及びデザインビューを表示して、どちらかの任意の場所を右クリックして、“コードの表示”を選択して、プログラムコード画面上部の中央列の“（TestHensu イベント）”をクリックします。

|  |
| --- |
|  |

次に、右側の“Disposed”と表示されている、ドロップダウンを開きます、下図

|  |
| --- |
|  |

次に上図中央付近の「Load」というイベント名を選択して下さい。

選択と同時にプログラムコード内に下表のようなイベントハンドラーが表示されます。

|  |
| --- |
|  |

このイベントは「TestHensu.aspx」ページが表示される時に発生するイベント名です。

作成した、“TestHensu\_Load”プロシージャ内に下記のコードを入力します。

赤いコード部分コピペ可能

|  |
| --- |
| Private Sub TestHensu\_Load(sender As Object, e As EventArgs) Handles Me.Load  '文字列型変数ａを宣言する  Dim a As String  '整数型変数ｂ.ｃを宣言する  Dim b, c As Integer  '長整数型変数ｆと、単精度浮動小数点型変数ｈを宣言する  Dim f As Long, h As Single  '値の代入  a = "test"  b = 365  c = 400  f = 1.5555  h = 2.22  '文字列型変数ｇの宣言と値の代入を同時に行う  Dim g As String = “test2”  End Sub |

次に、htmlコードビュー及びデザインビューを表示して、デザインビューの“左列”の直下に、テーブル（８行、２列）を挿入します。

（テーブルの挿入の手順はこの解説書の「付録」で解説してあります）

Htmlコードビューの<h2>左列</h2>のコードの下側に“kk”という文字を手入力（キーボードから）して下さい。下図

|  |
| --- |
|  |

次に、コードビューとデザインビューの間に、下図のように黄色の「デザインビューとソースビューと同期していません“という表示がでます。

|  |
| --- |
|  |

上図の黄色の部分をクリックして、同期して下さい。下図のように、追加で入力したコードがデザインビューに反映されます。

|  |
| --- |
|  |

上図の“ｋｋ”という文字の右側にマウスのポインタを置くと、VisualStudioの作業画面の最上部のメニューに「テーブル」というメニュー項目が自動で追加されます。次図

|  |
| --- |
|  |

上図の最上部のメニュー項目の「テーブル」をクリックします。下図のようなサブメニューが表示されます。

|  |
| --- |
|  |

上図の「テーブルの挿入」を選択します。下図のような「テーブルの挿入」ダイアログ（設定画面）が表示されます。

自動で設定されている以外は、次図のように設定します。

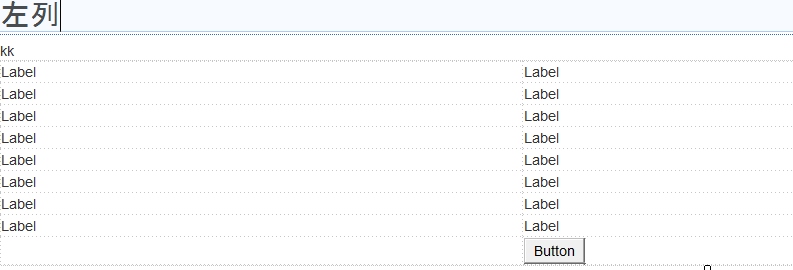
|  |
| --- |
|  |

「OK」ボタンをクリックします。

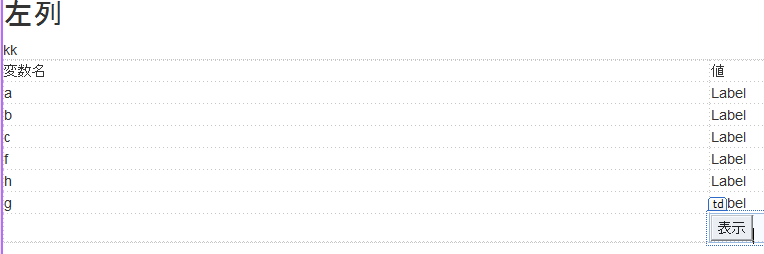
デザインビューは次図のようになります。

|  |
| --- |
|  |

上図のテーブルの各列、各行に下図のようにツールボックスからWebサーバーコントロール（以下コントロールと呼びます）をドラッグアンドドロップして下さい。



次に、テーブル内の各LabelコントロールのTextとButtonコントロールのTextを下図のようにします。



次にテーブルの右列の各Labelコントロールの「IDプロパティ」を下図のようにして下さい。



次に、プログラムコードに移って、Page\_Loadイベントに下記のようにコードを追加します。網掛け部分を追加します。

|  |
| --- |
| '文字列型変数ｇの宣言と値の代入を同時に行う  Dim g As String = “test2'”  LblA.Text = a  LblB.Text = b.ToString()  LblC.Text = c.ToString()  LblF.Text = f.ToString()  LblH.Text = h.ToString()  LblG.Text = g  End Sub |

ブラウザーで実行して下さい。下図のようになりましたか？

|  |
| --- |
|  |

前図の左列の直下の“ｋｋ”は、テーブルを挿入する時に使用した目印です。

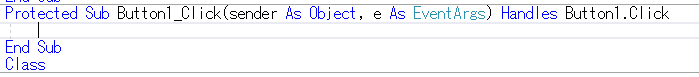
前図の、結果表示をプログラムしているのは、Loadイベントハンドラーの下記の部分です。

|  |
| --- |
| LblA.Text = a  LblB.Text = b.ToString()  LblC.Text = c.ToString()  LblF.Text = f.ToString()  LblH.Text = h.ToString()  LblG.Text = g |

次に、TestHensu\_Loadイベントハンドラー内の上記の網掛け部分を削除します。

**変数のスコープ**

次に、デザインビューの“表示”ボタンをダブルクリックして、Button1のClickイベントを作成して下さい。下図



このボタンコントロールには、特別にID名を設定していなかったから、自動で割り振られたID名をそのままに、Clickイベントが作成されました、状況に応じて、他のコントロールについても、特別にID名を付ける必要がないときは、自動で作成されるID名を使用して下さい。

