Бизнес-операции.

# Оглавление

[Оглавление 1](#_Toc453230582)

[1 Определения 2](#_Toc453230583)

[2 Решаемые задачи 2](#_Toc453230584)

[3 Общие сведения 2](#_Toc453230585)

[3.1 Типы бизнес-операций. 2](#_Toc453230586)

[3.2 Параметры бизнес-операций. 3](#_Toc453230587)

[3.3 Запуск бизнес-операций на выполнение. 3](#_Toc453230588)

[3.4 Реализации бизнес-операций. 4](#_Toc453230589)

[3.5 Бизнес-операции в модели приложения. 4](#_Toc453230590)

[4 Разработка и использование бизнес-операций. 10](#_Toc453230591)

[4.1 Подключение функционала бизнес-операций к прикладному приложению. 10](#_Toc453230592)

[4.2 Разработка прикладных бизнес-операций. 11](#_Toc453230593)

[4.2.1 Разработка класса бизнес-операции. 11](#_Toc453230594)

[4.2.2 Разработка класса управляемой бизнес-операции. 13](#_Toc453230595)

[4.2.3 Разработка класса реализации. 13](#_Toc453230596)

[4.2.4 Явное объявление классов бизнес-операций и классов реализации. 14](#_Toc453230597)

[4.2.5 Управление созданием, инициализацией, запуском. 15](#_Toc453230598)

[4.2.6 Создание экземпляров бизнес-операций и их выполнение в прикладном коде. 21](#_Toc453230599)

[5 Описание типов 21](#_Toc453230600)

[5.1 Атрибуты. 21](#_Toc453230601)

[5.1.1 Атрибуты, применяемые к классам бизнес-операций. 21](#_Toc453230602)

[5.1.2 Атрибуты, применяемые к полям классов бизнес-операций. 22](#_Toc453230603)

[5.1.3 Атрибуты, применяемые к классам реализации бизнес-операций. 23](#_Toc453230604)

[5.2 Классы. 23](#_Toc453230605)

[5.2.1 ContextBusinessOperationHelper 23](#_Toc453230606)

[5.2.2 BusinessOperationManager 23](#_Toc453230607)

[5.2.3 BOExecViewController 24](#_Toc453230608)

[5.2.4 BODynamicPropertiesObject 24](#_Toc453230609)

# Определения

* Бизнес-операция (БО), (BusinessOperation) – специальный программный модуль, декларирующий метод обработки данных.
* Реализация бизнес-операции (OperationService) – специальный класс, реализующий бизнес-операцию.

# Решаемые задачи

Бизнес-операции предназначены для преобразования одних данных (входов) в другие (выходы). Одни БО могут выполнять сложные расчеты с последующим изменением состояния бизнес - объектов, т.е. могут изменять данные системы. Другие же работают только «на чтение», т.е. получают параметры в качестве входных данных и на основании данных системы возвращают результат. Примерами БО являются:

• Процедура расчета зарплаты;

• Формирование накладной на основании ДО;

• Получение прайс-листа;

• Процедура MRP и пр.

Бизнес-операции являются составной частью интерфейса взаимодействия подсистем.

Бизнес-операции реализуют специальный механизм замены их реализации без перекомпиляции кода, тем самым они могут быть использованы как точки конфигурирования конечного приложения.

Бизнес-операции используются для поддержания совместимости между различными версиями подсистем.

# Общие сведения

Ознакомиться с функционалом бизнес-операций можно на примере демонстрационного приложения Northwind.DC, которое доступно для запуска из демо-центра, поставляемого с платформой XAFARI. В приложении использовано несколько контекстных и не контекстных бизнес-операций. Использованы различные варианты запуска бизнес-операций в рамках управляемых (длительных) операций.

## ­Типы бизнес-операций.

Бизнес-операция представляет собой класс, содержащий декларацию своих параметров. Фреймворк предоставляет API для доступа к метаинформации о самих бизнес-операциях и API для их выполнения.

Есть возможность создавать управляемые бизнес-операции. Такие бизнес операции самостоятельно создают управляемый объект (используя, например, класс [ManagedOperation](http://documentation.xafari.org/frlrfxafarimanagedoperationsmanagedoperationclasstopic.html)) под контролем которого бизнес-операция будет выполняться.

Для выполнения бизнес-операций применяются следующие методы:

* метод [Execute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationclassexecutetopic.html) - выполняет непосредственно саму операцию;
* метод [ExecuteManaged](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationmanagedclassexecutemanagedtopic.html) – присутствует в управляемых бизнес-операциях и выполняет бизнес-операцию, запуская метод Execute в рамках управляемой (длительной) операции.
* метод [Rollback](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationclassrollbacktopic.html) - выполняет обратную операцию к Execute, т.е. реализует откат операции. Конкретная реализация БО может и не содержать метод отката.
* метод [RollbackManaged](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationmanagedclassrollbackmanagedtopic.html) – присутствует в управляемых бизнес-операциях и выполняет «откат» бизнес-операции запуская метод Rollback в рамках управляемой (длительной) операции.

Бизнес операции могут выполняться как независимо (выполняются без контекста, например, пакетное формирование расходных ордеров, проверка корректности БД), так и в контексте определенного объекта (например «пересчет сумм документа», выполняется для экземпляра накладной).

Чтобы бизнес-операция стала ***контекстной***, у нее должно присутствовать доступное для записи паблик-свойство, которое отмечено специальным атрибутом [*ContextPropertyAttribute*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsattributescontextpropertyattributeclasstopic.html). Тип этого свойства определяет тип объекта, который и будет являться контекстом бизнес-операции. Атрибут содержит свойства (TypeMatchMode, ObjectsCriteria, ObjectsCriteriaMode) с помощью которых задается стратегия привязки бизнес-операции к контекстному бизнес-объекту.

Контекстные бизнес операции могут быть:

* ***одиночные*** – применяются к одному экземпляру сущности;
* ***списковые*** – применяются к списку экземпляров сущностей.

У одиночной контекстной бизнес-операции контекстное свойство имеет тип требуемого объекта. Контекстное свойство списковой бизнес-операции должно иметь тип любой обобщенной коллекции (абстрактной, например, [IEnumerable<T>](http://msdn.microsoft.com/en-us/library/9eekhta0%28v%3Dvs.110%29.aspx) или конкретной, например, [List<T>](http://msdn.microsoft.com/en-us/library/6sh2ey19%28v%3Dvs.110%29.aspx)). Параметром типа этой коллекции выступает тип объекта, для которого бизнес-операция будет выполняться.

## Параметры бизнес-операций.

Бизнес-операция может иметь произвольный набор входных параметров. Входным параметром бизнес-операции является любое доступное для записи паблик-свойство, объявленное в [классе](#_Разработка_бизнес-операции.) бизнес-операции. При запуске бизнес-операции пользователем из интерфейса приложения автоматически генерируется детальная форма для ввода входных параметров:



## Запуск бизнес-операций на выполнение.

Бизнес-операции, запущенные пользователем из интерфейса приложения (из списка действий или из общего списка бизнес-операций) по умолчанию выполняются в рамках управляемой (длительной) операции. Если бизнес-операция сама является управляемой – то вызываются ее методы [ExecuteManaged](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationmanagedclassexecutemanagedtopic.html) или [RollbackManaged](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationmanagedclassrollbackmanagedtopic.html). В противном случае объект управляемой (длительной) операции создается фреймворком самостоятельно. Синхронный или асинхронный метод выполнения конкретной бизнес-операции определяется либо атрибутом [*ExecutionWayAttribute*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesexecutionwayattributeclasstopic.html)*,* либо устанавливается в модели приложения.

В случае синхронного выполнения бизнес-операции генерируется представление с отображением процесса выполнения операции:



При активизации детального или спискового представления для какого-либо бизнес-объекта доступные для него контекстные бизнес-операции отображаются в списке действий категории Actions (отображается в виде кнопки «Действия» на тулбаре каждого представления):



Простые (не контекстные) бизнес операции отображаются в отдельном древовидном представлении «Бизнес-операции».



