# Programming ASP .Net web pages

**Kunikeyev Aidyn Dauletovich**

# Data types

| .NET | C# | VB.NET | DESCRIPTION |
|---|---|---|---|
| System.Byte | byte | Byte | Used to store small, positive whole numbers from 0 to 255. Defaults to 0 when no value is assigned explicitly. |
| System.Int16 | short | Short | Capable of storing whole numbers between –32,768 and 32,767. Defaults to 0. |
| System.Int32 | int | Integer | Capable of storing whole numbers between –2,147,483,648 and 2,147,483,647. Defaults to 0. |
| System.Int64 | long | Long | Holds whole large numbers between –9,223,372,036,854,775,808 and 9,223,372,036,854,775,807. Defaults to 0. |
| System.Single | float | Single | Stores large numbers with decimals between –3.4028235E+38 and 3.4028235E+38. Defaults to 0.0. |
| System.Double | double | Double | Can hold large fractional numbers. It's not as accurate as the Decimal when it comes to the fractional numbers but when extreme accuracy is not a requirement, you should prefer the Double over the Decimal, because the Double is a little faster. Defaults to 0.0. |
| System.Decimal | decimal | Decimal | Stores extremely large fractional numbers with a high accuracy. Defaults to 0. This data type is often used to store monetary values. |
| System.Boolean | bool | Boolean | Used to hold a simple boolean value: True or False in VB, and true or false in C#. Defaults to False. |

| .NET | C# | VB.NET | DESCRIPTION |
|---|---|---|---|
| System.DateTime | n/a | Date | VB.NET has an alias for the System.DateTime data type to store date and time values. C# doesn't define an alias for this type. Defaults to 1/1/0001: 12:00 a.m. |
| System.Char | char | Char | Holds a single character. Defaults to Nothing (null in C#). |
| System.String | string | String | Can hold text with a length of up to 2 billion characters. Defaults to Nothing (null in C#). |
| System.SByte | sbyte | SByte | Used to store small numbers from –128 to 127. Defaults to 0. |
| System.UInt16 | ushort | UShort | Similar to a System.Int16, but this data type can only store unsigned whole numbers, between 0 and 65,535. Defaults to 0. The other data types prefixed with a U are all unsigned as well. |
| System.UInt32 | uint | UInteger | Capable of storing whole numbers between 0 and 4,294,967,295. Defaults to 0. |
| System.UInt64 | ulong | ULong | Capable of storing whole numbers between 0 and 18,446,744,073,709,551,615. Defaults to 0. |
| System.Object | object | Object | The parent of all data types in .NET, including the CTS types and types you define yourself. Each data type is also an object, as you learn later in the book. Defaults to Nothing (null in C#). |

# Working with variables and objects. Converting and Casting Data Types.

- `<Data type name> <variable name> = <value>;`
- `<Class name> <variable name> = new <Class name>;`

- `Object a = 1; int b = (int) a;` **OK**
- `Object a = "a"; int b = (int) a;` **NOT OK**
- Class `Convert` has methods to convert objected to other data type:
  - `Convert.ToInt32();`
  - `Convert.ToBoolean();`

# Using Arrays, Collections and Generics.

```
• string[] roles = new string[2];
• roles[0] = "Administrators";
• roles[1] = "ContentManagers";
• Array.Resize(ref roles, 3);
• roles[2] = "Members";
```

```
• ArrayList roles = new
  ArrayList();
• roles.Add("Administrators");
```

```
• List<string> roles = new
  List<string>();
• roles.Add("Administrators");
• List<int> intList = new
  List<int>();
```

# Operators

| VB.NET | C# | USAGE |
|---|---|---|
| + | + | Adds two values to each other |
| – | – | Subtracts one value from another |
| * | * | Multiplies two values |
| / | / | Divides two values |
| \ | n/a | Divides two values but always returns a rounded integer |
| ^ | n/a | Raises one value to the power of another |
| Mod | % | Divides two whole numbers and returns the remainder |

```
someNumber1 += 3;
someNumber2 -= 3;
someNumber3 *= 3;
someNumber4 /= 3;
```

