

FastCube .Net

Руководство разработчика

Оглавление

1	Компоненты FastCube .Net	4
1.1.	Cube	5
	Загрузка куба из ранее сохраненного файла	6
	Загрузка куба из источника данных	6
1.2.	CubeGrid	6
1.2.1.	Настройка CubeGrid	7
1.3.	CubeGridToolbar	7
1.3.1.	Настройка CubeGridToolbar	7
1.4.	Slice	7
1.4.1.	Настройка среза	9
1.4.2.	Настройка структуры среза	9
1.5.	SliceGrid	10
1.5.1.	Настройка SliceGrid	10
1.6.	SliceGridToolbar	11
1.6.1.	Настройка SliceGridToolbar	12
1.7.	Chart	12
1.7.1.	Настройка Chart	12
1.8.	ChartToolbar	13
1.8.1.	Настройка ChartToolbar	13
1.9.	DataSource	13
1.9.1.	Настройка полей DataSource	14
1.9.2.	Настройка атрибутов DataSource	14
1.10.	DBDataSet	15
1.10.1.	Настройка подключения куба к базе данных	15
1.11.	DTDataSet	17
1.11.1.	Настройка подключения куба к DataTable	17
2	Настройка фильтров	18
3	Управление группами	19
4.	Настройка подсветки показателей	20
4.1	Выделение всех ячеек на основании их значений	21
4.2	Выделение только ячеек попадающих под условие	23
4.3	Выделение первых или последних ячеек	24
4.4	Выделение ячеек относительно среднего	24
4.5	Выделение повторяющихся или уникальных значений	25
4.6	Выделение ячеек, попадающих под выражение	25

1 Компоненты FastCube .Net

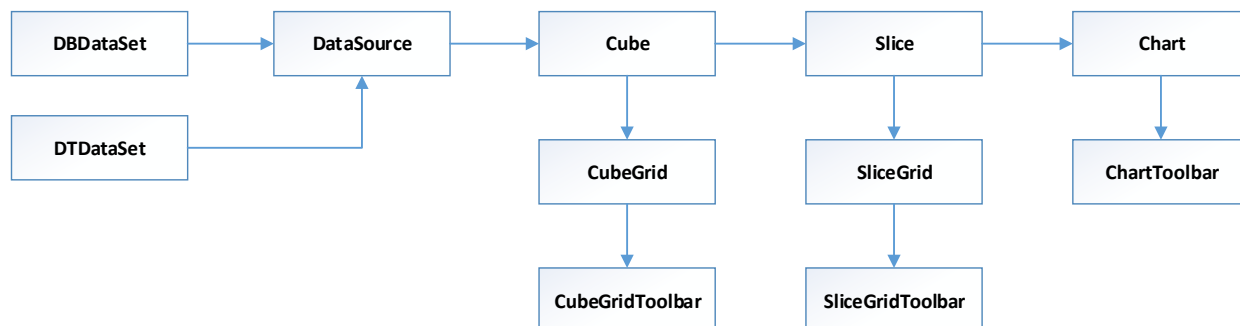
FastCube.Net представляет собой библиотеку компонентов для Visual Studio .Net. Компоненты можно разделить на визуальные и не визуальные.

Данное руководство призвано помочь вам создавать и редактировать куб, срез и другие компоненты и их свойства в коде приложения.

Список компонентов FastCube.Net:

- Cube – невидимый компонент, отвечающий за загрузку полей, уникальных значений и строк данных из файлов, баз данных и других источников, хранение и обработку структур в памяти приложения.
- CubeGrid – визуальный компонент, отображающий данные Cube в табличном виде с возможностью манипуляций со стороны пользователя.
- CubeGridToolbar – визуальный компонент, панель инструментов с функциями управления CubeGrid.
- Slice – невидимый компонент, отвечающий за выполнение OLAP-операций над данными Cube, расчет показателей и подготовку представлений для сводной таблицы и диаграммы.
- SliceGrid – визуальный компонент, отображающий данные Slice в виде сводной таблицы, с возможностью манипуляций со стороны пользователя.
- SliceGridToolbar – визуальный компонент, панель инструментов с функциями управления для SliceGrid.
- Chart – визуальный компонент, отображающий данные Slice в виде диаграммы.
- ChartToolbar – визуальный компонент, панель инструментов с функциями управления для Chart.
- DataSource – невидимый компонент, источник данных для куба.
- DBDataSet – невидимый компонент, поставляющий данные в DataSource из базы данных.
- DTDataSet – невидимый компонент, поставляющий данные в DataSource из DataTable.

Теперь, давайте рассмотрим схему взаимосвязи компонентов.



Рассматриваем схему слева направо. Объект DataSource имеет свойство DataSet. Значением этого свойства будет ссылка на один из двух объектов: DBDataSet или DTDataSet.

Объект Cube имеет свойство DataSource. А CubeGrid и Slice связаны с объектом Cube.

В свою очередь SliceGrid и Chart связаны с объектом Slice. В дальнейшем, эта схема поможет нам в настройке связи компонентов.

1.1. Cube

1) Компонент **Cube** – это основа всего FastCube. Он хранит данные куба и выступает поставщиком данных и схемы данных для других компонентов.

Данные в куб могут быть загружены из базы данных, из источника пользователя, либо из сохраненного куба.

Ниже будут приведены основные свойства и методы этого компонента.