При организации иерархии используется одноуровневая группировка бизнес-операций по категориям. Перечень категорий, к которым относится бизнес-операция определяются либо атрибутами [*BusinessOperationCategoryAttribute*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesbusinessoperationcategoryattributeclasstopic.html), либо в модели приложения.

Бизнес-операция может также быть вызвана в прикладном коде непосредственно. Для этого требуется создать экземпляр класса бизнес операции, задать входные параметры и вызвать требуемый метод (Execute, ExecuteManaged, Rollback, RollbackManaged).

## Реализации бизнес-операций.

У одной бизнес-операции может быть несколько [реализаций](#_Разработка_класса_реализации.) из которых только одна является текущей. Информация о текущей реализации БО задается атрибутом [*DefaultOperationServiceAttribute*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesdefaultoperationserviceattributeclasstopic.html), либо в модели приложения (свойство [*DefaultImplementation*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmodelimodelbusinessoperationclassdefaultimplementationtopic.html)узла «Application/Xafari/BusinessOperation/<Business\_Operation>» соответствующей бизнес-операции).

 Реализации могут быть размещены в различных модулях XAF. С помощью специальных атрибутов указывается принадлежность той или иной реализации конкретной бизнес операции. Подключая к приложению модули с реализацией, и указывая в бизнес-модели приложения реализацию по умолчанию, осуществляется возможность заменять реализацию бизнес операций без необходимости перекомпиляции исходного кода.

Реализация бизнес-операции может поддерживать возможность отката. Для чего класс реализации должен описать метод Rollback (который вводится интерфейсом [*IOperationServiceReversible*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsioperationservicereversibleclasstopic.html)).

## Бизнес-операции в модели приложения.

Бизнес-операции перечислены в виде дочерних узлов узла «*Xafari/BusinessOperations»* модели приложения.



Однако бизнес-операции поддерживают также работу с **Бизнес-Моделью**. Если к приложению подключен модуль XafariBCModelModule (находится в сборке Xafari.BC.Model.dll), то сервисы бизнес-операций начинают использовать вместо модели приложения **Бизнес-Модель**. Следует отметить, что данные **Бизнес-Модели** хранятся не в основной модели приложения, а в отдельных файлах. Доступ к этим данным осуществляется с помощью специального редактора **Бизнес-Модели** - Project.xafml.Editor.exe. Более детальную информацию можно получить в руководстве разработчика на платформе Xafari в разделе, описывающего **Бизнес-Модель**.

Представление бизнес-операций в модели:



Свойства узла не контекстной бизнес-операции:

|  |  |
| --- | --- |
| Имя свойства | Описание |
| ImageName | Строковое имя пиктограммы, используемое при отображении айтема, запускающего бизнес-операцию в пользовательском интерфейсе. |
| Visible | Свойство указывает видимость элементов для запуска бизнес-операции в пользовательском интерфейсе. Задается атрибутом [BrowsableAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.browsableattribute%28v%3Dvs.110%29.aspx). |
| DefaultImplementation | Позволяет выбрать реализацию «по умолчанию». |
| ExecutionWay | Способ выполнения бизнес-операции в рамках управляемой операции при запуске из пользовательского интерфейса:* [*Asyncronous*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsexecutionwaysclasstopic.html) – асинхронное выполнение. Бизнес-операция выполняется в фоне. Приложение не ожидает результатов выполнения.
* [*Synchronous*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsexecutionwaysclasstopic.html) – синхронное выполнение. Отображается интерфейс с ходом выполнения. Приложение ожидает окончания выполнения.
 |
| Caption | Наименование бизнес-операции в айтемах пользовательского интерфейса. |
| Description | Краткое описание бизнес-операции. |
| Id | Только для чтения. Уникальный строковый идентификатор бизнес-операции. По умолчанию используется наименование типа (класса) бизнес-операции. |
| Type | Только для чтения. Объект типа, представляющий класс бизнес-операции. |
| ParametersDialogSizeable | Свойство указывает, будет ли диалог ввода параметров бизнес-операции изменяемого размера или фиксированного.Значение по умолчанию - false (т.е. размер диалогов фиксирован). |

Большинство описанных свойств при разработке бизнес-операции могут быть заданы при помощи соответствующего [атрибута](#_Атрибуты,_применяемые_к).

Узлы, представляющие контекстные бизнес-операции кроме свойств простых бизнес-операций, содержат дополнительные свойства:

|  |  |
| --- | --- |
| Имя свойства | Описание |
| AutoRefreshView | определяет режим обновления данных на View после выполнения контекстной бизнес-операции. По умолчанию обновление данных не выполняется. Эта функциональность реализована в Xafari.BC.BusinessOperations.Controllers.BORefreshController. Доступны следующие значения:* Any - выполняется обновление данных на всех типах контекстных View: Detail View и List View;
* DetailView - выполняется обновление данных только на контекстном Detail View;
* ListView - выполняется обновление данных только на контекстном List View;
* None - Значение по умолчанию. Обновление данных не выполняется.

Значение данного свойства можно также указать в прикладном коде, с помощью атрибута **ModelDefaultAttribute**: [ModelDefault("AutoRefreshView", "DetailView")]public class MyBusinessOperation : BusinessOperationBase {...} Контроллер предоставляет точки расширения в виде событий **CustomRefreshBeforeExecute**, **CustomRefreshAfterExecute**, **CustomMatch**.События позволяют дополнить или заменить полностью логику обновления данных после выполнения бизнес операции.Для получения дополнительной информации см. справку по классам:Xafari.BC.BusinessOperations.Controllers.BOExecViewControllerXafari.BC.BusinessOperations.Controllers.BORefreshControllerXafari.BC.BusinessOperations.Model.IModelBusinessOperationContextXafari.BC.BusinessOperations.ContextViewType |
| ContextDataType | Только для чтения. Тип данных, представляющих контекст (например, тип бизнес-объекта, для которого предназначена данная бизнес-операция) |
| ContextProperty | Только для чтения. Ссылка на свойство бизнес-операции, являющееся контекстным. |
| ContextViewType | Тип представления, с которым может быть использована бизнес-операция:* [*DetailView*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontextviewtypeclasstopic.html) – бизнес-операция доступна только в детальном представлении;
* [*ListView*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontextviewtypeclasstopic.html) – бизнес-операция доступна только в списковом представлении;
* [*Any*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontextviewtypeclasstopic.html) - бизнес-операция доступна как в детальном, так и в списковом представлениях;
 |
| ContextTypeMatchMode | Режим проверки соответствия контекстной бизнес-операции типу бизнес-объекта, к которому бизнес-операция может применяться:* ObjectTypeOrSuccessors - Тип контекстного свойства бизнес-операции должен соответствовать типу бизнес-объекта и всем наследникам типа контекстного свойства. Это значение по умолчанию;
* ExactObjectType - Тип контекстного свойства бизнес-операции должен в точности соответствовать типу бизнес-объекта;
* SuccessorsOnly - Тип контекстного свойства бизнес-операции должен соответствовать всем наследникам типа контекстного свойства.

Значение данного свойства можно указать в прикладном коде, с помощью атрибута ContextPropertyAttribute, задав его свойство TypeMatchMode. |
| ContextObjectsCriteria | Содержит строку с критерием, который вычисляется на экземпляре выделенного (текущего) бизнес-объекта. Если результат вычисления False - бизнес-операция в списке действий становится не активной.Значение данного свойства можно указать в прикладном коде с помощью атрибута ContextPropertyAttribute, задав его свойств ObjectsCriteria. |
| ContextObjectsCriteriaMode | Режим, который определяет, все ли выделенные объекты должны удовлетворять критерию:* TrueAtLeastForOne - Хотя бы один выделенный объект должен удовлетворять критерию;
* TrueForAll - Все выделенные объекты должны удовлетворять критерию.

