# Глава 6

# Клиент Microsoft Dynamics AX

**В этой главе**

- Введение
- Работа с формами
- Работа с элементами управления
- Использование элементов форм
- Элементы навигации
- Изменение кода экранных форм
- Интеграция с клиентом Microsoft Office

## Введение

По своей сути клиент Microsoft Dynamics AX – это основанное на формах Windows-приложение, которое позволяет пользователю взаимодействовать с данными, хранящимися на сервере. В него можно внести изменения, позволяющие обрабатывать новые типы данных, или же изменить способ работы пользователей с уже существующими типами данных. Пользовательский интерфейс складывается из форм, объявленных в метаданных, и часто содержит в себе еще и сопутствующий код.

Клиент Microsoft Dynamics AX 2012 включает в себя несколько изменений и дополнений по сравнению с предыдущими версиями. Некоторые из наиболее существенных изменений стоит упомянуть сразу. Это новые способы представления записей заголовков документов (формы детализации заголовка) и строк документов (страницы списков и детализации строки документа); новый растягивающийся по вертикали элемент управления «экспресс-вкладка» (FastTab); использование панелей операций и полос панелей операций, позволяющих визуально выделить доступные действия. И, наконец, появление информационных панелей (FactBoxes) с выводимой в них релевантной информацией. Более подробную информацию см. в главе 5.

Данная глава в основном посвящена ключевым аспектам клиента Microsoft Dynamics AX. Тем не менее большая часть информации в ней носит обзорный характер. За более подробной информацией обратитесь к разделу «Client» в Microsoft Dynamics AX 2012 SDK по адресу: *http://msdn.microsoft.com/en-us/library/gg880996*.

## Работа с формами

Форма – это основная *единица отображения* в клиенте. В типичной форме отображаются поля, показывающие текущую запись, кнопки, дающие доступ к действиям над ней и доступные пользователю в текущий момент, а также механизм для перехода по записям.

Создавая форму, вы должны использовать AOT для определения метаданных формы. Если нужно, всегда есть возможность написать код, обрабатывающий любые события образом, недоступным для чисто декларативного описания путем изменения метаданных. Следующие шаги высокого уровня описывают процесс создания формы.

1. Создайте ресурс (элемент AOT) формы. Вы можете создать новую форму с чистого листа, но зачастую в качестве отправной точки можно взять уже существующую форму. Создав форму, убедитесь, что задали свойство *Caption* на узле *Design* формы. Это важный, но часто упускаемый из виду, шаг.
2. Добавьте источники данных и информацию об их связях. Если требуется, задайте особые запросы и фильтры.
3. Добавьте элементы управления на форму. Вы можете добавить элементы, привязанные к полям источников данных для отображения их содержимого, и активные элементы, такие как кнопки и панель операций, предоставляющие пользователю доступ к различным действиям над текущей записью.
4. Добавьте элементы формы, отображающие данные, относящиеся к главной записи. Элементы формы могут сократить путь к информации, нужной пользователю.
5. Добавьте навигационные элементы, чтобы пользователи смогли воспользоваться вашей формой. Создайте элемент управления *MenuItem* (пункт меню), указывающий на форму. Добавьте ссылку на этот пункт меню в раздел *Menus* или в другие формы, используя элемент *Menu-*

*ItemButton* (кнопка пункта меню), чтобы дать пользователю возможность открыть форму.

6. Переопределите методы формы или ее элементов управления, если вы не можете достичь нужного поведения, манипулируя метаданными.
7. Добавьте бизнес-логику, реализующую новую функциональность формы в классы приложения.

Последующие разделы в этой главе содержат более подробную информацию о составляющих каждого шага. Для получения обновленной информации и наиболее свежих примеров о построении и кастомизации форм посетите раздел «Client» в Microsoft Dynamics AX SDK по адресу: *http://msdn.microsoft.com/ru-ru/library/gg880996*.

## Способы проектирования форм

В предыдущих версиях Microsoft Dynamics AX использовалось некоторое количество неформальных способов проектирования форм. В Microsoft Dynamics AX 2012 несколько из них были формализованы и предлагаются теперь как шаблоны.

Когда вы создаете форму, выбирайте такой вариант проектирования, который наилучшим образом подходит к типу данных, выводимых в форме, и способу диалога с пользователем. В разделе «Form User Experience Guidelines» в MSDN (*http://msdn.microsoft.com/EN-US/library/gg886605*) обсуждается каждый из них. Эти рекомендации будут полезны для обеспечения нормального взаимодействия как новых форм, так и уже существующих в Microsoft Dynamics AX. После выбора варианта проектирования формы вы можете создать форму по какому-либо из шаблонов.

- В AOT щелкните правой кнопкой на узле Forms, далее – на New Form from Template, затем выберите нужный вам шаблон.

Microsoft Dynamics AX создаст форму из шаблона, описывающего в себе свойства и элементы управления, воплощающие структуру формы, определенную способом ее проектирования. Табл. 6-1 описывает доступные шаблоны форм и назначение каждого типа форм.

**Табл. 6–1.** Шаблоны форм

| **Вариант представления** | **Шаблон** | **Назначение** |
|---|---|---|
| Страница списка | ListPage | Поиск записи и осуществление действия над ней. Отдельная форма детализации – для показа подробностей записи. Этот способ предназначен для работы с записями-заголовками документов или записями-сущностями.<br>Пример: *CustTableListPage*.<br>Чтобы открыть эту форму, нажмите Расчеты с клиентами > Обычный > Клиенты > Все клиенты |
| Форма детализации | DetailsFormMaster | Просмотр, ввод, обновление и другие действия над конкретной записью. Этот способ предназначен для работы с записями-заголовками документов.<br>Пример: *CustTable*.<br>Чтобы открыть эту форму, нажмите Расчеты с клиентами > Обычный > Клиенты > Все клиенты и дважды щелкните на записи в списке |
| Форма детализации со строками | DetailsFormTransaction | Просмотр, ввод, обновление и другие действия над записью документа и его строками.<br>Пример: *SalesTable*.<br>Чтобы открыть эту форму, нажмите Расчеты с клиентами > Обычный > Заказы на продажу > Все заказы на продажу и дважды щелкните на записи в списке |
| Диалог | Dialog | Запуск задачи или процесса, требующий участия пользователя. Форма дает возможность пользователю продолжить или отменить его выполнение.<br>Пример: *DirPartyQuickCreateForm*.<br>Чтобы открыть эту форму, нажмите Расчеты с клиентами > Обычный > Клиенты > Все клиенты. На панели операций в группе Создать выберите Клиент |

**Табл. 6–1.** Шаблоны форм (*окончание*)

| **Вариант представления** | **Шаблон** | **Назначение** |
|---|---|---|
| Выпадающий диалог | DropDialog | Запуск задачи или процесса, требующий участия пользователя. Выпадающее диалоговое окно оперативно предоставляет небольшое количество дополнительной информации без необходимости покидать исходную форму.<br>Пример: *HcmWorkerNewWorker*.<br>Чтобы открыть эту форму, нажмите Управление организацией > Обычный > Работники > Сотрудники. На панели операций в группе Создать выберите Нанять нового работника |
| Простой список | SimpleList | Просмотр, ввод, обновление записей, представленных как список в табличном виде.<br>Пример: *CustGroup*.<br>Чтобы открыть эту форму, нажмите Расчеты с клиентами > Настройка > Клиенты > Группы клиентов |
| Простой список с детализацией | SimpleListDetails | Просмотр списка записей и одновременно подробностей о текущей записи. Этот вариант представления нацелен на простоту работы с записями различных справочников.<br>Пример: *CustPosting*.<br>Чтобы открыть эту форму, нажмите Расчеты с клиентами > Настройка > Профили разноски по клиенту |
| Оглавление | TableOfContents | Единый массив связанных настроек или опций конфигурации. Этот вариант представления обычно применяется на формах параметров для облегчения доступа к параметрам, используемым текущим модулем.<br>Пример: *CustParameters*.<br>Чтобы открыть эту форму, нажмите Расчеты с клиентами > Настройка > Параметры модуля расчеты с клиентами |

**Примечание.** Для журналов и форм запросов не существует формализованных вариантов представления, так как они очень сильно зависят от обрабатывающихся в них данных и процессов, в которых участвуют.