Основные свойства:

Свойство	Описание
<code>public bool Active</code>	Активность куба (загруженность данными) true или false. После открытия куба свойство Active имеет значение true. Только в этом случае визуальные компоненты отображают данные.
<code>public string Caption</code>	Заголовок куба. Виден в CubeGrid.
<code>public bool CompressCubeFile</code>	Использовать ли компрессию при сохранении данных куба для последующего использования.
<code>public string Description</code>	Описание.
<code>public bool SkipFieldsWithErrors</code>	Указывает пропускать ли поля источника данных с ошибками (неподдерживаемый тип и т.п.) или выводить Exception
<code>public SourceType SourceType</code>	Тип источника данных указывает откуда брать данные для заполнения куба. Может принимать одно из значений: • Empty • DataSource • File • Stream • Manual

Основные методы:

Метод	Описание
<code>public void ClearGroups()</code>	Отчистить группировки. При этом все примененные группировки будут очищены.
<code>public void Close()</code>	Закрыть куб. При этом данные куба и его структуры будут очищены.
<code>public int GetFieldsCount()</code>	Получить количество полей в кубе.
<code>public int GetSourceRecordsCount()</code>	Получить количество исходных записей данных в кубе.
<code>public Utils.Variant GetSourceValue(int recordIndex, int fieldIndex)</code>	Получить значение поля источника данных в указанной записи. Параметры – индекс записи и индекс поля.
<code>public Utils.Variant GetSourceValue(int recordIndex, CubeField cubeField)</code>	Получить значение поля источника данных в указанной записи. Параметры – индекс записи и поле куба.
<code>public string GetSourceValueAsString(int recordIndex, int fieldIndex) public string</code>	Получить значение поля источника данных в указанной записи в виде строки. Параметры – индекс записи и индекс поля.
<code>GetSourceValueAsString(int recordIndex, CubeField cubeField)</code>	Получить значение поля источника данных в указанной записи в виде строки. Параметры – индекс записи и поле куба.

<code>public int GetSourceValueId(int recordIndex, int fieldIndex)</code>	Получить идентификатор уникального значения поля источника данных в указанной записи. Параметры – индекс записи и индекс поля. Параметры – индекс записи и индекс поля.
<code>public int GetSourceValueId(int recordIndex, CubeField cubeField)</code>	Получить идентификатор уникального значения поля источника данных в указанной записи.
<code>public int GetSourceValueIdAndVariant(int recordIndex, CubeField cubeField, ref Utils.Variant value)</code>	Получить идентификатор уникального значения поля источника данных и само значение в указанной записи. Параметры – индекс записи, поле куба и значение.
<code>public int GetSourceValueIdAndVariant(int recordIndex, int fieldIndex, ref Utils.Variant value)</code>	Получить идентификатор уникального значения поля источника данных и само значение в указанной записи. Параметры – индекс записи, индекс поля и значение.
<code>void Load(Stream stream)</code>	Загружает сохраненные данные куба из потока. Куб очищается перед загрузкой.
<code>void Load(string fileName)</code>	Загружает сохраненные данные куба из указанного файла. Куб очищается перед загрузкой.
<code>public void Open()</code>	Открыть куб. Загрузка куба из источника данных указанного в настройках.
<code>public void Save(Stream s, bool compress = true, object filter = null)</code>	Записывает структуру куба и данные в поток.
<code>public void Save(string fileName)</code>	Записывает структуру куба и данные в указанный файл.

Загрузка куба из ранее сохраненного файла

```
FastReport.Olap.Cube.Cube cube = new FastReport.Olap.Cube.Cube();
cube.Load("C:\\Program Files (x86)\\FastReports\\FastCube.Net
Professional\\Demos\\Data\\Cubes\\2_0_sample_en1.mdc");
```

Загрузка куба из источника данных

```
cube.SourceType = SourceType.DataSource;
cube.DataSource = DataSource1;
cube.Open();
```

1.2. CubeGrid

2) Компонент **CubeGrid** представляет собой таблицу, заполняемую данными из куба. Проще говоря визуальное отображение куба.

Свойства:

Свойство	Описание
<code>public Cube.Cube Cube</code>	Объект куба.
<code>public CubeDataZone DataZone</code>	Настройки отображения данных в сетке.

Методы:

Метод	Описание
<code>public bool Export(ExportBase export)</code>	Экспорт куба в один из форматов: • HTML; • DBF; • CSV;

	• XML; • Open Document Spreadsheet; • Excel; • Excel 2007. Возвращает успех или неудачу.
<code>public override string GetClipboardText()</code>	Возвращает строковое представление выделенных в таблице данных куба.

1.2.1. Настройка CubeGrid

```
CubeGrid cubeGrid = new CubeGrid();
cubeGrid.Dock = DockStyle.Fill;
cubeGrid.Parent = this;
cubeGrid.Cube = cube;
```

Приведенный пример, показывает, как создать cubeGrid из кода приложения. Созданный объект нужно поместить на форму (свойство Parent), и настроить расположение (свойство Dock). Кроме того, необходимо указать куб, из которого брать данные.

1.3. CubeGridToolbar

Компонент **CubeGridToolbar** – представляет собой панель инструментов, которая работает в связке с компонентом cubeGrid.



Эта панель инструментов предоставляет только один элемент – экспорт. В таблице ниже будут приведены доступные форматы экспорта.

Свойства:

Свойство	Описание
<code>public CubeGrid Grid</code>	Грид, для которого действует панель инструментов.

1.3.1. Настройка CubeGridToolbar

```
CubeGridToolbar cubeGridToolbar = new CubeGridToolbar();
cubeGridToolbar.Dock = DockStyle.Top;
cubeGridToolbar.Parent = this;
cubeGridToolbar.Grid = cubeGrid;
```

1.4. Slice

Компонент **Slice** содержит в себе настройки среза куба и выполняет операции по группировки и вычислению данных. Управлять настройками среза можно через подключенный компонент SliceGrid, из кода программы, либо загрузить ранее сохраненную схему из файла или потока.

Свойства:

Свойство	Описание
<code>public bool AutoUniqueValuesFilter</code>	Указывает применять мгновенный пересчет фильтра по уникальному значению поля или обрабатывать его в пакетном режиме. Актуально для выпадающего списка уникальных значений в SliceGrid
<code>public Cube.Cube Cube</code>	Объект куба.

<code>public FieldComparerType FieldsOrder</code>	Порядок отображения списка полей: ByIndex, ByName, ByCaption.
<code>public int ColCount</code>	Возвращает количество колонок в сводной таблице.
<code>public bool HideColZeros</code>	Скрывать пустые колонки (колонки, в которых все значения равны 0 или пустые).
<code>public bool HideRowZeros</code>	Скрывать пустые строки (строки, в которых все значения равны 0 или пустые).
<code>public bool HideTotalForSingleValue</code>	Скрывать итог если узел содержит всего одно значение.
<code>public int RowCount</code>	Возвращает количество строк в сводной таблице.
<code>public Types.Language ScriptLanguage</code>	Язык скрипта. Представляет собой перечисление со значениями: CSharp = 0, Vb = 1
<code>public string ScriptText</code>	Текст скрипта.
<code>public PermissionSet ScriptRestrictions</code>	Настройка ограничений скрипта.
<code>public SliceFields SliceFields</code>	Контейнер, который содержит все поля среза.
<code>public AxisContainer XAxisContainer</code>	Контейнер для полей, размещающихся по оси X. Для измерений.
<code>public AxisContainer YAxisContainer</code>	Контейнер для полей, размещающихся по оси Y. Для измерений.
<code>public FiltersContainer FiltersContainer</code>	Контейнер для полей располагаемых в зоне фильтров.
<code>public MeasuresContainer MeasuresContainer</code>	Контейнер, который содержит показатели.

