Новые возможности EhLib.VCL 10.0

(Новые возможности ваших приложений)

Краткое описание новых возможностей этой версии:

* В класс TDatasetFeaturesEh добавлен метод WriteDataSetToMemTable для быстрой загрузки данных из разных DataSet’ов в TMemTableEh.
* В класс TXlsMemFileEh добавлена возможность загрузки Xlsx файлов.
* Расширены возможности класса TXlsMemFileEh в плане управления данными: Группировка строк, Параметры строк (Высота, Видимость), Масштаб, Параметры печати и т.д.
* Расширенная версия процедур выгрузки DBGridEh в Xlsx файл.
* Расширенная версия процедур выгрузки DBGridEh в текстовый файл.
* История поиска в DBGridEh.SeachPanel
* Поддержка High DPI мониторов
* Изменения в интерфейсной части модулей библиотеки
* Работа с классом TXlsMemFileEh

# ***В класс TDatasetFeaturesEh добавлен метод WriteDataSetToMemTable для быстрой загрузки данных из разных DataSet’ов в TMemTableEh***

В метод TMemTableEh.LoadFromDataSet добавлена возможность “быстрой” загрузки данных без использования классического варианта DataSet.Append, DataSet.Fields[i].Value := v.

Быстрая загрузки будет использоваться только для DataSet’ов, для которых зарегистрирован класс TDatasetFeaturesEh с перегруженной функцией WriteDataSetToMemTable.

В текущей версии библиотеки быстрая загрузка работает для DataSet’ов типа TADOQuery и TCustomADODataSet и его наследников.

Пример реализации метода WriteDataSetToMemTable смотрите в классе TADOSQLDatasetFeaturesEh (Модуль EhLibADO). Метод TADOSQLDatasetFeaturesEh.WriteDataSetToMemTable.

# ***В класс TXlsMemFileEh добавлена возможность загружать Xlsx файлы.***

В класс TXlsMemFileEh добавлен метод LoadFromFile.

Теперь есть возможность загрузить содержимое Xlsx файла в объект TXlsMemFileEh, изменить содержимое таблицы и сохранить обратно в файл.

При загрузке данных поддерживаются большинство основных (но не всех) элементов Xlsx файла.

В текущей версии библиотеки функция LoadFromFile и XlsMemFile поддерживают следующие элементы Xlsx файла.

* WorkSheet - рабочие листы. Имена и цвета рабочих листов.
* Строки и столбцы рабочего листа. Ширина\Высота, видимость строк и столбцов.
* Содержимое ячеек рабочего листа. Текст, числа, даты.
* Цвет и узор закраски ячейки.
* Цвет, название, размер и стиль шрифта.
* Выравнивание, перенос текста, объединение ячеек.
* Вертикальные, горизонтальные, наклонные разделительные линии.
* Свойства печати: Масштаб, заголовки листа.
* Автофильтр.
* Закрепленные-непрокручиваемые области листа.
* Формулы (только как текст, без расчета).
* Группировка строк, Группировка столбцов.

В текущей версии не поддерживается:

Градиентная закраска ячеек

Форматирование отдельных участков текста внутри ячейки (RichText в ячейке)

Условное форматирование (Форматирование на основе формул)

См. пример использования загрузки и изменения Xlsx файлов в Демо проектах:

<EhLib Archive>\Demos\XlsFileEh\Project1.dpr

<EhLib Archive>\Demos\DBGridEh.ExportToXlsx\Project1.dpr

<EhLib Archive>\Demos\MainDemo\Project1.dpr (Кнопка на тулбаре – Выгрузить в Excel).

# ***Расширены возможности класса TXlsMemFileEh в плане управления данными: Группировка строк, Параметры строк (Высота, Видимость), Масштаб, Параметры печати и т.д.***

Расширены возможности класса TXlsMemFileEh в плане управления данными: Группировка строк, Параметры строк (Высота, Видимость), Масштаб, Ширина колонки по умолчанию, Высота строки по умолчанию, Параметры печати, Цвет закладки рабочего листа, Отступы станицы.

# ***Расширенная версия процедур выгрузки DBGridEh в Xlsx файл***

В версии 9.5 присутствовала функция для выгрузки DBGridEh в Xlsx файла со следующими параметрами

procedure ExportDBGridEhToXlsx(DBGridEh: TCustomDBGridEh; const FileName: String; Options: TDBGridEhExportAsXlsxOptions; IsSaveAll: Boolean = True);

Где Options это комбинация значений

xlsxColoredEh Экспортировать цвета ячеек и фона.

xlsxDataAsDisplayText Выгружать все значения ячеек в виде текста.

xlsxDataAsEditText Выгружать все значения ячеек в виде текста. Использовать текст в режиме редактирование. Т.е. не использовать DisplayFormat.

В новую версию библиотеки добавлены две процедуры для расширенного управления форматированием данных при выгрузке данных в Xlsx файл.