Значение данного свойства можно указать в прикладном коде с помощью атрибута ContextPropertyAttribute, задав его свойств ObjectsCriteriaMode. |

Дочерние узлы модели бизнес-операции:

|  |  |
| --- | --- |
| Имя узла | Описание |
| Categories | Перечень категорий, к которым относится бизнес-операция. Применяется одноуровневая группировка бизнес-операций по категориям при отображении бизнес-операций в пользовательском интерфейсе (меню «Действия» и иерархический список бизнес-операций). Одна бизнес-операция может принадлежать нескольким категориям. Категория «по умолчанию» должна называться [Default](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesbusinessoperationcategoryattributeclassdefaulttopic.html). Бизнес-операции этой категории отображаются в корне списка. Категории можно добавлять, удалять, редактировать. |
| Implementations | Узел содержит перечень доступных реализаций. Для реализации можно менять заголовок и описание.Поле Id аналогично такому же полю бизнес-операции и представляет уникальный идентификатор класса-реализации.Поле Type – объект типа класса реализации. |
| Parameters | Содержит список параметров бизнес-операции. Как упоминалось ранее, параметром бизнес-операции является любое паблик-свойство класса бизнес-операции. Параметр в модели описывается следующими свойствами:* *CanWrite* – параметр доступен для записи. Установлен в true, если у соответствующего свойства класса БО имеется паблик-сеттер.
* *CanRead* – параметр доступен для чтения. Установлен в true, если у соответствующего свойства класса БО имеется паблик-геттер.
* *Caption* – наименование параметра. Используется в интерфейсе ввода параметров бизнес-операции.
* *Id* – идентификатор параметра. В качестве идентификатора используется имя соответствующего свойства класса БО.
* *Index* – задает порядок отображения параметра в интерфейсе ввода параметров БО.
* *ToolTip* – всплывающая подсказка для параметра в интерфейсе ввода параметров.
* *Type* – объект типа соответствующего свойства класса БО.

 |

Для отображения контекстных бизнес-операций в списке действий используется компонент типа SingleChoiceAction. Он имеет наименование BOExecuteAction и доступен для настройки в модели приложения в списке Actions:



Перечень простых (не контекстных) бизнес-операций отображается с помощью древовидного представления TreeListBusinessOperations\_ListView и представлен узлом модели приложения «Views/Xafari.BC.BusinessOperations.TreeList (а также соответствующим элементом NavigationItem):



# Разработка и использование бизнес-операций.

В разделе приводится описание процесса разработки и использования прикладных бизнес-операций.

## Подключение функционала бизнес-операций к прикладному приложению.

Для использования бизнес-операций в прикладном приложении необходимо подключить следующие XAF-модули:

|  |  |  |
| --- | --- | --- |
| **Сборка** | **Класс модуля** | **Описание** |
| Xafari.BC.BusinessOperations.dll | [XafariBusinessOperationsModule](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsxafaribusinessoperationsmoduleclasstopic.html) | Платформенно-независимый модуль. Подключается либо к самому приложению, либо к его платформенно-независимому модулю. |
| Xafari.BC.BusinessOperations.Web.dll | [XafariBusinessOperationsWebModule](http://documentation.xafari.org/frlrfxafaribcbusinessoperationswebxafaribusinessoperationswebmoduleclasstopic.html) | Модуль для Web версии приложения. Подключается либо к Web приложению, либо к его платформенно-зависимому модулю. |
| Xafari.BC.BusinessOperations.Win.dll | [XafariBusinessOperationsWinModule](http://documentation.xafari.org/frlrfxafaribcbusinessoperationswinxafaribusinessoperationswinmoduleclasstopic.html) | Модуль для Windows версии приложения. Подключается либо к Windows приложению, либо к его платформенно-зависимому модулю. |

Базовая функциональность бизнес-операций реализована в сборке Xafari.BC.dll (модуль XafariBCModule, сборка Xafari.BC.dll). Если в каком-либо модуле приложения всего лишь нужно объявить простую бизнес-операцию, при этом нет необходимости использовать ее в UI, то достаточно только подключения модуля базовой функциональности.

## Разработка прикладных бизнес-операций.

Разработка прикладной бизнес-операции подразумевает следующие этапы:

* [Разработка класса прикладной бизнес-операции](#_Разработка_бизнес-операции.);
* [Разработка одного или нескольких классов реализации бизнес-операции](#_Разработка_класса_реализации.);
* При необходимости, [Явное объявление типов в классе XAF-модуля](#_Регистрация_классов_бизнес-операций);
* При необходимости, [написание обработчиков событий для контроля входных данных, кастомизация представлений ввода параметров и вывода результатов, кастомизации способов создания экземпляров и запуска бизнес-операций](#_Кастомизация_создания,_инициализаци).

### Разработка класса бизнес-операции.

Минимальным требованием при разработке класса бизнес-операции является необходимость реализации интерфейса [*IBusinessOperation*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html).

Однако целесообразнее использовать готовый базовый класс [*BusinessOperationBase*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html), реализовав его наследника.

 [*BusinessOperationBase*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html) выполняет базовую функциональность бизнес-операций: генерацию и обработку событий, реализацию свойств и методов, декларируемых интерфейсом [*IBusinessOperation*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html)*.* Кроме того, данный класс предоставляет множество статических методов для доступа к метаинформации о самой бизнес-операции, фабричные методы создания экземпляров бизнес-операций. Для более детальной информации см. документацию класса [*BusinessOperationBase*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html)*.*

Когда разрабатывается класс бизнес-операции, который наследуется от [*BusinessOperationBase*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html)*,* в большинстве случаев достаточно описать параметры бизнес-операции (входные, выходные), контекст (при необходимости). Остальную работу выполнит [*BusinessOperationBase*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html)*.*

Рассмотрим пример [*контекстной списковой*](#_Типы_бизнес-операций.) бизнес-операции, использующей [*BusinessOperationBase*](http://documentation.xafari.org/index.html?frlrfxafaribcbusinessoperationsbusinessoperationbaseclasstopic.html) в качестве базового класса. Задача показанной в примере бизнес-операции - изменение стоимости фрахта в документе «заказ» на заданную величину в %.

В приведенном примере в целях демонстрации возможностей использования атрибутов применено множество атрибутов с указанием значений по умолчанию. В реальном коде часто их можно не указывать (см. [справочную информацию по конкретным атрибутам](#_Атрибуты,_применяемые_к)).

 /// <summary>

 /// Контекстная бизнес-операция. Выполняется в контексте бизнес-объектов "Заказы".

 /// Позволяет изменить стоимость фрахта требуемых заказов.

 /// </summary>

 [DisplayName("Изменение стоимости фрахта")]

 [Description("Изменение стоимости фрахта на заданную величину в %.")]

 [ImageName("BO\_Sale")]

 [ExecutionWay(ExecutionWays.Synchronous)]

 [ContextViewType(ContextViewType.Any)]

 [BusinessOperationCategory("Заказы"), BusinessOperationCategory]

 [DefaultOperationService(typeof(ChangeFreightContextDefaultImpl))]

 public class ChangeFreightContext : BusinessOperationBase

 {

 private List<string> \_processedOrdersLog;

 /// <summary>

 /// Контекстное свойство отмечено специальным атрибутом. Анализируя тип этого свойства фреймворк "поймет",

 /// для какого контекста предназначена бизнес-операция и инициализирует это свойство

 /// перед ее выполнением.

 /// </summary>

 [ContextProperty(ObjectsCriteria = "[Number] != '010248'",

 ObjectsCriteriaMode = TargetObjectsCriteriaMode.TrueForAll)]

 public ICollection<Order> Orders { get; set; }

 /// <summary>

 /// Входной параметр бизнес-операции. Фреймворк определяет входные параметры по наличию паблик-сеттера.

 /// В момент запуска бизнес-операции на основе входных параметров генерируется представление для

 /// их ввода пользователем.

 /// </summary>

 [DisplayName("Величина, %")]

 public double Percent { get; set; }

 /// <summary>

 /// Выходной параметр. Содержит текстовые сообщения об обработанных заказах.

 /// </summary>

 public List<string> ProcessedOrdersLog {

 get {return \_processedOrdersLog ?? (\_processedOrdersLog = new List<string>());}

 }

 }

Все паблик свойства бизнес-операции рассматриваются как ее параметры, часть которых инициализируется фреймворком перед выполнением бизнес-операции. Рассмотрим подробнее свойства из примера.

