Namespace BigExcelCreator

Classes

BigExcelWriter

This class writes Excel files directly using OpenXML SAX. Useful when trying to write tens of thousands of rows.

Class BigExcelWriter

Namespace: BigExcelCreator

Assembly: BigExcelCreator.dll

This class writes Excel files directly using OpenXML SAX. Useful when trying to write tens of thousands of rows.

```
public class BigExcelWriter : IDisposable
```

Inheritance

object ← BigExcelWriter

Implements

IDisposable

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

NuGet

Source

API

Site

Constructors

BigExcelWriter(Stream)

Initializes a new instance of the BigExcelWriter class with the specified stream and spreadsheet document type.

```
public BigExcelWriter(Stream stream)
```

Parameters

`stream` Stream

The stream to write the Excel document to.

Remarks

Initializes a new Workbook

BigExcelWriter(Stream, Stylesheet)

Initializes a new instance of the BigExcelWriter class with the specified stream, spreadsheet document type, and stylesheet.

```
public BigExcelWriter(Stream stream, Stylesheet stylesheet)
```

Parameters

`stream` Stream

The stream to write the Excel document to.

`stylesheet` Stylesheet

The stylesheet to apply to the Excel document. See GetStylesheet().

Remarks

Initializes a new Workbook

BigExcelWriter(Stream, SpreadsheetDocumentType)

Initializes a new instance of the BigExcelWriter class with the specified stream and spreadsheet document type.

```
public BigExcelWriter(Stream stream, SpreadsheetDocumentType
spreadsheetDocumentType)
```

Parameters

`stream` Stream

The stream to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

BigExcelWriter(Stream, SpreadsheetDocumentType, Stylesheet)

Initializes a new instance of the BigExcelWriter class with the specified stream, spreadsheet document type, and stylesheet.

```
public BigExcelWriter(Stream stream, SpreadsheetDocumentType
spreadsheetDocumentType, Stylesheet stylesheet)
```

Parameters

`stream` Stream

The stream to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

`stylesheet` Stylesheet

The stylesheet to apply to the Excel document. See GetStylesheet().

BigExcelWriter(Stream, SpreadsheetDocumentType, bool)

Initializes a new instance of the BigExcelWriter class with the specified stream, spreadsheet document type, and a flag indicating whether to skip cells when they are empty.

```
public BigExcelWriter(Stream stream, SpreadsheetDocumentType
spreadsheetDocumentType, bool skipCellWhenEmpty)
```

Parameters

`stream` Stream

The stream to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

`skipCellWhenEmpty` bool

A flag indicating whether to skip cells when they are empty. When true, writing an empty value to a cell moves the next cell to be written. When false, writing an empty value to a cell does nothing.

BigExcelWriter(Stream, SpreadsheetDocumentType, bool, Stylesheet)

Initializes a new instance of the BigExcelWriter class with the specified stream, spreadsheet document type, a flag indicating whether to skip cells when they are empty, and a stylesheet.

```
public BigExcelWriter(Stream stream, SpreadsheetDocumentType
spreadsheetDocumentType, bool skipCellWhenEmpty, Stylesheet stylesheet)
```

Parameters

`stream` Stream

The stream to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

`skipCellWhenEmpty` bool

A flag indicating whether to skip cells when they are empty. When true, writing an empty value to a cell moves the next cell to be written. When false, writing an empty value to a cell does nothing.

`stylesheet` Stylesheet

The stylesheet to apply to the Excel document. See GetStylesheet().

BigExcelWriter(Stream, bool)

Initializes a new instance of the BigExcelWriter class with the specified stream, spreadsheet document type, and a flag indicating whether to skip cells when they are empty.

```
public BigExcelWriter(Stream stream, bool skipCellWhenEmpty)
```

Parameters

`stream` Stream

The stream to write the Excel document to.

`skipCellWhenEmpty` bool

A flag indicating whether to skip cells when they are empty. When true, writing an empty value to a cell moves the next cell to be written. When false, writing an empty value to a cell does nothing.

Remarks

Initializes a new Workbook

BigExcelWriter(string)

Initializes a new instance of the BigExcelWriter class with the specified file path and spreadsheet document type.

```
public BigExcelWriter(string path)
```

Parameters

`path` string

The file path to write the Excel document to.

Remarks

Initializes a new Workbook

BigExcelWriter(string, Stylesheet)

Initializes a new instance of the BigExcelWriter class with the specified file path, spreadsheet document type, and stylesheet. Initializes a new Workbook

```
public BigExcelWriter(string path, Stylesheet stylesheet)
```

Parameters

`path` string

The file path to write the Excel document to.

`stylesheet` Stylesheet

The stylesheet to apply to the Excel document. See GetStylesheet()

BigExcelWriter(string, SpreadsheetDocumentType)

Initializes a new instance of the BigExcelWriter class with the specified file path and spreadsheet document type.

```
public BigExcelWriter(string path, SpreadsheetDocumentType spreadsheetDocumentType)
```

Parameters

`path` string

The file path to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

BigExcelWriter(string, SpreadsheetDocumentType, Stylesheet)

Initializes a new instance of the BigExcelWriter class with the specified file path, spreadsheet document type, and stylesheet.

```
public BigExcelWriter(string path, SpreadsheetDocumentType spreadsheetDocumentType,
Stylesheet stylesheet)
```

Parameters

`path` string

The file path to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

`stylesheet` Stylesheet

The stylesheet to apply to the Excel document. See GetStylesheet()

BigExcelWriter(string, SpreadsheetDocumentType, bool)

Initializes a new instance of the BigExcelWriter class with the specified file path, spreadsheet document type, and a flag indicating whether to skip cells when they are empty.

```
public BigExcelWriter(string path, SpreadsheetDocumentType spreadsheetDocumentType,
bool skipCellWhenEmpty)
```

Parameters

`path` string

The file path to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

`skipCellWhenEmpty` bool

A flag indicating whether to skip cells when they are empty. When true, writing an empty value to a cell moves the next cell to be written. When false, writing an empty value to a cell does nothing.

BigExcelWriter(string, SpreadsheetDocumentType, bool, Stylesheet)

Initializes a new instance of the BigExcelWriter class with the specified file path, spreadsheet document type, a flag indicating whether to skip cells when they are empty, and a stylesheet.

```
public BigExcelWriter(string path, SpreadsheetDocumentType spreadsheetDocumentType,
bool skipCellWhenEmpty, Stylesheet stylesheet)
```

Parameters

`path` string

The file path to write the Excel document to.

`spreadsheetDocumentType` SpreadsheetDocumentType

The type of the spreadsheet document (e.g., Workbook, Template).

`skipCellWhenEmpty` bool

A flag indicating whether to skip cells when they are empty. When true, writing an empty value to a cell moves the next cell to be written. When false, writing an empty value to a cell does nothing.

`stylesheet` Stylesheet

The stylesheet to apply to the Excel document. See GetStylesheet().

BigExcelWriter(string, bool)

Initializes a new instance of the BigExcelWriter class with the specified file path, spreadsheet document type, and a flag indicating whether to skip cells when they are

empty.

```
public BigExcelWriter(string path, bool skipCellWhenEmpty)
```

Parameters

`path` string

The file path to write the Excel document to.

`skipCellWhenEmpty` bool

A flag indicating whether to skip cells when they are empty. When true, writing an empty value to a cell moves the next cell to be written. When false, writing an empty value to a cell does nothing.

Remarks

Initializes a new Workbook

Properties

Document

Gets the SpreadsheetDocument object representing the Excel document.

```
public SpreadsheetDocument Document { get; }
```

Property Value

SpreadsheetDocument

Path

Gets the file path where the Excel document is being saved.

(null when not saving to file)

```
public string Path { get; }
```

Property Value

string

PrintGridLinesInCurrentSheet

Gets or sets a value indicating whether to print grid lines in the current sheet.

```
public bool PrintGridLinesInCurrentSheet { get; set; }
```

Property Value

bool

Remarks

When true, Prints gridlines. When false, Doesn't print gridlines (default).

Exceptions

NoOpenSheetException

When there is no open sheet

PrintRowAndColumnHeadingsInCurrentSheet

Gets or sets a value indicating whether to print row and column headings in the current sheet.

```
public bool PrintRowAndColumnHeadingsInCurrentSheet { get; set; }
```

Property Value

bool

Remarks

When true, Prints row and column headings. When false, Doesn't print row and column headings (default).

Exceptions

NoOpenSheetException

When there is no open sheet

ShowGridLinesInCurrentSheet

Gets or sets a value indicating whether to show grid lines in the current sheet.

```
public bool ShowGridLinesInCurrentSheet { get; set; }
```

Property Value

bool

Remarks

When true, shows gridlines on screen (default). When false, hides gridlines on screen.

Exceptions

NoOpenSheetException

When there is no open sheet

ShowRowAndColumnHeadingsInCurrentSheet

Gets or sets a value indicating whether to show row and column headings in the current sheet.

```
public bool ShowRowAndColumnHeadingsInCurrentSheet { get; set; }
```

Property Value

bool

Remarks

When true, shows row and column headings (default). When false, hides row and column headings.