その中に“LblA.Text = a”と入力して下さい。

|  |
| --- |
|  |

”a“の下に赤の波型のアンダーラインが表示されて、マウスを近づけると、”aは宣言されていません……”という表示が現れます、この意味は、“a”という変数をこのClickイベント内では、知らない状態なのです。“a”という変数は、Loadイベントハンドラー内で宣言されていますので、Loadイベントハンドラー内でだけ有効なのです、そのような規則になっています。したがってボタンのClickイベントは、“a”の変数を知らないのです。この現象を、見えない状態として、“スコープ外”と表現します。

“スコープ”の範囲は、色々あります。参考に大きく分けて次のようなスコープ範囲があります。

1. 変数のページ内のスコープ
2. 変数のサイト内のスコープ
3. 変数のクラス内のスコープ

等です。

**変数のページ内のスコープの実験**

プログラムコードのLoadイベントハンドラー内の下記の網掛け部分のコードを選択して、

|  |
| --- |
|  |
| '文字列型変数ｇの宣言と値の代入を同時に行う  Dim g As String = “test2” |

右クリックメニューから、“切り取り“を選択して切り取り、下記の部分に貼り付けて下さい。

|  |
| --- |
|  |

すると、Button1\_Clickイベント内の“a”の下の赤い波線が消えました。という事は“a”という変数がButton1\_Clickイベントから、見えたという事です。（かしこいですね～）

このことは、変数を“ページクラス内で宣言”したから、という事です。

上記の一行目の“Partial Class TestHensu”というコードで、なんとなく解るように、この

”TestHensu.aspx“というページは、“Class TestHensu”というクラスによって作成（インスタンス化）されていると、認識して下さい。（難しいことは、後回しにして….今はコードを実行して、コードの命令通りにコンピューターが動く事で満足しましょう）

それでは、Button1\_Clickイベントに、下記のコードを追加しましょう。赤のコード部分コピペ可能

|  |
| --- |
| Protected Sub Button1\_Click(sender As Object, e As EventArgs) Handles Button1.Click  LblA.Text = a 'aの型はString型の文字列だから、ToString()関数は使用しない  LblB.Text = b.ToString()　　'bはInteger（整数）型だからToString()関数で文字として表示する  LblC.Text = c.ToString()　　'同上  LblF.Text = f.ToString()　　'同上  LblH.Text = h.ToString()　　'同上  LblG.Text = g　　　　　　　'gもaと同じ  End Sub |

コードビュー及びデザインビューを表示して、ブラウザーで表示します。「表示」ボタンをクリックします。次図のように表示されます。

|  |
| --- |
|  |

これで、変数がページ内で有効であることを確認しました。

実行画面を閉じます。

ここまでの実験では、ページのロード時に、各、変数に値を代入していましたが、Laodイベントを空にして、Button1\_Clickイベント内に、各、変数に値を代入するコードを追加して、右列のLabelコントロールに表示する、プログラムをして見て下さい。

更に読者様の理解が深まります。

先程のButton1\_Clickイベント内のコードで、“ToString()”というコードがありました。

**ToString（）とは**

ToString関数と呼びます、

キーワード「“Object.ToString メソッド ()”」です、検索して確認して下さい。

（メソッドという、意味は“値を引き出す式（コード）”と理解して下さい）

VSのテクニカルヘルプで検索した結果、下記のような情報を確認しました。

Object.ToString メソッド ()

Returns a string that represents the current object.

Public Overridable Function ToString As String

2行目を直訳しますと、「現在のオブジェクトの値を“文字（型）”で表示（出力）する。

3行目の意味を、簡単に訳しますと、パブリックなオーバーライド可能な、ToStringという名前の“Functionプロシージャ”であり、String型のデータを返す。

**変数の使用法について**

例えば、変数aの値がString型で、値が“こんにちは”である場合、このような場合は、ToString()関数を使用して、String型に変換する必要はありません。aの値を表示する、LabelXコントロールに代入する場合のコードは、“LabelX.Text = a”となります。

変数fの値が、単精度浮動小数点数型（Single）で“3.14”という値であるときは、ｆの値の型がString型ではないので、ToString（）関数を使用して、LabelXのTextプロパティには、ToString関数を使用して、Single型の値‘3.14’ をString型に変換して、代入する必要があります。その場合の表示（出力）された型は、文字の”3.14“です。

また、変数ｄというDate型の変数があって、日付の値が格納されている場合は、ToString関数によって、“2025/04/15”のような日付を表す文字列に変換して表示します。

これが、ToString関数の役割です。

1. **配列：複数の値を表す変数**

変数とは、コンピュータープログラムにおいて、データを格納するのに、文字とか数字等のデータをそれぞれのデータの型に応じて、格納する領域を名前を付けて宣言した物（オブジェクト）の事である事を今まで解説し、実践してきました。

プログラマーとしてもう一つレベルアップするために、“変数の配列”という知識を習得する必要があります。

前章の「Dim TodoType As String() = {"仕事", "個人", "行事", "その他"} '大かっこ”{ }”で囲まれた中の文字列をString型の配列に初期化している」というこの部分です。

配列と呼ばれる別の種類の変数は、同じ型の複数の値を格納するのに便利です。

前章の付録の中で詳しく説明してありますので、重複する説明、解説は控えます、実践を通して、さらに理解を深めて、読者様のレベルを上げて下さい。

**TestHairetsu.aspxページを作成**

ソリューションエクスプローラのサイト名「TimeTable」を右クリックして、下図のように「追加」メニューにマウスを載せます。

|  |  |  |
| --- | --- | --- |
|  | 表示された左側のサブメニューの「Webフォーム（マスターあり）」を選択します。下図が表示されます。   |  | | --- | |  |   上図のように、項目名の入力欄に“TestHairetsu”と入力します。  次に、「OK」ボタンをクリックします。 |