Свойство ***public ICollection<Order> Orders*.**

Благодаря наличию атрибута [ContextPropertyAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributescontextpropertyattributeclasstopic.html) данное свойство рассматривается фреймворком как [*контекст*](#_Типы_бизнес-операций.), в котором будет доступна бизнес-операция.

Тип контекстного свойства может иметь любой, не являющийся коллекцией, тип. В этом случае бизнес-операция считается одиночной. Такая бизнес-операция может применяться только к одному объекту. Если в списковом представлении выделяется несколько объектов – одиночные бизнес-операции **станут недоступны** для запуска из пользовательского интерфейса. При выполнении бизнес-операции фреймворк инициализирует это свойство ссылкой на текущий объект.

Также в качестве типа контекстного свойства может быть указан **обобщенный тип коллекции**, в котором, как в данном примере, параметром обобщенного типа выступает бизнес-объект *Order*. Такая бизнес-операция рассматривается как списковая. Обобщенный тип коллекции может быть как абстрактным, например, [ICollection<T>](http://msdn.microsoft.com/en-us/library/92t2ye13%28v%3Dvs.110%29.aspx), так и конкретным, например, [List<T>](http://msdn.microsoft.com/en-us/library/6sh2ey19%28v%3Dvs.110%29.aspx). Список поддерживаемых типов приведен в описании класса [CollectionCreatorFactory](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscollectioncreatorscollectioncreatorfactoryclasstopic.html). Фреймворк создает экземпляр подходящего класса коллекции, наполняет его ссылками на выбранные объекты, либо ссылкой на текущий объект (если объекты не выбраны) и инициализирует им контекстное свойство бизнес-операции.

Поведение фреймворка при применении атрибута [ContextPropertyAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributescontextpropertyattributeclasstopic.html) сразу к нескольким свойствам бизнес-операции не определено и такая ситуация считается некорректной. К ошибкам компиляции или выполнения это не приведет, но фреймворк в качестве контекста может выбрать любое из них.

В узле модели приложения, представляющего бизнес-операцию (*Application\Xafari\BusinessOperations\<Бизнес-операция>*) контекст представлен свойствами [*ContextDataType*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmodelimodelbusinessoperationcontextclasscontextdatatypetopic.html)*,* [*ContextProperty*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmodelimodelbusinessoperationcontextclasscontextpropertytopic.html)*, ContextTypeMatchMode, ContextObjectsCriteria, ContextObjectsCriteriaMode* (см. [IModelBusinessOperationContext](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmodelimodelbusinessoperationcontextclasstopic.html)).

Свойство ***public double Percent*.**

Свойства, описанные в классе бизнес-операции и имеющие паблик-сеттер, фреймворк рассматривает как входные параметры бизнес-операции. Перед запуском бизнес-операции фреймворк генерирует детальную форму для ввода параметров. В форму ввода включаются все входные параметры, за исключением [контекстного](#_Типы_бизнес-операций.) свойства и свойства [*IBusinessOperationManaged.Process*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationmanagedclassprocesstopic.html). В форму ввода также не попадают параметры, не имеющие паблик-сеттера. Для генерации детальной формы используется объект класса [BODynamicPropertiesObject](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbodynamicpropertiesobjectclasstopic.html) (являющийся наследником класса [DynamicPropertiesObject](http://documentation.xafari.org/frlrfxafaribcdynamicpropertiesdynamicpropertiesobjectclasstopic.html)).

 Идентификатор детального представления генерируется по шаблону *BODPO\_InputParameters\_BusinessOperationId\_DetailView*, где *BusinessOperationId –* [идентификатор](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationclassidtopic.html) бизнес-операции, для которой сгенерировано представление. Для получения идентификатора представления в прикладном коде (например, для кастомизации представления в своем контроллере) можно использовать статический метод [BOMethodRunner.GetInputParametersDetailViewId](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbomethodrunnerclassgetinputparametersdetailviewidtopic.html)

Все параметры бизнес-операции перечислены в узле модели приложения *Application\Xafari\BusinessOperations\<Бизнес-операция>\Parameters* (см. [IModelBusinessOperation](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmodelimodelbusinessoperationclasstopic.html), [IModelBusinessOperationParameter](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmodelimodelbusinessoperationparameterclasstopic.html)).

Применяемые к параметрам бизнес операций атрибуты приведены в разделе [«Описание типов»](#_Описание_типов.).

### Разработка класса управляемой бизнес-операции.

Имеется возможность разработки бизнес-операций, которые могут выполняться в контексте управляемой (длительной) операции (используя, например, класс [ManagedOperation](http://documentation.xafari.org/frlrfxafarimanagedoperationsmanagedoperationclasstopic.html)). В этом случае класс бизнес-операции должен реализовать интерфейс [IBusinessOperationManaged](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationmanagedclasstopic.html). Интерфейс декларирует свойство Process – объект управляемой операции, в рамках которой выполняется бизнес-операция и два метода - ExecuteManaged(), и RollbackManaged(), запускающих бизнес-операцию в управляемом контексте. При вызове данные методы инициализируют свойство Process экземпляром создаваемой управляемой (длительной) операции.

При разработке класса управляемой бизнес-операции вместо непосредственно реализации интерфейса [IBusinessOperationManaged](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsibusinessoperationmanagedclasstopic.html) целесообразно воспользоваться готовым базовым классом [BusinessOperationManagedBase](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbusinessoperationmanagedbaseclasstopic.html). Класс реализует методы ExecuteManaged() и RollbackManaged(), а также предоставляет два защищенных виртуальных метода - CreateManagedOperation() и CreateManagedOperationStub().

Первый метод, CreateManagedOperation(), создает экземпляр класса [ManagedOperation](http://documentation.xafari.org/frlrfxafarimanagedoperationsmanagedoperationclasstopic.html), в контексте которой выполняется бизнес-операция. При необходимости кастомизации управляемой операции, метод можно перекрыть в классе разрабатываемой БО.

Второй метод, CreateManagedOperationStub(), служит для создания экземпляра класса [ManagedOperationStub](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmanagedoperationstubclasstopic.html). [ManagedOperationStub](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmanagedoperationstubclasstopic.html) - это класс-заглушка управляемой операции. Он используется, если происходит обращение к свойству Process без его предварительной инициализации. Управляемую бизнес-операцию допускается вызывать асинхронно, для чего вместо вызова ExecuteManaged(), вызывается метод Execute(). В коде класса-реализации управляемой бизнес-операции может выполняться доступ к свойству Process. Чтобы повсеместно не использовать проверки вида:

 public void Execute()

 {

 …

 If (Process != null)

 Process.NextStep(…

 …

и предназначен класс [ManagedOperationStub](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsmanagedoperationstubclasstopic.html). Свойство [Process](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbusinessoperationmanagedbaseclassprocesstopic.html) реализовано в базовом классе так, что оно всегда будет проинициализировано.

Если описанное поведение по каким-либо причинам не устраивает разработчика – можно в своем коде перекрыть метод CreateManagedOperationStub() (например, заставив возвращать null).

В остальном разработка управляемой бизнес-операции не отличается от разработки, описанной в [предыдущем разделе](#_Разработка_класса_бизнес-операции.).

### Разработка класса реализации.

Сам класс бизнес-операции описывает только ее интерфейс. Непосредственно же бизнес-логика располагается в классах ***реализации*** бизнес-операций. Любая бизнес-операция может иметь одну или несколько реализаций с возможностью их [выбора](#_Реализации_бизнес-операций.) в качестве реализации по умолчанию.

При разработке класса реализации бизнес-операции последний должен реализовать интерфейс [IOperationService](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsioperationserviceclasstopic.html). Базовая реализация интерфейса выполнена в абстрактном классе [OperationServiceBase](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsoperationservicebaseclasstopic.html). Его рекомендуется использовать в качестве базового класса при разработке собственной реализации бизнес-операции.