Exceptions

NoOpenSheetException

When there is no open sheet

SkipCellWhenEmpty

Gets or sets a value indicating whether to skip cells when they are empty.

```
public bool SkipCellWhenEmpty { get; set; }
```

Property Value

bool

Remarks

When true, writing an empty value to a cell moves the next cell to be written. When false, writing an empty value to a cell does nothing.

SpreadsheetDocumentType

Gets the type of the spreadsheet document (e.g., Workbook, Template).

only `SpreadsheetDocumentType.Workbook` is tested

```
public SpreadsheetDocumentType SpreadsheetDocumentType { get; }
```

Property Value

SpreadsheetDocumentType

Stream

Gets the Stream where the Excel document is being saved.

(null when not saving to Stream)

```
public Stream Stream { get; }
```

Property Value

Stream

Methods

AddAutofilter(CellRange, bool)

Adds an autofilter to the specified range in the current sheet.

```
public void AddAutofilter(CellRange range, bool overwrite = false)
```

Parameters

`range` CellRange

The range where the autofilter should be applied.

`overwrite` bool

If set to `true`, any existing autofilter will be replaced.

Remarks

The range height must be 1.

Only one filter per sheet is allowed.

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to add the autofilter to.

ArgumentNullException

Thrown when the `range` is `null`.

SheetAlreadyHasFilterException

Thrown when there is already an autofilter in the current sheet and `overwrite` is `false`.

ArgumentOutOfRangeException

Thrown when the height of the `range` is not 1.

AddAutofilter(string, bool)

Adds an autofilter to the specified range in the current sheet.

```
public void AddAutofilter(string range, bool overwrite = false)
```

Parameters

`range` string

The range where the autofilter should be applied.

`overwrite` bool

If set to `true`, any existing autofilter will be replaced.

Remarks

The range height must be 1.

Only one filter per sheet is allowed.

Exceptions

InvalidRangeException

Thrown when the `range` does not represent a valid range.

NoOpenSheetException

Thrown when there is no open sheet to add the autofilter to.

ArgumentNullException

Thrown when the `range` is `null`.

SheetAlreadyHasFilterException

Thrown when there is already an autofilter in the current sheet and `overwrite` is `false`.

ArgumentOutOfRangeException

Thrown when the height of the `range` is not 1.

AddConditionalFormattingCellIs(CellRange, ConditionalFormattingOperatorValues, string, int, string)

Adds a conditional formatting rule based on a cell value to the specified cell range.

```
public void AddConditionalFormattingCellIs(CellRange cellRange,
ConditionalFormattingOperatorValues @operator, string value, int format, string
value2 = null)
```

Parameters

`cellRange` CellRange

The cell range to apply the conditional formatting to.

`operator` ConditionalFormattingOperatorValues

The operator to use for the conditional formatting rule.

`value` string

The value to compare the cell value against.

`format` int

The format ID of the differential format in stylesheet to apply when the condition is met. See GetIndexDifferentialByName(string)

`value2` string

The second value to compare the cell value against, used for "Between" and "NotBetween" operators.

Exceptions

ArgumentNullException

Thrown when `cellRange`, `value`, or `value2` (if required) is null.

ArgumentOutOfRangeException

Thrown when `format` is negative.

NoOpenSheetException

Thrown when there is no open sheet to add the conditional formatting to.

AddConditionalFormattingCellIs(string, ConditionalFormattingOperatorValues, string, int, string)

Adds a conditional formatting rule based on a cell value to the specified cell range.

```
public void AddConditionalFormattingCellIs(string reference,
ConditionalFormattingOperatorValues @operator, string value, int format, string
value2 = null)
```

Parameters

`reference` string

The cell range to apply the conditional formatting to.

`operator` ConditionalFormattingOperatorValues

The operator to use for the conditional formatting rule.

`value` string

The value to compare the cell value against.

`format` int

The format ID of the differential format in stylesheet to apply when the condition is met. See GetIndexDifferentialByName(string)

`value2` string

The second value to compare the cell value against, used for "Between" and "NotBetween" operators.

Exceptions

ArgumentNullException

Thrown when `reference`, `value`, or `value2` (if required) is null.

ArgumentOutOfRangeException

Thrown when `format` is negative.

NoOpenSheetException

Thrown when there is no open sheet to add the conditional formatting to.

InvalidRangeException

Thrown when the `reference` does not represent a valid range.

AddConditionalFormattingDuplicatedValues(CellRange, int)

Adds a conditional formatting rule to highlight duplicated values in the specified cell range.

```
public void AddConditionalFormattingDuplicatedValues(CellRange cellRange,
int format)
```

Parameters

`cellRange` CellRange

The cell range to apply the conditional formatting to.

`format` int

The format ID of the differential format in stylesheet to apply when the condition is met. See GetIndexDifferentialByName(string)

Exceptions

ArgumentNullException

Thrown when `cellRange` is null.

ArgumentOutOfRangeException

Thrown when `format` is negative.

NoOpenSheetException

Thrown when there is no open sheet to add the conditional formatting to.

AddConditionalFormattingDuplicatedValues(string, int)

Adds a conditional formatting rule to highlight duplicated values in the specified cell range.

```
public void AddConditionalFormattingDuplicatedValues(string reference, int format)
```

Parameters

`reference` string

The cell range to apply the conditional formatting to.

`format` int

The format ID of the differential format in stylesheet to apply when the condition is met. See GetIndexDifferentialByName(string)

Exceptions

ArgumentNullException

Thrown when `reference` is null.

ArgumentOutOfRangeException

Thrown when `format` is negative.

NoOpenSheetException

Thrown when there is no open sheet to add the conditional formatting to.

InvalidRangeException

Thrown when the `reference` does not represent a valid range.

AddConditionalFormattingFormula(CellRange, string, int)

Adds a conditional formatting rule based on a formula to the specified cell range.

```
public void AddConditionalFormattingFormula(CellRange cellRange, string formula,
int format)
```

Parameters

`cellRange` CellRange

The cell range to apply the conditional formatting to.

`formula` string

The formula that determines the conditional formatting rule.

`format` int

The format ID of the differential format in stylesheet to apply when the condition is met. See GetIndexDifferentialByName(string)

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to add the conditional formatting to.

ArgumentNullException

Thrown when `cellRange` or `formula` is null.

ArgumentOutOfRangeException

Thrown when `format` is negative.

AddConditionalFormattingFormula(string, string, int)

Adds a conditional formatting rule based on a formula to the specified cell range.

```
public void AddConditionalFormattingFormula(string reference, string formula,
int format)
```

Parameters

`reference` string

The cell range to apply the conditional formatting to.

`formula` string

The formula that determines the conditional formatting rule.

`format` int

The format ID of the differential format in stylesheet to apply when the condition is met. See GetIndexDifferentialByName(string)

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to add the conditional formatting to.

ArgumentNullException

Thrown when `reference` or `formula` is null.

ArgumentOutOfRangeException

Thrown when `format` is negative.

InvalidRangeException

Thrown when the `reference` does not represent a valid range.

AddDecimalValidator(CellRange, decimal, DataValidationOperatorValues, bool, bool, bool, decimal?)

Adds a decimal data validation to the specified cell range.

```
public void AddDecimalValidator(CellRange range, decimal firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, decimal? secondOperand
= null)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`firstOperand` decimal

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` decimal?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

AddDecimalValidator(CellRange, double, DataValidationOperatorValues, bool, bool, bool, double?)

Adds a decimal data validation to the specified cell range.

```
public void AddDecimalValidator(CellRange range, double firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, double? secondOperand = null)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`firstOperand` double

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` double?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

AddDecimalValidator(CellRange, float, DataValidationOperatorValues, bool, bool, bool, float?)

Adds a decimal data validation to the specified cell range.

```
public void AddDecimalValidator(CellRange range, float firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, float? secondOperand = null)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`firstOperand` float

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` float?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

AddDecimalValidator(string, decimal, DataValidationOperatorValues, bool, bool, bool, decimal?)

Adds a decimal data validation to the specified cell range.

```
public void AddDecimalValidator(string range, decimal firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, decimal? secondOperand
= null)
```

Parameters

`range` string

The cell range to apply the validation to.

`firstOperand` decimal

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` decimal?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

AddDecimalValidator(string, double, DataValidationOperatorValues, bool, bool, bool, double?)

Adds a decimal data validation to the specified cell range.

```
public void AddDecimalValidator(string range, double firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, double? secondOperand = null)
```

Parameters

`range` string

The cell range to apply the validation to.

`firstOperand` double

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` double?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

AddDecimalValidator(string, float, DataValidationOperatorValues, bool, bool, bool, float?)

Adds a decimal data validation to the specified cell range.

```
public void AddDecimalValidator(string range, float firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, float? secondOperand = null)
```

Parameters

`range` string

The cell range to apply the validation to.

`firstOperand` float

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` float?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

AddIntegerValidator(CellRange, int, DataValidationOperatorValues, bool, bool, bool, int?)

Adds an integer data validation to the specified cell range.

```
public void AddIntegerValidator(CellRange range, int firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, int? secondOperand = null)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`firstOperand` int

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` int?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

AddIntegerValidator(CellRange, long, DataValidationOperatorValues, bool, bool, bool, long?)

