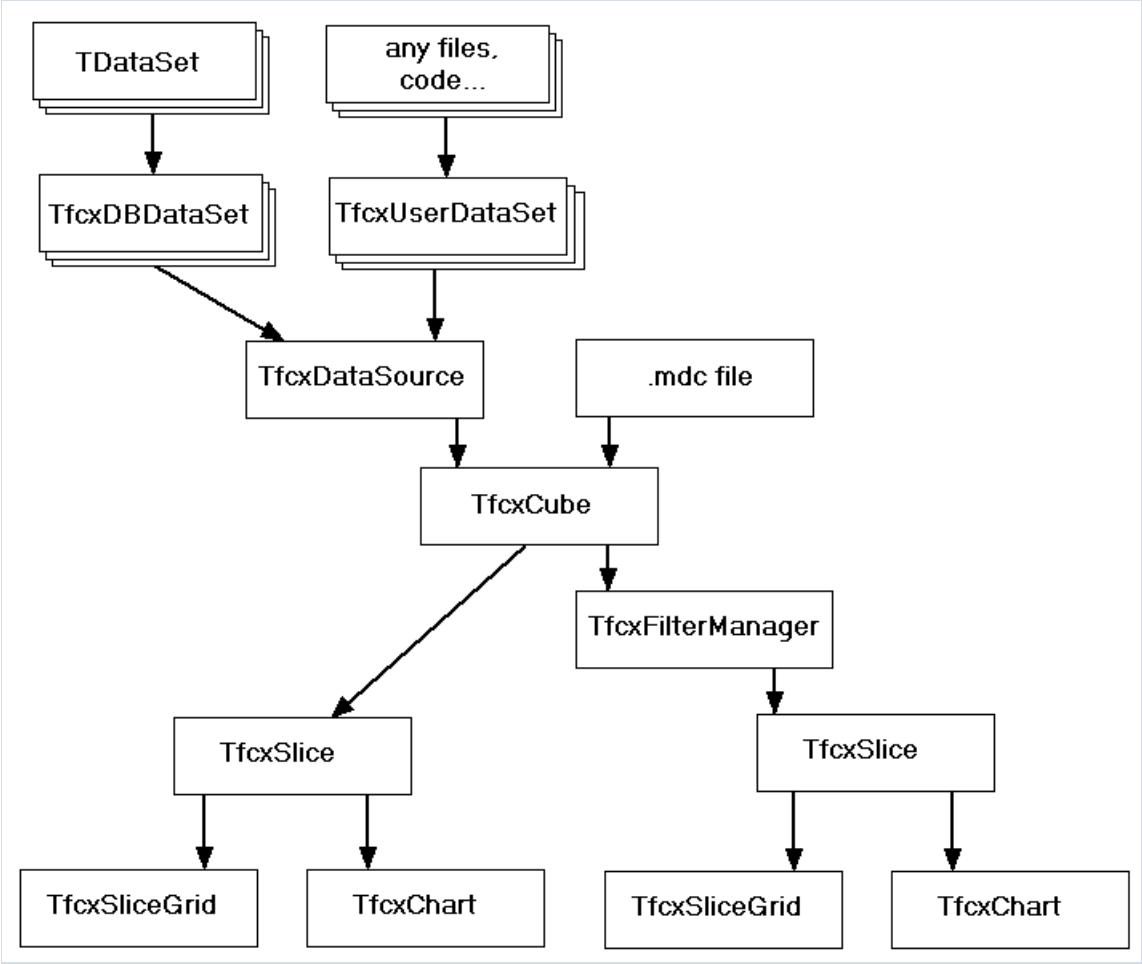
Руководство разработчика FastCube VCL

Версия 2021.1

Архитектура FastCube VCL

Библиотека компонент FastCube VCL - это набор невизуальных и визуальных компонент, назначение которых - обработка, хранение и визуализация многомерных данных. Архитектура компонент FastCube 2 представлена на следующей диаграмме.



