Namespace BigExcelCreator

Classes

BigExcelWriter

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

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. NuGet Source

```
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()

Constructors

BigExcelWriter(Stream, SpreadsheetDocumentType)

Creates a document into `stream`

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

Parameters

`stream` Stream

Where to store the document. `MemoryStream` is recommended

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

BigExcelWriter(Stream, SpreadsheetDocumentType, Stylesheet)

Creates a document into `stream`

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

Parameters

`stream` Stream

Where to store the document. `MemoryStream` is recommended

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

`stylesheet` Stylesheet

A Stylesheet for the document. See GetStylesheet()

BigExcelWriter(Stream, SpreadsheetDocumentType, bool)

Creates a document into `stream`

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

Parameters

`stream` Stream

Where to store the document. `MemoryStream` is recommended

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

`skipCellWhenEmpty` bool

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)

Creates a document into `stream`

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

Parameters

`stream` Stream

Where to store the document. `MemoryStream` is recommended

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

`skipCellWhenEmpty` bool

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

A Stylesheet for the document. See GetStylesheet()

BigExcelWriter(string, SpreadsheetDocumentType)

Creates a document into a file located in `path`

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

Parameters

`path` string

Path where the document will be saved

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

BigExcelWriter(string, SpreadsheetDocumentType, Stylesheet)

Creates a document into a file located in `path`

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

Parameters

`path` string

Path where the document will be saved

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

`stylesheet` Stylesheet

A Stylesheet for the document. See GetStylesheet()

BigExcelWriter(string, SpreadsheetDocumentType, bool)

Creates a document into a file located in `path`

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

Parameters

`path` string

Path where the document will be saved

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

`skipCellWhenEmpty` bool

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)

Creates a document into a file located in `path`

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

Parameters

`path` string

Path where the document will be saved

`spreadsheetDocumentType` SpreadsheetDocumentType

Document type. Only `SpreadsheetDocumentType.Workbook` is tested

`skipCellWhenEmpty` bool

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

A Stylesheet for the document. See GetStylesheet()

Properties

Document

The main document

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

Property Value

SpreadsheetDocument

Path

Created file will be saved to: ...

(null when not saving to file)

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

Property Value

string

PrintGridLinesInCurrentSheet

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

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

Property Value

bool

Exceptions

NoOpenSheetException

When there is no open sheet

PrintRowAndColumnHeadingsInCurrentSheet

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

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

Property Value

bool

Exceptions

NoOpenSheetException

When there is no open sheet

ShowGridLinesInCurrentSheet

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

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

Property Value

bool

Exceptions

NoOpenSheetException

When there is no open sheet

ShowRowAndColumnHeadingsInCurrentSheet

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

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

Property Value

bool

Exceptions

NoOpenSheetException

When there is no open sheet

SkipCellWhenEmpty

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.

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

Property Value

bool

SpreadsheetDocumentType

Document type

only `SpreadsheetDocumentType.Workbook` is tested

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

Property Value

SpreadsheetDocumentType

Stream

Created file will be saved to: ...

(null when not saving to Stream)

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

Property Value

Stream

Methods

AddAutofilter(CellRange, bool)

Adds autofilter. Only one filter per sheet is allowed.

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

Parameters

`range` CellRange

Where to add the filter (header cells)

`overwrite` bool

Replace active filter

Exceptions

ArgumentNullException

Null range

NoOpenSheetException

When no open sheet

SheetAlreadyHasFilterException

When there is already a filter an `overwrite` is set to false

ArgumentOutOfRangeException

When range height is not exactly one row

AddAutofilter(string, bool)

Adds autofilter. Only one filter per sheet is allowed.

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

Parameters

`range` string

Where to add the filter (header cells)

`overwrite` bool

Replace active filter

Exceptions

ArgumentNullException

Null range

NoOpenSheetException

When no open sheet

SheetAlreadyHasFilterException

When there is already a filter an `overwrite` is set to false

ArgumentOutOfRangeException

When range height is not exactly one row

InvalidRangeException

When `range` is not a valid range

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

Adds conditional formatting based on cell value

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