Основные методы:

Метод	Описание
<code>public void BeginUpdate()</code>	Включить режим обновления (пакетного изменения схемы).
<code>public void EndUpdate()</code>	Завершить обновление (пакетного изменения схемы).
<code>public void Clear()</code>	Отчистить срез. Сброс всех настроек.
<code>public void Save(Stream stream, SliceSaveExtras extras = SliceSaveExtras.None)</code>	Записывает схему среза в поток. Параметр SliceSaveExtras указывает дополнительные настройки сохранения (None, Filters, Groups, Charts).
<code>public void Save(XmlDocument doc, SliceSaveExtras extras = SliceSaveExtras.None)</code>	Записывает схему среза в XML документ. Параметр SliceSaveExtras указывает дополнительные настройки сохранения (None, Filters, Groups, Charts).
<code>public void Save(string fileName, SliceSaveExtras extras = SliceSaveExtras.None)</code>	Записывает схему среза в указанный файл. Параметр SliceSaveExtras указывает дополнительные настройки сохранения (None, Filters, Groups, Charts).
<code>public bool Load(string fileName)</code>	Загрузить схему среза из файла с заданным именем. Схема среза, настройки групп, фильтры и диаграммы сбрасываются перед загрузкой. При успехе возвращает true.
<code>public bool Load(XmlDocument doc)</code>	Загрузить схему среза из XML документа. Схема среза, настройки групп, фильтры и диаграммы сбрасываются перед загрузкой. При успехе возвращает true.
<code>public bool Load(Stream stream)</code>	Загрузить схему среза из потока. Схема среза, настройки групп, фильтры и диаграммы

	сбрасываются перед загрузкой. При успехе возвращает true.
<code>public void SetColsWidth(int value)</code>	Установить ширину всех колонок. Значение в пикселях.
<code>public void SetColWidth(int columnIndex, int value)</code>	Установить ширину заданной колонки. Значение в пикселях.
<code>public void SetRowHeight(int rowIndex, int value)</code>	Установить высоту заданной строки. Значение в пикселях.
<code>public void SetRowsHeight(int value)</code>	Установить высоту всех строк. Значение в пикселях.
<code>public void Transpose()</code>	Транспонировать срез (поменять оси).

1.4.1. Настройка среза

```
FastReport.Olap.Slice.Slice slice1 = new FastReport.Olap.Slice.Slice();
slice1.Cube = cube;
```

1.4.2. Настройка структуры среза

Настройка структуры среза заключается в добавлении полей и показателей в нужные контейнеры.

Для добавления и удаления полей в контейнеры оси (XAxisContainer, YAxisContainer), служат следующие методы:

<code>public int AddSliceField(SliceField sliceField)</code>	Добавить поле в срез. Возвращает индекс добавленного элемента.
<code>public int InsertSliceFieldToPosition(SliceField sliceField, int index)</code>	Вставить поле среза в указанную позицию. Возвращает индекс добавленного элемента.
<code>public void RemoveAxisField(AxisField axisField)</code>	Удалить поле из контейнера оси.
<code>public int AddMeasuresField()</code>	Добавляет в ось поле "Показатели".
<code>public void DeleteMeasuresField()</code>	Убирает из оси поле "Показатели".

Для добавления и удаления показателей в контейнер показателей (MeasuresContainer) служат следующие методы:

<code>public int AddMeasure(MeasureField measureField)</code>	Добавить показатель в контейнер. Возвращает индекс добавленного элемента.
<code>public void DeleteMeasure(MeasureField measureField)</code>	Удалить показатель из контейнера. При этом поле переносится в область фильтров.
<code>public void DeleteMeasures()</code>	Удалить все показатели из контейнера.
<code>public void DeleteMeasure(int measureIndex)</code>	Удалить показатель из контейнера по индексу.
<code>public void InsertMeasure(MeasureField measureField, int index)</code>	Вставить показатель в указанную позицию.

```
//Начать изменение структуры среза
slice1.BeginUpdate();
//Добавить поле в контейнер оси X
slice1.YAxisContainer.AddSliceField(slice1.SliceFields.GetFieldByIndex(0));
//Вставить поле SliceField1 в контейнер оси Y в 5 позицию
slice1.YAxisContainer.InsertSliceFieldToPosition(
    slice1.SliceFields.GetFieldByName("SliceField1"), 5);
//Добавить поле "Показатели" в контейнер оси X
slice1.XAxisContainer.AddMeasuresField();
//Добавить поле в контейнер "Показатели"
slice1.MeasuresContainer.AddMeasure(new MeasureField(
```

```

        slice1,
        FastReport.Olap.Types.AggregateFunction.Sum,
        slice1.SliceFields.GetFieldByIndex(1), null, null, "Measure1",
        "Measure1", false));
//Закончить обновление структуры среза
slice1.EndUpdate();

```

Показатели можно перемещать в списке для определения порядка их отображения с помощью метода `public bool MoveMeasure(int fromIndex, int toIndex)`. Параметры: индекс элемента который нужно переместить, индекс куда переместить. При успехе возвращает true.

Методы `SetAllVisible()` и `SetNoneVisible()` позволяют скрыть все показатели или отобразить все показатели.

Для того, чтобы скрыть или отобразить конкретный показатель, используйте конструкцию: `slice1.MeasuresContainer.GetMeasureFieldByIndex(1).Visible = true;`

1.5. SliceGrid

Компонент **SliceGrid** отображает срез в виде кросс-таблицы. Это основной инструмент аналитика. Позволяет настраивать расположение полей измерений и показателей, добавлять новые, сортировать, группировать и другое.