 Рассмотрим пример разработки класса ***реализации*** контекстной списковой бизнес-операции, рассмотренной в разделе [Разработка класса бизнес-операции](#_Разработка_бизнес-операции.).

 /// <summary>

 /// Реализация бизнес-операции изменения стоимости фрахта.

 /// </summary>

 [BusinessOperation(typeof(ChangeFreightContext))]

 [DisplayName("Реализация по умолчанию")]

 [Description("Выплняет изменение стоимости фрахта на заданную величину в %.")]

 public class ChangeFreightContextDefaultImpl : OperationServiceBase

 {

 public override void Execute(IBusinessOperation businessOperation)

 {

 var bo = (ChangeFreightContext)businessOperation;

 var rate = (decimal)(bo.Percent / 100);

 using (var objectSpace = BusinessOperationManager.Instance.Application.CreateObjectSpace())

 {

 foreach (var rawOrder in bo.Orders)

 {

 var order = objectSpace.GetObject(rawOrder); // Документ в текущем ObjectSpace.

 order.Freight = order.Freight + (order.Freight \* rate);

 var oldFreight = order.Freight;

 var msg = string.Format("Заказ # {0}. Предыдущая стоимость фрахта {1}, новая {2}",

 order.Number, oldFreight, order.Freight);

 bo.ProcessedOrdersLog.Add(msg);

 }

 objectSpace.CommitChanges();

 }

 }

 }

Как видно из примера, при использовании [OperationServiceBase](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsoperationservicebaseclasstopic.html) в качестве базового класса, часто достаточно перекрыть абстрактный метод Execute, который и содержит логику бизнес-операции.

Также, при необходимости, может быть реализован метод, выполняющий откат бизнес-операции. Для осуществления такой возможности нужно реализовать интерфейс [IOperationServiceReversible](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsioperationservicereversibleclasstopic.html). Интерфейс декларирует метод [Rollback](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsioperationservicereversibleclassrollbacktopic.html), который и требуется реализовать.

С помощью атрибута [BusinessOperationAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesbusinessoperationattributeclasstopic.html) указывается бизнес-операция, для которой разработана данная реализация.

По другим атрибутам см. информацию в разделе [«Описание типов»](#_Описание_типов.).

### Явное объявление классов бизнес-операций и классов реализации.

По умолчанию Xafari framework самостоятельно ищет все классы бизнес-операций и их реализаций в каждом модуле приложения. Для этого используется технология отражения (reflection).

В то же время у разработчиков есть возможность явно указать классы бизнес-операций и их реализаций, которые реализованы в модуле. Явное объявление классов бизнес-операций и их реализаций может немного оптимизировать время инициализации, т.к. при этом не будет использована технология отражения для поиска нужных классов.

Для того, чтобы выполнить явное объявление нужных классов (типов), необходимо явно (explicit) реализовать интерфейсы **ITypesProvider<IBusinessOperation>** и **ITypesProvider<IOperationService>** в соответствующем классе модуля. Интерфейс **ITypesProvider<T>** декларирует единственный метод - **IEnumerable<Type> GetTypes()**. Данный метод должен вернуть перечисляемую коллекцию типов, которые должны быть наследником типа, указанного в качестве обобщенного аргумента. Ниже приведен пример объявления бизнес-операции и ее реализации:

public sealed partial class MyModule : ModuleBase, ITypesProvider<IBusinessOperation>, ITypesProvider<IOperationService>

{

 /// <summary>

 /// Business Operations of current module.

 /// </summary>

 IEnumerable<Type> ITypesProvider<IBusinessOperation>.GetTypes()

 {

 return new[]

 {

 typeof (CalcOrderTotal)

 };

 }

 /// <summary>

 /// Business Operation Services of current module.

 /// </summary>

 IEnumerable<Type> ITypesProvider<IOperationService>.GetTypes()

 {

 return new[]

 {

 typeof (CalcWithoutDiscount),

 typeof (CalcWithDiscount),

 };

 }

}

### Управление созданием, инициализацией, запуском.

При запуске бизнес-операции из пользовательского интерфейса контроллер [BOExecViewController](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontrollersboexecviewcontrollerclasstopic.html) генерирует ряд событий, обрабатывая которые прикладной разработчик может управлять процессом создания экземпляров интересующих бизнес-операций, инициализацией параметров бизнес-операции, процессом запуска бизнес-операции на выполнение, создания представлений, отображаемых до запуска и после выполнения бизнес-операции и т.п. События доступны через экземпляр класса [BOExecViewController](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontrollersboexecviewcontrollerclasstopic.html) и через специальное индексируемое свойство BOExecViewController.BOEvents. Подписку на обработку событий можно выполнять как для конкретной бизнес-операции (передавая ее Id или тип в качестве индекса), так и для всех бизнес-операций (обращаясь непосредственно к полю BOEvents). Далее будут представлены примеры обработчиков. Рассмотрим эти события.

|  |  |
| --- | --- |
| Событие | Описание |
| [CreateCustomBusinessOperationInstance](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontrollersboexecviewcontrollerclasscreatecustombusinessoperationinstancetopic.html) | Доступно через свойсво BOEvents. Событие возникает, когда контроллеру BOExecViewController требуется создать экземпляр бизнес-операции. В аргументах передается идентификатор требуемой бизнес-операции.Обработчик события может создать экземпляр класса бизнес-операции и вернуть его, установив свойство BusinessOperationInstance аргумента. |
| BusinessOperationExecuting | Доступно через свойсво BOEvents. Возникает перед запуском бизнес-операции. В аргументах события передается информация о вызываемом методе (Execute или Rollback), объект бизнес-операции, и объект класса ShowViewParameters. В обработчике события можно самостоятельно организовать ввод параметров бизнес-операции (если они есть), инициализируя поля объекта ShowViewParameters(см. Пример 1). Также имеется возможность отменить запуск бизнес-операции, установив свойство Cancel в true. В этом случае принимается во внимание состояние объекта класса ShowViewParameters, с помощью которого можно сообщить пользователю о причине отмены запуска.  |
| BusinessOperationCustomExecute | Доступно через свойсво BOEvents. Событие возникает, когда контроллеру требуется выполнить бизнес-операцию. В аргументах события передается информация о вызываемом методе (Execute или Rollback), объект бизнес-операции, и объект класса ShowViewParameters.Обработчик события может самостоятельно вызвать требуемый метод бизнес-операции. Чтобы предотвратить запуск бизнес-операции контроллером, обработчик события должен установить свойство Handled аргумента события в true.При этом в обработчике события требуется самостоятельно организовать ввод параметров бизнес-операции (если они есть), инициализируя поля объекта ShowViewParameters. Также, после завершения бизнес-операции, если нужно отобразить результат, требуется самостоятельно вызвать метод контроллера BOExecViewController.CreateResultViewParameters и самостоятельно обработать полученный объект ShowViewParameters. |
| CustomizeResultViewParameters | Доступно через свойсво BOEvents. Событие возникает, после выполнения бизнес-операции. В обработчике можно инициализировать поля объекта ShowViewParameters для отображения результатов выполнения бизнес-операции. В аргументах события передается информация о выполненном методе (Execute или Rollback), объект бизнес-операции, и объект класса ShowViewParameters. |
| CustomizeContextBusinessOperationsList | Доступно через экземпляр класса [BOExecViewController](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontrollersboexecviewcontrollerclasstopic.html). Событие позволяет управлять списком контекстных бизнес-операций. Аргумент события содержит список контекстных бизнес-операций. Список может быть модифицирован обраотчиком события. |

Подписку на события контроллера BOExecViewController можно проводить в собственном контроллере. Ниже приведены примеры подписки на события и их обработки:

Пример 1:

/// <summary>

/// Example of custom handlers for Xafari.BC.BusinessOperations.Controllers.BOExecViewController events

/// </summary>

public class CustomizeBusinessOperationsController : ViewController

{

 private BOExecViewController \_boExecViewController;

 protected override void OnActivated()

 {

 base.OnActivated();

 boExecViewController = Frame.GetController<BOExecViewController>();

 if (\_boExecViewController != null)

 {

 // Subscribing to handling of controller's events for specific business operations.