次に、下図の「マスターページの選択」画面が表示されます。

|  |  |
| --- | --- |
|  | “Site.master”を選択して、「OK」ボタンをクリックします。 |

次に、表示されている、“TextHairetsu.aspx”のコードビューに下記のコードを追加します。赤のコード部分コピペ可能

|  |
| --- |
| <asp:Content ID="Content1" ContentPlaceHolderID="MainContent" Runat="Server">  　　　　<div class=”row”>  <div class=”col-md-12>  　　　　　　　　<h2>一列</h2>  </div>  </div>  </asp:Content> |

次に、プログラムコードファイルに移って。

TestHairetsu ページのLoadイベントハンドラーを作成して下さい。

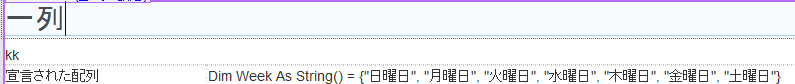
TestHairetsu.aspxが最初にLoadされた時に実行される、配列の初期化を行うコードを“TestHairetsu”のLoadイベントハンドラーに追加します。下記コード、赤のコード部分コピペ可能

|  |
| --- |
| If Not IsPostBack Then　'このページが最初にLoadされた時に実行する  Dim Week As String() = {"日曜日", "月曜日", "火曜日", "水曜日", "木曜日", "金曜日", "土曜日"}  '上記は、配列名”Week”を宣言して、日曜日から土曜日までの、曜日名を配列のデータとして代入しています、  '宣言と同時にデータを代入する事を,"初期化"といいます。  End If |

次に、

TestHairetsu.aspxのコードビュー及びデザインビューを表示して、”一列”の文字の直下に“ｋｋ”という文字を入力して目印にします。その“ｋｋ”の文字の右側にマウスのカーソルをおいて、テーブル（2行、2列）を挿入して下さ。1行目の左列にLabelコントロールを配置して、Textプロパティを“宣言された配列”にします。次に、右列にLabelコントロールを配置して、Textプロパティに

「'Dim Week As String() = {"日曜日", "月曜日", "火曜日", "水曜日", "木曜日", "金曜日", "土曜日"}'」の赤い部分をコピペして、貼り付けます。下図のようになります。

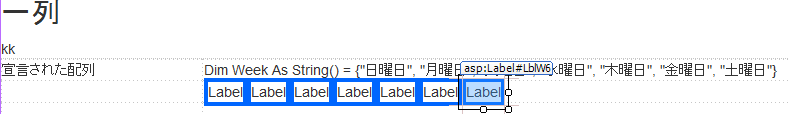


次に、テーブルの2行目の右列にLabelコントロールを連続して、7個配置して、各IDを左から“LblW0”,”LblW1”,“LblW2”,”LblW3”,“LblW4”,”LblW5”、“LblW6”にします。

そして、その7個のLabelコントロールのプロパティを下図のように設定します。

|  |
| --- |
| 青色 |

上図のBackColorは「#6666FF」“青色”（赤色部分コピペ可能）です。BorderStyleは“枠線”の事です。下図のようになります。



次に、htmlコードビューのの２行目の２列目のLabelコントロールが配置された、htmlコード（下記の部分）を表示して、手入力で<br />タグを挿入します。この場合<br　とタイプして、>をタイプすれば自動で“/”の記号は自動で挿入されます。

|  |
| --- |
|  |

デザイン画面は、下図のようにLabelコントロールが縦に並びます。

|  |
| --- |
|  |

次に、プログラムコードファイルに移ります。

TextHairetsu.aspxのLoadイベントに下記のコード、赤のコード部分を追加します。コピペ可能

|  |
| --- |
| Private Sub TestHairetsu\_Load(sender As Object, e As EventArgs) Handles Me.Load  Dim Week As String() = {"日曜日", "月曜日", "火曜日", "水曜日", "木曜日", "金曜日", "土曜日"}  '上記は、配列名”Week”を宣言して、日曜日から土曜日までの、曜日名を配列のデータとして代入しています、  '宣言と同時にデータを代入する事を,"初期化"といいます。  Dim len As Integer = Week.Length - 1 '配列の要素数から１を引く  Dim i As Integer  For i = 0 To len  If i = 0 Then  Me.LblW0.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) 'iの値と日曜日を表示　Week(0)の値です  ElseIf i = 1 Then  Me.LblW1.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) 'iの値と月曜日を表示　Week(0)の値です  ElseIf i = 2 Then  Me.LblW2.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) 'iの値と火曜日を表示　Week(0)の値です  ElseIf i = 3 Then  Me.LblW3.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) 'iの値と水曜日を表示　Week(0)の値です  ElseIf i = 4 Then  Me.LblW4.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) 'iの値と木曜日を表示　Week(0)の値です  ElseIf i = 5 Then  Me.LblW5.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) 'iの値と金曜日を表示　Week(0)の値です  ElseIf i = 6 Then  Me.LblW6.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) 'iの値と土曜日を表示　Week(0)の値です  End If  Next  End Sub |

要素数（データの数）は、日曜日から土曜日まで数えれば、データの数は、７個です、この場合コンピューターに配列として保存されている場所は、Weekという配列名の“Week(0～６)”の場所にあります。コンピューターの数字は、0から始まりますので、Weekの1番目の要素（データ）の場所はWeek(0)の場所（位置）です。したがって、Week配列の値を取得するには、日曜日の場合、Week(0)の値を参照（取得）すればいいのです。

上記のプログラムコードの変数“i”の値（数字）を０から６まで（7回）に変化させてWeek内の要素（データ）をとり出している事を是非理解して下さい。（理解できるまでこのコードを眺めるのも一つの方法です）

コードビュー及びデザインビューを表示して、ブラウザーで表示、実行して下さい。

次図が実行結果です

|  |
| --- |
|  |

目からウロコでしたか？

実行画面を閉じます。

**実行画面での、文字列の表示方法**

先程の実行結果を表示するのに、全てのコンピュータプログラム言語に共通する、文字列表示のコードを説明します。

先程のプログラムコードの例

Me.LblW0.Text = "Week(" & i.ToString() & ")" & " =" & Week(i) を解説します。

Me.LblW0.Text　これはLabelコントロールのID、「LblW0」のTextプロパティ

“Week(“ このダブルクオーテーション「“」マークで囲んだ中の部分をそのままの文字で表示すします。ダブルクオーテーション「“」マークで囲むと、囲まれたデータはString（文字）ですよ、と宣言したことと同等です。