**procedure** ExportDBGridEhToXlsx(DBGridEh: TCustomDBGridEh; **const** FileName: String; ExportOptions: TDBGridEhXlsMemFileExportOptions); **overload**;

**procedure** ExportDBGridEhToXlsx(DBGridEh: TCustomDBGridEh; **const** FileName: String; ExportOptions: TDBGridEhXlsMemFileExportOptions; ExporterClass: TDBGridEhToXlsMemFileExporterClass); **overload**;

Параметр ExportOptions: TDBGridEhXlsMemFileExportOptions

В новой версии процедуры вместо параметра Options используется параметр ExportOptions типа TDBGridEhXlsMemFileExportOptions.

Класс TDBGridEhXlsMemFileExportOptions содержит следующие свойства:

**property** IsExportSelecting: Boolean

Экспортировать только выделенную область грида. Если значение свойства = False то будет выгружен весь грид.

**property** ExportColumns: TColumnsEhList

Список столбцов для которые надо экспортировать. Если нужно экспортировать все видимые столбцы грида оставьте список ExportColumns пустым.

**property** IsExportTitle: Boolean

Необходимо ли экспортировать заголовки столбцов.

**property** IsExportFooter: Boolean

Необходимо ли экспортировать футер грида.

**property** IsExportFontFormat: Boolean

Необходимо ли экспортировать формат шрифта: Название, стиль, цвет.

**property** IsExportFillColor: Boolean

Необходимо ли экспортировать цвета ячеек.

**property** IsExportCellFormat: Boolean

Необходимо ли экспортировать формат ячеек. Вертикальное и горизонтальное выравнивание.

**property** IsExportDisplayFormat: Boolean

Необходимо ли экспортировать формат чисел и дат указанных в свойстве Column.DisplayFormat или Number/DateField.DisplayFormat

**property** IsCreateAutoFilter: Boolean

Необходимо ли создавать область автофильтра.

**property** IsExportFreezeZones: Boolean

Необходимо ли экспортировать фиксированные, не прокручиваемые зоны. Зона создаются на основе заголовка грида и Frozen колонок.

**property** IsFooterSumsAsFormula: Boolean

Необходимо ли пре экспорте футера переводить рассчитанные значения Sum и Count в формулы Excel.

**property** IsExportDataGrouping: Boolean

Необходимо ли экспортировать структуру группировки записей грида.

**property** GridHeaderText: String

Общий заголовок грида.

**property** GridHeaderFont: TFont;

Шрифт общего заголовка грида.

**property** GridHeaderFontStored: Boolean;

Шрифт общего заголовка грида присвоен. Если значение свойства содержит False, то будет использовать шрифт по умолчанию, используемый в Excel. При изменении свойства GridHeaderFont, значение данного свойства автоматически меняется на True.

**property** GridFooterText: String;

Общий футер грида. Данные текст добавляется в Worksheet в данных грида.

**property** GridFooterFont: TFont;

Шрифт общего футера грида.

**property** GridFooterFontStored: Boolean;

Шрифт общего футера грида присвоен.

**property** SheetName: String;

Название закладки Worksheet.

Параметр ExporterClass: TDBGridEhToXlsMemFileExporterClass

Данный параметр должен указывать на класс наследник от TDBGridEhToXlsMemFileExporter. Вызов процедуры ExportDBGridEhToXlsx с данным параметров нужно использовать когда необходимо изменить алгоритм экспорта или детали форматирования при экспорте данных из DBGridEh в Xlsx файл. Перед вызовом процедуры напишите наследник от класса TDBGridEhToXlsMemFileExporter и переопределите необходимые виртуальные функции ответственные за нужные детали экспорта. При вызове процедуры ExportDBGridEhToXlsx передайте указатель на наследуемый класс. Процедура создаст экземпляр указанного класса и будет использовать именно его для экспорта данных.

Пример вызова процедуры ExportDBGridEhToXlsx без параметра ExporterClass:

**procedure** TForm1.actExportToExcelExecute(Sender: TObject);

**var**

Grid: TDBGridEh;

Path: String;

FileName: String;

ExportOptions: TDBGridEhXlsMemFileExportOptions;

**begin**

Grid := TDBGridEh(ActiveControl);

GetDir(0, Path);

FileName := Path + '\DBGridEhAsXlsx.Xlsx';

ExportOptions := TDBGridEhXlsMemFileExportOptions.Create;

ExportOptions.IsExportAll := True;

ExportOptions.IsExportTitle := True;

ExportOptions.IsExportFooter := True;

ExportOptions.IsExportFontFormat := True;

ExportOptions.IsExportFillColor := True;

ExportOptions.IsCreateAutoFilter := True;

ExportOptions.IsExportFreezeZones := True;

ExportOptions.IsFooterSumsAsFormula := True;