FastCube VCL включает в себя следующие компоненты:

Невизуальные компоненты

- TfcxDBDataSet - компонент для связи с источником из базы данных;
- TfcxUserDataSet - компонент для связи с пользовательским источником (на основе событий);
- TfcxDataSource - компонент связывающий все источники данных для куба и описывающий свойства полей и атрибутов;
- TfcxCube - (куб) основное хранилище загруженных данных;
- TfcxSlice - (срез) отвечает за хранение структуры представления данных и их подготовку в соответствии с ней;
- TfcxFilterManager - управляет фильтрацией данных куба для среза;

Базовые визуальные компоненты:

- TfcxCubeGrid - (таблица данных) отображает исходные данные из куба;
- TfcxSliceGrid - (кросс-таблица) отображает данные в соответствии со структурой из среза и предоставляет пользователям возможность манипулирования данными и структурой;
- TfcxCubeGridToolbar - (панель инструментов таблицы данных) содержит набор кнопок, позволяющих выполнять операции с таблицей данных;

- TfcxSliceGridToolbar - (панель инструментов кросс-таблицы) содержит набор кнопок, позволяющих выполнять операции с кросс-таблицей, срезом и кубом;

Компоненты для отображения диаграмм:

- TfcxChart - (диаграмма) отвечает за представление данных в виде диаграммы.
- TfcxChartToolbar - (панель инструментов диаграммы) содержит набор кнопок, позволяющих выполнять операции с диаграммой.

Чтение и запись куба и среза

Загруженные в куб данные можно сохранить для последующего использования. Сохраненные данные можно будет загружать в куб без обращения к базе данных источника. Кроме самих данных можно сохранять настройку среза, групп, фильтров и диаграмм. Сохранить куб и/или настройки можно в файле или в BLOb-поле БД. Для это необходимо использовать методы компонентов TfcxCube, TfcxSlice, TfcxFilterManager и TfcxChart.

Данные куба

Данные куба (TfcxCube):

```
function LoadFromFile(ACubeFileName: String): Boolean;
```

Загружает данные куба из файла с заданным именем. Если файл загружен успешно, возвращает True.

Куб очищается перед загрузкой данных.

```
function LoadFromStream(ACubeStream: TStream): Boolean;
```

Загружает данные куба из потока. Если данные загружены успешно, возвращает True.

Куб очищается перед загрузкой данных.

```
function AppendFromFile(ACubeFileName: String): Boolean;
```

Загружает данные куба из файла с заданным именем, добавляя их к загруженным. Если файл загружен успешно, возвращает True.

Происходит объединение данных.

```
function AppendFromStream(ACubeStream: TStream): Boolean;
```

Загружает данные куба из потока, добавляя их к загруженным. Если данные загружены успешно, возвращает True.

Происходит объединение данных.

```
procedure SaveToFile(ACubeFileName: String; AFilter: TObject = nil);
```

Записывает данные куба в файл с заданным именем. Если в параметре AFilter указан объект TfcxFilterManager, то записываются только допущенные данным фильтром записи.

```
procedure SaveToStream(ACubeStream: TStream; ACompressionLevel: TCompressionLevel = clMax; AFilter:
TObject = nil);
```

Записывает данные куба в поток. Если в параметре AFilter указан объект TfcxFilterManager, то записываются только допущенные данным фильтром записи. Параметр ACompressionLevel указывает степень сжатия.

Вместе с данными в файл куба записываются настройки групп и схемы всех подключенных срезов. Состояние фильтров и настройки подключенных диаграмм вместе со схемами срезов не сохраняются.

Файл с данными куба имеет по умолчанию расширение mdc.

Примеры:

```
fcxCube1.LoadFromFile('c:\cube1.mdc');
fcxCube1.AppendFromFile('c:\cube1.mdc');
fcxCube1.SaveToFile('c:\cube2.mdc');
fcxCube1.SaveToFile('c:\cube2Filter.mdc', fcxFilterManager1);
```

Настройки среза

Настройка среза (TfcxSlice):

```
function LoadFromFile(AFileName: String): Boolean;
```

Загружает схему среза из файла с заданным именем. Если файл загружен успешно, возвращает True.

Схема среза сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о группах, настройка групп в кубе сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о фильтрах, настройка фильтров в менеджере фильтров сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о настройках диаграмм, настройка подключенных диаграмм сбрасывается перед загрузкой.

```
function LoadFromStream(ASliceStream: TStream): Boolean;
```

Загружает схему среза из потока. Если данные загружены успешно, возвращает True.

