# String Manipulation

## Basic Operations

**Concatenation:**

Dim result As String = "Hello " & "World"

**String Length:**

Dim length As Integer = myString.Length

**Substring:**

Dim subString As String = myString.Substring(startIndex, length)

' Example: "Hello".Substring(0, 2) gives "He"

**Index of a Character or Substring:**

Dim index As Integer = myString.IndexOf("text")

' Returns -1 if not found

**Last Index of a Character or Substring:**

Dim lastIndex As Integer = myString.LastIndexOf("text")

**Replace Substring:**

Dim replacedString As String = myString.Replace("oldText", "newText")

## Changing Case

**To Upper Case:**

Dim upperCase As String = myString.ToUpper()

**To Lower Case:**

Dim lowerCase As String = myString.ToLower()

**Capitalise First Letter:**

Dim capitalised As String = Char.ToUpper(myString(0)) & myString.Substring(1).ToLower()

## Trimming Strings

**Trim Leading and Trailing Whitespace:**

Dim trimmed As String = myString.Trim()

**Trim Leading Whitespace:**

Dim leftTrimmed As String = myString.TrimStart()

**Trim Trailing Whitespace:**

Dim rightTrimmed As String = myString.TrimEnd()

## Splitting and Joining

**Split String:**

Dim splitArray As String() = myString.Split(","c)

' Splits by comma; use "c" to specify character

**Join Array to String:**

Dim joinedString As String = String.Join(",", splitArray)

## Checking String Content

**Contains Substring:**

Dim contains As Boolean = myString.Contains("text")

**Starts with Substring:**

Dim startsWith As Boolean = myString.StartsWith("text")

**Ends with Substring:**

Dim endsWith As Boolean = myString.EndsWith("text")

## Comparing Strings

**Equals:**

Dim isEqual As Boolean = myString.Equals("text")

**Compare (Case-Insensitive):**

Dim compareResult As Integer = String.Compare(myString, "text", True)

' Returns 0 if equal, negative if myString < "text", positive if myString > "text"

## Finding and Extracting

**Get Character at Position:**

Dim charAt As Char = myString(2)

**Extract Number from String:**

Dim digits As String = New String(myString.Where(Function(c) Char.IsDigit(c)).ToArray())

## Formatting Strings

**String Format:**

Dim formatted As String = String.Format("Hello {0}, welcome to {1}", "User", "UiPath")

**Interpolated String:**

Dim name As String = "User"

Dim place As String = "UiPath"

Dim interpolated As String = $"Hello {name}, welcome to {place}"

## Regular Expressions

**Match:**

Dim match As Match = Regex.Match(myString, "pattern")

**Matches:**

Dim matches As MatchCollection = Regex.Matches(myString, "pattern")

**Replace using Regex:**

Dim replaced As String = Regex.Replace(myString, "pattern", "replacement")

**Split using Regex:**

Dim splitArray As String() = Regex.Split(myString, "pattern")

## String Conversion

**Convert String to Integer:**

Dim number As Integer = Convert.ToInt32(myString)

**Convert Integer to String:**

Dim strNumber As String = myInt.ToString()

**Convert String to DateTime:**

Dim dateValue As DateTime = DateTime.Parse(myString)

**Convert DateTime to String:**

Dim dateString As String = myDateTime.ToString("yyyy-MM-dd")

## Advanced String Manipulation

**Reverse String:**

Dim reversedString As String = New String(myString.Reverse().ToArray())

**Remove Characters:**

Dim result As String = myString.Remove(startIndex, length)

**Insert Substring:**

Dim result As String = myString.Insert(startIndex, "insertText")

# Datatables

## Creating a DataTable

**Empty DataTable:**

new DataTable()

**DataTable with Columns:**

new DataTable("TableName")

dt.Columns.Add("ColumnName", GetType(DataType))

## Adding Data to DataTable

**Add DataRow:**

dt.Rows.Add("Value1", "Value2", "Value3")

**Add DataRow with Array:**

dt.Rows.Add({Value1, Value2, Value3})