Свойства:

Свойство	Описание
<code>public SliceDataZone DataZone</code>	Настройки отображения данных в сетке.
<code>public Slice.Slice Slice</code>	Задается объект среза.
<code>public XAxisZone XAxisZone</code>	Настройки зоны отображения оси X.
<code>public YAxisZone YAxisZone</code>	Настройки зоны отображения оси Y.
<code>public SliceItemsZone FilterFieldsZone</code>	Настройки зоны фильтров.
<code>public SliceItemsZone XFieldsZone</code>	Настройки зоны отображения измерений по оси X.
<code>public SliceItemsZone YFieldsZone</code>	Настройки зоны отображения измерений по оси Y.
<code>public SliceFieldsZone FieldsZone</code>	Настройки зоны списка полей.

Методы:

Метод	Описание
<code>public bool Export(ExportBase export)</code>	Метод экспорта отчета в один из форматов: • HTML; • DBF; • CSV; • XML; • Open Document Spreadsheet; • Excel; • Excel 2007.
<code>public void ShowFieldsEditor()</code>	Открыть окно со списком полей среза.
<code>public override string GetClipboardText()</code>	Получить текстовое представление выделенных ячеек.

1.5.1. Настройка SliceGrid

```

SliceGrid sliceGrid = new SliceGrid();
sliceGrid.Dock = DockStyle.Fill;
sliceGrid.Parent = this;
sliceGrid.Slice = slice1;

```

Этот пример показывает, как создать объект sliceGrid в коде приложения. Если такой необходимости нет, то все настройки можно произвести в инспекторе свойств.

1.6. SliceGridToolbar

Компонент **SliceGridToolbar** представляет собой визуализированную панель инструментов для SliceGrid:



Состав инструментария следующий:

1. Сохранить:
 - Куб;
 - Схему.
2. Открыть:
 - Куб;
 - Дополнительный куб (дополнение данных в активный куб);
 - Схему.
3. Отчистить настройки;
4. Экспортировать в:
 - HTML;
 - DBF;
 - CSV;
 - XML;
 - Open Document Spreadsheet;
 - Excel;
 - Excel 2007.
5. Транспонировать – поменять оси X и Y местами;
6. Скрыть пустые строки;
7. Скрыть пустые столбцы;
8. Тип сортировки строк:
 - Сортировать по значению оси;
 - Сортировать по итогам показателей;
 - Сортировать по выделенной колонке.
9. Тип сортировки колонок:
 - Сортировать по значению оси;
 - Сортировать по итогам показателей;
 - Сортировать по выделенной строке.
10. Редактировать показатели – настройки показателей. Сюда входит настройка условного выделения данных;
11. Формат отображения данных – задается для выбранной колонки или строки;
12. Список полей – список всех полей, доступных в срезе;
13. Редактор формул – по сути редактор скрипта на языке программирования C# или VB;
14. Информация – информация о срезе;

Свойства:

Свойство	Описание
<code>public SliceGrid Grid</code>	Грид, для которого действует панель инструментов

<code>public override List<MetaItem> ToolItems</code>	Список элементов панели инструментов
<code>public string DialogsDefaultPath</code>	Путь по умолчанию для диалогов открытия/сохранения

1.6.1. Настройка SliceGridToolbar

```
FastReport.Olap.Controls.SliceGridToolbar toolbar = new
FastReport.Olap.Controls.SliceGridToolbar();
toolbar.Grid = sliceGrid1;
toolbar.Parent = sliceGrid1;
toolbar.Dock = DockStyle.Top;
```

1.7. Chart

Компонент **Chart** представляет собой график, основанный на данных из среза. Он строится автоматически, достаточно лишь задать свойство `Slice`.

Свойства:

Свойство	Описание
<code>public MarksShowStyle MarksShowStyle</code>	Стиль отображения меток
<code>public SeriesType SeriesType</code>	Тип Series (колончатая, круговая и др.)
<code>public bool SkipNullPoints</code>	Пропускать пустые точки на диаграмме
<code>public AxisDataType BaseAxisDataType</code>	Тип данных базовой оси
<code>public int MeasureFieldIndex</code>	Индекс поля показателя
<code>public int SeriesFieldCount</code>	Количество полей, используемых в качестве источника для Series
<code>public int CategoriesFieldCount</code>	Количество полей, используемых в качестве источника для категорий
<code>public AxisRegion SeriesAxis</code>	Ось – источник Series.
<code>public AxisRegion CategoriesAxis</code>	Ось – источник категорий.
<code>public ChartDataType DataType</code>	Способ получения данных в диаграмме: ByAxisAxis – категории и Series из измерений. Используется для отображения одного показателя; ByAxisMeasures – категории из измерений, Series из показателей; ByMeasuresAxis – категории из показателей, Series из измерений;
<code>public Slice.Slice Slice</code>	Ссылка на объект slice
<code>public bool Frozen</code>	Заморозка позволяет зафиксировать состояние графика, так, что последующие изменения в срезе не будут на нем отражаться

Методы:

Метод	Описание
<code>public void BeginUpdate()</code>	Включить режим пакетного редактирования
<code>public void EndUpdate()</code>	Завершить пакетного редактирование
<code>public void Load(XmlItem item)</code>	Загрузить настройки графика из XML элемента
<code>public void Save(XmlItem item)</code>	Сохранить настройки графика в XML элемента

1.7.1. Настройка Chart

Настройка компонента **Chart** сводится к выбору доступного среза в свойстве `Slice`. Чтобы создать и настроить объект **Chart** в коде приложения используйте код:

```
Chart chart = new Chart();
chart.Dock = DockStyle.Fill;
chart.Parent = tabPage3;
```

```
chart.Slice = slice1;
```

В этом случае нам нужно создать объект, настроить его отображение, привязать к родительскому объекту и задать срез.

1.8. ChartToolbar

Компонент **ChartToolbar** содержит инструменты для настройки отображения диаграммы.



Состав:

- 1) Chart style – стиль диаграммы:
 - Bar - столбцы;
 - Line - линейная;
 - Point - точки;
 - Area – цветные области;
 - Pie - круговая;
 - Horiz Bar – горизонтальные полосы.
- 2) Frozen chart – заморозить текущее состояние диаграммы;
- 3) Chart properties – свойства диаграммы в отдельном окне;
- 4) Marks – метки диаграммы;
- 5) Legend – легенда диаграммы;
- 6) Copy – копировать диаграмму в виде рисунка.