Схема среза сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о группах, настройка групп в кубе сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о фильтрах, настройка фильтров в менеджере фильтров сбрасывается перед загрузкой. При наличии в загружаемой схеме информации о настройках диаграмм, настройка подключенных диаграмм сбрасывается перед загрузкой.

```
procedure SaveToFile(AFileName: String; AStoreItems: TfcxItemsForStoreWithSlice = []);
```

Записывает схему среза в файл с заданным именем. Параметр AStoreItems описывает, какую информацию надо сохранять дополнительно (фильтры, группы, диаграммы).

```
procedure SaveToStream(ASliceStream: TStream; AStoreItems: TfcxItemsForStoreWithSlice = []);
```

Записывает схему среза в поток. Параметр AStoreItems описывает, какую информацию надо сохранять дополнительно (фильтры, группы, диаграммы).

Вместе со схемой в файл среза могут быть записаны настройки групп, фильтров и диаграмм. Файл схемы среза имеет xml структуру, расширение по умолчанию - mds.

Примеры:

```
fcxSlice1.LoadFromFile('c:\schema1.mds');
fcxSlice1.SaveToFile('c:\schema2.mds');
fcxSlice1.SaveToFile('c:\schema3.mds', [fcxiss_Groups, fcxiss_Filters, fcxiss_Charts]);
```

Настройка фильтров (TfcxFilterManager):

```
function LoadFromFile(AFileName: String): Boolean;
```

Загружает настройку фильтров из файла с заданным именем. Если файл загружен успешно, возвращает True.

Настройка фильтров сбрасывается перед загрузкой.

```
function LoadFromStream(AStream: TStream): Boolean;
```

Загружает настройку фильтров из потока. Если данные загружены успешно, возвращает True.

Настройка фильтров сбрасывается перед загрузкой.

```
procedure SaveToFile(AFileName: String);
```

Записывает настройку фильтров в файл с заданным именем.

```
procedure SaveToStream(AStream: TStream);
```

Записывает настройку фильтров в поток.

Настройка групп (TfcxCube):

```
function LoadGroupsFromFile(AGroupsFileName: String): Boolean;
```

Загружает настройку групп из файла с заданным именем. Если файл загружен успешно, возвращает True.

Настройка групп сбрасывается перед загрузкой.

```
function LoadGroupsFromStream(AStream: TStream): Boolean;
```

Загружает настройку групп из потока. Если данные загружены успешно, возвращает True.

Настройка групп сбрасывается перед загрузкой.

```
procedure SaveGroupsToFile(AGroupsFileName: String);
```

Записывает настройку групп в файл с заданным именем.

```
procedure SaveGroupsToStream(AStream: TStream);
```

Записывает настройку групп в поток.

Настройка диаграмм (TfcxChart):

```
function LoadFromFile(AFileName: String): Boolean;
```

Загружает настройку диаграмм из файла с заданным именем. Если файл загружен успешно, возвращает True.

Настройка диаграмм сбрасывается перед загрузкой.

```
function LoadFromStream(AStream: TStream): Boolean;
```

Загружает настройку диаграмм из потока. Если данные загружены успешно, возвращает True.

Настройка диаграмм сбрасывается перед загрузкой.

```
procedure SaveToFile(AFileName: String);
```

Записывает настройку диаграмм в файл с заданным именем.

```
procedure SaveToStream(AStream: TStream);
```

Записывает настройку диаграмм в поток.

Файл с настройкой диаграмм имеет по умолчанию расширение mdt.

Примеры:

```
fcxFilterManager1.LoadFromFile('c:\Filter1.fcf');
fcxFilterManager1.SaveToFile('c:\Filter2.fcf');
fcxCube1.LoadGroupsFromFile('c:\Group1.fcg');
fcxCube1.SaveGroupsToFile('c:\Group2.fcg');
fcxChart1.LoadFromFile('c:\Chart1.mdt');
fcxChart1.SaveToFile('c:\Chart2.mdt');
```

Загрузка данных

Загрузка данных из БД и пользовательских источников.

Загрузка в куб данных из одной таблицы базы данных

Набор компонентов FastCube предназначен для построения сводных таблиц на основе "плоских" данных.