## Метаданные форм

Метаданные форм в Microsoft Dynamics AX довольно обширны, но хорошо структурированы и легки в работе после небольшого знакомства с ними. Ниже мы выделим главные узлы метаданных в ресурсах формы.

- ***Form.DataSources***. Структуры данных, используемых в форме. Более подробно см. в разделе «Источники данных».
- ***Form.Designs.Design***. Элементы управления, отображающие данные в записях. Этот узел метаданных часто называют сокращенно Form. Design, или Дизайном Формы. Более подробно см. в разделе «Работа с элементами управления».
- ***Form.Parts***. Элементы формы, отображающие дополнительные данные. Более подробно см. в разделе «Работа с элементами формы».

На рис. 6-1 видны все эти узлы AOT на примере формы *CustGroup*.

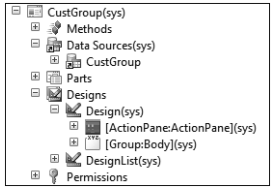



**Рис. 6–1.** Узлы метаданных для формы *CustGroup*

В идеале при изменении форм следует стремиться максимально использовать их метаданные. Такой способ изменения наиболее предпочтителен, потому что слияние изменений метаданных (так называемый *дельт*) провести проще, чем слияние изменений кода. При этом, чтобы обеспечить максимальный уровень повторного использования, любые изменения, делаемые в метаданных, должны быть сделаны на минимально возможном уровне, например на уровне таблицы, а не формы. При изменении форм следует помнить, что на вид формы и ее содержание также влияют связи метаданных и их наследование.

### Связи метаданных

Все метаданные в Microsoft Dynamics AX изменяются посредством редактирования AOT. И базовые определения форм, хранящиеся в узле *AOT\Forms,* основываются на иерархии метаданных, находящихся в других узлах AOT. Чтобы полностью понять структуру формы, вам нужно изучить связи метаданных, формируемые ею. Например, форма может использовать таблицу, определение которой задано в узле *AOT\Data Dictionary\Tables*, пункты меню, определенные в узле *AOT\Menu Items, запросы* из узла *AOT\Queries* и классы из *AOT\Classes*.

### Наследование метаданных

Кроме того, следует иметь в виду наследование в метаданных, используемых в формах. К примеру, таблица использует перечисления, расширенные типы данных (EDTs) и конфигурационные ключи. Еще один простой пример наследования – свойство *Image* кнопки пункта меню *MenuItemButton* наследуется от связанного с ним пункта меню *MenuItem*, если он не был явно переопределен на этой кнопке. Наследование также может быть среди форм. Элементы управления, содержащиеся внутри других элементов управления, автоматически получают значения некоторых свойств от своих родителей в случае, если в них не указаны другие значения этих свойств. Это относится к меткам, текстам подсказок, конфигурационным ключам, свойству Enabled и различным свойствам настройки шрифтов.

В табл. 6-2 показаны примеры метаданных, наследуемых из связанных метаданных.

**Табл. 6–2.** Примеры наследования метаданных

| Тип метаданных | Источник |
|---|---|
| Метки и тексты подсказок | Пункт меню > Кнопка пункта меню<br>Перечисление > Расширенный тип данных > Поле таблицы > Поле источника данных формы > Элемент управления формы<br>(Свойство перечисления *Help* это эквивалент свойства *HelpText* в остальных типах метаданных) |
| Длина отображения | Расширенный тип данных > Поле таблицы > Элемент управления формы |
| Конфигурационные ключи | Перечисление > Расширенный тип данных > Поле таблицы > Поле источника данных формы > Элемент управления формы |
| Свойства Image* (например *NormalImage*) | Пункт меню > Кнопка пункта меню |

## Источники данных

Microsoft Dynamics AX предоставляет богатый инструментарий доступа к данным, тем самым облегчая добавление данных на формы и привязку элементов управления к этим данным. Его основой является источник данных формы (form data source), который позволяет привязать таблицы и поля к форме.

Источник данных формы указывает на какую-либо таблицу, карту соответствия (map) или представление (view). Список полей в источнике данных автоматически заполняется полями, определенными на объекте, на который они ссылаются. Вы можете привязать к элементам управления любое поле или метод данных, существующие на таблице либо источнике данных формы.

Источники данных форм могут быть разделены на следующие категории.

- **Корневые**. Корневые источники данных не содержат заданного значения в свойстве *JoinSource* и поэтому не объединены и не связаны с любым другим источником данных. Подавляющее большинство форм имеют только один корневой источник данных, который ссылается на таблицу, являющуюся основным содержанием формы. Иногда корневые источники данных называют источниками данных верхнего уровня.
- **Главные**. **Главные источники данных – корневые или динамически связанные**. Для главного и объединенных с ним подчиненных источников данных для выборки данных из них используется один общий запрос. Можно представить главный источник данных как корень иерархии запроса.
- **Объединенные**. Объединенные источники данных – это те, которые объединены с другим источником данных. Они имеют значение свойства *LinkType* из ряда *InnerJoin*, *OuterJoin*, *ExistJoin* или *NotExistJoin*. Обычно объединение используется для связки источников данных так, чтобы данные извлекались одним запросом. Например, в форме *CustTable* источник данных *DirPartyTable* объединен с *CustTable*.
- **Связанные**. Связанные источники данных – это те, которые связаны с другим источником данных в форме. Они имеют значение свойства *LinkType* из ряда *Active*, *Delayed* или *Passive*. Такой тип связи используется для источников данных, имеющих отношение типа главный/подчиненный, для того чтобы данные выбирались отдельными запро-

сами. Например, в форме *SalesTable* источник данных *SalesLine* связан с *SalesTable*.

### Динамическое связывание

Термин *Динамическое связывание* относится к двум источникам данных, связь между которыми устанавливается динамически. Динамическая связь всегда имеет главный (родительский) источник данных и подчиненный ему.

К примеру, форма *SalesTable* (Заказы на продажу), где источник данных *SalesTable* (Заголовки заказов на продажу) является главным, а *SalesLine* (Строки заказов на продажу) – подчиненным. Если два источника данных имеют между собой динамическую связь, то при изменении записи в родительском источнике, подчиненный получает уведомление об этом. При этом запрос в подчиненном источнике выполняется заново для выборки соответствующих связанных данных.

Существуют следующие типы динамической связи.

- **Внутри формы.** Такие связи устанавливаются между источниками данных, у которых свойство *LinkType* у подчиненного источника установлено в *Active*, *Passive* или *Delayed*. Подчиненный источник имеет запрос, выполняющийся отдельно от родительского источника данных и выполняющийся в моменты времени, определяемые значением свойства *LinkType*.
- **Между формами.** Связи между формами устанавливаются между связанными источниками данных на формах, где одна из них (родительская) открывает другую (подчиненную). Для указания того, какой из источников данных будет главным, используется свойство *DataSource* кнопки пункта меню на главной форме. На открываемой же подчиненной форме связанным всегда выбирается первый по порядку корневой источник данных.