Adds an integer data validation to the specified cell range.

```
public void AddIntegerValidator(CellRange range, long firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, long? secondOperand = null)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`firstOperand` long

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` long?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

AddIntegerValidator(CellRange, uint, DataValidationOperatorValues, bool, bool, bool, uint?)

Adds an integer data validation to the specified cell range.

```
[CLSCompliant(false)]
public void AddIntegerValidator(CellRange range, uint firstOperand,
```

```
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, uint? secondOperand = null)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`firstOperand` uint

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` uint?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

AddIntegerValidator(CellRange, ulong, DataValidationOperatorValues, bool, bool, bool, ulong?)

Adds an integer data validation to the specified cell range.

```
[CLSCompliant(false)]
public void AddIntegerValidator(CellRange range, ulong firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, ulong? secondOperand = null)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`firstOperand` ulong

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` ulong?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

AddIntegerValidator(string, int, DataValidationOperatorValues, bool, bool, bool, int?)

Adds an integer data validation to the specified cell range.

```
public void AddIntegerValidator(string range, int firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, int? secondOperand = null)
```

Parameters

`range` string

The cell range to apply the validation to.

`firstOperand` int

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` int?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

AddIntegerValidator(string, long, DataValidationOperatorValues, bool, bool, bool, long?)

Adds an integer data validation to the specified cell range.

```
public void AddIntegerValidator(string range, long firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, long? secondOperand = null)
```

Parameters

`range` string

The cell range to apply the validation to.

`firstOperand` long

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` long?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

AddIntegerValidator(string, uint, DataValidationOperatorValues, bool, bool, bool, uint?)

Adds an integer data validation to the specified cell range.

```
[CLSCompliant(false)]
public void AddIntegerValidator(string range, uint firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, uint? secondOperand = null)
```

Parameters

`range` string

The cell range to apply the validation to.

`firstOperand` uint

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` uint?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

AddIntegerValidator(string, ulong, DataValidationOperatorValues, bool, bool, bool, ulong?)

Adds an integer data validation to the specified cell range.

```
[CLSCompliant(false)]
public void AddIntegerValidator(string range, ulong firstOperand,
DataValidationOperatorValues validationType, bool allowBlank = true, bool
showInputMessage = true, bool showErrorMessage = true, ulong? secondOperand = null)
```

Parameters

`range` string

The cell range to apply the validation to.

`firstOperand` ulong

The first operand for the validation.

`validationType` DataValidationOperatorValues

The type of validation to apply.

`allowBlank` bool

If set to `true`, blank values are allowed.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

`secondOperand` ulong?

The second operand for the validation, if required by the validation type.

Exceptions

ArgumentNullException

Thrown when the validation type requires a second operand but `secondOperand` is `null`.

ArgumentNullException

Thrown when the `range` is `null`.

NoOpenSheetException

Thrown when there is no open sheet to add the validation to.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

AddListValidator(CellRange, string, bool, bool, bool)

Adds a list data validation to the specified cell range.

```
public void AddListValidator(CellRange range, string formula, bool allowBlank =
true, bool showInputMessage = true, bool showErrorMessage = true)
```

Parameters

`range` CellRange

The cell range to apply the validation to.

`formula` string

The formula defining the list of valid values.

`allowBlank` bool

If set to `true`, blank values are considered valid.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to add the autofilter to.

ArgumentNullException

Thrown when the `range` is `null`.

AddListValidator(string, string, bool, bool, bool)

Adds a list data validation to the specified cell range.

```
public void AddListValidator(string range, string formula, bool allowBlank = true,
bool showInputMessage = true, bool showErrorMessage = true)
```

Parameters

`range` string

The cell range to apply the validation to.

`formula` string

The formula defining the list of valid values.

`allowBlank` bool

If set to `true`, blank values are considered valid.

`showInputMessage` bool

If set to `true`, an input message will be shown.

`showErrorMessage` bool

If set to `true`, an error message will be shown when invalid data is entered.

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to add the autofilter to.

ArgumentNullException

Thrown when the `range` is `null`.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

BeginRow()

Begins a new row in the currently open sheet.

```
public void BeginRow()
```

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

OutOfOrderWritingException

Thrown when writing rows out of order is attempted.

BeginRow(bool)

Begins a new row in the currently open sheet.

```
public void BeginRow(bool hidden)
```

Parameters

`hidden` bool

Indicates whether the row should be hidden.

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

OutOfOrderWritingException

Thrown when writing rows out of order is attempted.

BeginRow(int)

Begins a new row in the currently open sheet.

```
public void BeginRow(int rownum)
```

Parameters

`rownum` int

The row number to begin.

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

OutOfOrderWritingException

Thrown when writing rows out of order is attempted.

BeginRow(int, bool)

Begins a new row in the currently open sheet.

```
public void BeginRow(int rownum, bool hidden)
```

Parameters

`rownum` int

The row number to begin.

`hidden` bool

Indicates whether the row should be hidden.

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

OutOfOrderWritingException

Thrown when writing rows out of order is attempted.

CloseDocument()

Closes the current document, ensuring all data is written and resources are released.

```
public void CloseDocument()
```

Remarks

This method will end any open rows and sheets, write shared strings and sheets, and save the document and worksheet part writer. If saving to a stream, it will reset the stream position to the beginning.

CloseSheet()

Closes the currently open sheet.

```
public void CloseSheet()
```

Exceptions

NoOpenSheetException

Thrown when there is no open sheet to close.

Comment(string, CellRange, string)

Adds a comment to a specified cell range.

```
public void Comment(string text, CellRange cellRange, string author
= "BigExcelCreator")
```

Parameters

`text` string

The text of the comment.

`cellRange` CellRange

The cell range where the comment will be added. Must be a single cell range.

`author` string

The author of the comment. Default is "BigExcelCreator".

Exceptions

ArgumentOutOfRangeException

Thrown when `author` is null or empty, or when `cellRange` is not a single cell range.

ArgumentNullException

Thrown when `cellRange` is null.

NoOpenSheetException

Thrown when there is no open sheet to add the comment to.

Comment(string, string, string)

Adds a comment to a specified cell range.

```
public void Comment(string text, string reference, string author
= "BigExcelCreator")
```

Parameters

`text` string

The text of the comment.

`reference` string

The cell range where the comment will be added. Must be a single cell range.

`author` string

The author of the comment. Default is "BigExcelCreator".

Exceptions

ArgumentOutOfRangeException

Thrown when `author` is null or empty, or when `reference` is not a single cell range.

ArgumentNullException

Thrown when `reference` is null.

NoOpenSheetException

Thrown when there is no open sheet to add the comment to.

InvalidRangeException

Thrown when the `reference` does not represent a valid range.

CreateAndOpenSheet(string)

Creates and opens a new sheet with the specified name, and prepares the writer to use it.

```
public void CreateAndOpenSheet(string name)
```

Parameters

`name` string

The name of the sheet to create and open.

Exceptions

SheetAlreadyOpenException

Thrown when a sheet is already open and not closed before opening a new one.

SheetNameCannotBeEmptyException

Thrown when the sheet name is null or empty.

SheetWithSameNameAlreadyExistsException

Thrown when a sheet with the same name already exists.

CreateAndOpenSheet(string, SheetStateValues)

Creates and opens a new sheet with the specified name, and sheet state, and prepares the writer to use it.

```
public void CreateAndOpenSheet(string name, SheetStateValues sheetState)
```

Parameters

`name` string

The name of the sheet to create and open.

`sheetState` SheetStateValues

Sets sheet visibility. `SheetStateValues.Visible` to list the sheet. `SheetStateValues.Hidden` to hide it. `SheetStateValues.VeryHidden` to hide it and prevent unhiding from the GUI.

Exceptions

SheetAlreadyOpenException

Thrown when a sheet is already open and not closed before opening a new one.

SheetNameCannotBeEmptyException

Thrown when the sheet name is null or empty.

SheetWithSameNameAlreadyExistsException

Thrown when a sheet with the same name already exists.

CreateAndOpenSheet(string, IList<Column>)

Creates and opens a new sheet with the specified name and columns, and prepares the writer to use it.

```
public void CreateAndOpenSheet(string name, IList<Column> columns)
```

Parameters

`name` string

The name of the sheet to create and open.

`columns` IList<Column>

The columns to add to the sheet. Can be null. Use this to set the columns' width.

Exceptions

SheetAlreadyOpenException

Thrown when a sheet is already open and not closed before opening a new one.

SheetNameCannotBeEmptyException

Thrown when the sheet name is null or empty.

SheetWithSameNameAlreadyExistsException

Thrown when a sheet with the same name already exists.

CreateAndOpenSheet(string, IList<Column>, SheetStateValues)

Creates and opens a new sheet with the specified name, columns, and sheet state, and prepares the writer to use it.