「＆」記号は　文字列と文字列をつなぐという役割です、「＆」記号の代わりに「＋」記号を使用しても、同じです。

i.ToString() は、変数iの値をString型に変換して、表示しています。このようなi等のデータの値を表示する時は、「“」で囲みません。「”」で囲みますと、iをデータとして認識しないで、文字として表示してしまいます。

このような、記法をマスターするのに、筆者の場合、「LabelX.Text = “ｘｘ” + “)” & i.ToString()」等というように、何度もキーボードから手入力して、身に着けたことを思い出します。

プログラマー必須の定型文

## プロシージャ（関数）とは？

コンピュータ―プログラムの世界では、“メソッド”、“プロシージャ”または“関数”という言葉が頻繁に出てきます。“プロシージャ”、“関数”双方とも“メソッド”と呼びます、今までに出てきた“Button1\_Click”や“PageLoadイベントハンドラー”もメソッドです。

“メソッド”を直訳しますと、“順序、筋道、秩序”です。

“プロシージャ（procedure）”を直訳しますと、“（進行上の）手続き、手順、順序”等の意味があります。

技術情報：[基本的なメソッド定義と呼び出し](https://msdn.microsoft.com/ja-jp/library/dd314348.aspx)　のリンクを参照して詳細を読んで理解を深めて下さい。

日本語で“関数”と呼ぶのは、数学の関数から来ているように思われますが。これから解説する、“ファンクションプロシージャ”を理解すれば、“関数”の語源が理解できるでしょう。

ここから以後“プロシージャ”の事を、日本語の“関数”または、“メソッド”、“サブ関数”、“ファンクション関数”等と呼んで、解説を進めます。

関数、メソッドには大きく分けて、次の2つに分かれます。

値を返さない“サブ（Sub） 関数”

値を返す“ファンクション（Function）関数”

サブ関数は、イベントハンドラーに象徴されるような、関数名の前に“Sub”という語をつけて、”値を返さないメソッド“であると宣言します。

のファンクション関数は、関数名の前に“Function”という語をつけて、”値を返すメソッド“であると宣言します。読者様にとってはまだ見たこともない関数ですが、これから実践する“TestProcedure.aspx”のページで体験していただきます。

それでは、この章の、「サイト内に新規にWebページを作成する」項を参考にして、「Webフォームマスターあり」

項目名”の入力欄に“TestProcedure”と入力して、“OK”ボタンをクリックします。マスターページは「Site.master」を選択します。

TestProcedure.aspxページのコードビューが表示されましたら、「並べて表示」をクリックして、コードビュー及びデザインビューを表示して下さい。

下記はTestProcedure.aspxのｈｔｍｌコードビューです。

|  |
| --- |
| <asp:Content ID="Content1" ContentPlaceHolderID="MainContent" runat="server">  <div class="container">  <h2>関数について</h2>  <h4>値を2倍にして返す関数</h4>  <table class="table" >  <tr>  <td class="modal-sm" style="width: 139px">値</td>  <td style="width: 119px">  <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>  </td>  <td style="width: 109px">答え</td>  <td>  <asp:Label ID="Label2" runat="server"></asp:Label>  </td>  </tr>  <tr>  <td class="modal-sm" style="width: 139px">&nbsp;</td>  <td style="width: 119px">  <asp:Button ID="Button1" runat="server" Text="処理" />  </td>  <td style="width: 109px">&nbsp;</td>  <td>&nbsp;</td>  </tr>  </table>  </div>  </asp:Content> |

デザインビュー

|  |
| --- |
| テーブル  自動的に生成された説明 |

プログラムコード

|  |
| --- |
| グラフィカル ユーザー インターフェイス, テキスト, アプリケーション, メール  自動的に生成された説明 |

講義中に講師と共に実践します。

あなたが、もし長い間コンピュータプログラミングと関わるような事であれば下記URLを何十回も訪れることになります。そして訪れる度に少しづつ霧が晴れるようになるでしょう。下記URLを是非大事にして下さい。

URL　<https://learn.microsoft.com/ja-jp/dotnet/visual-basic/language-reference/functions/>

関数というと、数学っぽくて難しそうなイメージがありますけど、プログラミングの関数はそんなに難しいものではありません。

関数とは、処理のまとまりのことです。

プログラミングをしていると、1つのプログラムの中に何回か同じ処理を行う時があります。

その同じ処理っていうのを処理のまとまり「関数」として扱うことができます。

同じ処理なんだったら、コピペしたらいいじゃん！って思う人がいるかもしれませんが、関数にはいろいろとメリットがあるので、紹介していきます。

VB.NETにおいて、Sub関数とFunction関数は、プログラムの処理をまとめるために使われます。Sub関数は戻り値を持たず、Function関数は戻り値を持ちます。以下にそれぞれの関数の基本的な使い方を示します。

' Sub関数の定義

Private Sub 関数名 (ByVal 引数 As 型)

処理を書く

End Sub

' Sub関数の呼び出し

関数名(引数)

' Function関数の定義

Private Function 関数名 (ByVal 引数 As 型) As 返却する値の型

処理を書く

End Function

' Function関数の呼び出し

Dim 戻り値 As 返却する値の型 = 関数名(引数)

Sub関数は、処理をまとめることで、プログラムの見通しを良くし、修正やテストが楽になります。一方、Function関数は、戻り値を返すことができるため、呼び出し元に値を返すことができます

## クラス（Class）とは？

.NET 全体の情報　URL　 <https://dobon.net/vb/dotnet/index.html>

永年プログラミングを実践していると、自分で作成した使い勝手のいい関数が増えてきます。例えばその関数は複雑な計算用の関数であったり、表示用に特別に作成した関数であったり、汎用的に捨てがたい関数がある場合、複数の関数を役割に応じた区分けして置くと便利です。そのような場合役割に応じた名前のクラスに保存して、それをプロジェクト内の何処からでも呼び出すことが出来るようにします。VisualStudioではそのようなクラスを一元的に管理できるような仕組みがあります。それはプロジェクト内に「App\_Code」というフォルダを作成してその中にクラスのファイルを格納して置くようになっています。なおクラス内に変数（プロパティ）を宣言することも出来ます。