Простейшим источником для куба является таблица базы данных.

Для загрузки данных в куб необходимо подключение к базе через компонент - наследник TDataSet. Его выбор зависит от используемых в приложении компонентов доступа к БД .

TfcxDBDataSet служит для связи компонента доступа с описателем источника данных для куба: TfcxDataSource.

TfcxDataSource содержит в себе полное описание структуры данных, загружаемых в куб. В нём описываются все источники, поля этих источников, взаимосвязи между источниками, правила конвертирования данных и т.п. Один из источников является основным, он указывается в свойстве TfcxDataSource.DataSet. В случае, когда загружаем данные только из одной таблицы, достаточно указать только основной источник.

Далее надо связать компоненты TfcxDataSource - TfcxCube - TfcxSlice - TfcxSliceGrid. TfcxCube и TfcxSlice можно связывать через менеджер фильтров TfcxFilterManager, если он упущен, то в TfcxSlice автоматически создастся внутренний объект TfcxFilterManager. Отдельно выделять TfcxFilterManager нужно только в случае, если один менеджер фильтров будет использоваться в нескольких срезах.

В TfcxDataSource можно описать поля, загружаемые из источника. Если описание полей отсутствует, то будут автоматически загружаться все поля.

Описание полей основного источника хранит свойство TfcxDataSource.Fields. Очистить описание можно методом TfcxDataSource.DeleteFields. Заполнить описание полей из источника можно методом TfcxDataSource.AddFields, но заполнение происходит только если в источнике созданы объекты полей (определены в дизайнере или созданы автоматически при открытии DataSet). Если Вы не планируете менять описание полей, вызывать AddFields не нужно, т.к. его вызов произойдет автоматически при открытии источника данных.

В качестве источника для куба могут служить: компонент TfcxDataSource, файл с сохранением куба, поток. Выбор источника происходит на основании свойства TfcxCube.CubeSource: TfcxCubeSource.

```
TfcxCubeSource =
(
  fccs_None,           // None
  fccs_DataSource,     // load from fcxDataSource
  fccs_CubeFile,       // load from file
  fccs_CubeStream      // load from Stream
);
```

В нашем случае надо присвоить fccs_DataSource.

Загрузка данных куба вызывается методом TfcxCube.Open. При этом куб автоматически открывает необходимые источники и загружает из них данные из описанных полей.

После загрузки данных кросс-таблица готова к работе с пользователем.

Пример:

```
//  Создание необходимых компонентов, если они отсутствуют на форме.
fcxDBDataSet1 := TfcxDBDataSet.Create(Self);
fcxDataSource1 := TfcxDataSource.Create(Self);
fcxCube1 := TfcxCube.Create(Self);
fcxSlice1 := TfcxSlice.Create(Self);
fcxSliceGrid1 := TfcxSliceGrid.Create(Self);
fcxSliceGrid1.Parent := Self;
fcxSliceGrid1.Align := alClient;
// Настраиваем связи между компонентами
fcxDBDataSet1.DataSet := DataSet1;
fcxDataSource1.DataSet := fcxDBDataSet1;
fcxCube1.DataSource := fcxDataSource1;
fcxSlice1.Cube := fcxCube1;
fcxSliceGrid1.Slice := fcxSlice1;
// Очищаем описание полей
fcxDataSource1.DeleteFields;
// Указываем тип источника данных куба
fcxCube1.CubeSource := fccs_DataSource;
// Загружаем данные в куб
fcxCube1.Open;
```

Настройка полей в TfcxDataSource

Настройка полей источника необходима в случае, если нужно загружать только часть полей, конвертировать данные, управлять разбивкой полей типа дата-время на составляющие, загружать данные из нескольких связанных источников.

Описание полей основного источника хранит свойство TfcxDataSource.Fields. Поле описывается объектом TfcxSourceField.