Parameters

`cellRange` CellRange

Cell to apply format to

`operator` ConditionalFormattingOperatorValues

`value` string

Compare cell value to this

`format` int

Index of differential format in stylesheet. See GetIndexDifferentialByName(string)

`value2` string

When `operator` requires 2 parameters, compare cell value to this as second parameter

Exceptions

ArgumentOutOfRangeException

When format is less than 0

ArgumentNullException

When `value` is null OR `operator` requires 2 arguments and `value2` is null

NoOpenSheetException

When there is no open sheet

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

Adds conditional formatting based on cell value

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

Parameters

`reference` string

Cell to apply format to

`operator` ConditionalFormattingOperatorValues

`value` string

Compare cell value to this

`format` int

Index of differential format in stylesheet. See GetIndexDifferentialByName(string)

`value2` string

When `operator` requires 2 parameters, compare cell value to this as second parameter

Exceptions

ArgumentOutOfRangeException

When format is less than 0

ArgumentNullException

When `value` is null OR `operator` requires 2 arguments and `value2` is null

NoOpenSheetException

When there is no open sheet

InvalidRangeException

When `reference` is not a valid range

AddConditionalFormattingDuplicatedValues(CellRange, int)

Adds conditional formatting to duplicated values

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

Parameters

`cellRange` CellRange

Cell to apply format to

`format` int

Index of differential format in stylesheet. See GetIndexDifferentialByName(string)

Exceptions

ArgumentOutOfRangeException

When format is less than 0

NoOpenSheetException

When there is no open sheet

AddConditionalFormattingDuplicatedValues(string, int)

Adds conditional formatting to duplicated values

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

Parameters

`reference` string

Cell to apply format to

`format` int

Index of differential format in stylesheet. See GetIndexDifferentialByName(string)

Exceptions

ArgumentOutOfRangeException

When format is less than 0

NoOpenSheetException

When there is no open sheet

InvalidRangeException

When `reference` is not a valid range

AddConditionalFormattingFormula(CellRange, string, int)

Adds conditional formatting based on a formula

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

Parameters

`cellRange` CellRange

Cell to apply format to

`formula` string

Formula. Format will be applied when this formula evaluates to true

`format` int

Index of differential format in stylesheet. See GetIndexDifferentialByName(string)

Exceptions

ArgumentNullException

When formula is null or empty string

ArgumentNullException

`cellRange` is `null`.

ArgumentOutOfRangeException

When format is less than 0

NoOpenSheetException

When there is no open sheet

AddConditionalFormattingFormula(string, string, int)

Adds conditional formatting based on a formula

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

Parameters

`reference` string

Cell to apply format to

`formula` string

Formula. Format will be applied when this formula evaluates to true

`format` int

Index of differential format in stylesheet. See GetIndexDifferentialByName(string)

Exceptions

ArgumentNullException

When formula is null or empty string

ArgumentOutOfRangeException

When format is less than 0

NoOpenSheetException

When there is no open sheet

InvalidRangeException

When `reference` is not a valid range

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

Adds a decimal number validator to a 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

`firstOperand` decimal

`validationType` DataValidationOperatorValues

`allowBlank` bool

`showInputMessage` bool

`showErrorMessage` bool

`secondOperand` decimal?

Exceptions

ArgumentNullException

NoOpenSheetException

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

Adds a decimal number validator to a 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

`firstOperand` decimal

`validationType` DataValidationOperatorValues

`allowBlank` bool

`showInputMessage` bool

`showErrorMessage` bool

`secondOperand` decimal?

Exceptions

ArgumentNullException

NoOpenSheetException

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

Adds an integer (whole) number validator to a 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

`firstOperand` int

`validationType` DataValidationOperatorValues

`allowBlank` bool

`showInputMessage` bool

`showErrorMessage` bool

`secondOperand` int?

Exceptions

ArgumentNullException

NoOpenSheetException

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

Adds an integer (whole) number validator to a 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

`firstOperand` int

`validationType` DataValidationOperatorValues

`allowBlank` bool

`showInputMessage` bool

`showErrorMessage` bool

`secondOperand` int?

Exceptions

ArgumentNullException

NoOpenSheetException

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

Adds a list validator to a range based on a formula

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

Parameters

`range` CellRange

Cells to validate

`formula` string

Validation formula

`allowBlank` bool

`showInputMessage` bool

`showErrorMessage` bool

Exceptions

ArgumentNullException

When `range` is null

NoOpenSheetException

When there is no open sheet

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

Adds a list validator to a range based on a formula

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

Parameters

`range` string

Cells to validate

`formula` string

Validation formula

`allowBlank` bool

`showInputMessage` bool

`showErrorMessage` bool

Exceptions

ArgumentNullException

When `range` is null

NoOpenSheetException

When there is no open sheet

InvalidRangeException

When `range` is not a valid range

BeginRow()

Creates a new row