# Comparison Operators

| VB.NET | C# | USAGE |
|---|---|---|
| = | == | Checks if two values are equal to each other |
| <> | != | Checks if two values are not equal |
| < | < | Checks if the first value is less than the second |
| > | > | Checks if the first value is greater than the second |
| <= | <= | Checks if the first value is less than or equal to the second |
| >= | >= | Checks if the first value is greater than or equal to the second |
| Is | is | In VB.NET: Compares two objects; In C#: Checks if a variable is of a certain type |

# Logical Operators

| VB.NET | C# | USAGE |
|---|---|---|
| And | & | Returns True when both expressions result in a True value. |
| Or | \| | Returns True if at least one expression results in a True value. |
| Not | ! | Reverses the outcome of an expression. |
| AndAlso | && | Enables you to short-circuit your logical And condition checks. |
| OrElse | \|\| | Enables you to short-circuit your logical Or condition checks. |

# If and If Else Constructs

```
if (User.IsInRole("Administrators") == true)
{
    DeleteButton.Visible = true;
}
```

```
if (User.IsInRole("Administrators"))
{
    DeleteButton.Visible = true;
}
else
{
    DeleteButton.Visible = false;
}
```

# Select Case/switch Constructs

```
switch (today.DayOfWeek)
{
    case DayOfWeek.Monday:
        discountPercentage = 40;
    break;

    default:
        discountPercentage = 0;
    break;
}
```

# Loops

```
for (startCondition; endCondition; step definition)
{
    // Code that must be executed    for each iteration
}
```

```
while (!success)
{
    success = SendEmailMessage();
}
```

```
foreach (string role in roles)
{
    Label1.Text += role + "<br />";
}
```

# Object and Constructors

```
public class Person
{
    public Person(string firstName, string lastName, DateTime dateOfBirth)
    {
        _firstName = firstName;
        _lastName = lastName;
        _dateOfBirth = dateOfBirth;
    }
}

Person myPerson = new Person("Imar", "Spaanjaars", new DateTime(1971, 8, 9));
```

Constructors are special methods in a class that help you create an instance of your object.

# Methods: Functions and Subroutines

```
// Define a function
public DataType FunctionName([parameterList])
{
    return DataType
}


// Define a subroutine
public void SubName([parameterList])
{
}
```

# Methods: Functions and Subroutines

```
public class Person
{
    private string _firstName;

    public string FirstName
    {
        get { return _firstName; }
        set { _firstName = value; }
    }
}
```

```
Person myPerson = new Person();
myPerson.FirstName = "imar";
```

If you want to make read-only field: just include `get`
If you want to make write-only field: just include `set`

# Events

```
protected void Button1_Click(object sender, EventArgs e)
{
}
```

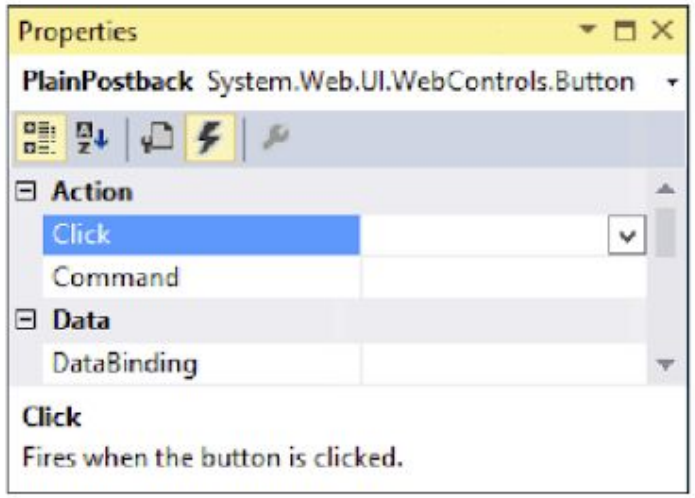


FIGURE 5-9

# The end

- Laboratory work #3
- All tasks on chapter #5, Imar Spaanjaars