### Наследование таблиц

Наследование таблиц в Microsoft Dynamics AX работает во многом аналогично наследованию классов в любом объектно-ориентированном языке. Кроме того, такое наследование имеет дополнительное преимущество – возможность полиморфных запросов данных из таблиц. (Более подробно о наследовании таблиц см. в главе 17.)

Если вы создадите источник данных на базе таблицы, являющейся базовым типом, производные типы автоматически будут развернуты в

подузле *Derived Data Sources*. Этот узел нередактируемый и порождается внутренним фреймворком форм. Производные источники данных сами по себе не имеют свойств или методов, так как все их характеристики унаследованы от базового источника данных. Например, источник данных *DirPartyTable* на рис. 6-2 это – часть иерархии наследования таблиц.

На рис. 6-2 показаны все автоматически сгенерированные источники данных, по одному для каждого производного типа. Узел *Fields* для каждого производного типа содержит список полей, принадлежащих этому типу. В случае если существует только одна цепь иерархии базовых типов, все базовые поля будут собраны в один узел *Fields* под источником данных формы. Например, если вы создадите источник данных, основывающийся на типе *DirPerson*, узел *Fields* будет содержать все поля из цепи.

Методы экземпляра источника данных формы также подчиняются иерархии наследования таблиц. В качестве примера можно привести активацию пользователем события *validateWrite* при текущей активной записи типа *DirPerson*. При этом будет вызван метод *FormDataSource.validateWrite*, который вызовет метод *DirPerson.validateWrite*, который в свою очередь вызовет *DirPartyTable.validateWrite*. Завершит цепочку вызовов метод ядра *Common.validateWrite*. Обратите внимание, что независимые от экземпляра таблицы методы, такие как *executeQuery,* работающие на источниках данных в целом, не приведут к вызову никаких методов базовых типов таблиц.

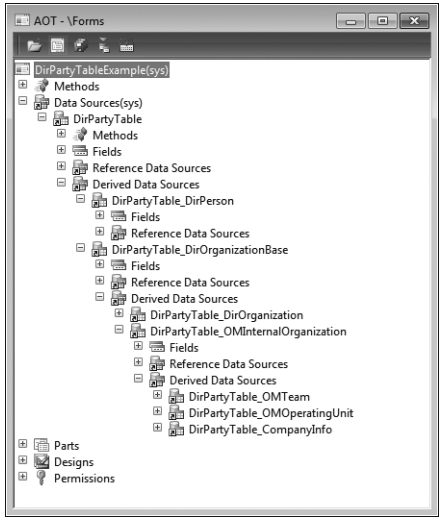



**Рис. 6–2.** Источник данных *DirPartyTable* в AOT

Так как допустимы полиморфные запросы, то допустимо и полиморфное создание записей. Когда пользователь нажимает Создать на форме с источником данных без производных типов, то конкретный тип создаваемой записи известен. Но, когда источник данных имеет производные типы, пользователю должен быть выдан запрос о том, запись какого типа он хочет создать.

Традиционно, чтобы создать запись в коде X++, вам было достаточно вызвать метод *FormDataSource.create*. Но этот метод не позволяет выбрать тип. Чтобы реализовать сценарий полиморфного создания записей, используйте следующий метод:

```
FormRun.createRecord(str _formDataSourceName [, boolean _append = false])
```

Все действия ядра по созданию записей проводятся путем использования этого метода. Так что правильнее вместо вызова *create* использовать именно его. Первый параметр указывает имя источника данных, в котором нужно создать запись, второй содержит тоже значение *append*, которое ранее передавалось методу *create*. Метод *createRecord* можно перекрыть для реализации кода, зависящего от создаваемого типа. Вызов *super* исполняет логику создания записи на основании типа создаваемой записи.

Если тип (или любые типы в иерархии объединения) является полиморфным, то пользователю будет выдан запрос о типе той записи, которую он желает создать, как показано на рис. 6-3.

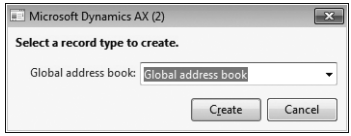



**Рис. 6–3.** Диалоговое окно запроса о типе создаваемой записи

Метод *createRecord* позволяет переопределить поведение *super* для того, чтобы явно указать типы записей, доступные для создания пользователем, или чтобы отобразить собственное диалоговое окно. Чтобы ограничить пользователя в типах создаваемых записей, следует передать ядру список нужных типов в следующем методе:

```
FormDataSource.createTypes(Map _concreteTypesToCreate [, boolean _append = false])
```

Первый параметр содержит карту пар типа *string* вида ключ/значение, где ключ – это название источника данных формы, а значение – имя табличного типа, доступного для создания. Например, если ваша цель – всегда создавать записи типа *CompanyInfo*, вам следует написать следу-

ющий код. Обратите внимание, что имя источника данных – это то имя, которое указано для него в самой форме, а не имя таблицы, на которой он основывается.

```
public void createRecord(str _formDataSourceName, boolean _append = false)
{
    Map typestoCreate = new Map(Types::String, Types::String);

    If (_formDataSourceName == "DirPartyTable")
    {
        typestoCreate.insert("DirPartyTable", "CompanyInfo");
        DirPartyTable_ds.createTypes(typestoCreate, _append);
    }
     else
    {
        super(_formDataSourceName, _append);
    }
}
```

**Пакет изменения**

Сохранение записей в источниках данных формы может происходить двумя путями. Традиционный путь – сохранение записей, соединенных внутренним и внешним объединением, производится вместе одним процессом, однако каждая запись сохраняется отдельными вызовами RPC и транзакциями. Это может вызывать проблемы, так как иногда хотелось бы, чтобы все записи сохранялись в одной транзакции.

Этого можно достичь, установив на узле *Data Sources* формы свойство *ChangeGroupMode* в *ImplicitInnerOuter*.

(По умолчанию оно установлено в *None*, что соответствует поведению, описанному в начале.) При значении же *ImplicitInnerOuter* все записи, вовлеченные во внутреннее либо внешнее объединение, будут сгруппированы в один вызов RPC к серверу и произойдут в одной транзакции. Если по какой-либо причине транзакция не будет успешно завершена, все изменения будут отменены. Эта возможность и называется «пакетом изменений». Более подробно о пакетах изменений вы можете прочитать в главе 17.

При использовании этого нового подхода методы *write* и *delete* становятся неприменимыми, так как действия над записью производятся в вызове *super* этих методов. В случае с режимом группового изменения следует использовать методы *writing*, *written*, *deleting* и *deleted*. Для каждого типа операций, после выполнения методов *validate* для каждого из источников данных на клиенте, вызываются методы, заканчивающиеся на *-ing*, такие

как *writing*. Затем происходит транзакция на сервере, где вызываются табличные методы *insert*, *update* или *delete*. И в конце вызываются методы на *-ed* или *-en*, такие как *deleted* или *written*.