```
public void BeginRow()
```

Exceptions

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

BeginRow(bool)

Creates a new row

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

Parameters

`hidden` bool

Hides the row when true

Exceptions

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

BeginRow(int)

Creates a new row

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

Parameters

`rownum` int

Row index

Exceptions

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

OutOfOrderWritingException

If attempting to write rows out of order

BeginRow(int, bool)

Creates a new row

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

Parameters

`rownum` int

Row index

`hidden` bool

Hides the row when true

Exceptions

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

OutOfOrderWritingException

If attempting to write rows out of order

CloseDocument()

Closes the document

```
public void CloseDocument()
```

CloseSheet()

Closes a sheet

```
public void CloseSheet()
```

Exceptions

NoOpenSheetException

When there is no open sheet

Comment(string, CellRange, string)

Adds a comment to a cell

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

Parameters

`text` string

Comment text

`cellRange` CellRange

Commented cell

`author` string

Comment Author

Exceptions

ArgumentOutOfRangeException

When `author` is null or an empty string OR `cellRange` is not a single cell

NoOpenSheetException

When there is no open sheet

ArgumentNullException

`cellRange` is `null`.

Comment(string, string, string)

Adds a comment to a cell

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

Parameters

`text` string

Comment text

`reference` string

Commented cell

`author` string

Comment Author

Exceptions

ArgumentOutOfRangeException

When `author` is null or an empty string OR `reference` is not a single cell

NoOpenSheetException

When there is no open sheet

InvalidRangeException

When `reference` is not a valid range

CreateAndOpenSheet(string)

Creates a new sheet and prepares the writer to use it.

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

Parameters

`name` string

Names the sheet

Exceptions

SheetAlreadyOpenException

When a sheet is already open

CreateAndOpenSheet(string, SheetStateValues)

Creates a new sheet and prepares the writer to use it.

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

Parameters

`name` string

Names the sheet

`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

When a sheet is already open

CreateAndOpenSheet(string, IList<Column>)

Creates a new sheet and prepares the writer to use it.

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

Parameters

`name` string

Names the sheet

`columns` IList<Column>

Use this to set the columns' width

Exceptions

SheetAlreadyOpenException

When a sheet is already open

CreateAndOpenSheet(string, IList<Column>, SheetStateValues)

Creates a new sheet and prepares the writer to use it.

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

Parameters

`name` string

Names the sheet

`columns` IList<Column>

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

When a sheet is already open

Dispose()

Saves and closes the document.