 \_boExecViewController.BOEvents[typeof(BusinessOperation)].BusinessOperationExecuting += BusinessOperationExecuting;

 }

 }

 protected override void OnDeactivated()

 {

 if (\_boExecViewController != null)

 {

 \_boExecViewController.BOEvents[typeof(BusinessOperation)].BusinessOperationExecuting -= BusinessOperationExecuting;

 }

 base.OnDeactivated();

 }

 // Custom BusinessOperation business operation executing.

 // Changes the view settings without changing the business operation method call.

 void BusinessOperationExecuting(object sender, BC.BusinessOperations.BusinessOperationExecutingEventArgs e)

 {

 e.ShowViewParameters.CreatedView = Application.CreateDetailView(Application.CreateObjectSpace(), ((BusinessOperation)e.BusinessOperation).Parameters);

 ((DetailView)e.ShowViewParameters.CreatedView).ViewEditMode = ViewEditMode.Edit;

 e.ShowViewParameters.TargetWindow = TargetWindow.NewModalWindow;

 e.ShowViewParameters.Context = TemplateContext.PopupWindow;

 }

}

Пример 2:

/// <summary>

/// Example of custom handlers for Xafari.BC.BusinessOperations.Controllers.BOExecViewController events

/// </summary>

public class CustomizeBusinessOperationsController : ViewController

{

 private BOExecViewController \_boExecViewController;

 protected override void OnActivated()

 {

 base.OnActivated();

 \_boExecViewController = Frame.GetController<BOExecViewController>();

 if (\_boExecViewController != null)

 {

 // Subscribing to handling of controller's events for specific business operations.

 \_boExecViewController.BOEvents[typeof(CalcOrderTotalList)].BusinessOperationExecuting += CalcOrderTotalListExecuting;

 \_boExecViewController.BOEvents[typeof(ReversibleOperation)].BusinessOperationCustomExecute += ReversibleOperationCustomExecute;

 \_boExecViewController.BOEvents[typeof(ChangeFreightContext)].BusinessOperationCustomExecute += ChangeFreightContextCustomExecute;

 \_boExecViewController.BOEvents[typeof(ChangeFreightContext)].CreateCustomBusinessOperationInstance += CreateCustomChangeFreightContextInstance;

 \_boExecViewController.BOEvents[typeof(CalcOrderTotal)].ContextPropertyInitializing += CalcOrderTotalContextPropertyInitializing;

 \_boExecViewController.BOEvents[typeof(CalcOrderTotal)].ProcessResult += CalcOrderTotalProcessResult;

 \_boExecViewController.BOEvents[typeof(CalcOrderTotal)].BusinessOperationCustomExecute += CalcOrderTotalCustomExecute;

 // Subscribing to handling of controller's events for all business operations.

 \_boExecViewController.BOEvents.ProcessResult += ProcessResult;

 }

 }

 protected override void OnDeactivated()

 {

 if (\_boExecViewController != null)

 {

 \_boExecViewController.BOEvents[typeof(CalcOrderTotalList)].BusinessOperationExecuting -= CalcOrderTotalListExecuting;

 \_boExecViewController.BOEvents[typeof(ReversibleOperation)].BusinessOperationCustomExecute -= ReversibleOperationCustomExecute;

 \_boExecViewController.BOEvents[typeof(ChangeFreightContext)].BusinessOperationCustomExecute -= ChangeFreightContextCustomExecute;

 \_boExecViewController.BOEvents[typeof(ChangeFreightContext)].CreateCustomBusinessOperationInstance -= CreateCustomChangeFreightContextInstance;

 \_boExecViewController.BOEvents[typeof(CalcOrderTotal)].ContextPropertyInitializing -= CalcOrderTotalContextPropertyInitializing;

 \_boExecViewController.BOEvents[typeof(CalcOrderTotal)].ProcessResult -= CalcOrderTotalProcessResult;

 \_boExecViewController.BOEvents[typeof(CalcOrderTotal)].BusinessOperationCustomExecute -= CalcOrderTotalCustomExecute;

 \_boExecViewController.BOEvents.ProcessResult -= ProcessResult;

 }

 base.OnDeactivated();

 }

 private const int BoundaryYear = 1996;

 // Before starting a business operation (in this case CalcOrderTotalList) you can change its parameters.

 // In this example, from the list of selected orders are deleted elements that do not meet a certain condition.

 // The situation is fictional, it used just to show the possibility of implementing something similar.

 // In the case of the order in the list, which is less than a year RequiredDate 1995, business operation

 // "Cost of orders (list)" won't performed, and an appropriate message will shown.

 private void CalcOrderTotalListExecuting(object sender, BusinessOperationExecutingEventArgs e)

 {

 var businessOperation = (CalcOrderTotalList)e.BusinessOperation;

 businessOperation.Orders.RemoveAll(order => order.RequiredDate.Year < BoundaryYear);

 if (businessOperation.Orders.Count == 0)

 {

 MessageObject.CreateMessageWarning(

 e.ShowViewParameters,

 Application,

 Resources.Attention,

 Resources.CostOrderProhibited + Environment.NewLine +

 Resources.DeleteYouOrder + Environment.NewLine +

 Resources.SelectOtherOrders,

 BoundaryYear

 );

 e.Cancel = true;

 }

 }

 // Displaying the results of executing a business operation CalcOrderTotalList.

 private void CalcOrderTotalListProcessResult(ProcessResultEventArgs e)

 {

 var businessOperation = (CalcOrderTotalList)e.BusinessOperation;

 var resultMsg = new StringBuilder();

 resultMsg.AppendLine(string.Format(Resources.TotalValueOrder, businessOperation.Total));

 resultMsg.AppendLine();

 resultMsg.AppendLine(Resources.ProcessedOrders);

 foreach (var order in businessOperation.OrderList)

 resultMsg.AppendLine(order);

 MessageObject.CreateMessageInfo(

 e.ShowViewParameters,

 Application,

 Resources.Result,

 resultMsg.ToString()

 );

 }

 private static readonly string CalcOrderTotalListId = BusinessOperationBase.GetId(typeof(CalcOrderTotalList));

 // Custom displaying the results of executing a business operation.

 // To demonstrate the possibilities, handler is subscribed to events for all business operations.

 // Required business operation is determined directly in the code of the event handler.

 private void ProcessResult(object sender, ProcessResultEventArgs e)

 {

 if (e.Method != BusinessOperationMethod.Execute)

 return; // Displaying the results of other methods of BO is not expected.

 // Preferred business operations can be defined either by type or by its ID.

 if (e.BusinessOperation.Id == CalcOrderTotalListId)

 CalcOrderTotalListProcessResult(e);

 else if (e.BusinessOperation is ChangeFreightContext)

 ChangeFreightContextProcessResult(e);

 }

 // Custom creation of instance of the business operation.

 // This example is used just to show the possibility of implementing something similar.

 private void CreateCustomChangeFreightContextInstance(object sender, CreateCustomBusinessOperationInstanceEventArgs e)

 {

 // Any desired constructor can be used.

 e.BusinessOperationInstance = new ChangeFreightContext();

 }

 // Custom initialization of business operation context property.

 // This example is used just to show the possibility of implementing something similar.

 private void CalcOrderTotalContextPropertyInitializing(object sender, ContextPropertyInitializingEventArgs e)

 {

 var currentObject = View.CurrentObject;

 // DataAccessMode = DataView behavior.

 var order = currentObject is XafDataViewRecord ? (Order)ObjectSpace.GetObject(currentObject) : (Order)currentObject;

 if (order.Number == "010248")

 {

 order = View.ObjectSpace.FindObject<Order>(CriteriaOperator.Parse("Number != '010248'"));

 if (order != null)

 {

 ((CalcOrderTotal)e.BusinessOperation).Order = order;

 e.Handled = true;

 }

 }

 }

 // Displaying the results of executing a business operation CalcOrderTotal.

 private void CalcOrderTotalProcessResult(object sender, ProcessResultEventArgs e)

 {

 var businessOperation = (CalcOrderTotal)e.BusinessOperation;

 var resultMsg = new StringBuilder();

 resultMsg.AppendLine(string.Format(Resources.OrderNumber, businessOperation.OrderNum));

 resultMsg.AppendLine(string.Format(Resources.DateOrder, businessOperation.OrderDate.ToShortDateString()));

 resultMsg.AppendLine(string.Format(Resources.TotalCostOrder, businessOperation.Total));

 MessageObject.CreateMessageInfo(

 e.ShowViewParameters,

 Application,

 Resources.Result,

 resultMsg.ToString()

 );

 }

 // Custom executing of business operation CalcOrderTotal.