*Пакет изменения имеет также дополнительный режим, называемый OptionalRecord, который* экономит место в базе данных, вставляя записи, находящиеся во внешнем объединении, только если значения были изменены относительно значений по умолчанию. Для режима *OptionalRecord* существуют две возможных опции – *ImplicitCreate* и *ExplicitCreate*. В случае сценария с *ImplicitCreate* механизм управления формами автоматически создает запись во внешнем объединении, если такая запись не существует в базе данных. Однако она сохраняется, только если ее значения не были изменены относительно значений по умолчанию. В случае сценария *ExplicitCreate* вы можете создать на форме флажок (check-box), который явно будет задавать поведение при создании и удалении записи, находящейся во внешнем объединении.

#### Действительные даты

Действительные даты позволяют отследить, как данные изменяются во времени. Функциональность действительных дат позволяет пользователям с соответствующими правами доступа увидеть всю историю изменений записи. Также она предоставляет возможность создать запись, которая вступит в действие, начиная с указанной даты. Процентные ставки или курсы обмена валют – вот хорошие примеры записей, актуальных с какой-либо даты, так как имеют смысл только на вполне определенных отрезках времени. Чтобы узнать больше, прочитайте главу 17 или скачайте официальное описание «Using date effective data patterns» по адресу: *http://download.microsoft.com/download/4/E/3/4E36B655-568E-4D4A-B161-152B28BAAF30/Using_Date_Effective_Patterns_AX2012.pdf*.

#### Суррогатные внешние ключи

Традиционные внешние ключи в Microsoft Dynamics AX обычно строились на естественных ключах или каких-либо типах сложных ключей. Такое построение имело много недостатков, например, происходило разрушение ссылочной целостности, если значение естественного ключа было изменено. Для решения этих проблем и улучшения производительности, благодаря использованию целочисленных ключей, в Microsoft Dynamics AX 2012 была добавлена поддержка суррогатных ключей. Последняя основана на использовании ссылочных источников данных и элементов управления Reference Group. В главе 17 можно получить больше информации.

### Метаданные источников данных

В табл. 6-3 описаны некоторые из наиболее важных свойств источника данных формы.

**Табл. 6–3.** Метаданные свойств источника данных формы

| Свойство | Описание |
|---|---|
| *Name* | Указывает наименование ссылки на источник данных. В качестве значения рекомендуется использовать название таблицы, на которую указывает ссылка |
| *Table* | Задает таблицу, используемую как источник данных |
| *CrossCompanyAutoQuery* | Определяет, будет ли источник данных выбирать их из:<br>• ***No*** – только текущей компании (по умолчанию);<br>• ***Yes*** – всех компаний в текущем разделе (например, для показа справочника всех клиентов из всех компаний) |
| *JoinSource* | Устанавливает, с каким источником данных будет связан этот источник данных, или он войдет как часть запроса. К примеру, на форме *SalesTable* источник данных *SalesLine* связан с источником данных *SalesTable*. Разница в связанных и объединенных источниках в том, будут они представлены отдельными запросами в БД либо одним, соответственно |
| *LinkType* | Назначает тип связи или объединения этого источника данных и указанного в свойстве *JoinSource*. ***Объединение обычно требуется при отображении данных из нескольких источников данных в одном и том же элементе управления Grid.*** Разница в связанных и объединенных источниках в том, будут они представлены отдельными запросами в БД либо одним, соответственно.<br>**Динамические связи**<br>Возможны следующие варианты.<br>• ***Delayed*** (по умолчанию). Перед обновлением связанных подчиненных источников данных будет сделана пауза, что обеспечит быстрый переход по записям в родительском источнике данных, так как записи в подчиненных источниках данных не будут выбираться для каждой выбранной записи родительского источника. Например, пользователь может перейти на несколько заказов ниже без отображения строк каждого заказа.<br>• ***Active.*** Подчиненные источники данных обновляются немедленно при выборе пользователем записи в главном источнике данных. Такие постоянные обновления потребляют довольно много ресурсов. |

**Табл. 6–3.** Метаданные свойств источника данных формы (*продолжение*)

| Свойство | Описание |
|---|---|
| | • ***Passive***. Связанные источники данных не будут автоматически обновляться при выборе новой записи в главном источнике данных. Связь источников отслеживается ядром системы, однако для запуска запроса на выборку разработчик должен явно вызвать метод *executeQuery* на связанном источнике данных.<br>**Объединения**<br>Возможны следующие варианты.<br>• ***InnerJoin***. Выбирает только те записи из основной таблицы, для которых существуют связанные записи в объединенной таблице, и наоборот. Для каждого совпадения выбирается одна комбинация записей. Записи, для которых нет соответствия в объединенном источнике данных, в результирующую выборку не попадают.<br>• ***OuterJoin***. Выбирает записи из главной таблицы без учета наличия соответствующих им записей в подчиненной объединенной таблице. Внешнее объединение не требует наличия соответствующей записи в подчиненной таблице для каждой записи в главной таблице.<br>• ***ExistJoin.*** Выбирает запись в главной таблице, только если есть соответствующая запись в подчиненной объединенной таблице. В выборку-результат данные из подчиненной таблицы не выбираются.<br>• ***NotExistJoin.*** Выбирает записи из главной таблицы, для которых нет соответствующих записей в объединенной подчиненной таблице. В выборку-результат данные из подчиненной таблицы не выбираются |
| *InsertIfEmpty* | Возможны следующие варианты.<br>• ***Yes*** (по умолчанию). Если в источнике данных нет записей, то автоматически будет создана новая запись для ввода пользователем.<br>• ***No.*** Пользователь должен будет создать новую запись вручную. Обычно этот режим используется, если в процессе создания записи нужно использовать специальный интерфейс или процесс |

**Табл. 6–3.** Метаданные свойств источника данных формы (*окончание*)

| Свойство | Описание |
|---|---|
| *AutoSearch* | • ***Yes*** (по умолчанию). Во время вызова ***super*** в *FormRun.run* будет автоматически вызван *executeQuery*.<br>• ***No.*** Во время вызова ***super*** в *FormRun.run executeQuery* не будет автоматически вызван.<br>Это свойство имеет значение только для корневых источников данных |
| *AutoQuery* | • ***Yes*** (по умолчанию). Запрос в источнике данных будет создан автоматически механизмом *FormRun*. Запрос будет содержать информацию по объекту *QueryBuildDataSource* для каждого объекта источника данных формы *(FormDataSource)* в иерархии объединения.<br>• ***No.*** Механизм *FormRun* не будет автоматически создавать запрос. Разработчику нужно явно указать используемый запрос, присваивая значение объекту *FormDataSource.query* во время исполнения метода *FormDataSource.init*.<br>Это свойство имеет значение только для главных источников данных |
| *OnlyFetchActive* | • ***No*** (по умолчанию). Все поля *FormDataSource будут* выбраны в *QueryBuildDataSource*. Эквивалентом будет установка значения свойства *Dynamic* объекта *QueryBuildDataSource.FieldList* в *Yes.*<br>• ***Yes***. В поля для выборки *QueryBuildDataSource* будут добавлены только поля, связанные с элементами управления. Это сокращает объем данных, вовлекаемых в обработку, и увеличивает быстродействие на таблицах с большим средним объемом одной записи |

## Запросы в формах

Одним из наиболее часто используемых путей кастомизации форм – это изменение запросов, используемых формой. Есть два пути сделать это: использовать свойства *AutoQuery* или прямо указать нужный запрос. Во время загрузки формы в источниках данных происходят следующие события.

1. Создаются объекты источников данных формы, ссылающиеся на данные, выбирающиеся из базы данных.
2. Выполняются запросы формы, выбирающие данные.
3. Элементы управления, связанные с полями, отображают выбранные данные.