ExportOptions.IsExportDisplayFormat := True;

ExportOptions.IsExportDataGrouping := True;

ExportOptions.SheetName := 'Text';

ExportOptions.GridHeaderText := 'GridHeaderText';

ExportOptions.GridHeaderFont := Grid.Font;

ExportOptions.GridHeaderFont.Size := 24;

ExportOptions.GridFooterText := 'GridFooterText'

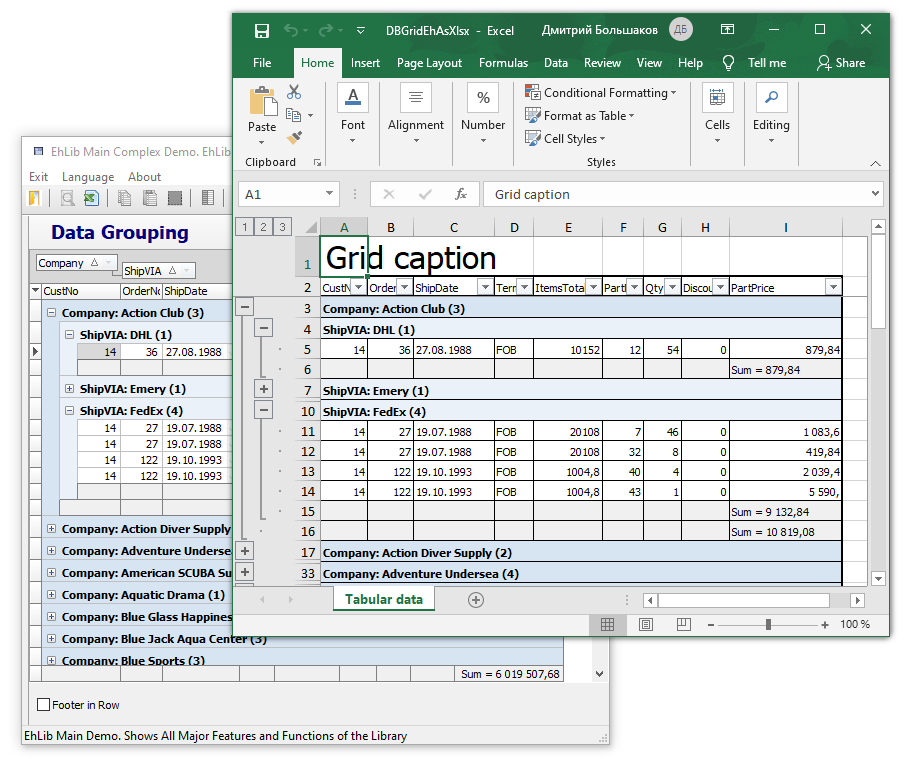
ExportDBGridEhToXlsx(Grid, FileName, ExportOptions);

ExportOptions.Free;

ShellExecute(Handle, nil, PChar(FileName), nil, nil, SW\_SHOWNORMAL);

**end**;

Результат экспорта данных:



В новой версии также добавлены процедуры для выгрузки данных в промежуточный объект TXlsMemFileEh.

**procedure** ExportDBGridEhToXlsMemFile(DBGridEh: TCustomDBGridEh;

XlsFile: TXlsMemFileEh; ExportOptions: TDBGridEhXlsMemFileExportOptions);

**procedure** ExportDBGridEhToXlsMemFile(DBGridEh: TCustomDBGridEh; XlsFile: TXlsMemFileEh; ExportOptions: TDBGridEhXlsMemFileExportOptions; ExporterClass: TDBGridEhToXlsMemFileExporterClass);

Вы можете использовать данные функции для выгрузки данных вначале в объект типа TXlsMemFileEh, замет выполнить дополнительные изменения в объекте XlsMemFile, и уже после это сохранять данные в файл на диске.

## Экспорт данных из DBGridEh с помощью класса TDBGridEhToXlsMemFileExporter.

Используя класс TDBGridEhToXlsMemFileExporter напрямую, вы получаете еще более высокий уровень гибкости при экспорте данных в XlsMemFile с последующим сохранением данных в Xlsx файл.

Например класс TDBGridEhToXlsMemFileExporter имеет такие свойства как Worksheet, FromCol и FromRow. С помощью свойства Worksheet вы можете указать в какую конкретно рабочую книгу необходимо выполнять экспорт. Свойства FromCol и FromRow задают начальный столбец и строку с которой необходимо начать формировать экспортные данные.

Написав наследника от TDBGridEhToXlsMemFileExporter вы можете перегрузить виртуальные методы класса чтобы изменить формат выгрузки данных.

Ниже приведена реализация процедуры ExportDBGridEhToXlsMemFile, которой можно понять как использовать класс TDBGridEhToXlsMemFileExporter.