 // If business operation does not implement interface IBusinessOperationManaged, it is executed directly

 // in main thread of application unlike default behavior. Otherwise, it is used the default

 // way: business operation is executed as managed operation.

 private void CalcOrderTotalCustomExecute(object sender, BusinessOperationExecuteEventArgs e)

 {

 BOMethodRunner runner;

 if (e.Method == BusinessOperationMethod.Execute)

 runner = new CustomBOMethodRunnerExecute(Application);

 else

 runner = new CustomBOMethodRunnerRollback(Application);

 runner.Run(((BOExecViewController)sender).ProcessResult, e.BusinessOperation, e.ShowViewParameters);

 e.Handled = true;

 }

 // Custom executing of business operation ReversibleOperation with request for confirmation of the executing.

 private void ReversibleOperationCustomExecute(object sender, BusinessOperationExecuteEventArgs e)

 {

 var businessOperation = (ReversibleOperation)e.BusinessOperation;

 var msg = e.Method == BusinessOperationMethod.Execute

 ? string.Format(Resources.Execute, businessOperation.Name)

 : string.Format(Resources.Rollback, businessOperation.Name);

 ConfirmationMessage.Create(e.ShowViewParameters, Application, Resources.Confirm, msg);

 var dialogController = e.ShowViewParameters.Controllers.OfType<DialogController>().First();

 var runner = BOMethodRunner.Create(Application, e.Method);

 dialogController.Accepting += (s, args) => runner.Run(((BOExecViewController)sender).ProcessResult, businessOperation, args.ShowViewParameters);

 e.Handled = true;

 }

 // Custom executing of business operation ChangeFreightContext. It is executed synchronously.

 // After it finish, the results are displayed(see CustomizeResultViewParameters handler).

 private void ChangeFreightContextCustomExecute(object sender, BusinessOperationExecuteEventArgs e)

 {

 var businessOperation = (ChangeFreightContext)e.BusinessOperation;

 if (e.Method != BusinessOperationMethod.Execute)

 return;

 var dpo = BODynamicPropertiesObject.CreateWithParametersOf(businessOperation, BOParameterType.In);

 var viewId = BOMethodRunner.GetInputParametersDetailViewId(businessOperation.Id);

 e.ShowViewParameters.CreatedView = dpo.CreateDetailView(Application.CreateObjectSpace(), Application, true, modelDetailView => modelDetailView.Id = viewId);

 var dialogController = Application.CreateController<DialogController>();

 // You can control the launch of business operations. In this case, after the execution, you must call

 // the method BOExecViewController.CreateResultViewParameters to support the possibility of submission of view to display the results.

 dialogController.Accepting += (s, args) =>

 {

 businessOperation.Execute();

 ((BOExecViewController)sender).ProcessResult(e.Method, businessOperation, args.ShowViewParameters);

 };

 e.ShowViewParameters.Controllers.Add(dialogController);

 e.ShowViewParameters.CreatedView.Caption = string.Format(Resources.UserInterfaceParameter, businessOperation.Name);

 e.ShowViewParameters.TargetWindow = TargetWindow.NewModalWindow;

 e.ShowViewParameters.Context = TemplateContext.PopupWindow;

 e.ShowViewParameters.CreateAllControllers = true;

 ((DetailView)e.ShowViewParameters.CreatedView).ViewEditMode = ViewEditMode.Edit; // Actual for Web. Framework ignores this property at Win.

 // Another way to run a business operation - use the library classes. In this case, do not specify a handler for dalogController.Accepting.

 // BOMethodRunner, found that e.ShowViewParameters.CreatedView been specified custom View, create a self-handler and execute

 // the business operation at default mode.

 //

 // var runner = BOMethodRunner.Create(Application, e.Method);

 // runner.Run(controller, businessOperation, e.ShowViewParameters);

 e.Handled = true;

 }

 private void ChangeFreightContextProcessResult(ProcessResultEventArgs e)

 {

 var businessOperation = (ChangeFreightContext)e.BusinessOperation;

 var resultMsgBuilder = new StringBuilder();

 foreach (var order in businessOperation.ProcessedOrdersLog)

 {

 resultMsgBuilder.AppendLine(order);

 }

 var resultMsg = resultMsgBuilder.Length != 0 ? resultMsgBuilder.ToString() : Resources.NoOrdersMessage;

 MessageObject.CreateMessageInfo(

 e.ShowViewParameters,

 Application,

 Resources.ProcessedOrders,

 resultMsg

 );

 }

 private class CustomBOMethodRunnerExecute : BOMethodRunnerExecute

 {

 public CustomBOMethodRunnerExecute(XafApplication application) : base(application) {}

 protected override void RunOperationCore(BusinessOperationAction createResultViewParameters, IBusinessOperation businessOperation, ShowViewParameters showViewParameters)

 {

 if (businessOperation is IBusinessOperationManaged)

 {

 RunAsManagedOperation(createResultViewParameters, businessOperation, showViewParameters);

 }

 else

 {

 businessOperation.Execute();

 createResultViewParameters(Method, businessOperation, showViewParameters);

 }

 }

 }

 private class CustomBOMethodRunnerRollback : BOMethodRunnerRollback

 {

 public CustomBOMethodRunnerRollback(XafApplication application) : base(application) { }

 protected override void RunOperationCore(BusinessOperationAction createResultViewParameters, IBusinessOperation businessOperation, ShowViewParameters showViewParameters)

 {

 if (businessOperation is IBusinessOperationManaged)

 {

 RunAsManagedOperation(createResultViewParameters, businessOperation, showViewParameters);

 }

 else

 {

 businessOperation.Rollback();

 createResultViewParameters(Method, businessOperation, showViewParameters);

 }

 }

 }

}

### Создание экземпляров бизнес-операций и их выполнение в прикладном коде.

В прикладном коде может потребоваться непосредственное создание экземпляра и выполнение бизнес-операции. Этот процесс не отличается от создания любых других классов и вызова их методов:

 // Создание, инициализация и выполнение бизнес-операции ChangeFreightContext.

 void DirectExecuteChangeFreightContext(ICollection<Order> OrderList)

 {

 var businessOperation = new ChangeFreightContext();

 businessOperation.Orders = OrderList;

 businessOperation.Percent = 10;

 businessOperation.Execute();

 }

 // Создание и выполнение произвольной бизнес-операции.

 void DirectCreateAndExecuteBusinessOperationById(string businessOperationId)

 {

 var businessOperation = BusinessOperationBase.CreateInstance(businessOperationId);

 businessOperation.Execute();

 }

# Описание типов

В разделе приводится описание некоторых типов, используемых при работе с бизнес-операциями. Дополнительную информацию см. в [онлайн-документации](http://documentation.xafari.org).