```
public void CreateAndOpenSheet(string name, IList<Column> columns,
SheetStateValues sheetState)
```

Parameters

`name` string

The name of the sheet to create and open.

`columns` IList<Column>

The columns to add to the sheet. Can be null. Use this to set the columns' width.

`sheetState` SheetStateValues

Sets sheet visibility. `SheetStateValues.Visible` to list the sheet. `SheetStateValues.Hidden` to hide it. `SheetStateValues.VeryHidden` to hide it and prevent unhiding from the GUI.

Exceptions

SheetAlreadyOpenException

Thrown when a sheet is already open and not closed before opening a new one.

SheetNameCannotBeEmptyException

Thrown when the sheet name is null or empty.

SheetWithSameNameAlreadyExistsException

Thrown when a sheet with the same name already exists.

CreateSheetFromObject<T>(IEnumerable<T>, string, SheetStateValues, bool, bool, IList<Column>)

Creates a new sheet from a collection of objects with a specified sheet state, automatically mapping object properties to columns.

```
[SuppressMessage("Performance", "CA1851:Possible multiple enumerations of
'IEnumerable' collection", Justification = "I need to have it separated")]
public void CreateSheetFromObject<T>(IEnumerable<T> data, string sheetName,
SheetStateValues sheetState, bool writeHeaderRow = true, bool
addAutoFilterOnFirstColumn = false, IList<Column> columns = null) where T : class
```

Parameters

`data` IEnumerable<T>

The collection of objects to write to the sheet.

`sheetName` string

The name of the sheet to create.

`sheetState` SheetStateValues

Sets sheet visibility. `SheetStateValues.Visible` to list the sheet. `SheetStateValues.Hidden` to hide it. `SheetStateValues.VeryHidden` to hide it and prevent unhiding from the GUI.

`writeHeaderRow` bool

If set to `true`, writes column headers as first row. Default is `true`.

`addAutoFilterOnFirstColumn` bool

If set to `true`, adds an autofilter to the first row. Default is `false`.

`columns` IList<Column>

The column definitions to use for the sheet. If not provided, columns will be generated automatically from the object type. Default is `null`.

Type Parameters

`T`

The type of objects in the collection. Must be a reference type.

Remarks

This method automatically discovers properties from type `T` and writes them as sheet columns.

Properties can be decorated with the following attributes to customize their behavior:

- ExcelIgnoreAttribute - Excludes a property from being written to the sheet.
- ExcelColumnNameAttribute - Sets a custom column header name.
- ExcelColumnOrderAttribute - Controls the column order.
- ExcelColumnTypeAttribute - Specifies the cell data type (Text, Number, or Formula).
- ExcelColumnWidthAttribute - Sets a custom column width.
- ExcelColumnHiddenAttribute - Hides the column from view.

Exceptions

ArgumentNullException

Thrown when `data` is `null`.

SheetAlreadyOpenException

Thrown when a sheet is already open and not closed before opening a new one.

SheetNameCannotBeEmptyException

Thrown when `sheetName` is null or empty.

SheetWithSameNameAlreadyExistsException

Thrown when a sheet with the same name already exists.

CreateSheetFromObject<T>(IEnumerable<T>, string, bool, bool, IList<Column>)

Creates a new sheet from a collection of objects, automatically mapping object properties to columns.

```
public void CreateSheetFromObject<T>(IEnumerable<T> data, string sheetName, bool
writeHeaderRow = true, bool addAutoFilterOnFirstColumn = false, IList<Column>
columns = null) where T : class
```

Parameters

`data` IEnumerable<T>

The collection of objects to write to the sheet.

`sheetName` string

The name of the sheet to create.

`writeHeaderRow` bool

If set to `true`, writes column headers as first row. Default is `true`.

`addAutoFilterOnFirstColumn` bool

If set to `true`, adds an autofilter to the first row. Default is `false`.

`columns` IList<Column>

The column definitions to use for the sheet. If not provided, columns will be generated automatically from the object type. Default is `null`.

Type Parameters

`T`

The type of objects in the collection. Must be a reference type.

Remarks

This method automatically discovers properties from type `T` and writes them as sheet columns.

Properties can be decorated with the following attributes to customize their behavior:

- ExcelIgnoreAttribute - Excludes a property from being written to the sheet.
- ExcelColumnNameAttribute - Sets a custom column header name.
- ExcelColumnOrderAttribute - Controls the column order.
- ExcelColumnTypeAttribute - Specifies the cell data type (Text, Number, or Formula).
- ExcelColumnWidthAttribute - Sets a custom column width.
- ExcelColumnHiddenAttribute - Hides the column from view.

The sheet state is set to Visible by default.

Exceptions

ArgumentNullException

Thrown when `data` is `null`.

SheetAlreadyOpenException

Thrown when a sheet is already open and not closed before opening a new one.

SheetNameCannotBeEmptyException

Thrown when `sheetName` is null or empty.

SheetWithSameNameAlreadyExistsException

Thrown when a sheet with the same name already exists.

Dispose()

Closes the current document, ensuring all data is written and resources are released.

```
public void Dispose()
```

Remarks

This method will end any open rows and sheets, write shared strings and sheets, and save the document and worksheet part writer. If saving to a stream, it will reset the stream position to the beginning.

Dispose(bool)

Closes the current document, ensuring all data is written and resources are released.

```
protected virtual void Dispose(bool disposing)
```

Parameters

`disposing` bool

Remarks

This method will end any open rows and sheets, write shared strings and sheets, and save the document and worksheet part writer. If saving to a stream, it will reset the stream position to the beginning.

EndRow()

Ends the currently open row in the sheet.

```
public void EndRow()
```

Exceptions

NoOpenRowException

Thrown when there is no open row to end.

~BigExcelWriter()

Finalizes an instance of the BigExcelWriter class.

```
protected ~BigExcelWriter()
```

MergeCells(CellRange)

Merges the specified cell range in the current sheet.

```
public void MergeCells(CellRange range)
```

Parameters

`range` CellRange

The cell range to merge.

Exceptions

ArgumentNullException

Thrown when `range` is null.

NoOpenSheetException

Thrown when there is no open sheet to merge the cells into.

OverlappingRangesException

Thrown when the specified range overlaps with an existing merged range.

MergeCells(string)

Merges the specified cell range in the current sheet.

```
public void MergeCells(string range)
```

Parameters

`range` string

The cell range to merge.

Exceptions

ArgumentNullException

Thrown when `range` is null.

NoOpenSheetException

Thrown when there is no open sheet to merge the cells into.

OverlappingRangesException

Thrown when the specified range overlaps with an existing merged range.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

WriteFormulaCell(string, int)

Writes a formula cell to the currently open row in the sheet.

```
public void WriteFormulaCell(string formula, int format = 0)
```

Parameters

`formula` string

The formula to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteFormulaRow(IEnumerable<string>, int, bool)

Writes a row of formula cells to the currently open sheet.

```
public void WriteFormulaRow(IEnumerable<string> formulas, int format = 0, bool
hidden = false)
```

Parameters

`formulas` IEnumerable<string>

The collection of formulas to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the formulas collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberCell(byte, int)

Writes a numerical value to the currently open row in the sheet.

```
public void WriteNumberCell(byte number, int format = 0)
```

Parameters

`number` byte

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(decimal, int)

Writes a numerical value to the currently open row in the sheet.

```
public void WriteNumberCell(decimal number, int format = 0)
```

Parameters

`number` decimal

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(double, int)

Writes a numerical value to the currently open row in the sheet.

```
public void WriteNumberCell(double number, int format = 0)
```

Parameters

`number` double

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(short, int)

Writes a numerical value to the currently open row in the sheet.

```
public void WriteNumberCell(short number, int format = 0)
```

Parameters

`number` short

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(int, int)

Writes a numerical value to the currently open row in the sheet.

```
public void WriteNumberCell(int number, int format = 0)
```

Parameters

`number` int

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(long, int)

Writes a numerical value to the currently open row in the sheet.

```
public void WriteNumberCell(long number, int format = 0)
```

Parameters

`number` long

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(sbyte, int)

Writes a numerical value to the currently open row in the sheet.

```
[CLSCompliant(false)]
public void WriteNumberCell(sbyte number, int format = 0)
```

Parameters

`number` sbyte

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(float, int)

Writes a numerical value to the currently open row in the sheet.

```
public void WriteNumberCell(float number, int format = 0)
```

Parameters

`number` float

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(ushort, int)

Writes a numerical value to the currently open row in the sheet.

```
[CLSCompliant(false)]
public void WriteNumberCell(ushort number, int format = 0)
```

Parameters

`number` ushort

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(uint, int)

Writes a numerical value to the currently open row in the sheet.

```
[CLSCompliant(false)]
public void WriteNumberCell(uint number, int format = 0)
```

Parameters

`number` uint

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberCell(ulong, int)

Writes a numerical value to the currently open row in the sheet.