**procedure** ExportDBGridEhToXlsMemFile(DBGridEh: TCustomDBGridEh; XlsFile: TXlsMemFileEh;

ExportOptions: TDBGridEhXlsMemFileExportOptions; ExporterClass: TDBGridEhToXlsMemFileExporterClass);

**var**

Exporter: TDBGridEhToXlsMemFileExporter;

**begin**

**if** ExporterClass <> nil

**then** Exporter := ExporterClass.Create

**else** Exporter := TDBGridEhToXlsMemFileExporter.Create;

Exporter.XlsFile := XlsFile;

Exporter.Grid := DBGridEh;

**if** ExportOptions <> nil **then**

Exporter.ExportOptions := ExportOptions;

Exporter.ExportGrid;

Exporter.Free;

**end**;

Если ExporterClass не указан, то используется класс TDBGridEhToXlsMemFileExporter по умолчанию.

Присваивается объект XlsFile, DBGridEh и ExportOptions.

Метод Exporter.ExportGrid выполняет экспорт данных.

После экспорта объект Exporter больше не нужен и он удаляется. В классе XlsFile остаются сформированные данные.

К коде не присваивается свойство TDBGridEhToXlsMemFileExporter.Worksheet. Поэтому Exporter формирует данные на первой рабочей книге из коллекции Workbook.Worksheets.

# ***Расширенная версия процедур выгрузки DBGridEh в текстовый файл.***

В версии 9.5 присутствовала выгрузки DBGridEh в текстовые представления использовался набор классов TDBGridEhExportAsText, TDBGridEhExportAsUnicodeText, TDBGridEhExportAsCSV с процедурой SaveDBGridEhToExportFile.

В версии EhLib 10.0 предлагается использовать один класс TDBGridEhTextExportOptions с процедурой SaveDBGridEhToTextFile.

Простейший вызов процедуры SaveDBGridEhToTextFile выглядит следующим образом:

**procedure** TfrImportExport.btnExportAsTxtClick(Sender: TObject);

**var**

Path: string;

FileName: string;

ExportOptions: TDBGridEhTextExportOptions;

**begin**

ExportOptions := TDBGridEhTextExportOptions.Create;

ExportOptions.IsExportTitle := True;

ExportOptions.IsExportFooter := False;

ExportOptions.Encoding := TEncoding.UTF8;

ExportOptions.WriteBOM := True;

ExportOptions.QuoteChar := '"';

ExportOptions.CellDelimiter := #09;

ExportOptions.TrailingLineDelimiter := True;

ExportOptions.UseEditFormat := False;

GetDir(0,Path);

FileName := Path + '\DBGridEh1Export.Txt';

SaveDBGridEhToTextFile(DBGridEh1, FileName, ExportOptions);

ExportOptions.Free;

ShellExecute(Handle, nil, PChar(FileName), nil, nil, SW\_SHOWNORMAL);

**end**;

В данном примере перед вызовом процедуры SaveDBGridEhToTextFile создается класс TDBGridEhTextExportOptions в котором прописываются настройки экспорта данных грида.

В классе TDBGridEhTextExportOptions присутствует обширный свойств, с помощью которых управлять объемом форматом выгрузки данных.

Для экспорта данных можно также использовать следующие процедуры и функции.

**procedure** SaveDBGridEhToTextFile(DBGridEh: TCustomDBGridEh; **const** FileName: String; ExportOptions: TDBGridEhTextExportOptions);

Сохранение данных грида в файл в виде текста.

**procedure** WriteDBGridEhToTextStream(DBGridEh: TCustomDBGridEh; Stream: TStream; ExportOptions: TDBGridEhTextExportOptions);

Экспорт данных грида в поток в текстовом формате.

**function** WriteDBGridEhToString(DBGridEh: TCustomDBGridEh; ExportOptions: TDBGridEhStringExportOptions): String;

Экспорт данных грида в строку.

Класс TDBGridEhStringExportOptions содержит следующие свойства

**property** ExportSelecting: Boolean;

Определяет, что нужно выгружать только выделенную область грида. Если свойство = False, то будет выгружен весь грид не зависимо от наличия выделенной области MultiSelect.

**property** CellDelimiter: String;

Строка-разделитель ячеек. Значение по умолчанию #9 (Код табуляции)

**property** LineDelimiter: String;

Строка-разделитель строк. Значение по умолчанию sLineBreak.

**property** TrailingLineDelimiter: Boolean;

Свойство задает, необходимо ли выгружать разделитель строк после последний выгруженной строки.

**property** QuoteChar: Char;

Используйте QuoteChar, чтобы получить или установить символ кавычки, который используется для заключения отдельных значений ячеек если значение содержит подстроку совпадающую по значению со свойством CellDelimiter, LineDelimiter или QuoteChar.

**property** IsExportTitle: Boolean;

Свойство указывает, необходимо ли выгружать заголовки столбцов в первой строке экспорта.