**Create a new DataRow:**

Dim newRow As DataRow = dt.NewRow()

newRow("ColumnName1") = "Value1"

newRow("ColumnName2") = "Value2"

dt.Rows.Add(newRow)

## Selecting Data

**Select Rows**:

dt.Select("ColumnName = 'Value'")

**Select with Multiple Conditions**:

dt.Select("ColumnName1 = 'Value1' AND ColumnName2 > 5")

**Select with LIKE (for partial matches)**:

dt.Select("ColumnName LIKE '%Value%'")

**LINQ Query to Select Rows:**

dt.AsEnumerable().Where(Function(row) row("ColumnName").ToString = "Value")

## Filtering DataTable

**Filter and Copy to New DataTable**:

dt.Select("ColumnName = 'Value'").CopyToDataTable()

**Filter with LINQ and Copy to New DataTable**:

dt.AsEnumerable().Where(Function(row) row("ColumnName").ToString = "Value").CopyToDataTable()

## Modifying DataTable

**Update a Cell**:

dt.Rows(RowIndex)("ColumnName") = "NewValue"

**Delete a Row**:

dt.Rows(RowIndex).Delete()

**Remove Rows based on Condition**:

For Each row As DataRow In dt.Select("ColumnName = 'Value'")

dt.Rows.Remove(row)

## Looping through DataTable

**For Each Row in DataTable**:

For Each row As DataRow In dt.Rows

' Access column data as row("ColumnName")

**For Each with LINQ**:

dt.AsEnumerable().ToList().ForEach(Sub(row)

' Perform operations on each row

End Sub)

## Sorting DataTable

**Sort DataTable**:

dt = dt.Select("", "ColumnName ASC").CopyToDataTable()

**Sort DataTable with Multiple Columns**:

dt = dt.Select("", "ColumnName1 ASC, ColumnName2 DESC").CopyToDataTable()

## Merging DataTables

**Merge Two DataTables**:

dt1.Merge(dt2)

## Cloning and Copying DataTable

**Clone DataTable Structure (without data)**:

dt.Clone()

**Copy DataTable with Data**:

dt.Copy()

## Aggregating Data

**Compute Sum**:

dt.Compute("SUM(ColumnName)", "")

**Compute Average**:

dt.Compute("AVG(ColumnName)", "")

**Count Rows with Condition**:

dt.Compute("COUNT(ColumnName)", "ColumnName > 5")

## Checking for Duplicates

**Check for Duplicate Rows**:

dt.DefaultView.ToTable(True, "ColumnName")

## Converting DataTable to String

**Join DataTable Rows to String (for a single column)**:

String.Join(",", dt.AsEnumerable().[Select](Function(row) row("ColumnName").ToString()).ToArray())

**Concatenate all Rows and Columns:**

String.Join(Environment.NewLine, dt.AsEnumerable().[Select](Function(row) String.Join(",", row.ItemArray)).ToArray())

## Adding Columns with Calculated Values

**Add Calculated Column**:

Dim dt As New DataTable

dt.Columns.Add("OriginalColumn", GetType(Integer))

dt.Columns.Add("CalculatedColumn", GetType(Integer), "OriginalColumn \* 2")

## Renaming Columns

**Rename a Column**:

dt.Columns("OldColumnName").ColumnName = "NewColumnName"

## Exporting DataTable

**To CSV**:

dt.WriteToCSVFile("path\to\file.csv")

**To Excel**:

dt.WriteToExcelFile("path\to\file.xlsx", "SheetName")

# Invoke Code Versus Invoke Method

* using a **Target Object**, you can call an instance method on the object provided
* using a **Target Type**, you can call a static method of the provided type

**List of methods**:

<https://chatgpt.com/share/007c794b-9566-43ff-b5d4-f90717407789>

Both approaches achieve the same result, but **Invoke Code** is typically more versatile for larger blocks of code, while **Invoke Method** is more direct and structured for calling specific methods.

# Global Exception Handler - Done

* Add global exception handler
* Configure the settings
* Add the process that manages errors

# Testing – Done

* Verify Expression
* Verify Expression with Operator
* Verify Range
* Verify Control Attribute

# Integration Services – Done

* Trigger
* Connectors
* Connections
* Events