```
public void Dispose()
```

Dispose(bool)

Saves and closes the document.

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

Parameters

`disposing` bool

EndRow()

Closes a row

```
public void EndRow()
```

Exceptions

NoOpenRowException

When there is no open row

~BigExcelWriter()

The finalizer

```
protected ~BigExcelWriter()
```

MergeCells(CellRange)

Merges cells

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

Parameters

`range` CellRange

Cells to merge

Exceptions

ArgumentNullException

When `range` is null

NoOpenSheetException

When there is no open sheet

OverlappingRangesException

When trying to merge already merged cells

MergeCells(string)

Merges cells

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

Parameters

`range` string

Cells to merge

Exceptions

InvalidRangeException

When `range` is not a valid range

NoOpenSheetException

When there is no open sheet

OverlappingRangesException

When trying to merge already merged cells

WriteFormulaCell(string, int)

Writes a formula to a cell

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

Parameters

`formula` string

formula to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteFormulaRow(IEnumerable<string>, int, bool)

Writes an entire formula row at once

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

Parameters

`formulas` IEnumerable<string>

List of formulas to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberCell(byte, int)

Writes a numerical value to a cell

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

Parameters

`number` byte

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(decimal, int)

Writes a numerical value to a cell

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

Parameters

`number` decimal

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(double, int)

Writes a numerical value to a cell

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

Parameters

`number` double

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(short, int)

Writes a numerical value to a cell

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

Parameters

`number` short

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(int, int)

Writes a numerical value to a cell

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

Parameters

`number` int

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(long, int)

Writes a numerical value to a cell

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

Parameters

`number` long

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(sbyte, int)

Writes a numerical value to a cell

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

Parameters

`number` sbyte

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(float, int)

Writes a numerical value to a cell

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

Parameters

`number` float

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(ushort, int)

Writes a numerical value to a cell

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

Parameters

`number` ushort

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(uint, int)

Writes a numerical value to a cell

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

Parameters

`number` uint

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberCell(ulong, int)

Writes a numerical value to a cell

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

Parameters

`number` ulong

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

WriteNumberRow(IEnumerable<byte>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<byte>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<decimal>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<decimal>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<double>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<double>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<short>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<short>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<int>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<int>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<long>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<long>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<sbyte>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<sbyte>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<float>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<float>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<ushort>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<ushort>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<uint>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<uint>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteNumberRow(IEnumerable<ulong>, int, bool)

Writes an entire numerical row at once

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

Parameters

`numbers` IEnumerable<ulong>

Lists of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

WriteTextCell(string, int, bool)

Writes a string to a cell

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

Parameters

`text` string

value to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`useSharedStrings` bool

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.

Exceptions

ArgumentOutOfRangeException

When `format` is less than 0

NoOpenRowException

When there is no open row

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

Writes an entire text row at once

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

Parameters

`texts` IEnumerable<string>

List of values to be written

`format` int

Format index inside stylesheet. See GetIndexByName(string)

`hidden` bool

Hides the row when true

`useSharedStrings` bool

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.

Exceptions

ArgumentNullException

When list is null

NoOpenSheetException

If there is no open sheet

RowAlreadyOpenException

If already inside a row

ArgumentOutOfRangeException

When `format` is less than 0

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

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

Namespace BigExcelCreator.Ranges

Classes

CellRange

Range in Excel spreadsheets

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

Range in Excel spreadsheets

```
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()

Constructors

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

Creates a range 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?

`fixedStartingColumn` bool

`startingRow` int?

`fixedStartingRow` bool

`endingColumn` int?

`fixedEndingColumn` bool

`endingRow` int?

`fixedEndingRow` bool

`sheetname` string

Exceptions

ArgumentOutOfRangeException

If any index is less than 1

InvalidRangeException

If a range makes no sense

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

Creates a fixed single cell range using coordinates indexes

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

Parameters

`column` int?

`fixedColumn` bool

`row` int?

`fixedRow` bool

`sheetname` string

Exceptions

ArgumentOutOfRangeException

If any index is less than 1

InvalidRangeException

If a range makes no sense

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

Creates a range using coordinates indexes

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

Parameters

`startingColumn` int?

`startingRow` int?

`endingColumn` int?

`endingRow` int?

`sheetname` string

Exceptions

ArgumentOutOfRangeException

If any index is less than 1

InvalidRangeException

If a range makes no sense

CellRange(int?, int?, string)

Creates a single cell range using coordinates indexes

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

Parameters

`column` int?