Свойства поля источника SourceFieldProperties зависят от его типа SourceFieldType: TfcxAttributeType.

```
TfcxAttributeType =
  (
    fcxsft_Reference,   // Поле из источника
    fcxsft_Custom,      // Пользовательское поле
    fcxsft_Date,        // Поле типа Дата (разбивается на составляющие)
    fcxsft_Time         // Поле типа Время (разбивается на составляющие)
  );
```

Свойство DataField описывает тип данных источника, имя и название поля в источнике, необходимость конвертирования с указание целевого типа данных, имя и название результирующего поля в кубе. Класс свойства DataField зависит от SourceFieldType.

Примеры:

```
// Загружаем описание полей
fcxDataSource1.AddFields;
// изменяем отображаемое в кубе название поля для поля с индексом 2
fcxDataSource1.Fields[2].DataField.CubeFieldDisplayLabel := 'Покупатель';
// устанавливаем правило конвертирования в строковый тип для поля с именем поля в источнике равным
'Population'.
TfcxReferenceDataField(fcxDataSource1.Fields.FieldByName['Population'].DataField).Convert := True;
TfcxReferenceDataField(fcxDataSource1.Fields.FieldByName['Population'].DataField).CubeFieldType :=
fcdt_String;
```

Создание и настройка атрибутов в TfcxDataSource

SplitProperty: TfcxSplitProperty описывает разбивку поля на составляющие "атрибуты". Атрибутами являются части полей даты и времени (год, день, час и т.д.) и поля зависимых источников. Атрибут в свою очередь может иметь собственные атрибуты. Вложенность атрибутов не ограничена.

Для поля можно указывать имя атрибута, содержащего отображаемое значение поля CaptionSourceAttribute и имя атрибута по которому производится сортировка значений поля OrderSourceAttribute. Если эти атрибуты не указаны, то в качестве отображаемого значения и ключа сортировки используется значение самого поля.

Атрибут как и основное поле описывается объектом TfcxSourceField. Список атрибутов - TfcxSplitProperty.Attributes.

DateSplitPaths и TimeSplitPaths описывают какие составляющие частей даты и времени надо создавать для поля (в случае если это поле имеет тип дата или время).

Примеры:

```
var
  ARefField: TfcxReferenceAttributeSFProperties;
  AAttribute: TfcxSourceField;
begin
  // Загружаем описание полей
  fcxDataSource1.AddFields;
  // настраиваем создание атрибутов День, Месяц, Год для поля 'Date1'
  fcxDataSource1.Fields.FieldByName['Date1'].SplitProperty.DateSplitPaths := [odt_Day, odt_Month,
odt_Year];
  // сделаем атрибут, хранящий отображаемое значения поля 'IdClient':
  // создаем новый атрибут для поля 'IdClient'.
  AAttribute :=
TfcxSourceField(fcxDataSource1.Fields.FieldByName['IdClient'].SplitProperty.Attributes.Add);
  // тип атрибута - Поле из источника
  AAttribute.SourceFieldType := fcxsft_Reference;
  ARefField := TfcxReferenceAttributeSFProperties(AAttribute.SourceFieldProperties);
  // источник атрибута совпадает с источником основного поля
  ARefField.DataSet :=
TfcxReferenceSourceFieldProperties(fcxDataSource1.Fields.FieldByName['IdClient'].SourceFieldProperties).D
ataSet;
  // имя поля атрибута в источнике - 'FullName'
  ARefField.DataField.DataFieldName := 'FullName';
  // Указываем имя созданного атрибута в качестве источника отображаемого значения
  fcxDataSource1.Fields.FieldByName['IdClient'].SourceFieldProperties.CaptionSourceAttribute :=
'FullName';
  // та же задача, но имя клиента берётся из другого источника данных
  // создаем новый атрибут для поля 'IdClient'.
  AAttribute :=
TfcxSourceField(fcxDataSource1.Fields.FieldByName['IdClient'].SplitProperty.Attributes.Add);
  // тип атрибута - Поле из источника
  AAttribute.SourceFieldType := fcxsft_Reference;
  ARefField := TfcxReferenceAttributeSFProperties(AAttribute.SourceFieldProperties);
  // источник атрибута - fcxDataSet2, ссылается на словарную таблицу, хранящую код 'Id' и полное имя
'FullName'
  ARefField.DataSet := fcxDataSet2;
  // имя ключевого поля в источнике атрибута - 'Id'
  ARefField.IdField.DataFieldName := 'Id';
  // имя поля атрибута в источнике атрибута - 'FullName'
  ARefField.DataField.DataFieldName := 'FullName';
  // Указываем имя созданного атрибута в качестве источника отображаемого значения
  fcxDataSource1.Fields.FieldByName['IdClient'].SourceFieldProperties.CaptionSourceAttribute :=
'FullName';
end;
```

Настройка среза

Для управления настройками среза случит компонент TfcxSlice.