```
[CLSCompliant(false)]
public void WriteNumberCell(ulong number, int format = 0)
```

Parameters

`number` ulong

The number to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

Exceptions

ArgumentOutOfRangeException

Thrown when `format` is less than 0

NoOpenRowException

Thrown when there is no open row to write the cell to.

WriteNumberRow(IEnumerable<byte>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
public void WriteNumberRow(IEnumerable<byte> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<byte>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<decimal>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
public void WriteNumberRow(IEnumerable<decimal> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<decimal>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<double>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
public void WriteNumberRow(IEnumerable<double> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<double>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<short>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
public void WriteNumberRow(IEnumerable<short> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<short>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<int>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
public void WriteNumberRow(IEnumerable<int> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<int>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<long>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
public void WriteNumberRow(IEnumerable<long> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<long>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<sbyte>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
[CLSCompliant(false)]
public void WriteNumberRow(IEnumerable<sbyte> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<sbyte>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<float>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
public void WriteNumberRow(IEnumerable<float> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<float>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<ushort>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
[CLSCompliant(false)]
public void WriteNumberRow(IEnumerable<ushort> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<ushort>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<uint>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
[CLSCompliant(false)]
public void WriteNumberRow(IEnumerable<uint> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<uint>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteNumberRow(IEnumerable<ulong>, int, bool)

Writes a row of cells with numerical values to the currently open sheet.

```
[CLSCompliant(false)]
public void WriteNumberRow(IEnumerable<ulong> numbers, int format = 0, bool hidden
= false)
```

Parameters

`numbers` IEnumerable<ulong>

The collection of numbers to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

Exceptions

ArgumentNullException

Thrown when the numbers collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

WriteTextCell(string, int, bool)

Writes a text cell to the currently open row in the sheet.

```
public void WriteTextCell(string text, int format = 0, bool useSharedStrings
= false)
```

Parameters

`text` string

The text to write in the cell.

`format` int

The format index to apply to the cell. Default is 0. See GetIndexByName(string).

`useSharedStrings` bool

Indicates whether to write the value to the shared strings table. This might help reduce the output file size when the same text is shared multiple times among sheets. Default is false.

Exceptions

NoOpenRowException

Thrown when there is no open row to write the cell to.

ArgumentOutOfRangeException

When `format` is less than 0

WriteTextRow(IEnumerable<string>, int, bool, bool)

Writes a row of text cells to the currently open sheet.

```
public void WriteTextRow(IEnumerable<string> texts, int format = 0, bool hidden =
false, bool useSharedStrings = false)
```

Parameters

`texts` IEnumerable<string>

The collection of text strings to write in the row.

`format` int

The format index to apply to each cell. Default is 0. See GetIndexByName(string)

`hidden` bool

Indicates whether the row should be hidden. Default is false.

`useSharedStrings` bool

Indicates whether to write the value to the shared strings table. This might help reduce the output file size when the same text is shared multiple times among sheets. Default is false.

Exceptions

ArgumentNullException

Thrown when the texts collection is null.

NoOpenSheetException

Thrown when there is no open sheet to write a row to.

RowAlreadyOpenException

Thrown when a row is already open. Use EndRow to close it.

ArgumentOutOfRangeException

Thrown when `format` is less than 0

Namespace BigExcelCreator.Class Attributes

Classes

ExcelColumnHiddenAttribute
: Marks a property as hidden in Excel exports. When applied to a property, the corresponding column will not be visible in the generated Excel spreadsheet.

ExcelColumnNameAttribute
: Specifies the name of an Excel column for a property.

ExcelColumnOrderAttribute
: Specifies the column order for a property when exporting to Excel.

ExcelColumnTypeAttribute
: Specifies the Excel cell type for a property when exporting to Excel.

ExcelColumnWidthAttribute
: Specifies the width of an Excel column for a property.

ExcelIgnoreAttribute
: Marks a property to be ignored when generating Excel output.

Class ExcelColumnHiddenAttribute

Namespace: BigExcelCreator.ClassAttributes

Assembly: BigExcelCreator.dll

Marks a property as hidden in Excel exports. When applied to a property, the corresponding column will not be visible in the generated Excel spreadsheet.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public sealed class ExcelColumnHiddenAttribute : Attribute
```

Inheritance

object ← Attribute ← ExcelColumnHiddenAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,

Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) , Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Class ExcelColumnNameAttribute

Namespace: BigExcelCreator.ClassAttributes

Assembly: BigExcelCreator.dll

Specifies the name of an Excel column for a property.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public sealed class ExcelColumnNameAttribute : Attribute
```

Inheritance

object ← Attribute ← ExcelColumnNameAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,

Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ExcelColumnNameAttribute(string)

Specifies the name of an Excel column for a property.

```
public ExcelColumnNameAttribute(string name)
```

Parameters

`name` string

The name of the Excel column.

Properties

Name

Gets the name of the Excel column.

```
public string Name { get; }
```

Property Value

string

Class ExcelColumnOrderAttribute

Namespace: BigExcelCreator.ClassAttributes

Assembly: BigExcelCreator.dll

Specifies the column order for a property when exporting to Excel.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public sealed class ExcelColumnOrderAttribute : Attribute
```

Inheritance

object ← Attribute ← ExcelColumnOrderAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,

Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This attribute is used to control the order in which properties are exported as columns in an Excel worksheet. Properties are ordered by their Order value in ascending order.

Constructors

ExcelColumnOrderAttribute(int)

Specifies the column order for a property when exporting to Excel.

```
public ExcelColumnOrderAttribute(int order)
```

Parameters

`order` int

The zero-based position of the column within the Excel sheet.

Remarks

This attribute is used to control the order in which properties are exported as columns in an Excel worksheet. Properties are ordered by their Order value in ascending order.

Properties

Order

Gets the zero-based position of the item within its containing collection.

```
public int Order { get; }
```

Property Value

int

Class ExcelColumnTypeAttribute

Namespace: BigExcelCreator.ClassAttributes

Assembly: BigExcelCreator.dll

Specifies the Excel cell type for a property when exporting to Excel.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public sealed class ExcelColumnTypeAttribute : Attribute
```

Inheritance

object ← Attribute ← ExcelColumnTypeAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,

Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ExcelColumnTypeAttribute(CellDataType)

Specifies the Excel cell type for a property when exporting to Excel.

```
public ExcelColumnTypeAttribute(CellDataType type)
```

Parameters

`type` CellDataType

Properties

Type

Gets the Excel cell type for the attributed property.

```
public CellDataType Type { get; }
```

Property Value

CellDataType

Class ExcelColumnWidthAttribute

Namespace: BigExcelCreator.ClassAttributes

Assembly: BigExcelCreator.dll

Specifies the width of an Excel column for a property.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public sealed class ExcelColumnWidthAttribute : Attribute
```

Inheritance

object ← Attribute ← ExcelColumnWidthAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,

Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This attribute is used to define the column width when exporting data to Excel. The width value must be a non-negative integer.

Constructors

ExcelColumnWidthAttribute(int)

Initializes a new instance of the ExcelColumnWidthAttribute class.

```
public ExcelColumnWidthAttribute(int width)
```

Parameters

`width` int

The width of the Excel column. Must be non-negative.

Exceptions

ArgumentOutOfRangeException

Thrown when `width` is negative.

Properties

Width

Gets the width of the Excel column.

```
public int Width { get; }
```

Property Value

int

Class ExcelIgnoreAttribute

Namespace: BigExcelCreator.ClassAttributes

Assembly: BigExcelCreator.dll

Marks a property to be ignored when generating Excel output.

```
[AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
public sealed class ExcelIgnoreAttribute : Attribute
```

Inheritance

object ← Attribute ← ExcelIgnoreAttribute

Inherited Members

Attribute.Equals(object) , Attribute.GetCustomAttribute(Assembly, Type) ,
Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) ,
Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) ,
Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) ,
Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) ,
Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) ,
Attribute.GetCustomAttributes(Module) , Attribute.GetCustomAttributes(Module, bool) ,
Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) ,
Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,

Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) , Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.ReferenceEquals(object, object) , object.ToString()

Namespace BigExcelCreator.Enums

Enums

CellDataType

Specifies the Excel cell type for a spreadsheet cell.

Enum CellDataType

Namespace: BigExcelCreator.Enums

Assembly: BigExcelCreator.dll

Specifies the Excel cell type for a spreadsheet cell.

```
public enum CellDataType
```

Fields

`Formula = 2`

Formula cell type for Excel formulas.

`Number = 1`

Number cell type for numeric values.

`Text = 0`

Text cell type for string values.