### クラスを作って、使用してみる

講座での実践で時間をかけて、クラスの作成から使用までを実践します。

#### 重要な事：

VisualStudioで開発中に、自分で定義してないのに使えるクラスやメソッド、プロパティというのは.NET側で予め用意されたクラスの中に記述されているから使用できるんです。これが継承の効果です。頭の片隅にでも入れておいてください。

### インスタンスとは？

Private Array As New ArrayList()

上記の例ではArrayListクラスをインスタンス化をして、Arrayという変数を定義しています。Newキーワードが必要

|  |
| --- |
| クラスのインスタンス化  ArrayListクラス  既存のクラス  　　　　　　　　　　　　　　　　　　　　　　　　　Private Array As New ArrayList()  Newをつけることで変数名「Array」は「ArrayListクラス」になる。  　　　　　　　　　　　　　　　　　　　　　　　　　以後、Array.xxxxxの記述で (ArrayList)のProperryやメソッドが使用できる。 |

感覚的に説明すると

○○クラスで「計算クラス」のメソッドを使用したいなら

Private 任意インスタンス名 As New 「計算クラス」

任意インスタンス名.使用したいメソッド名()

**.NET 名前空間（クラス）**

.NET Framework には、多数の名前空間（クラス）が用意されています。以下に、一部の名前空間を紹介します:

System.Collections: コレクションを扱うためのクラスが含まれています。

System.Data: データベースにアクセスするためのクラスが含まれています。

System.Drawing: 画像を扱うためのクラスが含まれています。

System.IO: ファイルの読み書きを行うためのクラスが含まれています。

System.Linq: LINQ (Language-Integrated Query) を使用するためのクラスが含まれています。

System.Net: ネットワーク通信を行うためのクラスが含まれています。

System.Reflection: リフレクションを使用するためのクラスが含まれています。

System.Text: 文字列を操作するためのクラスが含まれています。

System.Threading: スレッドを扱うためのクラスが含まれています。

以上のように、.NET Framework には多数の名前空間が用意されており、それぞれの名前空間には特定の目的に応じたクラスが含まれています。使用するクラスやメソッドは、アプリケーションの目的に応じて適切に選択する必要があります

**htmlコードとプログラム言語の関係**

技術情報：[ASP.NET Webページの概要](https://msdn.microsoft.com/ja-jp/library/428509ah(v=vs.100).aspx)　（日本語に翻訳するボタンが出現します）

特に、上記のヘルプドキュメントの“ASP.NET　Webページを使用して出来ること”の項に目を通しておいてください。

HTML（ハイバー.テキスト..マークアップ.ランゲージ）の知識を得るにはインターネットに溢れるほどの情報がありますので、是非検索して知識を習得して下さい。

前章でも、解説済ですが、Webページそのものは、htmlコードによって表示されています、解り易く言えば、VisualStudioという開発ツール（多言語の）を使用して、プログラムされたWebアプリは最終的には、HTMLコードにコンパイルされて、Webブラウザーによって表示されます。そのプログラムの過程においてプログラマーの困難な部分、例えば入力用のWebサーバーコントロール等、マイクロソフトのASP.NETフレームワークが、プログラマーが膨大な労力を要するWeb用の資産（VSのツールボックス内のツール）を既に作成してくれているのです、そのことは、VSのツールボックスから、コードビュー及びデザインビューにWebサーバーコントロールをドラッグアンドドロップした後の、コードビューを確認すると、<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-Size="1.2em" Text="タイトル"></asp:Label> のようにhtmlコードでマークアップされます。

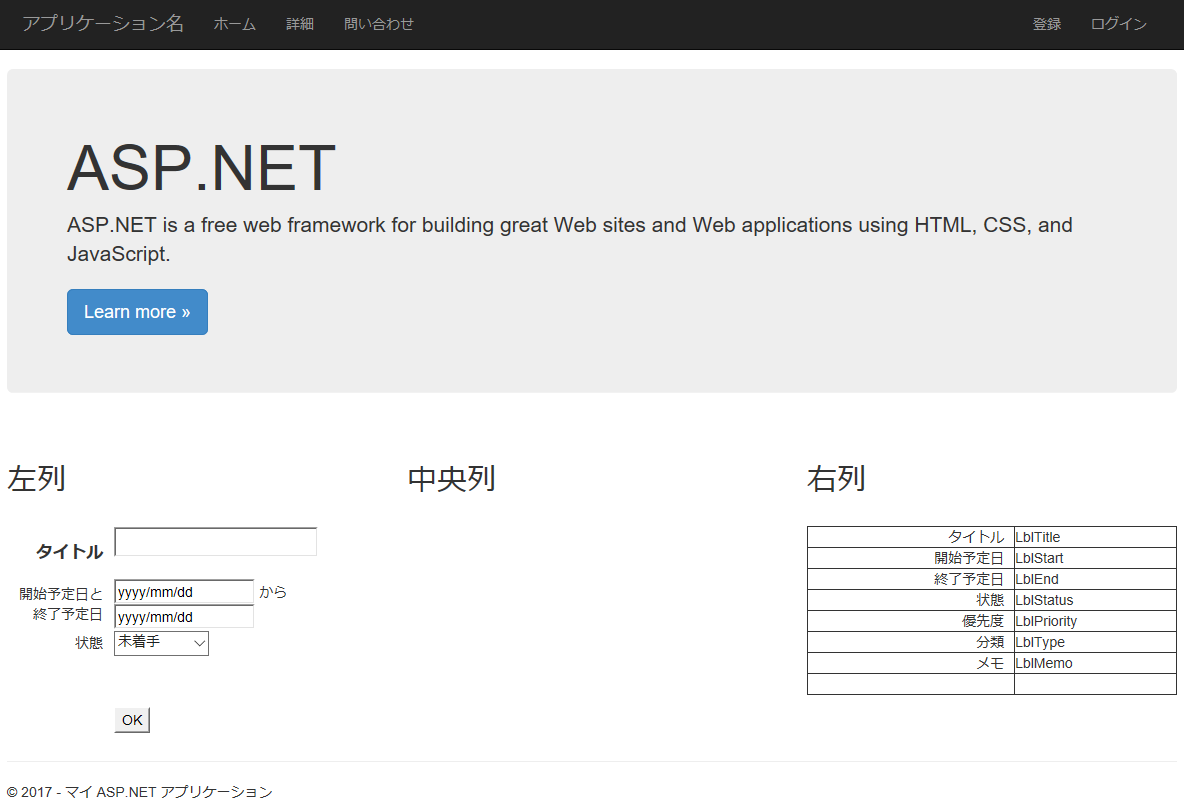
また、使用するプログラム言語とプログラムコードを参照するために、VSの各ページの1行目に下記のような記述があります。赤色の部分