**property** IsExportFooter: Boolean;

Свойство указывает, необходимо ли выгружать футеры грида после выгрузки разделан данных.

**property** UseEditFormat: Boolean;

Свойство указывает, что при получении данных ячейки в виде строки для числовых данных и данных типа DateTime необходимо использовать формат строки используемый при редактировании ячейки. Если свойство = False, то для форматирования будет использоваться свойство Column.DisplayFormat.

**property** FormatSettings: TFormatSettings;

Свойство задает формат для числовых значений и значений типа DateTime, которые будут использоваться при преобразовании чисел и дат в текстовой представление.

**property** UseFormatSettings: Boolean;

Установите свойство в True, чтобы указать, что пре преобразовании числовых данных необходимо использовать свойство FormatSettings.

**property** ExportColumns: TColumnsEhList;

Список колонок, которые будут экспортироваться. Если свойство не заполнить, то будут экспортировать все видимые строки грида.

Класс TDBGridEhTextExportOptions наследуется от класса TDBGridEhStringExportOptions и имеет следующие дополнительные свойтсва.

**property** Encoding: TEncoding;

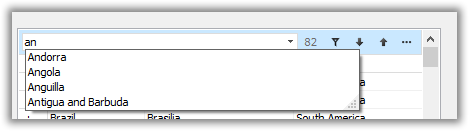
Свойство задает кодировку символов, которая будет использовать при записи экспортированной строки данных в поток или файл.

**property** WriteBOM: Boolean;

Установите свойство в True чтобы указать алгоритму, что при записи данных в поток, в начало потока необходимо записать Byte Order Mark, который содержит код кодировки, в которой были записаны текстовый данные. Описание Byte Order Mark смотрите здесь: <https://en.wikipedia.org/wiki/Byte_order_mark>

# ***История поиска в DBGridEh.SeachPanel***

В DBGridEh.SeachPanel добавлено свойство MRUList. Данное свойств позволяет настроит историю “Most Recently Used values” значений поиска в редакторе текста раздела SeachPanel.



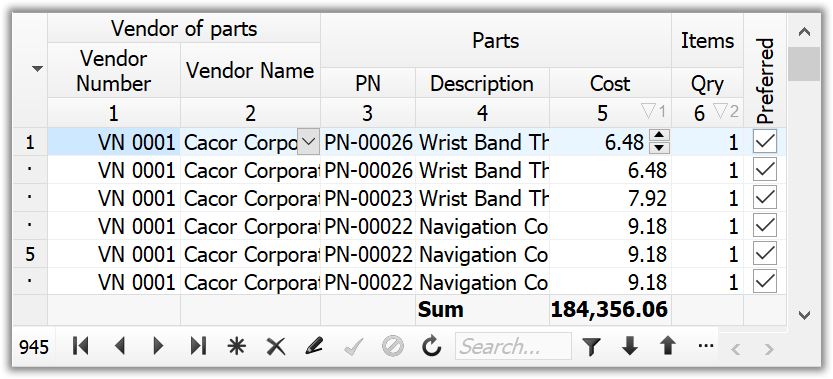
Для активации списка установите свойство DBGridEh.SeachPanel.MRUList.Active в True.

Описание остальных подсвойств свойства DBGridEh.SeachPanel.MRUList смотрите в разделе “Working with lists MRUList (last entry list)”.

По умолчанию новые значения редактора текста добавляются в список когда пользователь наживает Enter или редактор текста теряет фокус ввода.

# ***Поддержка High DPI мониторов***

В новой версии улучшена поддержка отображения данных на мониторах большого разрешения (High DPI displays). Для масштабирования данных задействуется свойство TControl.ScaleFactor, добавленное в VCL XE10.3.



# ***Изменения в интерфейсной части модулей библиотеки.***

---

Свойство IXlsFileCellsRangeEh.InsideBorder удалено.

Вместо него используйте свойства IXlsFileCellsRangeEh.Border.InsideHorizontal и IXlsFileCellsRangeEh.Border.InsideVertical.

---

При вызове процедуры MergeCell

procedure TXlsWorksheetEh.MergeCell(Col, Row, ColCount, RowCount: Integer);

параметры ColCount и RowCount должны содержать количество столбцов и строк с учетом базового столбца.

Т.е. чтобы объединить две ячейки по горизонтали надо вызвать метод с параметрами

Sheet.MergeCell(Col, Row, 2, 1);

В версии 9.5 аналогичный код выглядел как:

Sheet.MergeCell(Col, Row, 1, 0);

---

Тип TFooterValues перенесен в модуль DBGridEh

---

Тип TDBGridMultiTitleExportNodeEh перенесен в модуль DBGridEh

---

Тип TDBGridMultiTitleExportNodeMatrixEh перенесен в модуль DBGridEh

---

В библиотеку добавлен модуль DBGridEhXlsMemFileExporters. Модуль содержит класс TDBGridEhToXlsMemFileExporter, которые предназначен для экспорта компоненты TDBGridEh в объект типа TXlsMemFileEh и последующего сохранении данных в файл Xlsx.