Namespace BigExcelCreator.Exceptions

Classes

NoOpenRowException
: When attempting to write to a row when there is none open

NoOpenSheetException
: When attempting to write to a sheet when there is none open

OutOfOrderWritingException
: When attempting to write to a previous row / a row before another already written to

RowAlreadyOpenException
: When attempting to open a row when there is another already open

SheetAlreadyHasFilterException
: When attempting to create a filter to a sheet that already has one, without indicating to overwrite the old one

SheetAlreadyOpenException
: When attempting to open a sheet when there is another already open

SheetNameCannotBeEmptyException
: When attempting to open a sheet when there is another already open

SheetWithSameNameAlreadyExistsException
: When attempting to open a sheet when there is another already open

UnsupportedSpreadsheetDocumentTypeException
: When attempting to open a sheet when there is another already open

Class NoOpenRowException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to write to a row when there is none open

```
[Serializable]
public class NoOpenRowException : InvalidOperationException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← NoOpenRowException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

NoOpenRowException()

The constructor for NoOpenRowException

```
public NoOpenRowException()
```

NoOpenRowException(SerializationInfo, StreamingContext)

The constructor for NoOpenRowException

```
protected NoOpenRowException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

NoOpenRowException(string)

The constructor for NoOpenRowException

```
public NoOpenRowException(string message)
```

Parameters

`message` string

NoOpenRowException(string, Exception)

The constructor for NoOpenRowException

```
public NoOpenRowException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class NoOpenSheetException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to write to a sheet when there is none open

```
[Serializable]
public class NoOpenSheetException : InvalidOperationException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← NoOpenSheetException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

NoOpenSheetException()

The constructor for NoOpenSheetException

```
public NoOpenSheetException()
```

NoOpenSheetException(SerializationInfo, StreamingContext)

The constructor for NoOpenSheetException

```
protected NoOpenSheetException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

NoOpenSheetException(string)

The constructor for NoOpenSheetException

```
public NoOpenSheetException(string message)
```

Parameters

`message` string

NoOpenSheetException(string, Exception)

The constructor for NoOpenSheetException

```
public NoOpenSheetException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class OutOfOrderWritingException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to write to a previous row / a row before another already written to

```
[Serializable]
public class OutOfOrderWritingException : InvalidOperationException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← OutOfOrderWritingException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

OutOfOrderWritingException()

The constructor for OutOfOrderWritingException

```
public OutOfOrderWritingException()
```

OutOfOrderWritingException(SerializationInfo, StreamingContext)

The constructor for OutOfOrderWritingException

```
protected OutOfOrderWritingException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

OutOfOrderWritingException(string)

The constructor for OutOfOrderWritingException

```
public OutOfOrderWritingException(string message)
```

Parameters

`message` string

OutOfOrderWritingException(string, Exception)

The constructor for OutOfOrderWritingException

```
public OutOfOrderWritingException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class RowAlreadyOpenException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to open a row when there is another already open

```
[Serializable]
public class RowAlreadyOpenException : InvalidOperationException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← RowAlreadyOpenException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

RowAlreadyOpenException()

The constructor for RowAlreadyOpenException

```
public RowAlreadyOpenException()
```

RowAlreadyOpenException(SerializationInfo, StreamingContext)

The constructor for RowAlreadyOpenException

```
protected RowAlreadyOpenException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

RowAlreadyOpenException(string)

The constructor for RowAlreadyOpenException

```
public RowAlreadyOpenException(string message)
```

Parameters

`message` string

RowAlreadyOpenException(string, Exception)

The constructor for RowAlreadyOpenException

```
public RowAlreadyOpenException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class SheetAlreadyHasFilterException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to create a filter to a sheet that already has one, without indicating to overwrite the old one

```
[Serializable]
public class SheetAlreadyHasFilterException : InvalidOperationException,
ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← SheetAlreadyHasFilterException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

SheetAlreadyHasFilterException()

The constructor for SheetAlreadyHasFilterException

```
public SheetAlreadyHasFilterException()
```

SheetAlreadyHasFilterException(SerializationInfo, StreamingContext)

The constructor for SheetAlreadyHasFilterException

```
protected SheetAlreadyHasFilterException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

SheetAlreadyHasFilterException(string)

The constructor for SheetAlreadyHasFilterException

```
public SheetAlreadyHasFilterException(string message)
```

Parameters

`message` string

SheetAlreadyHasFilterException(string, Exception)

The constructor for SheetAlreadyHasFilterException

```
public SheetAlreadyHasFilterException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class SheetAlreadyOpenException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to open a sheet when there is another already open

```
[Serializable]
public class SheetAlreadyOpenException : InvalidOperationException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← SheetAlreadyOpenException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

SheetAlreadyOpenException()

The constructor for SheetAlreadyOpenException

```
public SheetAlreadyOpenException()
```

SheetAlreadyOpenException(SerializationInfo, StreamingContext)

The constructor for SheetAlreadyOpenException

```
protected SheetAlreadyOpenException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

SheetAlreadyOpenException(string)

The constructor for SheetAlreadyOpenException

```
public SheetAlreadyOpenException(string message)
```

Parameters

`message` string

SheetAlreadyOpenException(string, Exception)

The constructor for SheetAlreadyOpenException

```
public SheetAlreadyOpenException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class SheetNameCannotBeEmptyException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to open a sheet when there is another already open

```
[Serializable]
public class SheetNameCannotBeEmptyException : InvalidOperationException,
ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← SheetNameCannotBeEmptyException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

SheetNameCannotBeEmptyException()

The constructor for SheetNameCannotBeNullException

```
public SheetNameCannotBeEmptyException()
```

SheetNameCannotBeEmptyException(SerializationInfo, StreamingContext)

The constructor for SheetNameCannotBeNullException

```
protected SheetNameCannotBeEmptyException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

SheetNameCannotBeEmptyException(string)

The constructor for SheetNameCannotBeNullException

```
public SheetNameCannotBeEmptyException(string message)
```

Parameters

`message` string

SheetNameCannotBeEmptyException(string, Exception)

The constructor for SheetNameCannotBeNullException

```
public SheetNameCannotBeEmptyException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class SheetWithSameNameAlreadyExistsException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to open a sheet when there is another already open

```
[Serializable]
public class SheetWithSameNameAlreadyExistsException : InvalidOperationException,
ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← SheetWithSameNameAlreadyExistsException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

SheetWithSameNameAlreadyExistsException()

The constructor for SheetWithSameNameAlreadyExistsException

```
public SheetWithSameNameAlreadyExistsException()
```

SheetWithSameNameAlreadyExistsException(SerializationInfo, StreamingContext)

The constructor for SheetWithSameNameAlreadyExistsException

```
protected SheetWithSameNameAlreadyExistsException(SerializationInfo
serializationInfo, StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

SheetWithSameNameAlreadyExistsException(string)

The constructor for SheetWithSameNameAlreadyExistsException

```
public SheetWithSameNameAlreadyExistsException(string message)
```

Parameters

`message` string

SheetWithSameNameAlreadyExistsException(string, Exception)

The constructor for SheetWithSameNameAlreadyExistsException

```
public SheetWithSameNameAlreadyExistsException(string message,
Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class UnsupportedSpreadsheetDocumentTypeException

Namespace: BigExcelCreator.Exceptions

Assembly: BigExcelCreator.dll

When attempting to open a sheet when there is another already open

```
[Serializable]
public class UnsupportedSpreadsheetDocumentTypeException :
NotSupportedException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← NotSupportedException ← UnsupportedSpreadsheetDocumentTypeException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

UnsupportedSpreadsheetDocumentTypeException()

The constructor for UnsupportedSpreadsheetDocumentTypesException

```
public UnsupportedSpreadsheetDocumentTypeException()
```

UnsupportedSpreadsheetDocumentTypeException(SerializationInfo, StreamingContext)

The constructor for UnsupportedSpreadsheetDocumentTypesException

```
protected UnsupportedSpreadsheetDocumentTypeException(SerializationInfo
serializationInfo, StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

UnsupportedSpreadsheetDocumentTypeException(string)

The constructor for UnsupportedSpreadsheetDocumentTypesException

```
public UnsupportedSpreadsheetDocumentTypeException(string message)
```

Parameters

`message` string

UnsupportedSpreadsheetDocumentTypeException(string, Exception)

The constructor for UnsupportedSpreadsheetDocumentTypesException

```
public UnsupportedSpreadsheetDocumentTypeException(string message,
Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Namespace BigExcelCreator.Ranges

Classes

CellRange

Represents a range of cells in an Excel sheet.

InvalidRangeException

When unable to parse a range from a string or a range is not valid

OverlappingRangesException

When 2 or more ranges overlaps one another

Class CellRange

Namespace: BigExcelCreator.Ranges

Assembly: BigExcelCreator.dll

Represents a range of cells in an Excel sheet.

```
public class CellRange : IEquatable<CellRange>, IComparable<CellRange>
```

Inheritance

object ← CellRange

Implements

IEquatable<CellRange>, IComparable<CellRange>

Inherited Members

object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Remarks

This class provides properties and methods to handle cell ranges, including their dimensions, overlap checks, and string representations.

Constructors

CellRange(int?, bool, int?, bool, int?, bool, int?, bool, string)

Initializes a new instance of the CellRange class using coordinates indexes.

```
public CellRange(int? startingColumn, bool fixedStartingColumn, int? startingRow,
bool fixedStartingRow, int? endingColumn, bool fixedEndingColumn, int? endingRow,
bool fixedEndingRow, string sheetname)
```

Parameters

`startingColumn` int?

The starting column of the cell range.

`fixedStartingColumn` bool

Indicates whether the starting column is fixed.

`startingRow` int?

The starting row of the cell range.

`fixedStartingRow` bool

Indicates whether the starting row is fixed.

`endingColumn` int?

The ending column of the cell range.

`fixedEndingColumn` bool

Indicates whether the ending column is fixed.

`endingRow` int?

The ending row of the cell range.

`fixedEndingRow` bool

Indicates whether the ending row is fixed.

`sheetname` string

The name of the sheet.

Exceptions

ArgumentOutOfRangeException

Thrown when any of the column or row values are less than 1.

InvalidRangeException

Thrown when the range is invalid.

CellRange(int?, bool, int?, bool, string)

Initializes a new instance of the CellRange class using coordinates indexes. This creates a single cell range

```
public CellRange(int? column, bool fixedColumn, int? row, bool fixedRow,
string sheetname)
```

Parameters

`column` int?

The column of the cell range.

`fixedColumn` bool

Indicates whether the column is fixed.

`row` int?

The row of the cell range.

`fixedRow` bool

Indicates whether the row is fixed.

`sheetname` string

The name of the sheet.

Exceptions

ArgumentOutOfRangeException

Thrown when any of the column or row values are less than 1.

InvalidRangeException

Thrown when the range is invalid.

CellRange(int?, int?, int?, int?, string)

Initializes a new instance of the CellRange class using coordinates indexes.