Свойства:

Свойство	Описание
<code>public Chart Chart</code>	Объект Chart для которого действует эта панель инструментов
<code>public override List<MetaItem> ToolItems</code>	Список элементов панели инструментов
<code>public string DialogsDefaultPath</code>	

1.8.1. Настройка ChartToolbar

При настройке визуального компонента необходимо задать свойство Chart – это диаграмма к которой будет привязана панель инструментов.

```
ChartToolbar chartToolbar = new ChartToolbar();  
chartToolbar.Dock = DockStyle.Top;  
chartToolbar.Parent = this;  
chartToolbar.Chart = chart;
```

1.9. DataSource

Компонент **DataSource** – источник данных для куба.

Свойства:

Свойство	Описание
<code>public IBaseDataSet DataSet</code>	Набор данных – DBDataSet или DTDataSet
<code>public BaseFields<DataSourceField> Fields</code>	Список полей источника данных

Методы:

Метод	Описание
<code>public void AddFields()</code>	Загрузить поля куба из источника данных
<code>public bool Check(StringBuilder msg, bool</code>	Проверить поля на дублирование и другие

skipFieldsWithErrors)	ошибки
<code>public void Close()</code>	Обнуляет источник данных
<code>public void DeleteFields()</code>	Очищает список полей
<code>public void InitFields(bool loaded = false)</code>	Инициализация полей
<code>public bool Open()</code>	Открыть источник данных. При этом данные будут загружены.

В настройках этого компонента необходимо определить набор данных. Это может быть DBDataSet или DTDataSet.

Ниже, для перечисленных компонентов будет показана настройка из кода приложения всей цепочки получения данных.

1.9.1. Настройка полей DataSource

Настройка полей источника данных нужна, когда требуется конвертировать поле в другой тип данных, форматировать поле, или просто требуется загрузка определенных полей.

Свойство DataSource.Fields хранит описание полей источника. Само поле описывается объектом DataSourceField.

Свойства источника DataFieldProperties зависят от его типа AttributeType:

```
public enum AttributeType
{
    None = 0,
    Custom = 1,
    Reference = 2,
    DateTime = 3
}
```

Свойство DataField описывает тип данных, имя и название поля в источнике данных, признак конвертирования с указанием конечного типа данных, имя и название поля в кубе.

Примеры:

```
//Загружаем описание полей
```

```
dataSource1.AddFields();
```

```
//Изменяем название поля в кубе для поля с индексом 2
```

```
dataSource1.Fields[2].DataField.CubeFieldCaption = "Customer";
```

```
//Устанавливаем правило конвертирования для поля с именем Population
```

```
dataSource1.Fields.GetByName("Population").DataField.Convert = true;
```

```
    dataSource1.Fields.GetByName("Population").DataField.CubeFieldType =
FastReport.Olap.Types.DataType.Int;
```

1.9.2. Настройка атрибутов DataSource

Поля данных можно разбивать на атрибуты. Это работает для полей типа дата время и полей зависимых источников.

Для разбивки поля на атрибуты используется свойство поля источника SplitProperty. Атрибут может иметь вложенные атрибуты. Уровень вложенности не ограничен.

Для поля можно указывать имя атрибута, содержащего значение CaptionSource и имя атрибута для сортировки OrderSource.

```
FastReport.Olap.Cube.DataSourceField aAttribute;
```

```

        dataSource1.Fields.GetByName("123").SplitProperty.SplitPaths =
FastReport.Olap.Types.DateTimePart.Day;
        aAttribute = new
FastReport.Olap.Cube.DataSourceField(dataSource1.Fields.GetByName("Client").SplitProperty
.Attributes.Add());

```

1.10. DBDataSet

DBDataSet – набор данных для DataSource, получаемый из базы данных.

Свойства:

Свойство	Описание
<code>public IDbCommand DbCommand</code>	Команда для базы данных, которая содержит SQL запрос

Методы:

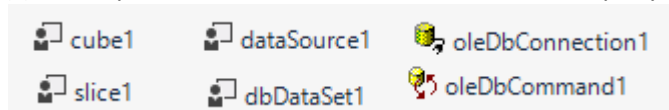
Метод	Описание
<code>public override bool AssignedSource()</code>	Проверка на наличие привязанного источника (DbCommand).
<code>public override void Close()</code>	Закрывает набор данных.
<code>public override bool Open()</code>	Открывает набор данных для чтения.

1.10.1. Настройка подключения куба к базе данных

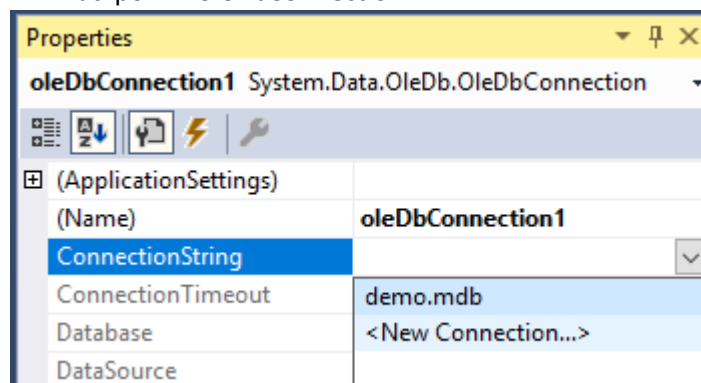
1) С помощью визуальных компонентов:

Чтобы настроить подключение к данным через DBDataSet требуется сформировать DBCommand с помощью компонента OleDbCommand. В свою очередь, для компонента OleDbCommand требуется задать подключение к базе данных с помощью компонента OleDbConnection.