<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master" AutoEventWireup="true" CodeFile="Default.aspx.vb" Inherits="\_Default" %>

使用するプログラム言語が“C#”の場合は、”Languege=”C#”“になります。

**プログラマーなら“CSS”を理解しよう**

VisualStudioでWebサイトを作成して、マスターページを親とする子のページを表示した場合の画面をざっくりとプログラマー目線で見ると、下図のように解剖できます。



コンテンツの中の <div class=”row”>内

コンテンツの中の　ヘッダー部

コンテンツ部（各ページ）

フッター部

メニューバー

前図の「メニューバー」と「フッター部」はマスターページ内で構成されています。

コンテンツ部は、各ページでプログラムしたものが表示されています。

次に、マスターページと各ページの関係を図にしました。下図

マスターページ

メニューバー

各ページのコンテンツ

ID=“MainContent”

フッター

前図のように、「メニューバー」、「各ページのコンテンツ」、「フッター」は、マスターページ内に、あります。

サイト内に“新規にWebページを作成”する時に、“Webフォーム（マスターあり）”のメニューを選択した場合に作成したページは、前図の「各ページのコンテンツ」部に表示されます。

**マスターページの構造**

マスターページも他のWebページと同様、htmlコードで作成されて表示されます。

ソリューションエクスプローラウインドウから、“Site.master”をクリックして、Site.masterのコードビューを開いてください。

メニュー表示部分が下記のコードで表示されています。

**メニュー表示部分**

|  |
| --- |
|  |

43行目の<div…はメニューを表示す範囲の背景色、等を定義しています。

45行目から50行目までは、モバイル用にメニューを表示するための各種の設定をしています。５４行目から、58行目までがパソコン、タブレット用のメニュー表示のコードです。

**ページの表示部分とフッター部分**

「ページの表示部分とフッター部分」を担っているのが、下記の部分です。

|  |
| --- |
|  |

「<asp:ContentPlaceHolder>～</asp:ContentPlaceHolder>の中に、各ページが表示されます。<footer></footer>も後で、変更して見ましょう。

その他の、htmlコードについては、わが社の実践解説書の高度化に合わせて、解説しています。

ざっくり（大雑把）に理解して下さい。コンピュータ―プログラムの世界は膨大な量の知識が必要です、優秀なプログラマーは、どの情報がどの辺にあったかを知っている（経験している）ことなんです。

**HTMLとCSSの関係**

Webページの本体がHTML（ハイパーターミナルマークアップランゲージ）です。以前はこれだけで作るのが主流でしたが、現在は、CSS（カスケードスタイルシート）を絡めて作るのが多くなってきています。下欄はヘルプの記事からの引用です。

|  |
| --- |
| カスケード スタイル シート (CSS: Cascading Style Sheet) には、Web ページの要素に適用されるスタイル ルールが格納されます。CSS スタイルは、ページでの要素の表示方法と配置を定義します。ページの各要素に属性を個別に割り当てる代わりに、要素のインスタンスや、特定のスタイル クラスに割り当てられた要素が Web ブラウザで見つかるたびに属性を適用する一般ルールを作成します。 |

１、CSSは文章に直接styleで書き込む事もできる。

２、HTMLの上部（head部）に指定する事もできる。

３、外部にCSSファイルを置きlinkする事もできる。

１，２、３のどれかの方法です（混合も可能）

多くのＨＰを作る場合、「３」の一つの外部CSSにlinkさせると一つのCSSだけで全部のＨＰのHTMLを同様に指定できます。

例えば　「赤い大きい文字」が１０箇所あってそこの部分だけを「黄色の小さい文字」に変えたい時HTMLだけで作ってある場合は、その１０箇所を一箇所づつ変えなければなりませんが、「２」「３」の様にCSSで指定していた場合は

CSSの一箇所だけを変えれば１０箇所全部が変更可能になるのです。

また、HTMLだけでもＨＰは作れます。CSSを使うことでプログラム効率も上がると言う事です。

当然必要最低限だけCSSを使う事もできます。

（これらの情報は、インターネットで検索すれば詳しく解説してあります。）

CSS,スタイルシートの詳細については市販の解説書をご用意して下さい。本解説書はプログラミングについての解説ですので、CSS,スタイルシートについては、必要最小限にとどめています。

**スタイルの定義位置の検索の仕方**

まずメニュー表示部分のコードを見てみます。下記

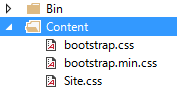
|  |
| --- |
| <ul class="nav navbar-nav">  <li><a runat="server" href="~/">ホーム</a></li>  <li><a runat="server" href="~/TestHairetsu.aspx">配列テスト</a></li>  <li><a runat="server" href="~/About">詳細</a></li>  <li><a runat="server" href="~/Contact">問い合わせ</a></li>  </ul> |

<ul> <li> </ul>によって、箇条書きを指定しています、<ul class=”……..”>によって箇条書きのスタイルを指定しています。この場合<li></li>の間の項目を横並びに指定しています。

そのスタイルとは、前項のマスターページでのCSSファイルのリンクにより“bootstrap.css”ファイル内に定義されています。その定義されている場所の見つけ方を説明します。