```
public CellRange(int? startingColumn, int? startingRow, int? endingColumn, int?
endingRow, string sheetname)
```

Parameters

`startingColumn` int?

The starting column of the cell range.

`startingRow` int?

The starting row of the cell range.

`endingColumn` int?

The ending column of the cell range.

`endingRow` int?

The ending row of the cell range.

`sheetname` string

The name of the sheet.

Exceptions

ArgumentOutOfRangeException

Thrown when any of the column or row values are less than 1.

InvalidRangeException

Thrown when the range is invalid.

CellRange(int?, int?, string)

Initializes a new instance of the CellRange class using coordinates indexes. This creates a single cell range

```
public CellRange(int? column, int? row, string sheetname)
```

Parameters

`column` int?

The column of the cell range.

`row` int?

The row of the cell range.

`sheetname` string

The name of the sheet.

Exceptions

ArgumentOutOfRangeException

Thrown when any of the column or row values are less than 1.

InvalidRangeException

Thrown when the range is invalid.

CellRange(string)

Initializes a new instance of the CellRange class from a string representation of a range.

```
public CellRange(string range)
```

Parameters

`range` string

The range string to initialize the cell range.

Exceptions

ArgumentNullException

If `range` is null

ArgumentOutOfRangeException

Thrown when any of the column or row values are less than 1.

InvalidRangeException

Thrown when the `range` does not represent a valid range.

Properties

EndingColumn

Gets the ending column of the cell range.

```
public int? EndingColumn { get; }
```

Property Value

int?

The ending column of the cell range, or null if the ending column is not specified.

EndingColumnIsFixed

true if the ending column is fixed

Represented by '$' in the string representation

```
public bool EndingColumnIsFixed { get; }
```

Property Value

bool

EndingRow

Gets the ending row of the cell range.

```
public int? EndingRow { get; }
```

Property Value

int?

The ending row of the cell range, or null if the ending row is not specified.

EndingRowIsFixed

true if the ending row is fixed

Represented by '$' in the string representation

```
public bool EndingRowIsFixed { get; }
```

Property Value

bool

Height

Gets the height of the cell range.

```
public int Height { get; }
```

Property Value

int

IsInfiniteCellRange

Gets a value indicating whether the cell range is infinite.

```
public bool IsInfiniteCellRange { get; }
```

Property Value

bool

True if the cell range is infinite; otherwise, false.

IsInfiniteCellRangeCol

Gets a value indicating whether the cell range represents an entire column.

```
public bool IsInfiniteCellRangeCol { get; }
```

Property Value

bool

True if the cell range represents an entire column; otherwise, false.

IsInfiniteCellRangeRow

Gets a value indicating whether the cell range represents an entire row.

```
public bool IsInfiniteCellRangeRow { get; }
```

Property Value

bool

True if the cell range represents an entire row; otherwise, false.

IsSingleCellRange

Gets a value indicating whether the cell range represents a single cell.

```
public bool IsSingleCellRange { get; }
```

Property Value

bool

True if the cell range represents a single cell; otherwise, false.

RangeString

Gets the range string representation of the cell range, including the sheet name if available.

```
public string RangeString { get; }
```

Property Value

string

The range string representation of the cell range, including the sheet name if available.

RangeStringNoSheetName

Gets the range string representation of the cell range without the sheet name.

```
public string RangeStringNoSheetName { get; }
```

Property Value

string

The range string representation of the cell range without the sheet name.

Sheetname

Gets or sets the sheet name of the cell range.

```
public string Sheetname { get; set; }
```

Property Value

string

The sheet name of the cell range.

Exceptions

InvalidRangeException

Thrown when the sheet name contains invalid characters.

StartingColumn

Gets the starting column of the cell range.

```
public int? StartingColumn { get; }
```

Property Value

int?

The starting column of the cell range, or null if the starting column is not specified.

StartingColumnIsFixed

true if the starting column is fixed

Represented by '$' in the string representation

```
public bool StartingColumnIsFixed { get; }
```

Property Value

bool

StartingRow

Gets the starting row of the cell range.

```
public int? StartingRow { get; }
```

Property Value

int?

The starting row of the cell range, or null if the starting row is not specified.

StartingRowIsFixed

true if the starting row is fixed

Represented by '$' in the string representation

```
public bool StartingRowIsFixed { get; }
```

Property Value

bool

Width

Gets the width of the cell range.

```
public int Width { get; }
```

Property Value

int

Methods

CompareTo(CellRange)

Compares the current instance with another CellRange and returns an integer that indicates whether the current instance precedes, follows, or occurs in the same position in the sort order as the other CellRange.

```
public int CompareTo(CellRange other)
```

Parameters

`other` CellRange

The CellRange to compare with the current instance.

Returns

int

A value that indicates the relative order of the objects being compared. The return value has these meanings:

- Less than zero: This instance precedes `other` in the sort order.
- Zero: This instance occurs in the same position in the sort order as `other`.
- Greater than zero: This instance follows `other` in the sort order.

Equals(CellRange)

Determines whether the specified CellRange is equal to the current CellRange instance.

```
public virtual bool Equals(CellRange other)
```

Parameters

`other` CellRange

The CellRange to compare with the current instance.

Returns

bool

True if the specified CellRange is equal to the current instance; otherwise, false.

Equals(object)

Determines whether the specified object is equal to the current CellRange instance.

```
public override bool Equals(object obj)
```

Parameters

`obj` object

The object to compare with the current instance.

Returns

bool

True if the specified object is a CellRange and is equal to the current instance; otherwise, false.

GetHashCode()

Returns the hash code for this instance.

```
public override int GetHashCode()
```

Returns

int

A 32-bit signed integer hash code.

RangeOverlaps(CellRange)

Determines whether the current CellRange overlaps with another specified CellRange.

```
public bool RangeOverlaps(CellRange other)
```

Parameters

`other` CellRange

The CellRange to compare with the current CellRange.

Returns

bool

`true` if the current CellRange overlaps with the `other` CellRange; otherwise, `false`.

Exceptions

ArgumentNullException

`other` is `null`.

Operators

operator ==(CellRange, CellRange)

Determines whether two specified CellRange objects have the same value.

```
public static bool operator ==(CellRange left, CellRange right)
```

Parameters

`left` CellRange

The first CellRange to compare.

`right` CellRange

The second CellRange to compare.

Returns

bool

`true` if the value of `left` is the same as the value of `right`; otherwise, `false`.

operator >(CellRange, CellRange)

Determines whether one specified CellRange is greater than another specified CellRange.

```
public static bool operator >(CellRange left, CellRange right)
```

Parameters

`left` CellRange

The first CellRange to compare.

`right` CellRange

The second CellRange to compare.

Returns

bool

`true` if the value of `left` is greater than the value of `right`; otherwise, `false`.

operator >=(CellRange, CellRange)

Determines whether one specified CellRange is greater than or equal to another specified CellRange.

```
public static bool operator >=(CellRange left, CellRange right)
```

Parameters

`left` CellRange

The first CellRange to compare.

`right` CellRange

The second CellRange to compare.

Returns

bool

`true` if the value of `left` is greater than or equal to the value of `right`; otherwise, `false`.

operator !=(CellRange, CellRange)

Determines whether two specified CellRange objects have different values.

```
public static bool operator !=(CellRange left, CellRange right)
```

Parameters

`left` CellRange

The first CellRange to compare.

`right` CellRange

The second CellRange to compare.

Returns

bool

`true` if the value of `left` is different from the value of `right`; otherwise, `false`.

operator <(CellRange, CellRange)

Determines whether one specified CellRange is less than another specified CellRange.

```
public static bool operator <(CellRange left, CellRange right)
```

Parameters

`left` CellRange

The first CellRange to compare.

`right` CellRange

The second CellRange to compare.

Returns

bool

`true` if the value of `left` is less than the value of `right`; otherwise, `false`.

operator <=(CellRange, CellRange)

Determines whether one specified CellRange is less than or equal to another specified CellRange.