`row` int?

`sheetname` string

Exceptions

ArgumentOutOfRangeException

If any index is less than 1

InvalidRangeException

If a range makes no sense

CellRange(string)

Parses a string into a range

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

Parameters

`range` string

Exceptions

ArgumentNullException

If `range` is null

ArgumentOutOfRangeException

If any index is less than 1

InvalidRangeException

If a range makes no sense

Properties

EndingColumn

Index of the range's last column

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

Property Value

int?

EndingColumnIsFixed

true if the ending column is fixed

Represented by '$' in the string representation

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

Property Value

bool

EndingRow

Index of the range's last row

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

Property Value

int?

EndingRowIsFixed

true if the ending row is fixed

Represented by '$' in the string representation

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

Property Value

bool

Height

Range's height

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

Property Value

int

IsInfiniteCellRange

true if the range is infinite in any direction

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

Property Value

bool

IsInfiniteCellRangeCol

true if the range is infinite in any column

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

Property Value

bool

IsInfiniteCellRangeRow

true if the range is infinite in any row

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

Property Value

bool

IsSingleCellRange

true if the range represents a single column

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

Property Value

bool

RangeString

A string representing a range with the sheet's name (if available)

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

Property Value

string

RangeStringNoSheetName

A string representing a range without the sheet's name

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

Property Value

string

Sheetname

The name of the range's sheet (if available)

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

Property Value

string

StartingColumn

Index of the range's first column

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

Property Value

int?

StartingColumnIsFixed

true if the starting column is fixed

Represented by '$' in the string representation

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

Property Value

bool

StartingRow

Index of the range's first row

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

Property Value

int?

StartingRowIsFixed

true if the starting row is fixed

Represented by '$' in the string representation

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

Property Value

bool

Width

Range's width

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

Property Value

int

Methods

CompareTo(CellRange)

Comparison method

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

Parameters

`other` CellRange

Another range

Returns

int

See CompareTo(object)

Equals(CellRange)

Range equals

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

Parameters

`other` CellRange

Another range

Returns

bool

true if ranges are equal. false otherwise.

Equals(object)

Range equals

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

Parameters

`obj` object

Another range

Returns

bool

true if ranges are equal. false otherwise.

GetHashCode()

Returns the hash code for this range

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

Returns

int

RangeOverlaps(CellRange)

Compares ranges and returns true if they share any cell

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

Parameters

`other` CellRange

Returns

bool

Exceptions

ArgumentNullException

Operators

operator ==(CellRange, CellRange)

The equality operator

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

Parameters

`left` CellRange

`right` CellRange

Returns

bool

operator >(CellRange, CellRange)

The greater than operator

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

Parameters

`left` CellRange

`right` CellRange

Returns

bool

operator >=(CellRange, CellRange)

The greater or equal than operator

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

Parameters

`left` CellRange

`right` CellRange

Returns

bool

operator !=(CellRange, CellRange)

The inequality operator

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

Parameters

`left` CellRange

`right` CellRange

Returns

bool

operator <(CellRange, CellRange)

The less than operator

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

Parameters

`left` CellRange

`right` CellRange

Returns

bool

operator <=(CellRange, CellRange)

The less or equal than operator

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

Parameters

`left` CellRange

`right` CellRange

Returns

bool

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 : Exception, ISerializable
```

Inheritance

object ← Exception ← 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

Style to look for

Returns

int

GetIndexByName(string, out StyleElement)

Gets the index of a named style

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

Parameters

`name` string

Style to look for

`styleElement` StyleElement

A copy of the found style

Returns

int

GetIndexDifferentialByName(string)

Gets the index of a named differential style

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

Parameters

`name` string

Style to look for

Returns

int

GetIndexDifferentialByName(string, out DifferentialStyleElement)

Gets the index of a named differential style

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

Parameters

`name` string

Style to look for

`differentialStyleElement` DifferentialStyleElement

A copy of the found style

Returns

int

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.

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

```
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