Для настройки подключения к базе данных потребуются следующие компоненты:



– Настройки OleDbConnection:



Создаем новое подключение:

Add Connection ? X

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:

Database file name:

Log on to the database

User name:

Password:

☐ Save my password

– Настройка OleDbCommand:

Properties ▾ ⚙ X

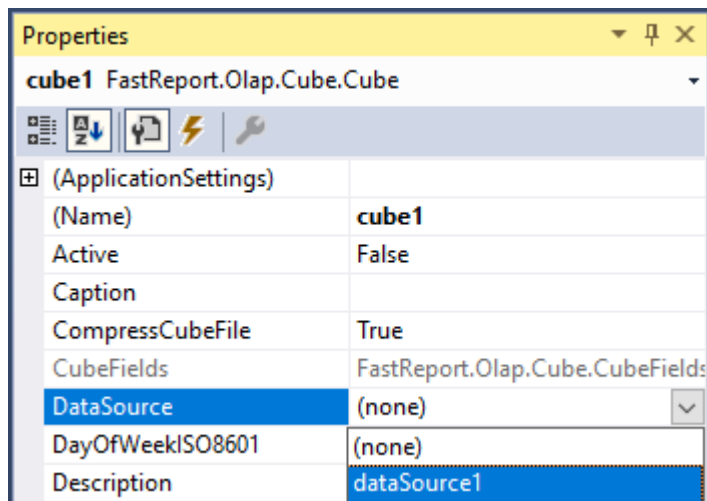
oleDbCommand1 System.Data.OleDb.OleDbCommand ▾

⚙

(ApplicationSettings)	
(Name)	oleDbCommand1
CommandText	SELECT items.OrderNo, item
CommandTimeout	30
CommandType	Text
Connection	(none) ▾
GenerateMember	Existing oleDbConnection1
Modifiers	New
Parameters	(None)
UpdatedRowSource	

Необходимо задать команду – sql запрос. А также, нужно выбрать подключение к данным.

– Настройка cube:



2) Из кода приложения:

```
OleDbCommand command = new OleDbCommand();
command.CommandText = "Select * from Sales";
dbDataSet1.DbCommand = command;
dataSource1.DataSet = dbDataSet1;
cube1.DataSource = dataSource1;
cube1.Open();
```

1.11. DTDataset

DTDataSet – набор данных для DataSource, получаемый из объекта DataTable. В свою очередь DataTable может быть заполнен данными из базы, текстового файла, кода приложения.

Свойства:

Свойство	Описание
<code>public DataTable</code> DataTable	Ссылка на таблицу

Методы:

Метод	Описание
<code>public override bool</code> AssignedSource()	Проверка на наличие привязанного источника (dataTable).
<code>public override bool</code> Open()	Открыть набор данных для чтения.

Процедура настройки подключения к DataTable с помощью визуальных компонентов очень проста.

1.11.1. Настройка подключения куба к DataTable:

- 1) Настройка визуальных компонентов подобна настройке подключения к базе данных, с той лишь разницей, что в настройке DataSet у компонента DataSource нужно выбрать dtDataSet1. Соответственно не требуются OleDbConnection и OleDbCommand.
- 2) Из кода приложения:

```
DataTable dataTable = new DataTable(); //Создаем таблицу и наполняем ее данными
...
cube1.Close(); //Закрываем куб, чтобы выгрузить из него данные (если были ранее
загружены)
```

```
dtDataSet1.DataTable = dataTable; //Для набора данных присваиваем свойству DataTable
созданную таблицу
dataSource1.DeleteFields(); //Очищаем поля в источнике данных (если были ранее загружены)
dataSource1.DataSet = dtDataSet1;
cube1.Open(); //Открываем куб, чтобы загрузить в него данные
```

2 Настройка фильтров

Фильтры служат для отбора данных для расчета в соответствии с заданными критериями.

Управлять фильтрами можно через методы и свойства поля среза.

Примеры:

```
// Начать пакетное изменение фильтра
slice1.SliceFields.GetFieldByIndex(0).BeginUpdateField();

// Закончить пакетное изменение фильтра (применить изменения)
slice1.SliceFields.GetFieldByIndex(0).EndUpdateField();

// Снять признак фильтрации у уникального значения под индексом 3 поля среза с индексом 0
slice1.SliceFields.GetFieldByIndex(0).SetAllowedUniqueValue(3, false);

// Снять признак фильтрации со всех значений поля 'FirstName'
slice1.SliceFields.GetFieldByName("FirstName").SetAllDenied();

// Установить признак фильтрации у значения 'Sergey' поля 'FirstName'
slice1.SliceFields.GetFieldByName("FirstName").SetAllowedUniqueValue(slice1.SliceFields.G
etFieldByName("FirstName").CubeField.Values.GetValueIdAtValue("Sergey"), true);

// Установить признак фильтрации у значения с индексом 12 поля 'FirstName'
slice1.SliceFields.GetFieldByName("FirstName").SetAllowedUniqueValue(12, true);

// Устанавливает признак фильтрации только у значения с индексом 4 поля среза с индексом 0
slice1.SliceFields.GetFieldByIndex(0).UniqueValueSingleIndex = 4;

// Инвертирует признак фильтрации значений поля среза с индексом 0
slice1.SliceFields.GetFieldByIndex(0).InverseFilter();

// Устанавливает признак фильтрации поля среза с индексом в соответствии с критерием,
заданным в range
slice1.SliceFields.GetFieldByIndex(0).SetRangeFilter(range);

// Установить фильтру тип "переключатель"
slice1.SliceFields.GetFieldByIndex(0).UVFilterType = UniqueValuesFilterType.Single;
```

3 Управление группами

Группы служат для повышения наглядности представления данных. С их помощью можно группировать значения измерений.

Управлять группами можно через методы и свойства поля среза и менеджера групп (GroupsManager) поля среза.

Примеры:

```
// Создать новую группу MyGroup для поля с индексом 1
```

```
var groupIndex =  
slice1.SliceFields.GetFieldByIndex(1).GroupsManager.CreateGroup("MyGroup");
```

```
// Добавить уникальное значение с индексом 1 в группу индексом groupIndex
```

```
slice1.SliceFields.GetFieldByIndex(1).GroupsManager.AddUniqueValueIdToGroup(1,  
groupIndex);
```

```
// Добавить все не группированные значения в группу OtherGroup
```

```
slice1.SliceFields.GetFieldByIndex(1).GroupsManager.CreateOtherGroup("OtherGroup");
```

4. Настройка подсветки показателей

Свойство `MeasureField.Highlights` задает правила подсветки показателей. Правила применяются последовательно, и при одновременной установке одного и того же свойства форматирования двумя и более правилами победит настройка последнего правила из списка.