まず、<ul class=”nav navbar-nav”>の“navbar-nav”という文字だけをコピー状態にします、

次に、ソリューションエクスプローラのツリー構造の中の、Site.masterでリンク先をしていた、“~/Content/css”を探します。下図の位置にありました。



上図の“bootstrap.css”を選択して、CSSファイルの中身を表示します。

次に、VSのメニューバーの「編集」メニューをクリックして、表示されたドロップダウンメニューの上から二番目の“検索と置換”にマウスを乗せて一番上の“クイック検索”をクリックします、下図のように“navbar-nav”と貼り付けか、または入力して、右矢印をクリックして、検索を開始します、下図



すると次図のように、その文字が使用されているのコードの位置にジャンプします。

|  |
| --- |
|  |

上記のオレンジ色に選択されたブロック（｛～｝の間）の内容が、下記の、マスターページ内のhtmコード　<ul　のスタイルが定義されている位置になります。

|  |
| --- |
| <div class="navbar-collapse collapse">  <ul class="nav navbar-nav">  <li><a runat="server" href="~/">ホーム</a></li>  <li><a runat="server" href="~/About">詳細</a></li>  <li><a runat="server" href="~/Contact">問い合わせ</a></li>  </ul> |

ここまでの説明で、

1. 外部CSSのリンクの記述位置
2. 外部CSSのソリューションエクスプローラ内のフォルダの位置とファイルの位置
3. CSSファイルの開き方
4. CSSファイル内のスタイルの定義位置の検索の仕方

等を実践しました。尚CSSの詳細（スタイルの定義）等は、インターネットの情報などで、知識を習得して下さい。図書も本屋さんで、数多く出版されています。

**HTMLの<a>~</a>要素を理解する**

Webページのhtmlコードを見ると、コードビューの１行目から<>で囲まれた文の連続で構成されています。例えば<head runat=”server”></head>,<body></body>等という記法で記述されています。<body>~</body>の内にも複数の<>~</>の記述があります、この<>~</>の事を要素と呼びます。<body>~</body>の呼び名を“<body>要素”または”<body>~</body>”要素という呼び方をします。これ以後チュートリアルを進めるのに”<a>要素“だけはここで理解しておきましょう。Site.masterのメニュー表示のマークアップの中で次のような行があります。

<li><a runat="server" href="~/About">詳細</a></li>　を直訳しますと、

「箇条書きとして表示される“詳細”という文字は主ルートの“About.aspx”のページへ遷移する」という意味になります。

[runat=”server”]　はサーバー側で実行、「href」は遷移先、「About」はaspxという拡張子を省略したWebページ名です。この<a>要素は頻繁にでてきますが、記法を暗記する必要はありません、様々なシーンで使い分ける必要があります。要は「<a …..href=”…”>」の形によって、ページ遷移を実行できる（ハイバーリンク）だと記憶しておいて下さい。

## これだけは、ながめておこう！！

### If文とは

|  |
| --- |
| If 条件式 Then  　　 処理 End If |
| ' ElseIfとElseを書く場合 If 条件式 Then  　　 処理 ElseIf 条件式 Then   　　 処理 Else  　　処理 End If |

* 条件が合致する場合、配下の処理が実行され、if文の処理は終了します。
* 条件が合致することをtrue(真)ともいいます。
* 条件が合致しないことをfalse(偽)ともいいます。
* 条件が合致しない場合、配下の処理は実行されず、その下(ElseIf/Else/if文終了)に進みます。
* ElseIfは、複数記述できます。省略可です。
* Elseの処理は、どの条件も合致しなかったときに実行されます。Elseは１つのみ記述できます。省略可能です。

### If文のサンプル

Module Module1

Sub Main()

Dim a As Integer = 1

If a = 0 Then

Console.WriteLine("0")

ElseIf a = 1 Then

Console.WriteLine("1") '1が表示される

Else

Console.WriteLine("other")

End If

End Sub

End Module

5行目からif文が始まります。  
8行目のElseIf(if文)はtrueになるのでその配下の処理が実行されます。  
9行目の処理が実行されif文を抜けます。

**String型の値の比較**

|  |
| --- |
| Public Shared Operator =(a As [String], b As [String]) As Boolean |

string型の値の比較は、==で可能です。

Module Module1

Sub Main()

Dim a As String = "test"

Dim b As String = "test"

If a = b Then

Console.WriteLine("true") 'trueが出力される

End If

End Sub

End Module

String型の値を比較するときも比較演算子の=で可能です。  
Stringクラスにメソッドがあります。  
<https://docs.microsoft.com/ja-jp/dotnet/api/system.string.op_equality?view=netcore-3.1>

**ネスト構造 if文の中にif文**

Module Module1

Sub Main()

Dim a As Integer = 1

Dim b As Integer = 1

If a = 1 Then

If b = 1 Then

Console.WriteLine("1") ' 1が出力される

End If

End If

End Sub

End Module

7行目は、if文の中にif文を記述しています。ネストの構造です。入れ子ともいいます。  
6,7行目は、trueになり文字列が表示されます。

**値の一部で比較する(like演算子)**

like演算子とアスタリスク(\*)で、前方一致、部分一致、後方一致の比較ができます。

Module Module1

Sub Main()

Dim str1 As String

str1 = "みかん"

If str1 Like "み\*" Then

Console.WriteLine("OK") 'OKが表示される

End If

If str1 Like "み\*ん" Then

Console.WriteLine("OK") 'OKが表示される

End If

If str1 Like "\*かん" Then

Console.WriteLine("OK") 'OKが表示される

End If

If str1 Like "みかん\*" Then

Console.WriteLine("OK") 'OKが表示される

End If

End Sub

End Module

アスタリスク(\*)は、0以上の任意の文字を表します。

**null(nothing)を比較**

Module Module1

Sub Main()