По умолчанию, когда свойство *FormDataSource.**AutoQuery установлено в Yes*, созданный запрос основывается на источниках данных формы. Запрос создается в вызове метода *super* формы *init*. Запрос содержит по одному объекту *QueryBuildDataSource **для каждого источника данных в прямой иерархии объединения***. Для динамически связанных источников данных создаются отдельные запросы, которые привязаны к текущим данным их связанных родителей, чтобы обеспечить правильность запросов. Такое разделение таблиц по нескольким запросам к БД имеет преимущество в виде возможности увеличить производительность и убрать перемножение результатов, которое может произойти в объединениях вида 1:*n*. На рис. 6-4 показаны источники данных для иллюстрации свойства *AutoQuery*.

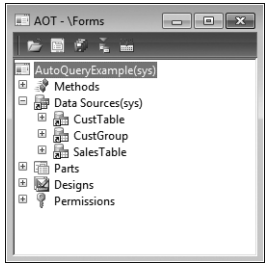



**Рис. 6–4.** Форма *AutoQueryExample* в AOT

В этом примере свойства на источниках данных установлены следующим образом.

- Источник данных *CustTable* – корневой источник данных, следовательно его свойство *JoinSource* пусто.
- Источник данных *CustGroup* имеет свойство *JoinSource*, установленное в значение *CustTable*, и *LinkType* – в *OuterJoin*.
- Наконец, источник данных *SalesTable* имеет *JoinSource* равным *CustTable* и *LinkType* равным *Delayed*.

Согласно поведению *AutoQuery,* будет создан запрос с корневым объектом *QueryBuildDataSource* с таблицей *CustTable*, который имеет подчиненный источник данных *CustGroup*. Так как источник данных *SalesTable* связан динамической связью, он имеет собственный запрос с единственным корневым объектом *QueryBuildDataSource*. Объект *QueryBuildDataSource* с таблицей *SalesTable* имеет динамическую связь, добавленную к нему с помощью вызова метода *addDynalink* на источнике данных *CustTable*. Когда загружается форма, сначала запускается запрос на выборку

из *CustTable* и *CustGroup*. После того как будут получены его результаты, запустится запрос на выборку данных из *SalesTable*, основываясь на выбранной записи в текущий момент в *CustTable*.

Второй путь моделирования доступа к данным из формы – использование запроса как базиса для структуры источников данных. Вы можете сделать это, перетащив запрос на узел *Data* ***Sources, либо установив значение свойства Query этого узла***. В этом сценарии запрос заставляет форму сгенерировать соответствующие источники данных. При этом добавление других источников данных блокируется. Этот метод моделирования обычно применяется только для страниц списков, потому что он требует создания двух объектов метаданных для реализации формы. Эта дополнительная работа в случае со страницами списков имеет преимущество в создании сложных запросов, содержащих фильтры для вторичных форм списков, повторно использующих исходную страницу списка. Например, форма Клиенты на удержании, которая и есть вторичная страница списка, повторно использует страницу списка Клиенты.

#### Методы *QueryBuildDataSource* и *QueryRunQueryBuildDataSource*

Важное изменение в Microsoft Dynamics AX 2012 – это добавление методов *FormDataSource.queryBuildDataSource* и *FormDataSource.**queryRunQueryBuildDataSource***. Они открывают доступ к соответствующим объектам *Query* и *QueryBuildDataSource,* которые используются формой. Данные методы дают разработчикам X++ возможность быстро обратиться к правильному объекту *QueryBuildDataSource* источника данных формы. Это особенно полезно, если форма имеет несколько источников данных одного и того же типа, как, например, в форме *CustTable*.

#### Объекты *Query* и *QueryRun*

Важная часть взаимодействия формы с запросом – это доступ к правильному запросу. Для каждого источника данных на форме есть два запроса: *FormDataSource.query* и *FormDataSource.queryRun.query*. Когда форма открывается впервые, объект *Query* создается в вызове *super в* методе *FormDataSource.init*. Если вы хотите внести изменения, которые останутся в запросе при любой его модификации и изменении фильтров пользователем, то они должны быть внесены именно в этот запрос. В вызове *super* метода *executeQuery* создается объект *QueryRun,* содержащий копию исходного объекта *Query*. Когда пользователь применяет фильтр, то он применяется к объекту *QueryRun.query,* и тут же вызывается метод *research*. Устанавливается внутренний флажок, указывающий не создавать повторно объ-

ект *QueryRun*, затем вызывается метод *executeQuery*. Когда пользователь сбрасывает фильтрацию, снова происходит вызов *executeQuery,* который по умолчанию пересоздает объект *QueryRun из исходного объекта Query*.

### Свойство *CopyCallerQuery*

*CopyCallerQuery – это* свойство элемента управления *MenuItemButton* и ресурса *MenuItem*, которое указывает, нужно ли копировать запрос из исходной формы в вызываемую. При использовании *CopyCallerQuery*, используются те же правила, которые применяются при программном назначении запроса через код.

- Корневой источник данных в запросах должен совпадать с источником данных формы.
- Объединенные источники данных в запросе также должны быть совместимы.

В Microsoft Dynamics AX 2012 ядро автоматически добавит любые недостающие объекты *QueryBuildDataSource,* необходимые для источников данных формы. Это делает запросы настолько совместимыми, насколько возможно.

### Фильтры запросов (*QueryFilter*)

В Microsoft Dynamics AX 2012 фильтрация в формах осуществляется путем использования объектов *QueryFilter* вместо объектов *QueryBuildRange в предыдущих версиях*. Эти *QueryBuildRanges* применяются к выражению *ON* в Transact-SQL, которые корректно работают во внутренних объединениях, поскольку выражения *ON* и выражения *WHERE* ведут себя одинаково. Однако это не работает ожидаемым образом для внешних объединений. Ожидаемое поведение для источников данных во внешнем объединении – применение выражения *WHERE* ко всему запросу не так, как действует выражение *ON*, которое ограничивает только объединение. Чтобы решить эту проблему, и был создан класс *QueryFilter*. Вся внутренняя логика ядра в механизме форм, которая модифицирует запросы, работает с объектами *QueryFilter*. Для обеспечения единства подхода, рекомендуется всю новую логику на формах основывать на использовании объектов *QueryFilter* вместо объектов *QueryBuildRange*.

Для расширенной информации о фильтрации запросов обратитесь к главе 17.

# Работа с элементами управления

В Microsoft Dynamics AX входит широкий набор элементов управления, который позволяет быстро создавать управляемые данными формы.

Когда вы добавляете на форму элементы управления, старайтесь установить минимальное количество их свойств. Тогда они смогут использовать все преимущества значений по умолчанию и автоматических значений. Значения свойств *Default* и *Auto* на элементах управления дают Microsoft Dynamics AX возможность использовать предопределенную функциональность для задания характеристик отображения и поведения.

**Примечание**. Само приложение является превосходным источником информации о построении форм. Вы можете изучить построение форм в приложении и узнать, какие элементы управления там использованы.

## Перегрузка элементов управления

Каждый элемент управления имеет набор методов, которые вы можете перекрыть. Старайтесь, чтобы в перегруженном методе на форме было как можно меньше кода. Если возможно, вызывайте сразу метод какого-либо экземпляра класса или статический метод.

## Привязка данных к элементам управления

Большая часть элементов управления может быть напрямую привязана к источникам данных и их полям для отображения значения из поля данных. Остальные элементы управления, такие как кнопки, могут быть привязаны к источникам данных для получения контекста данных. Если элемент управления не привязан явно к источнику данных, его контекст данных определяется контекстом его родительского элемента или умолчаниями формы, если и на родительском элементе управления источник данных не указан.