---

В библиотеку добавлен модуль XlsFileReadersEh. Модуль содержит класс TXlsFileReaderEh, который предназначен для загрузки данных из Xlsx файлов в объект типа TXlsMemFileEh. Класс TXlsFileReaderEh используется в методе TXlsMemFileEh.LoadFromFile.

---

В DBGridEh сортмаркер типа smstClassicEh теперь отображается как smstFrameEh, т.к. картинка для smstClassicEh плохо масштабируется под HighDPI мониторы.

# ***Работа с классом TXlsMemFileEh***

Класс TXlsMemFileEh предназначен для работы с файлами в формате Xlsx.

TXlsMemFileEh позволяет загружать данные из Xlsx файлов, изменять содержимое и сохранять измененные данные в Xlsx файл.

Класс TXlsMemFileEh не использует внешних библиотек или приложений и работает с Xlsx файлами напрямую через внутренний код библиотеки.

Xlsx файл, по сути, являет упакованным архивом в формате Zip. Архив содержит набор файлов в формате Xml.

Используя стандартные процедуры библиотеки VCL класс TXlsMemFileEh имеет возможность распаковывать Xlsx файлы и выполнять разбор Xml файлов архива.

В результат разбора TXlsMemFileEh создает в себе следующие объекты и свойства.

## Свойство Workbook: TXlsWorkbookEh

Рабочая книга. TXlsMemFileEh содержит один экземпляр рабочей книги типа TXlsWorkbookEh. Рабочая книга содержит набор таблиц (TXlsWorksheetEh) и стилей TXlsFileStylesEh.

Стили Styles содержат описание формата ячеек данных в Worksheet. Каждая ячейка должна ссылаться на один из стилей из списка Styles. При заполнении данных в коде программы вам не требуется использовать свойство TXlsWorkbookEh.Styles. Внутренний код класса TXlsMemFileEh автоматически создает стили и ссылки на стили при изменении формата в ячейках таблицы.

## Свойства и методы класса TXlsWorkbookEh

Для работы с рабочими страницами (таблицами Worksheet) используется следующие свойства TXlsWorkbookEh

**function** AddWorksheet(WorksheetName: string): TXlsWorksheetEh;

Добавить новую таблице с указанным именем.

**function** FindWorksheet(WorksheetName: string): TXlsWorksheetEh;

Найти таблицу по имени.

**procedure** MoveWorksheet(FromIndex, ToIndex: Integer);

Передвинуть таблицу с позиции FromIndex в позицию ToIndex.

**procedure** RemoveWorksheet(WorksheetEh: TXlsWorksheetEh);

Удалить таблицу Worksheet.

**property** Worksheets[WorksheetId: Variant]: TXlsWorksheetEh;

Получить ссылку на таблицу по имени или по индексу. Для получения значение по имени в параметр WorksheetId передайте имя таблицы в виде строки.

**property** WorksheetCount: Integer;

Получить количество таблиц в коллекции Worksheets.

## Класс TXlsWorksheetEh

Класс TXlsWorksheetEh содержит табличные данные для одной таблицы.

Табличные данные включают в себя:

Определение колонок (Свойства Columns, DefaultColWidth)

Определение строк (Свойства Rows, DefaultRowHeight)

Значения в ячейках таблицы (Свойство Cells)

Формат для каждой ячейки (Цвет, Шрифт, Толщина линий и т.д.) (Через интерфейс IXlsFileCellsRangeEh)

Определение группировки строк (Задается через свойства Rows, OutlineRowsSummaryBelow)

Определение группировки столбцов (Задается через свойства Columns, OutlineColsSummaryRight)

Параметры печати (Подсвойства свойства PrintParams)

Задание не прокручиваемых областей (Свойства FrozenColCount, FrozenRowCount)

Класс TXlsWorksheetEh имеет следующие свойства

**property** Name: String;

Название таблица Worksheet. В Excel название отображается внизу в списке закладок.

**property** TabColor: TColor;

Цвет закладки таблицы.

**property** AutoFilterRange: TXlsFileWorksheetCellsRectEh;

Область автофильтрации

**property** CellDataExists[Col, Row: Integer]: Boolean;

Определить, создан ли объект TXlsFileCellEh в указанных координатах.

**property** Cells[Col, Row: Integer]: TXlsFileCellEh;

Доступ к ячейкам таблицы с координатами Col, Row. Координаты начинаются с нуля.

**property** Columns: TXlsFileColumnsEh;

Параметры столбцов страницы.

**property** DefaultColWidth: Double;

Ширина столбца по умолчанию.

**property** DefaultRowHeight: Double;

Высота строки по умолчанию.