Dim a As String = Nothing

If a Is Nothing = True Then

Console.Write("true") ' trueが出力される

End If

If IsNothing(a) Then

Console.Write("true") ' trueが出力される

End If

If a IsNot Nothing = True Then

Console.Write("true3") '出力されない

End If

If String.IsNullOrEmpty(a) Then

Console.Write("true") ' trueが出力される

End If

End Sub

End Module

13行目は、Nothingの否定です。IsNotを使用します。  
17行目のIsNullOrEmptyは、nullと空文字のときにtrueになります。

以下は、MicrosoftのNot演算子のリンクです。  
<https://docs.microsoft.com/ja-jp/dotnet/visual-basic/language-reference/operators/not-operator>

**三項演算子**

|  |
| --- |
| 変数 = if(条件式, 値1, 値2) |

* 条件式がtrueのとき、値1を返します。
* 条件式がfalseのとき、値2を返します。

Module Module1

Sub Main()

Dim a As Integer = 1

Dim b As String

b = If(a = 0, "true", "false")

Console.WriteLine(b) ' falseが表示される

If a = 0 Then

Console.WriteLine("true")

Else

Console.WriteLine("false") 'falseが表示される

End If

End Sub

End Module

6行目は、三項演算子です。  
6,7行目をif文で書くと、9～13行目のようになります。

**比較演算子**

比較演算子は、2つの値を比べて条件を満たしていればtrue,そうでない場合はfalseを返します。

等しくないとする場合は、<>とします。

|  |  |
| --- | --- |
| 比較演算子 | 説明 |
| a = b | aとbの値は等しい |
| a <> b | aとbの値は等しくない |
| a > b | aはbより大きい |
| a >= b | aはbより大きい または 等しい |
| a < b | aはbより小さい |
| a <= b | aはbより小さい または 等しい |

以下は、Microsoftの比較演算子 (Visual Basic)のリンクです。  
<https://docs.microsoft.com/ja-jp/dotnet/visual-basic/language-reference/operators/comparison-operators>

**論理積(And) AかつB**

|  |
| --- |
| If  左の条件式 And 右の条件式 Then |
| If  左の条件式 AndAlso 右の条件式 Then |

* 論理演算子のAndとAndAlsoは、左の式と右の式が両方ともtrueの場合、trueになります。それ以外はfalseです。
* AndAlsoは、左の式がfalseの場合、右の式は判定(評価)されません。これを短絡評価(ショートサーキット評価 : Short-Circuit Evaluation)といいます。パフォーマンスの向上を図れます。

Module Module1

Sub Main()

Dim a As Integer = 1

Dim b As Integer = 1

If a = 1 And b = 1 Then

Console.WriteLine("True") 'Trueが出力される

Else

Console.WriteLine("False")

End If

If a = 1 AndAlso b = 1 Then

Console.WriteLine("True") 'Trueが出力される

Else

Console.WriteLine("False")

End If

End Sub

End Module

6,12行目は、左の式と右の式が共にtrueなので、trueになります。  
12行目は、左の式がfalseの場合、右の式は判定されません。

**論理和(Or) AまたはB**

|  |
| --- |
| If  左の条件式 Or 右の条件式 Then |
| If  左の条件式 OrElse 右の条件式 Then |

* 論理演算子のOrとOrElseは、左の式または右の式のどちらかがtrueの場合、trueになります。
* OrElseは、左の式がtrueの場合、右の式は判定(評価)されません。これを短絡評価(ショートサーキット評価 : Short-Circuit Evaluation)といいます。パフォーマンスの向上を図れます。

Module Module1

Sub Main()

Dim a As Integer = 1

Dim b As Integer = 2

If a = 2 Or b = 2 Then

Console.WriteLine("True") 'Trueが出力される

Else

Console.WriteLine("False")

End If

If a = 2 OrElse b = 2 Then

Console.WriteLine("True") 'Trueが出力される

Else

Console.WriteLine("False")

End If

End Sub

End Module

6,12行目は、左の式はfalseですが右の式がtrueなのでtrueになります。  
12行目は、左の式がtrueの場合、右の式は判定されません。

**排他的論理和(Xor)**

|  |
| --- |
| If  左の条件式 Xor 右の条件式 Then |

* 左の式がtrueで右の式がfalseの場合、trueになります。
* 左の式がfalseで右の式がtrueの場合、trueになります。
* 左の式と右の式が両方ともtrueの場合、falseになります。
* 左の式と右の式が両方ともfalseの場合、falseになります。

Module Module1

Sub Main()

Dim a As Integer = 2

Dim b As Integer = 2

If a = 2 Xor b = 3 Then

Console.WriteLine("True") 'Trueが出力される

Else

Console.WriteLine("False")

End If

If a = 3 Xor b = 2 Then

Console.WriteLine("True") 'Trueが出力される

Else

Console.WriteLine("False")

End If

If a = 2 Xor b = 2 Then

Console.WriteLine("True")

Else

Console.WriteLine("False") 'Falseが出力される

End If

If a = 3 Xor b = 3 Then

Console.WriteLine("True")

Else

Console.WriteLine("False") 'Falseが出力される

End If

End Sub

End Module

6行目は、左の式がtrueで右の式がfalseなので、trueになります。  
12行目は、左の式がfalseで右の式がtrueなので、trueになります。  
18行目は、左の式と右の式が両方ともtrueなので、falseになります。  
24行目は、左の式と右の式が両方ともfalseなので、falseになります。

**論理否定(Not) Aではない**

|  |
| --- |
| If  Not (条件式)  Then |

* 条件式がfalseの場合にtrueになります。
* 条件式がtrueの場合にfalseになります。

Module Module1

Sub Main()

Dim a As Integer = 1

If Not (a = 2) Then

Console.WriteLine("True") 'Trueが出力される

End If

If Not (a = 1) Then

Console.WriteLine("True")

Else

Console.WriteLine("False") 'Falseが出力される

End If

If a <> 2 Then

Console.WriteLine("True") 'Trueが出力される

End If

End Sub

End Module

5行目は値の比較でfalseになり、それを否定するのでtrueになります。  
9行目は値の比較でtrueになり、それを否定するのでfalseになります。  
15行目の<>でも否定になります。

以下は、Microsoftの論理演算子のリンクです。  
<https://docs.microsoft.com/ja-jp/dotnet/visual-basic/programming-guide/language-features/operators-and-expressions/logical-and-bitwise-operators>

**関連の記事**