Подразумеваемый контекст источника данных чрезвычайно важен при исполнении действий и сохранении данных.

- При выполнении действия оно выполняется в контексте определенного источника данных. Например, когда инициируется действие по созданию новой записи, то, какая запись будет создана, зависит от текущего контекста данных.

- Если изменения в записи не были сохранены, то при перемещении фокуса с элемента управления в одном контексте данных на элемент управления в другом контексте данных запись будет автоматически сохранена.

## Свойства узла *Design*

Узел формы *Design* содержит элементы управления, которые отображают данные записей. Этот узел имеет свойства, наиболее важные из которых перечислены в табл. 6-4.

**Табл. 6–4.** Свойства метаданных узла *Design*

| Свойство | Описание |
|---|---|
| *Caption* | Указывает текст, отображаемый в заголовке окна стандартной формы или на панели фильтров страницы списка |
| *TitleDataSource* | Задает, из какого источника данных будет показываться информация в заголовке стандартной формы, а также какой источник будет использован для отображения текущего фильтра на панели фильтрации на страницах списков |
| *WindowType* | Определяет тип формы. Возможны следующие типы.<br>• ***Standard*** (по умолчанию). Стандартная SDI-форма, открывающаяся в отдельном окне с индивидуальным заголовком на панели задач Windows. Это тип по умолчанию.<br>• ***ContentPage***. Форма, которая заполняет область содержания рабочего пространства.<br>• ***ListPage.***Специальный тип *ContentPage,* который отображает записи в простой форме, предоставляя быстрый доступ к фильтрации и возможным действиям. Этот тип формы требует как минимум элементов управления панели операций (Action pane) и списка записей (Grid).<br>• ***Workspace***. MDI-форма, открывающаяся внутри рабочего пространства. Этот вид форм следует использовать только для форм, нацеленных на разработчиков.<br>• ***Popup***. Форма, открывающаяся как дочерняя для ее родителя. Всплывающие формы не имеют своего заголовка на панели задач Windows и не могут быть перекрыты другими окнами приложения |

**Табл. 6–4.** Свойства метаданных узла *Design (окончание)*

| Свойство | Описание |
|---|---|
| *AllowFormCompanyChange* | Здесь указывается, допускает ли форма смену компании при ее использовании как подчиненной с динамической связью, пересекающей границу компании. Доступны следующие варианты.<br>• ***No*** (по умолчанию). Форма закрывается, если главная форма сменяет свою компанию.<br>• ***Yes.*** Форма по мере надобности динамически изменяет свою компанию |

## Изменения в процессе выполнения

Можно добавлять и удалять элементы управления прямо во время исполнения в ответ на действия пользователя. Например, можно по необходимости добавлять и удалять элементы управления для фильтрации данных.

В момент изменения формы во время исполнения нужно ее заблокировать, вызвав метод *element.lock* и затем метод *element.unlock,* чтобы все изменения были отображены одновременно, а не были видимы последовательно, приближая форму к нужному виду. Кроме того, это увеличивает быстродействие, так как форма будет перерисована только один раз.

## Элементы управления действия

Для выполнения пользователем действий над текущей записью предназначены элементы управления, такие как кнопки и панели действий.

### Кнопки

Для выполнения действия пользователь должен нажать на кнопку. Существуют кнопки следующих типов.

- ***MenuItemButton*** – кнопка пункта меню. Активизирует пункт меню (*MenuItem*) для открытия формы, запуска класса или отображения отчета. Это наиболее часто используемый тип кнопок, так как поведение кнопки задается в модели, вместо явного определения посредством кода X++. Для более подробной информации обратитесь к разделу «Элементы навигации» ниже в этой главе .
- ***CommandButton*** – выполняет системную команду, такую как **OK**, **Экспорт в Excel** и **Закрыть**. Этот тип кнопок используется там, где для действия, которое вы хотите реализовать, существует системная команда.

- ***Button*** – просто кнопка. Предоставляет метод *clicked*, который вы можете переопределить по своему желанию. Этот тип используется нечасто. Избегайте его использования, так как оно означает, что код будет располагаться прямо на форме.
- ***DropDialogButton*** – открывает форму раскрывающегося диалогового окна. Раскрывающиеся диалоговые окна позволяют пользователю быстро узнать нужную информацию или сделать выбор, необходимый для выполнения какого-либо действия. Например, с помощью такого диалогового окна пользователь может выбрать определенный тип удержания клиента.
- ***MenuButton*** – отображает меню. Этот тип кнопки может содержать в себе любой тип кнопок, кроме другого *MenuButton*.

Для типов *Button* или *MenuButton* нужно заполнить свойство *Text*. Но для типов командной кнопки (*CommandButton)* или кнопки пункта меню (*MenuItemButton)* надпись на кнопке неявно наследуется из команды или пункта меню, указанного на ней, соответственно.

Можно помещать кнопки прямо на форму или же внутрь группы (*ButtonGroup)*. В общем случае кнопки обычно располагаются внутри панели действий или полосы панели действий.

### Панель действий и полоса панели действий

Панель действий (рис. 6-5) отображает организованные в определенном порядке кнопки, представляющие собой действия, доступные в форме. Действие – это задача или операция, которые выполняются при щелчке пользователя на кнопке панели действий. Эти действия дают пользователям возможность выполнять команды, открывать связанные формы или исполнять другой X++-код, используя данные, отображаемые в текущей форме.

Используйте панель действий наверху больших форм, где вам потребуются несколько вкладок, чтобы отобразить все доступные в форме действия.


**Рис. 6–5.** Панель действий

Используйте полосы панелей действий наверху небольших форм, в которых можно совершить лишь небольшое количество действий. Такой тип панели действий можно создать, установив свойство *ActionPane.Style* в *Strip*.

Можно также использовать полосу панели действий для показа действий, имеющих определенный контекст. Типичное размещение полосы панели действий – наверху страницы вкладок (*TabPage*), экспресс-вкладок (*FastTab*) или элемента управления Группа (*Group*). В роли контекста могут выступать поля одной категории внутри записи или набор связанных дочерних записей. Чаще всего контекстно-зависимые полосы панелей действия используются при отображении действий Добавить и Удалить над списком записей, содержащим дочерние записи, как показано на рис. 6-6. При использовании полос панели действий в конкретном контексте данных, необходимо задать значение свойства *DataSource*.


**Рис. 6–6.** Полоса панели действий

Для более подробной информации об использовании панелей действий обратитесь к главе 5.

## Элементы управления размещением

Существуют три элемента управления для размещения внутри них других элементов.

- ***Group*** – элемент управления группа, для управления группировкой и разделения по категориям элементов внутри формы. Узел Design формы, поскольку также может содержать элементы, тоже обладает многими свойствами и поведением, подобными этому элементу управления.
- ***TabPage*** – элемент управления страница вкладок организует другие элементы управления и поля на форме так, что в отдельный момент времени из них видна только одна из частей.
- ***Grid*** – элемент управления список записей представляет расположенные на нем поля ввода в компактном простом виде строк и столбцов, одновременно отображая несколько полей из нескольких записей одного типа.

### Группа (Group)

Используйте элемент управления *Group* для организации в форме связанных полей и других управляющих элементов в логические группы. Вы можете создать и назвать (используя свойство *Caption*) группу вручную, а также создать элемент управления *Group,* используя свойство *DataGroup,* чтобы указать на группу полей, определенную на таблице (источнике данных).