Все правила наследуются от абстрактного класса `CustomHighlight`, который задает лишь базовый каркас.

Свойства:

Свойство	Описание
<code>public HighlightApply ApplyTo</code>	Задаёт виды ячеек, к которым можно применять правило.
<code>public HighlightScanDirection ScanDirection</code>	Задаёт направление анализа в котором следует рассматривать соседние ячейки для вычисления при необходимости агрегатов по группе.
<code>public virtual bool IsCustomDrawn</code>	Определяет необходимость отрисовки подсветки классом подсветки. В случае <code>false</code> настройки подсветки задаются через объект <code>Style</code> .
<code>public virtual string Caption</code>	Текстовое представление подсветки для отображения в редакторе.
<code>public virtual bool HideValue</code>	Определяет необходимость скрытия текстовое представление значения.
<code>public CustomHighlights Owner</code>	Ссылка на список <code>Highlights</code> .
<code>public int Index</code>	Индекс правила в списке <code>Highlights</code> .

Методы:

Метод	Описание
<code>public virtual void BeginUpdate()</code>	Начать групповую операцию изменения правила. При изменении свойств правила реакция не последует до вызова метода <code>EndUpdate()</code> . Рекомендуется использовать этот метод только если меняется более одного свойства.
<code>public virtual void EndUpdate()</code>	Закончить групповую операцию изменения.
<code>public virtual void Load(XmlItem item)</code>	Загрузить правило из узла XML.
<code>public virtual void Save(XmlItem item)</code>	Сохранить правило в узел XML.
<code>public virtual void Assign(CustomHighlight source)</code>	Скопировать настройки из <code>source</code> .
<code>public abstract bool AcceptCell(MeasureCell cell)</code>	Проверка применимости правила к ячейке показателя.
<code>public virtual Expression[] GetExpressions()</code>	Получить массив выражений, требуемый для работы правила (используется внутренними механизмами выполнения выражений).

Абстрактный класс `GraphicHighlight`, от которого фактически наследуются все классы подсветки добавляются следующие свойства и методы.

Свойства:

Свойство	Описание
----------	----------

<code>public Style Style</code>	Задаёт стиль ячеек.
---------------------------------	---------------------

Методы:

Метод	Описание
<code>public virtual void DrawExample(Graphics g, Rectangle r)</code>	Отрисовать пример подсветки на переданный Graphic в заданную Rectangle область. Используется в редакторе подсветки для отрисовки примера.
<code>public virtual void DrawValue(Graphics g, Rectangle r, MeasureCell Value, ref bool CanDrawImage, ref bool CanDrawText)</code>	Отрисовать значение ячейки. Только для Highlight с IsCustomDrawn = true.
<code>public virtual Style GetStyleFor(MeasureCell value)</code>	Получить стиль отрисовки для ячейки.

4.1 Выделение всех ячеек на основании их значений

За функциональность выделения всех ячеек на основании их значений отвечает класс `ContinuousHighlight`. Класс позволяет задать настройки как изменением отдельных свойств, так и применить их из заранее определенного шаблона.

Свойства:

Свойство	Описание
<code>public ContinuousHighlightKind Kind</code>	Вид подсветки.
<code>public double MinValue</code>	Минимальное значение (для шкал и гистограммы).
<code>public double MaxValue</code>	Максимальное значение (для шкал и гистограммы).
<code>public double MidValue</code>	Среднее значение (для трехцветной шкалы).
<code>public ContinuousHighlightValueType MinValueType</code>	Тип минимального значения.
<code>public ContinuousHighlightValueType MaxValueType</code>	Тип максимального значения.
<code>public ContinuousHighlightValueType MidValueType</code>	Тип среднего значения.
<code>public Color MinValueColor</code>	Цвет минимального значения.
<code>public Color MaxValueColor</code>	Цвет максимального значения.
<code>public Color MidValueColor</code>	Цвет среднего значения.
<code>public Color BarColor</code>	Цвет гистограммы.
<code>public Color FrameColor</code>	Цвет рамки гистограммы.
<code>public bool GradientDraw</code>	Использовать градиент для гистограммы.
<code>public bool ShowCellValue</code>	Показывать значение ячейки (для гистограммы и набора иконок).
<code>public string IconSet</code>	Текущий набор иконок для типа IconSet.
<code>public ImageList Images</code>	Ссылка на полный набор картинок, из которого загружаются иконки для типа IconSet.
<code>public int ImageCount</code>	Количество иконок для типа IconSet.

Методы:

Метод	Описание
<code>public void DrawIconSet(Graphics g, Rectangle r, string name)</code>	Отрисовать набор иконок.

<code>public int GetImageIndex(int icon)</code>	Получить индекс изображения для переданного индекса иконки из текущего набора иконок.
<code>public void SetImageIndex(int icon, int imageIndex)</code>	Установить индекс изображения для переданного индекса иконки из текущего набора иконок.
<code>public double GetImageValue(int icon)</code>	Получить значение для переданного индекса иконки из текущего набора иконок.
<code>public void SetImageValue(int icon, double value)</code>	Установить значение для переданного индекса иконки из текущего набора иконок.
<code>public ContinuousHighlightValueType GetImageValueType(int icon)</code>	Получить тип значения для переданного индекса иконки из текущего набора иконок.
<code>public void SetImageValueType(int icon, ContinuousHighlightValueType value)</code>	Установить тип значения для переданного индекса иконки из текущего набора иконок.
<code>public ContinuousHighlightIconCondition GetImageValueCondition(int icon)</code>	Получить тип сравнения для переданного индекса иконки из текущего набора иконок.
<code>public void SetImageValueCondition(int icon, ContinuousHighlightIconCondition value)</code>	Установить тип сравнения для переданного индекса иконки из текущего набора иконок.
<code>public string GetImageInfo(int icon)</code>	Получить текстовое представление условия для переданного индекса иконки для отображения в редакторе.
<code>public void SetImageReverseOrder()</code>	Поменять порядок следования иконок для текущего набора иконок.
<code>public bool LoadPreset(ContinuousHighlightKind kind, string name)</code>	Загрузить настройки подсветки из шаблона.

Определены следующие шаблоны подсветки (в зависимости от типа).