На основе полей куба в срезе автоматически создаются поля среза (TfcxSliceField).

В срезе существует контейнеры, которые могут содержать поля регионов (TfcxCommonFieldOfRegion):

- XAxisContainer - ось X, содержит поля с типом TfcxAxisField
- YAxisContainer - ось Y, содержит поля с типом TfcxAxisField
- PageContainer - область фильтров, содержит поля с типом TfcxAxisField
- MeasuresContainer - показатели, содержит поля с типом TfcxMeasureField

Поля TfcxAxisField создаются на основе полей среза.

Показатели TfcxMeasureField могут быть созданы на основе полей среза, либо на основе скриптов FastScript.

Любое поле среза может быть перемещено в любой из контейнеров. Создание показателя на основе поля среза не запрещает этому полю находиться в другом контейнере.

Создание структуры среза

Срез настраивается путем добавления полей среза в контейнеры.

Поле региона создается автоматически при добавлении в контейнер поля среза.

Для добавление измерения (поля) в регион оси или фильтров служат следующие методы контейнера:

```
function AddDimension(ASliceField: TfcxSliceField; AName: TfcxString = ''; ACaption: TfcxString = ''):
integer;
```

Функция добавляет в указанный регион измерение на основе поля ASliceField в конец списка полей. Возвращает позицию поля в списке полей региона.

Если поле региона на основе ASliceField уже существует, то оно переносится в указанный регион на нужную позицию с сохранением настроек, иначе поле региона создается.

```
procedure InsertDimension(ASliceField: TfcxSliceField; AIndex: integer; AName: TfcxString = ''; ACaption:
TfcxString = '');
```

Метод вставляет в указанный регион измерение на основе поля ASliceField на заданную позицию в списке полей. Если поле региона на основе ASliceField уже существует, то оно переносится в указанный регион на нужную позицию с сохранением настроек, иначе поле региона создается.

```
procedure DeleteDimension(AIndex: integer);
```

Метод удаляет из указанного региона поле, находящееся на заданной позиции. Поле региона уничтожается.

Для управления уровнем показателей (поле "Показатели") служат методы контейнера:

```
function AddMeasuresField: integer;
```

Функция переносит поле "Показатели" в указанный регион, присваивая ему крайнюю позицию. Возвращает позицию поле "Показатели".

```
function InsertMeasuresField(AIndex: TfcxSmallCount): integer;
```

Функция переносит поле "Показатели" в указанный регион, присваивая ему указанную позицию. Возвращает позицию поле "Показатели".

```
procedure DeleteMeasuresField;
```

Функция удаляет поле "Показатели" из указанный региона. При этом поле "Показатели" автоматически

переносится в регион фильтров на первую позицию.

ВАЖНО!

Поле "Показатели" является виртуальным полем, оно всегда создано и не присутствует в списке полей соответствующего контейнера.

Для обращения к его свойствам надо обращаться к свойству среза - MeasuresContainer.

Его положение в регионе определяется свойством MeasuresContainer.Position. Все поля региона, индекс которых равен или меньше MeasuresContainer.Position, отображаются в регионе после поля "Показатели".

Свойство MeasuresContainer.Container определяет контейнер, в котором находится поле "Показатели".

Операции по настройке структуры среза лучше объединять методами среза BeginUpdate и EndUpdate. Это позволит избежать лишних пересчетов и перестроений в срезе.