Старайтесь использовать табличные группы полей везде, где это возможно. Группы полей на таблицах облегчают обслуживание системы, потому что одно изменение группы приведет к внесению изменений сразу во все формы и отчеты, где используется эта группа.

Если же вы вручную добавили элементы управления и назначили заголовок группе, убедитесь, что этот заголовок достаточно понятен и ясно описывает ее суть.

### Страница вкладок (TabPage)

Чтобы снизить сложность формы, используйте страницы вкладок путем скрытия полей, ненужных пользователю в текущий момент. Элементы управления *TabPage* содержатся в родительском элементе управления Вкладки (*Tab)*. Обычно этот элемент управления называют «группой вкладок» *(tab group),* чтобы отличать его от содержащихся в нем элементов вкладок.

Для вкладок доступны следующие стили.

- ***Standard.*** Показывает элементы *TabPage* в виде стопки один поверх другого. На экране видно содержимое только одной вкладки. Этот стиль используется по всему программному продукту и хорошо знаком большей части пользователей. Он проиллюстрирован на рис. 6-7.


**Рис. 6–7.** Закладки стиля *Standard*

- ***VerticalTabs.*** Показывает вкладки в виде вертикального списка ссылок слева от отображаемой вкладки. На экране видно содержимое только одной вкладки. Этот стиль обычно используется для форм параметров (рис. 6-8). Формы, имеющие такой стиль, часто называют «формами оглавления» (table of contents).

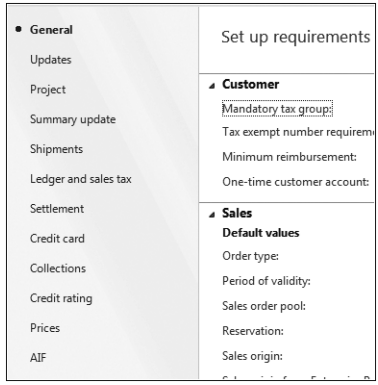



**Рис. 6–8.** Вертикальные закладки

- ***IndexTabs.*** Показывает вкладки в виде горизонтального списка вкладок под отображаемой частью вкладки. На экране видно содержимое только одной вкладки. Этот стиль, показанный на рис. 6-9, обычно применяется для вывода деталей строки в формах, выводящих подробную информацию о каком-либо документе.


**Рис. 6–9.** Индексные вкладки

- ***FastTabs (***экспресс-вкладки). Показывает вкладки в виде вертикального списка и позволяет пользователю увидеть одновременно несколько вкладок. Такой стиль дает возможность раскрывать (или оставлять свернутыми) нужные в текущий момент вкладки. Кроме того, на рис. 6-10 видно, что можно показать содержание ключевых полей даже на свернутой вкладке.

Чтобы стиль *FastTabs* у вкладок было действительно полезен пользователям, не раздувайте список полей на вкладках, присваивайте им понятные названия и не забывайте отображать на свернутых вкладках ключевые поля. Еще больше советов по созданию эффективных экспресс-вкладок вы найдете в главе 5.

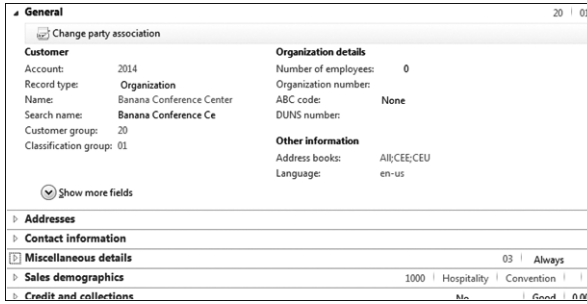



**Рис. 6–10.** Экспресс-вкладки *FastTabs*

### Список записей (Grid)

Список записей используется для вывода набора записей, связанных с главной записью формы. Например, вы можете вывести в него список контактов или адресов клиента.

Если вам не нужно, чтобы список записей занимал всю форму, можете ограничить его размер, задав свойство *VisibleRows*. Для того же примера с адресами – многие из клиентов имеют всего один или два адреса. Так что установка *VisibleRows* равному трем – это один из путей отобразить релевантную информацию, заняв минимум места.

## Элементы ввода

Элементы ввода могут быть как привязанными, так и в свободной форме. Привязанными они могут быть к полям, либо к методам.

### Элементы управления, привязанные к полям

Элементы ввода представляют собой поля на форме. Чтобы создать привязанные к полям элементы ввода, вы можете вручную вставить необходимые элементы управления в форму и затем указать в них нужные поля источника данных. Другая альтернатива состоит в перетаскивании полей из источника данных в форму следующим образом.

1. Нажмите правую кнопку мыши на узле Form.DataSources и щелкните Open In New Window.
2. Поместите новое окно, содержащее источники данных формы рядом с исходным окном AOT.
3. Перетащите нужные поля из источников данных и установите их в подходящее место формы – в группе, странице списка или в списке

записей. Это действие создаст нужный тип элемента ввода (StringEdit для строк, IntEdit для целых чисел и т. д.) и тут же привяжет его к полю источника данных.

#### Элементы управления, привязанные к методам

Если вы привязываете элемент ввода к методу отображения, у вас появляется возможность вывести данные, создаваемые и обрабатываемые внутри кода. Методы отображения, основывающиеся на какой-либо таблице, вы должны реализовать на этой таблице вместо источника данных формы.

Чтобы связать элемент управления с нужным методом отображения, используйте его свойства *Datasource и Datamethod*.

Методы отображения используют ключевое слово *display* в определении метода. Наилучшие примеры методов отображения можно почерпнуть в уже существующем коде Microsoft Dynamics AX. Используйте поиск по AOT, чтобы отыскать примеры методов отображения на таблицах, задав для поиска ключевое слово *display* с пробелом после него.

Типичный display-метод выглядит следующим образом:

```
display SomeEDT myDisplayMethod()
{
    //Code here...
    return "returnValue";
}
```

#### Свободные элементы управления

Также возможно просто поместить элементы ввода вроде *StringEdit и IntEdit* на форму и, манипулируя ими из кода X++, обеспечить тот способ взаимодействия с пользователем, который вы хотите. Пример с таким вариантом управления можно увидеть в форме *AxdWizard*. Кроме того, свободные элементы управления часто применяются как поля управления фильтрацией. Пример такого использования можно посмотреть в форме *SalesLineBackOrder*.

## Элемент управления *ManagedHost*

Если, используя встроенные в Microsoft Dynamics AX элементы управления, вам все же не удается достичь каких-то специфических целей или у вас уже есть готовый компонент и вы хотите сэкономить время, используйте внешний элемент управления. Используя элемент управления *ManagedHost*, в формах Microsoft Dynamics AX становится возможным применять элементы управления .NET. Использование элемента управления

*ManagedHost* – это наилучшее решение, когда вам нужен сторонний элемент управления, потому что с ним это делается легко и без усилий.

Чтобы в форме Microsoft Dynamics AX было возможно использовать элемент управления .NET, в AOT должна содержаться ссылка на сборку .NET с искомым элементом управления. Чтобы добавить новые элементы управления .NET, добавьте сборку (или сборки), реализующие их, в узел *Reference в* AOT, щелкнув на этом узле правой кнопкой и выбрав Add Reference.

Чтобы обратиться к этому элементу управления .NET из формы Microsoft Dynamics AX, добавьте на нее элемент управления *ManagedHost* и выберите в окне Managed Control Selector тот элемент управления, который вам нужен. Щелкните правой кнопкой на новом элементе управления, чтобы перехватить события от него.