```
public static bool operator <=(CellRange left, CellRange right)
```

Parameters

`left` CellRange

The first CellRange to compare.

`right` CellRange

The second CellRange to compare.

Returns

bool

`true` if the value of `left` is less than or equal to the value of `right`; otherwise, `false`.

Class InvalidRangeException

Namespace: BigExcelCreator.Ranges

Assembly: BigExcelCreator.dll

When unable to parse a range from a string or a range is not valid

```
[Serializable]
public class InvalidRangeException : Exception, ISerializable
```

Inheritance

object ← Exception ← InvalidRangeException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

InvalidRangeException()

Constructor for InvalidRangeException

```
public InvalidRangeException()
```

InvalidRangeException(SerializationInfo, StreamingContext)

Constructor for InvalidRangeException

```
protected InvalidRangeException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

InvalidRangeException(string)

Constructor for InvalidRangeException

```
public InvalidRangeException(string message)
```

Parameters

`message` string

InvalidRangeException(string, Exception)

Constructor for InvalidRangeException

```
public InvalidRangeException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Class OverlappingRangesException

Namespace: BigExcelCreator.Ranges

Assembly: BigExcelCreator.dll

When 2 or more ranges overlaps one another

```
[Serializable]
public class OverlappingRangesException : InvalidOperationException, ISerializable
```

Inheritance

object ← Exception ← SystemException ← InvalidOperationException ← OverlappingRangesException

Implements

ISerializable

Inherited Members

Exception.GetBaseException() , Exception.GetObjectData(SerializationInfo, StreamingContext) , Exception.GetType() , Exception.ToString() , Exception.Data , Exception.HelpLink , Exception.HResult , Exception.InnerException , Exception.Message , Exception.Source , Exception.StackTrace , Exception.TargetSite , Exception.SerializeObjectState , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.MemberwiseClone() , object.ReferenceEquals(object, object)

Constructors

OverlappingRangesException()

The constructor for OverlappingRangesException

```
public OverlappingRangesException()
```

OverlappingRangesException(SerializationInfo, StreamingContext)

The constructor for OverlappingRangesException

```
protected OverlappingRangesException(SerializationInfo serializationInfo,
StreamingContext streamingContext)
```

Parameters

`serializationInfo` SerializationInfo

`streamingContext` StreamingContext

OverlappingRangesException(string)

The constructor for OverlappingRangesException

```
public OverlappingRangesException(string message)
```

Parameters

`message` string

OverlappingRangesException(string, Exception)

The constructor for OverlappingRangesException

```
public OverlappingRangesException(string message, Exception innerException)
```

Parameters

`message` string

`innerException` Exception

Namespace BigExcelCreator.Styles

Classes

DifferentialStyleElement

A style to be converted to an entry of a stylesheet.

Used in conditional formatting

StyleElement

A style to be converted to an entry of a stylesheet

StyleList

Manages styles and generates stylesheets

Class DifferentialStyleElement

Namespace: BigExcelCreator.Styles

Assembly: BigExcelCreator.dll

A style to be converted to an entry of a stylesheet.

Used in conditional formatting

```
public class DifferentialStyleElement
```

Inheritance

object ← DifferentialStyleElement

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Alignment

A Alignment to overwrite when the differential format is applied

```
public Alignment Alignment { get; }
```

Property Value

Alignment

Border

A Border to overwrite when the differential format is applied

```
public Border Border { get; }
```

Property Value

Border

DifferentialFormat

A DifferentialFormat representing this style

```
public DifferentialFormat DifferentialFormat { get; }
```

Property Value

DifferentialFormat

Fill

A Fill to overwrite when the differential format is applied

```
public Fill Fill { get; }
```

Property Value

Fill

Font

A Font to overwrite when the differential format is applied

```
public Font Font { get; }
```

Property Value

Font

Name

Given name of a differential style

```
public string Name { get; }
```

Property Value

string

NumberingFormat

A NumberingFormat to overwrite when the differential format is applied

```
public NumberingFormat NumberingFormat { get; }
```

Property Value

NumberingFormat

Class StyleElement

Namespace: BigExcelCreator.Styles

Assembly: BigExcelCreator.dll

A style to be converted to an entry of a stylesheet

```
public class StyleElement
```

Inheritance

object ← StyleElement

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

StyleElement(string, int?, int?, int?, int?, Alignment)

The constructor for StyleElement

```
public StyleElement(string name, int? fontIndex, int? fillIndex, int? borderIndex,
int? numberFormatIndex, Alignment alignment)
```

Parameters

`name` string

`fontIndex` int?

`fillIndex` int?

`borderIndex` int?

`numberFormatIndex` int?

`alignment` Alignment

Properties

BorderIndex

Border index in the border list of StyleList

```
public int BorderIndex { get; }
```

Property Value

int

FillIndex

Fill index in the fill list of StyleList

```
public int FillIndex { get; }
```

Property Value

int

FontIndex

Font index in the font list of StyleList

```
public int FontIndex { get; }
```

Property Value

int

Name

Given name of a style

```
public string Name { get; }
```

Property Value

string

NumberFormatIndex

NumberFormat index in the Number format list of StyleList

```
public int NumberFormatIndex { get; }
```

Property Value

int

Style

A CellFormat object representing a style

```
public CellFormat Style { get; }
```

Property Value

CellFormat

Class StyleList

Namespace: BigExcelCreator.Styles

Assembly: BigExcelCreator.dll

Manages styles and generates stylesheets

```
public class StyleList
```

Inheritance

object ← StyleList

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

StyleList()

Creates a style list and populates with default styles

```
public StyleList()
```

Properties

DifferentialStyleElements

Differential styles.

Used in COnditional formatting

```
public IList<DifferentialStyleElement> DifferentialStyleElements { get; }
```

Property Value

IList<DifferentialStyleElement>

Styles

Main styles

```
public IList<StyleElement> Styles { get; }
```

Property Value

IList<StyleElement>

Methods

GetIndexByName(string)

Gets the index of a named style

```
public int GetIndexByName(string name)
```

Parameters

`name` string

The name of the style to look for.

Returns

int

The index of the named style, or -1 if not found.

GetIndexByName(string, out StyleElement)

Gets the index of a named style.

```
public int GetIndexByName(string name, out StyleElement styleElement)
```

Parameters

`name` string

The name of the style to look for.

`styleElement` StyleElement

A copy of the found style.

Returns

int

The index of the named style, or -1 if not found.

GetIndexDifferentialByName(string)

Gets the index of a named differential style.

```
public int GetIndexDifferentialByName(string name)
```

Parameters

`name` string

The name of the differential style to look for.

Returns

int

The index of the named differential style, or -1 if not found.

GetIndexDifferentialByName(string, out DifferentialStyleElement)

Gets the index of a named differential style.

```
public int GetIndexDifferentialByName(string name, out DifferentialStyleElement
differentialStyleElement)
```

Parameters

`name` string

The name of the differential style to look for.

`differentialStyleElement` DifferentialStyleElement

A copy of the found differential style.

Returns

int

The index of the named differential style, or -1 if not found.

GetStylesheet()

Generates a Stylesheet to include in an Excel document

```
public Stylesheet GetStylesheet()
```

Returns

Stylesheet

Stylesheet: A stylesheet

NewDifferentialStyle(string, Font, Fill, Border, NumberingFormat, Alignment)

Generates, stores and returns a new differential style

```
public DifferentialStyleElement NewDifferentialStyle(string name, Font font = null,
Fill fill = null, Border border = null, NumberingFormat numberingFormat = null,
```

```
Alignment alignment = null)
```

Parameters

`name` string

A unique name to find the inserted style later

`font` Font

Font

`fill` Fill

Fill

`border` Border

Border

`numberingFormat` NumberingFormat

NumberingFormat

`alignment` Alignment

Alignment

Returns

DifferentialStyleElement

The DifferentialStyleElement generated

NewStyle(Font, Fill, Border, NumberingFormat, Alignment, string)

Generates, stores and returns a new style

```
public StyleElement NewStyle(Font font, Fill fill, Border border, NumberingFormat
numberingFormat, Alignment alignment, string name)
```

Parameters

`font` Font

Font

`fill` Fill

Fill

`border` Border

Border

`numberingFormat` NumberingFormat

NumberingFormat

`alignment` Alignment

Alignment

`name` string

A unique name to find the inserted style later

Returns

StyleElement

The StyleElement generated

NewStyle(Font, Fill, Border, NumberingFormat, string)

Generates, stores and returns a new style

```
public StyleElement NewStyle(Font font, Fill fill, Border border, NumberingFormat
numberingFormat, string name)
```

Parameters

`font` Font

Font

`fill` Fill

Fill

`border` Border

Border

`numberingFormat` NumberingFormat

NumberingFormat

`name` string

A unique name to find the inserted style later

Returns

StyleElement

The StyleElement generated

NewStyle(int?, int?, int?, int?, Alignment, string)

Generates, stores and returns a new style.

```
public StyleElement NewStyle(int? fontId, int? fillId, int? borderId, int?
numberingFormatId, Alignment alignment, string name)
```

Parameters

`fontId` int?

Index of already inserted font

`fillId` int?

Index of already inserted fill

`borderId` int?

Index of already inserted border

`numberingFormatId` int?

Index of already inserted numbering format

`alignment` Alignment

Alignment

`name` string

A unique name to find the inserted style later

Returns

StyleElement

The StyleElement generated

Remarks

If the inserted indexes don't exist when the stylesheet is generated, the file might fail to open

To avoid such problems, use NewStyle(Font, Fill, Border, NumberingFormat, string) or NewStyle(Font, Fill, Border, NumberingFormat, Alignment, string) instead

This method should be private, but it's kept public for backwards compatibility reasons.

Exceptions

ArgumentOutOfRangeException

Thrown when any of the provided indexes are less than 0