Примеры:

```
// Начинаем обновление структуры
fcxSlice1.BeginUpdate;
// Добавляем в ось Y поле среза с индексом 0
fcxSlice1.YAxisContainer.AddDimension(fcxSlice1.SliceField[0]);
// Вставляем в ось Y поле среза с индексом 1 на позицию 0
fcxSlice1.YAxisContainer.InsertDimension(fcxSlice1.SliceField[1], 0);
// Добавляем в ось X поле среза с именем 'FullName'
fcxSlice1.XAxisContainer.AddDimension(fcxSlice1.SliceFieldByName['FullName']);
// Добавляем поле 'Показатели' в ось X
fcxSlice1.XAxisContainer.AddMeasuresField;
// Заканчиваем обновление структуры, запускаем внутренние расчета в срезе.
fcxSlice1.EndUpdate;
```

Управление показателями

Показатели настраиваются путем добавления в контейнер MeasuresContainer.

Показатели создаются автоматически при добавлении в контейнер. При необходимости их можно создавать вручную и добавлять в контейнер созданный показатель.

Показатель создается на основе поля среза либо на основе скрипта FastScript.

Добавление показателя на основе ASliceField со статистической функцией AAgrFunc. Возвращает позицию показателя в контейнере:

```
function AddMeasure(ASliceField: TfcxSliceField; AName, ACaption: TfcxString; AAgrFunc: TfcxAgrFunc):
Integer;
```

Добавление вычисляемого показателя на основе скриптовой функции AScriptFunctionName с типом показателя AAgrFunc. AScriptFunctionCode - код фукнции. Возвращает позицию показателя в контейнере:

```
function AddCalcMeasure(AName, ACaption: TfcxString; AAgrFunc: TfcxAgrFunc; AScriptFunctionName: String;
AScriptFunctionCode: TfcxString): Integer;
```

Добавление созданного показателя AField. Возвращает позицию показателя в контейнере:

```
function AddMeasure(AField: TfcxMeasureField): Integer;
```

Вставка показателя на основе ASliceField со статистической функцией AAgrFunc на заданную позицию в контейнере:

```
procedure InsertMeasure(ASliceField: TfcxSliceField; AName, ACaption: TfcxString; AAgrFunc: TfcxAgrFunc;
AIndex: TfcxSmallCount);
```

Вставка вычисляемого показателя на основе скриптовой функции AScriptFunctionName с типом показателя AAgrFunc на заданную позицию в контейнере. AScriptFunctionCode - код фукнции:

```
procedure InsertCalcMeasure(AName, ACaption: TfcxString; AAgrFunc: TfcxAgrFunc; AScriptFunctionName:
String; AScriptFunctionCode: TfcxString; AIndex: TfcxSmallCount);
```

Вставка созданного показателя на заданную позицию в контейнере:

```
procedure InsertMeasure(AField: TfcxMeasureField; AIndex: TfcxSmallCount);
```

Удаление показателя с заданным индексом

```
procedure DeleteMeasure(AMeasureIndex: TfcxSmallCount; ADoStopChange: Boolean = False);
```

Для доступа к показателям и управления показателями необходимо пользоваться свойствами и методами MeasuresContainer.

Показатели могут быть скрыты. В этом случае они рассчитываются, но не отображаются.

Перемещение показателя в контейнере:

```
function MoveMeasure(AFromIndex, AToIndex: TfcxSmallCount): boolean;
```

Доступ к показателю:

```
property Measures[AIndex: TfcxSmallCount]: TfcxMeasureField;
```

Для управления свойствами показателя служат методы класса TfcxMeasureField:

Видимость показателя:

```
property Visible: Boolean;
```

Тип отображения:

```
property DisplayAs: TfcxDisplayAs;
```

Примеры:

```
// Начинаем обновление структуры
fcxSlice1.BeginUpdate;
// Добавляем показатель на основе поля среза с индексом 3 и статистической функцией af_Sum (сумма)
fcxSlice1.MeasuresContainer.AddMeasure(fcxSlice1.SliceField[3],'Sum1', 'Сумма поступлений', af_Sum);
// Добавляем вычисляемый показатель, рассчитывающий половину суммы поступлений
fcxSlice1.MeasuresContainer.AddCalcMeasure('Calc1', 'Половина сумма поступлений', af_Formula,
'CalcScript1', 'Result := measures[''Sum1''].currentvalue / 2');
// Перемещаем показатель с индексом 1 на позицию с индексом 0
fcxSlice1.MeasuresContainer.MoveMeasure(1, 0)
// Скрыть показатель с индексом 1
fcxSlice1.MeasuresContainer.Measures[1].Visible := False;
//  Заканчиваем обновление структуры, запускаем внутренние расчета в срезе.
fcxSlice1.EndUpdate;
```

Управление фильтрами

Фильтры служат для сокращения объема рассчитываемых данных в соответствии с заданными критериями.