**property** DefaultRowHeight: Double;

Размер области которая содержит непустые ячейки

**property** FrozenColCount: Integer;

Количество замороженных столбцов.

**property** FrozenRowCount: Integer;

Количество замороженных строк.

**property** OutlineRowsSummaryBelow: Boolean;

Заголовки групп для столбцов находится под группой. Свойство используется при группировке строк.

**property** OutlineColsSummaryRight: Boolean

Заголовки групп строк находится справа от группы. Свойство используется при группировке столбцов.

**property** PrintParams: TXlsFileWorksheetPrintParamsEh;

Параметры печати: Ориентация страницы, верхний нижний колонтитул и т.д.

**property** Rows: TXlsFileRowsEh;

Параметры строк страницы.

**property** ZoomScale: Integer;

Масштабирование. Устанавливается в процентах. Значение по умолчание 100.

## Определение колонок таблицы.

Для определения свойств колонок таблицы используйте коллекцию Columns.

Коллекция не требует добавления колонок для задания списка колонок. Колонки в коллекции создаются автоматически при обращении к элементам коллекции.

Например код:

Sheet.Columns[5].Width := 20;

Создает в коллекции список из пяти колонок (Если их еще не было в коллекции) и присваивает пяткой колонки ширину 20 Единиц ширины. Нумерация колонок в коллекции начинается с нуля.

Размер единицы ширина колонок в Excel задается по следующему правилу:

Column width measured as the number of characters of the maximum digit width of the numbers 0, 1, 2, …, 9 as rendered in the normal style's font. There are 4 pixels of margin padding (two on each side), plus 1 pixel padding for the gridlines.

Для перевода ширины экранной колонки в ширину в единицах экселя используйте функцию TXlsFileColumnsEh.ScreenToXlsWidth.

Sheet.Columns[1].Width := Sheet.Columns.ScreenToXlsWidth(DBGridEh1.Columns[1].Width);

Коллекция Sheet.Columns содержит объекты типа TXlsFileColumnEh.

Класс TXlsFileColumnEh содержит следующие свойства:

**property** OutlineLevel: Integer;

Уровень группировки. Свойство используется при группировке столбцов в таблице. Столбец не участвует в группировке, то значение содержит 0.

**property** OutlineNodeCollapsed;

Заголовок группы свернут. Свойство используется при группировке столбцов в таблице. Если столбец является заголовком группы и группа свернута то свойство содержит True. Для столбцов, которые находится в свернутой группе также необходимо установить свойство Visible в False.

**property** Width: Double;

Ширина столбца в Единицах Excel ширины.

**property** Visible: Boolean;

Столбец виден.

## Определение строк (Свойства Rows, DefaultRowHeight)

Для определения свойств строк таблицы используйте коллекцию Rows.

Коллекция не требует добавления строк для задания списка строк. Строки в коллекции создаются автоматически при обращении к элементам коллекции.

Работы со строками таблицы аналогична работе со столбцами.

## Значения в ячейках таблицы (Свойство Cells)

Для задания значений ячеек используйте свойство двумерный массив Sheet.Cells.

Массив не требует добавления ячеек для расширения размера массива. Элементы массива создаются автоматически при обращении.

Значений ячейки задается свойством Value типа Variant. Свойство поддерживает числовые, строковые, булевые типы данные и тип DateTime для задания даты и времени.

Ниже приведен пример задания значений таблицы

Sheet.Cells[0, i + 4].Value := DBGridEh1.Columns[0].Field.Value;

Sheet.Cells[1, i + 4].Value := DBGridEh1.Columns[1].Field.Value;

Sheet.Cells[2, i + 4].Value := DBGridEh1.Columns[2].Field.Value;

Кроме значений, ячейка также может содержать формулу. Для задания формулы используйте свойство Formula.

Ниже приведен пример задания формулы для одной ячейки таблицы

Sheet.Cells[1, i+4].Formula := 'SUM(E5:' + 'E' + IntToStr(i+4) + ')';

Класс TXlsMemFileEh не производит вычисления ячейки формул. При задании свойства Formula свойство Value не меняется. Однако если вы формируете файл который планируется открывать в программе MS Excel, то при открытии файла в MS Excel пользователь увидит рассчитанные данные, т.к. MS Excel пересчитывает значение формул при открытии файла.

## Форматирование ячеек в таблице.

Под форматированием ячеек понимается задание Цвета, Шрифта, Разделительный линий, Выравнивание, Отступ, Цифровой формат для одной или нескольких ячеек данных.

Для форматирования ячеек используется метод GetCellsRange и интерфейс IXlsFileCellsRangeEh.

С помощью метода Sheet.GetCellsRange необходимо получить ссылку на заданный массив ячеек в таблице (GetCellsRange возвращает массив ячеек в виде ссылки на интерфейс IXlsFileCellsRangeEh).