ContinuousHighlightKind.TwoColorScale:

- White - Red
- Red - White
- Green - White
- White - Green
- Green - Yellow
- Yellow - Green

ContinuousHighlightKind.ThreeColorScale:

- Green - Yellow - Red
- Red - Yellow - Green
- Green - White - Red
- Red - White - Green
- Blue - White - Red
- Red - White - Blue

ContinuousHighlightKind.BarChart:

- Blue
- Green
- Red

- Orange
- Light Blue
- Purple
- Blue Gradient
- Green Gradient
- Red Gradient
- Orange Gradient
- Light Blue Gradient
- Purple Gradient

ContinuousHighlightKind.IconSet:

- Arrows (Colored)
- Arrows (Gray)
- Flags (Colored)
- 3 Traffic Lights (Unrimmed)
- 3 Traffic Lights (Rimmed)
- 3 Signs
- 3 Symbols (Circled)
- 3 Symbols (Uncircled)
- 3 Stars
- 3 Triangles
- Arrows (Colored)
- 4 Arrows (Gray)
- Red To Black
- 4 Ratings
- 4 Traffic Lights
- Arrows (Colored)
- 5 Arrows (Gray)
- 5 Ratings
- 5 Quarters
- 5 Boxes

Пример:

```
// Создаем новую подсветку ContinuousHighlight для первого показателя
ContinuousHighlight h = new
ContinuousHighlight(slice1.MeasuresContainer.MeasureFields[0].Highlights);
// Загружаем из шаблона BarChart "Orange Gradient"
h.LoadPreset(ContinuousHighlightKind.BarChart, "Orange Gradient");

// Создаем новую подсветку ContinuousHighlight для второго показателя
ContinuousHighlight h = new
ContinuousHighlight(slice1.MeasuresContainer.MeasureFields[1].Highlights);
// Загружаем из шаблона IconSet "3 Flags (Colored)"
h.LoadPreset(ContinuousHighlightKind.IconSet, "3 Flags (Colored)");
```

4.2 Выделение только ячеек попадающих под условие

RangeHighlight - отбор ячеек, по условию в свойстве Range.

Свойства:

Свойство	Описание
<code>public Range</code> Range	Условие отбора.

Пример:

```
// Создаем новую подсветку RangeHighlight с подсветкой ячеек Fuchsia для значений > 1000
RangeHighlight h = new
RangeHighlight(slice1.MeasuresContainer.MeasureFields[0].Highlights);
// Начинаем группу изменений
h.BeginUpdate();
// Задаем цвет заливки ячеек
h.Style.FillColor = Color.Fuchsia;
// Сравниваем по значению ячейки
h.Range.CompareObject = RangeCompareObject.Value;
// Условие >
h.Range.ValueCondition = RangeValueCondition.Greater;
// Сравниваем со значением 1000
h.Range.LowRange = 1000;
// Заканчиваем группу изменений
h.EndUpdate();
```

4.3 Выделение первых или последних ячеек

TopHighlight - отбор первых или последних ячеек.

Свойство	Описание
<code>public TopType</code> TopType	Первые или последние ячейки.
<code>public int</code> TopCount	Количество первых или последних.
<code>public bool</code> TopPercent	Признак использования процентов.

Пример:

```
// Создаем новую подсветку TopHighlight подсветкой Fuchsia для первых 30% значений
TopHighlight h = new TopHighlight(slice1.MeasuresContainer.MeasureFields[0].Highlights);
// Начинаем группу изменений
h.BeginUpdate();
// Задаем цвет заливки ячеек
h.Style.FillColor = Color.Fuchsia;
// Задаем первые ячейки
h.TopType = TopType.Max;
// Задаем количество
h.TopCount = 30;
// Указываем что количество в процентах
h.TopPercent = true;
// Заканчиваем группу изменений
h.EndUpdate();
```

4.4 Выделение ячеек относительно среднего

AverageHighlight - отбор значений с отклонением от среднего.

Свойство	Описание
<code>public AverageValueCondition</code> Condition	Условие сравнения со средним значением.
<code>public decimal</code> StdDev	Количество стандартных отклонений от среднего.

Пример:

```
// Создаем новую подсветку AverageHighlight подсветкой Fuchsia для значений больших
// среднего на 0.1 стандартных отклонения
AverageHighlight h = new
AverageHighlight(slice1.MeasuresContainer.MeasureFields[0].Highlights);
// Начинаем группу изменений
h.BeginUpdate();
// Задаем цвет заливки ячеек
h.Style.FillColor = Color.Fuchsia;
// Задаем условие сравнения
h.Condition = AverageValueCondition.Above;
// Задаем количество стандартных отклонений
h.StdDev = 0.1M;
// Заканчиваем группу изменений
h.EndUpdate();
```

4.5 Выделение повторяющихся или уникальных значений

UniqueHighlight - отбор повторяющихся и уникальных значений.

Свойство	Описание
<code>public UniqueValueCondition Condition</code>	Условие сравнения.

Пример:

```
// Создаем новую подсветку UniqueHighlight подсветкой Fuchsia для повторяющихся значений
UniqueHighlight h = new
UniqueHighlight(slice1.MeasuresContainer.MeasureFields[0].Highlights);
// Начинаем группу изменений
h.BeginUpdate();
// Задаем цвет заливки ячеек
h.Style.FillColor = Color.Fuchsia;
// Задаем условие сравнения
h.Condition = UniqueValueCondition.Repeatable;
// Заканчиваем группу изменений
h.EndUpdate();
```

4.6 Выделение ячеек, попадающих под выражение

ExpressionHighlight - отбор значений по условию Expression.

Свойство	Описание
<code>public string Expression</code>	Выражение сравнения.

Пример:

```
// Создаем новую подсветку ExpressionHighlight с подсветкой Fuchsia для выражения
ExpressionHighlight h = new
ExpressionHighlight(slice1.MeasuresContainer.MeasureFields[0].Highlights);
// Начинаем группу изменений
h.BeginUpdate();
// Задаем цвет заливки ячеек
h.Style.FillColor = Color.Fuchsia;
// Задаем условие сравнения
h.Expression = "Substring([Manager], 0, 1) == \"В\"";
// Заканчиваем группу изменений
h.EndUpdate();
```