Управлять фильтрами можно через методы и свойства поля среза.

Примеры:

```
// Снять признак фильтрации у значения 3 поля среза с индексом 0
fcxSlice1.SliceField[0].UVFilterOfValue[3] := False;
// Начать пакетное изменение фильтра
fcxSlice1.SliceFieldByName['FirstName'].BeginUpdateFieldFilter;
// Снять признак фильтрации со всех значений поля 'FirstName'
fcxSlice1.SliceFieldByName['FirstName'].SetNoneFilter;
// Установить признак фильтрации у значения 'Sergey' поля 'FirstName'
fcxSlice1.SliceFieldByName['FirstName'].UVFilterOfValue['Sergey'] := True;
// Установить признак фильтрации у значения с индексом 12 поля 'FirstName'
fcxSlice1.SliceFieldByName['FirstName'].UVFilterOf[12] := True;
// Закончить пакетное изменение фильтра (применить изменения)
fcxSlice1.SliceFieldByName['FirstName'].EndUpdateFieldFilter;
// Устанавливает признак фильтрации только у значения с индексом 4 поля среза с индексом 0
fcxSlice1.SliceField[0].UVSingleIndex := 4;
// Инвертирует признак фильтрации значений поля среза с индексом 0
fcxSlice1.SliceField[0].InverseFilter;
// Устанавливает признак фильтрации в соответствии с критерием, заданным в ARange
fcxSlice1.SliceField[0].SetRangeFilter(ARange);
// Установить фильтру тип "переключатель"
SliceField[1].UVFilterType := uvft_Single;
```

Управление группами

Группы служат для повышения наглядности представления данных.

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

Работать с группами (создавать, изменять и т.д.) можно после включения режима группировки в поле среза CanGroup.

Примеры:

```
// Включаем режим группировки
ASliceField.CanGroup := True;
// Проверяем на возможность группировки
if ASliceField.CanGroup then
begin
  // создаем группу с именем 'Group1'
  AGroupIndex := ASliceField.GroupManager.CreateGroup('Group1').Index;
  // добавляем значение с индексом 3 в группу с индексом AGroupIndex
  ASliceField.GroupManager.AddUVInGroup(3, AGroupIndex);
  // добавляем значение 30 в группу с индексом AGroupIndex
  ASliceField.GroupManager.AddUVValueInGroup(30, AGroupIndex);
  // создаем группы "Другие"
  ASliceField.GroupManager.CreateOtherGroup;
end;
```

Информация о выпусках

Версия 2021.1

Информация о выпуске FastCube VCL 2021.1

Новые возможности

Из новых возможностей - поддержка новой Rad Studio 11 Sydney. Кроме этого мы улучшили работу с интерфейсом - появились новые пункты в контекстных меню, улучшили возможности поиска значений. Также обновлены языковые ресурсы и исправлены ошибки.

Добавлена поддержка Embarcadero Rad Studio 11 Sydney

Начиная с этой версии мы добавляем поддержку Embarcadero Rad Studio 11 Sydney.

Изменения в SliceGrid

- Добавлен пункт "Копировать" в контекстное меню оси, который копирует в буфер обмена значение измерения
- Добавлена возможность позиционирования в оси на значение выбранное по двойному клику в выпадающем списке значений
- Добавлен поиск в выпадающем списке значений по вставке из буфера обмена

Изменения в отчетах

- Добавлены свойства PreviewOptions, ReportOptions, PrintOptions в класс TfcxpSliceGridReport

Другие изменения

- Изменен шрифт некоторых форм с "MS Sans Serif" на "Tahoma"
- Обновлены ресурсы Чешской локали
- Обновлены ресурсы Греческой локали

Исправления ошибок

- Исправлена отрисовка в выпадающем списке значений при измененном стиле
- Замена отрисовки заголовка (D10.4 bug)
- Access violation при двойном клике на редакторе функции в настройках изменения
- Stack overflow при выпадении длинного списка