Затем с помощью методов интерфейса IXlsFileCellsRangeEh выполнить изменения формата ячеек и вызвать метод IXlsFileCellsRangeEh.ApplyChages, чтобы применить выполненные изменения в таблицу.

Следующий код получает ссылку на диапазон из четырех ячеек (LeftCol=0, TopRow=0, ColCount=1, RowCount=1) и устанавливает для выбранного диапазона размер шрифта равным 24.

**var**

cr: IXlsFileCellsRangeEh;

**begin**

cr := XlsFile.Workbook.Worksheets[0].GetCellsRange(0, 0, 1, 1);

cr.Font.Size := 24;

cr.ApplyChages;

**end**;

Для задания других свойств формата используйте следующие свойства интерфейса IXlsFileCellsRangeEh.

**property** Font: TXlsFileCellsRangeFontEh;

Задает свойства шрифта для выбранного диапазона.

**property** Fill: TXlsFileCellsRangeFillEh;

Задает свойства закраски ячейки для выбранного диапазона.

**property** Border: TXlsFileCellsRangeLinesEh;

Задает свойства цветов и толщины разделительных линей ячеек для выбранного диапазона.

**property** HorzAlign: TXlsFileCellHorzAlign;

Задает горизонтальное текста в ячейках для выбранного диапазона.

**property** VertAlign: TXlsFileCellVertAlign;

Задает вертикальное выравнивание текста в ячейках для выбранного диапазона.

**property** WrapText: Boolean;

Задает необходимость переноса текста но новую строку, если текст не умещается в ширину ячейки.

**property** Rotation: Integer; //Degrees

Задает поворот текста.

**property** Indent: Integer;

Задает отступ текста слева внутри ячейки.

**property** CharsFlowDirection: TXlsFileCharsFlowDirectionEh;

Задает способ расположения букв текста относительно друг друга.

**property** NumberFormat: String **read** GetNumberFormat **write** SetNumberFormat;

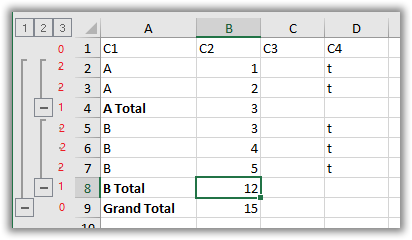
Задает числовой формат для ячеек которые содержит числа или дату/время. Формат задается по правилам MS Excel.  
Описание формата чисел смотрите в документации по MS Excel.  
<https://support.microsoft.com/en-us/office/number-format-codes-5026bbd6-04bc-48cd-bf33-80f18b4eae68>

## Настройка группировки в строках и столбцах таблицы.

Для настройки группировочных строк в таблице необходимо корректно заполнить свойства OutlineLevel и OutlineNodeCollapsed класса TXlsFileRowEh в коллекции Worksheet.Rows или Worksheet.Columns если вы группируете колонки.

OutlineLevel задает уровень группировки записи.

На скриншоте ниже красным шрифтом указаны значения, которые должны быть присвоены свойству OutlineLevel для каждой записи таблицы.



Запись 1 и 9 имеют нулевой уровень группировки.

Запись с номером 1 это заголовок таблицы и не участвует в группировке.

Запись с номером 9 это заголовок группы уровня 1. Заголовок уровня должен иметь значение в OutlineLevel на единицу меньше чем уровень группы, которую он описывает (Заголовок группы 1 - 1 = 0)

Запись 4 и 8 – этого заголовки для группы уровня 2. Соответственно в OutlineLevel содержат значение 1.

Для записей 4, 8 и 9 можно также указать что группа свернута. Для этого надо установить свойство OutlineNodeCollapsed в True. В этом случае, также, для записей, которые входят в свернутую группу надо установить свойство Visible в False.

Если заголовочные записи групп находятся над группой, то необходимо установить свойство TXlsWorksheetEh.OutlineRowsSummaryBelow в False.

Аналогично заполняются свойства OutlineLevel и OutlineNodeCollapsed для классов TXlsFileColumnEh при группировке по столбцам.

## Объединение ячеек.

Используйте методы TXlsWorksheetEh.MergeCell и TXlsWorksheetEh.UnmergerCell для объединения ячеек и удаления объединения.

Процедура MergeCell имеет следующий интерфейс вызова:

procedure MergeCell(Col, Row, ColCount, RowCount: Integer);

Параметры Col и Row указывают на базовый столбец и строку. Значения начинаются с 0.

Параметры ColCount и RowCount должны содержать количество столбцов и строк для объединения с учетом базового столбца.  
Т.е. чтобы объединить две ячейки по горизонтали надо вызвать метод с параметрами:

Sheet.MergeCell(Col, Row, 2, 1);

Процедура UnmergerCell имеет следующие параметры вызова:

procedure UnmergerCell(Col, Row: Integer);