## Атрибуты.

### Атрибуты, применяемые к классам бизнес-операций.

|  |  |
| --- | --- |
| [DisplayNameAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.displaynameattribute.aspx) | Отображаемое наименование бизнес-операции. Значение атрибута используется в качестве значения поля Caption узла модели приложения, представляющего бизнес-операцию. Для контекстных бизнес-операций это значение используется в качестве наименования пункта меню, с помощью которого запускается на выполнение бизнес-операция. Если атрибут не указан, то в качестве наименования используется имя экземпляра типа класса бизнес-операции. |
| [DescriptionAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.descriptionattribute%28v%3Dvs.110%29.aspx) | Текстовое описание бизнес-операции. Значение атрибута используется в качестве значения поля ToolTip узла модели приложения, представляющего бизнес-операцию. Для контекстных бизнес-операций это значение используется в качестве всплывающей подсказки пункта меню, с помощью которого запускается на выполнение бизнес-операция. |
| [BrowsableAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.browsableattribute%28v%3Dvs.110%29.aspx) | Указывает, следует ли отображать элементы для запуска бизнес-операции в пользовательском интерфейсе.  |
| [DefaultOperationServiceAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesdefaultoperationserviceattributeclasstopic.html) | Реализация «по умолчанию» бизнес-операции. В параметре атрибута передается экземпляр типа класса - реализации бизнес-операции. Значение атрибута используется в качестве значения поля DefaultImplementation узла модели приложения, представляющего бизнес-операцию. При выполнении методов Execute и Rollback бизнес-операция перенаправляет запросы экземпляру класса реализации, указанному в этом поле. |
| [ContextViewTypeAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributescontextviewtypeattributeclasstopic.html) | Тип представления, с которым может использоваться контекстная бизнес-операция. |
| [ImageNameAttribute](http://documentation.devexpress.com/#Xaf/clsDevExpressPersistentBaseImageNameAttributetopic) | Строковое имя пиктограммы, используемое при отображении айтема, запускающего бизнес-операцию в пользовательском интерфейсе. |
| [BusinessOperationCategoryAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesbusinessoperationcategoryattributeclasstopic.html) | Позволяет задать одну или несколько категорий, к которым относится бизнес-операция. Применяется одноуровневая группировка бизнес-операций по категориям при отображении бизнес-операций в пользовательском интерфейсе (меню «Действия» и иерархический список бизнес-операций). Одна бизнес-операция может принадлежать нескольким категориям. Атрибут без параметров задает категорию «по умолчанию». Также, экземпляр такого атрибута можно получить через статическое свойство [Default](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesbusinessoperationcategoryattributeclassdefaulttopic.html). Бизнес-операции этой категории отображаются в корне списка. |
| [ExecutionWayAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesexecutionwayattributeclasstopic.html) | Указывает способ выполнения бизнес-операции, запущенной через пользовательский интерфейс. Доступны синхронный и асинхронный способы выполнения.При синхронном способе выполнения бизнес-операция запускается в контексте управляемой операции и отображается диалог, показывающий ход выполнения длительной операции. По окончании выполнения и закрытии диалога отображаются результаты выполнения бизнес операции, если таковые были предусмотрены.При асинхронном способе выдается только сообщение, что бизнес-операция запущена. Бизнес-операция запускается под контролем управляемой операции. Ожидание окончания выполнения не производится. Результат операции отдельным представлением не отображается. Всю информацию о процессе выполнения можно увидеть в списке управляемых (длительных) операций. |

### Атрибуты, применяемые к полям классов бизнес-операций.

|  |  |
| --- | --- |
| [BrowsableAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.browsableattribute%28v%3Dvs.110%29.aspx) | Указывает, следует ли отображать поле в модели приложения. По умолчанию общедоступное поле класса бизнес-операции отображается в списке параметров узла модели приложения, представляющего бизнес-операцию. Атрибут со значением false используется, если нет необходимости отображать поле в списке параметров узла модели БО. |
| [ContextPropertyAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributescontextpropertyattributeclasstopic.html) | Этим атрибутом помечается [контекстное свойство](#_Типы_бизнес-операций.) бизнес-операции. Атрибут содержит дополнительные свойства, которые могут быть заданы прикладным разработчиком:* TypeMatchMode - режим проверки соответствия контектсной бизнес-операции типу бизнес-объекта, к которому бизнес-операция может применяться. Свойство может принимать значения перечисления BusinessOperationContextTypeMatchMode.
* ObjectsCriteria - критерий, который вычисляется на экземпляре выделенного (текущего) бизнес-объекта. Если результат вычисления False - бизнес-операция в списке действий становится не активной.
* ObjectsCriteriaMode - режим, который определяет, все ли выделенные объекты должны удовлетворять критерию. Свойство может принимаеть значения перечисления TargetObjectsCriteriaMode.
 |

### Атрибуты, применяемые к классам реализации бизнес-операций.

|  |  |
| --- | --- |
| [BusinessOperationAttribute](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsattributesbusinessoperationattributeclasstopic.html) | Бизнес-операция, для которой предназначена реализация. В параметре атрибута передается экземпляр типа класса бизнес-операции. Значение атрибута используется при формировании списка доступных реализаций бизнес-операции узла модели приложения, представляющего конкретную бизнес-операцию. Список доступных реализаций представляется узлом модели Implemenations. При смене пользователем значения поля DefaultImplementation в модели, выпадающий список доступных реализаций содержит перечень реализаций текущей бизнес-операции. |
| [DisplayNameAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.displaynameattribute.aspx) | Отображаемое наименование реализации бизнес-операции. Значение атрибута используется в качестве значения поля Caption узла модели приложения, представляющего реализацию бизнес-операции. Если атрибут не указан, то в качестве наименования используется имя экземпляра типа класса реализации бизнес-операции. |
| [DescriptionAttribute](http://msdn.microsoft.com/en-us/library/system.componentmodel.descriptionattribute%28v%3Dvs.110%29.aspx) | Текстовое описание реализации бизнес-операции. Значение атрибута используется в качестве значения поля ToolTip узла модели приложения, представляющего реализацию бизнес-операции. |

## Классы.

В разделе приводится описание некоторых классов, используемых при работе с бизнес-операциями. Дополнительную информацию см. в [онлайн-документации](http://documentation.xafari.org).

### [ContextBusinessOperationHelper](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontextbusinessoperationhelperclasstopic.html)

Вспомогательный класс, позволяющий инициализировать контекстное свойство произвольной бизнес-операции, для чего служит метод [SetContext](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontextbusinessoperationhelperclasssetcontexttopic_overloads--.html).

### [BusinessOperationManager](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbusinessoperationmanagerclasstopic.html)

Менеджер бизнес-операций. Доступ к глобальному экземпляру менеджера осуществляется через статическое свойство [BusinessOperationManager.Instance](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbusinessoperationmanagerclassinstancetopic.html).

Члены класса:

* Свойство [*Model*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbusinessoperationmanagerclassmodeltopic.html) - содержит ссылку на узел модели «Application/Xafari/BusinessOperations» - список бизнес-операций, зарегистрированных в приложении.
* Свойство [*CollectionCreatorFactory*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbusinessoperationmanagerclasscollectioncreatorfactorytopic.html) - экземпляр фабрики создателя коллекций, используемой для генерации значений контекстных свойств бизнес-операций. Это поле используется в качестве параметра конструктора при создании экземпляров класса [ContextBusinessOperationHelper](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontextbusinessoperationhelperclasstopic.html).
* Свойство [*Application*](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbusinessoperationmanagerclassapplicationtopic.html) – ссылка на экземпляр текущего приложения. Часто бывает удобно при реализации бизнес-операции иметь ссылку на приложение. В таких случаях может быть использовано это свойство менеджера.

### [BOExecViewController](http://documentation.xafari.org/frlrfxafaribcbusinessoperationscontrollersboexecviewcontrollerclasstopic.html)

Формирует элементы управления для запуска *контекстных* и *не контекстных* бизнес-операций. Выполняет запуск бизнес-операций из пользовательского интерфейса. Поддерживает набор событий, позволяющих контролировать создание, инициализацию, запуск бизнес-операций пользователем, а также события, позволяющие создать собственные представления для ввода параметров бизнес-операции и отображения результатов их выполнения.

### [BODynamicPropertiesObject](http://documentation.xafari.org/frlrfxafaribcbusinessoperationsbodynamicpropertiesobjectclasstopic.html)

Поддержка динамического набора свойств бизнес операций. Генерация представления для ввода параметров произвольной бизнес-операции. Является наследником класса [DynamicPropertiesObject](http://documentation.xafari.org/frlrfxafaribcdynamicpropertiesdynamicpropertiesobjectclasstopic.html), добавляя специфику работы с параметрами бизнес-операций.