В этом простом примере мы разберем вставку на форму кнопки из .NET.

1. Щелкните в AOT правой кнопкой мыши на узле Forms и выберите New.
2. На узле формы Design снова щелкните правой кнопкой мыши и затем укажите New Control > ManagedHost.
3. В диалоговом окне Managed Control Selector в верхнем списке выберите сборку *System.Windows.Forms*, затем в списке Controls выберите Button и щелкните OK.

**Примечание.** Заметьте, что сборка *System.Windows.Forms* уже включена в ссылки по умолчанию, так что добавлять ее не нужно.

4. Укажите название элемента управления *ManagedButton*.
5. Щелкнув правой кнопкой мыши на элементе управления *ManagedButton*, откройте диалоговое окно *Events* и добавьте событие *Click*.
6. Раскройте узел формы *Methods*, откройте метод init и, чтобы указать надпись на кнопке, замените его код на следующий:

```
public void init()
{
    super();
    _ManagedButton_Control = ManagedButton.control();
    _ManagedButton_Control.add_Click(new ManagedEventHandler(this, 'ManagedButton_
Click'));
```

```
        _ManagedButton_Control.set_Text("Managed button");
            }
```

7. Откройте исходный код метода *Click* и, чтобы по нажатию кнопки отобразить текст об этом в списке сообщений, замените его реализацию на следующую:

```
void ManagedButton_Click(System.Object sender, System.EventArgs e)
{
    info("Managed button clicked");
}
```

Запустите форму и нажмите на кнопку. Результат вы можете увидеть на рис. 6-11.

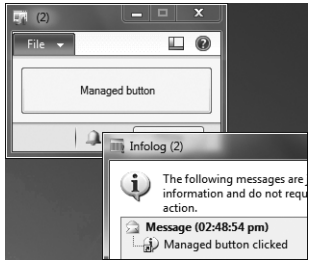



**Рис. 6–11.** Пример с элементом управления *ManagedHost*

### Прочие элементы управления

Чтобы отобразить дополнительную информацию либо увеличить интерактивность формы, на нее можно добавить и многие другие элементы управления.

Например, элемент управления *static text* может содержать в себе пояснения пользователю, а элемент управления *Window* – отображать картинки, помогающие быстро ориентироваться в продуктах или услугах.

Более детальную информацию см. в разделе «Controls in Microsoft Dynamics AX» по адресу: *http://msdn.microsoft.com/en-us/library/gg881259*.

## Использование элементов форм

Типичное использование элементов форм – выборка и отображение данных, относящихся к текущей записи в форме, содержащей элемент. Элементы могут быть использованы в информационных панелях (*FactBox*) любой формы, а также в области просмотра страницы списка.

## Типы элементов форм

Существуют следующие типы.

- **Элементы данных**, отображаемые в процессе выполнения формы. Во время разработки элементы данных используют упрощенный набор метаданных, позволяющий им работать одновременно в клиенте Microsoft Dynamics AX и веб-клиенте на Enterprise Portal. Они имеют простой вид и, по существу, представляют собой набор полей из какого-либо запроса. В элементах данных может задаваться набор действий, отображаемых внизу полей данных. Области просмотра страниц списков всегда проектируются в виде элементов данных.
- **Группы подсказок** – это наборы подсказок. Подсказки – это механизм для отображения числа записей, выбранных в запросе. Часто запрос, используемый в подсказке, основывается на текущей, отображаемой в основной форме записи. Подсказка содержит в себе три вещи: запрос, возвращающий количество; свойство *MenuItemName*, указывающее действие, которое будет осуществлено по щелчку на подсказке, и текст, поясняющий, к чему относится подсчитанное количество (если не указано, будет использован заголовок пункта меню). В главе 5 можно почерпнуть больше информации об использовании подсказок.
- **Элементы форм** – это указатели на существующие формы, которые могут быть отображены как информационные панели. Свойство *Form* задает форму, которая будет показана, а в свойстве *Caption* указывается заголовок информационной панели. Разрабатывая форму, которая будет использована как информационная панель, установите ее свойство *Style* в *FormPart*, свойство *ViewEditMode* – в *View*, а свойство *Width* – в *ColumnWidth*, чтобы обеспечить ей корректный внешний вид.

## Использование элементов форм в формах

В формах узел *Parts* указывает на элементы форм, используемые для отображения данных, относящихся к показываемой на форме записи. В этом узле *Parts*, чтобы обеспечить корректный контекст, вы должны создать ссылки на эти элементы.

Чтобы создать стандартную ссылку на элемент формы, выполните следующие действия.

1. Задайте значение *MenuItemName* пункту меню, вызывающему элемент формы. Пункты меню используются здесь для того, чтобы обеспечить работу механизма контроля доступа по пунктам меню.
2. Установите свойства *DataSourceName* и *DataSourceRelationName* для указания правильного режима связывания данных (динамической связи) между основной формой и элементом формы.
3. Определите значение свойству *PartLocation*, чтобы указать режим отображения элемента формы как информационной панели (по умолчанию) или как области просмотра.
4. (Необязательно. Только для страниц списков.) Задайте свойство *DisplayTarget* там, где элемент формы будет отображаться – в клиенте Microsoft Dynamics AX или на Enterprise Portal (или и там и там).
5. (Необязательно.) Свойством *Visible* можно скрыть информационную панель по умолчанию. Но она будет по-прежнему доступна пользователям, если они захотят ее отобразить.

## Элементы навигации

Чтобы пользователи могли воспользоваться созданными вами формами, вы должны вставить ссылки на них в различные меню.

### Пункт меню (*MenuItem*)

В Microsoft Dynamics AX пункт меню – это созданный указатель на какой-либо ресурс: форма, класс или отчет. Вы определяете метаданные для пунктов меню в узле AOT *Menu Items*. Этот узел имеет три подузла, служащие целям категоризации. Так, типы *Display*, *Action* и *Output* обычно ссылаются на формы, классы и отчеты, соответственно. Тем не менее существует общепринятое исключение в виде пункта меню *display* для класса, который служит для инициализации и открытия формы.

### Меню (*Menu*)

В Microsoft Dynamics AX *Menu* – это структурированный набор ссылок на пункты меню и другие меню. Область переходов, страницы области и строка адреса – механизмы, визуализирующие пользователю метаданные меню, определяемые вами в узлах AOT *Menus* и *Menu Items*. Меню модулей собраны в узле AOT *Menus\MainMenu*. С этой стартовой точки вы можете проследить всю структуру меню. Например, модуль Расчеты с клиентами

представлен ссылкой на меню (*MenuReference),* указанной в ***Menus\Main-Menu\AccountsReceivable,*** и определен в *Menus\AccountsReceivable*.

Добавляя в меню новый пункт, убедитесь в корректном указании свойства *IsDisplayedInContentArea*. Для страниц списков и содержания, отображаемых в клиенте, установите это свойство в *Yes*, чтобы в строке адреса отображался правильный путь.

## Создание меню

В предыдущих версиях Microsoft Dynamics AX формы обычно были специфичны для какого-либо модуля. Однако в Microsoft Dynamics AX 2012 приложение было реорганизовано в сторону большей ориентированности на роли. Как результат, были созданы новые модули, такие как Продажи и маркетинг и Управление запасами и складами. Многие из часто используемых форм теперь могут быть в нескольких модулях одновременно; например, форма Клиенты или Заказы на продажу теперь находится и в модуле Расчеты с клиентами, и в модуле Продажи и маркетинг.

Создавая меню модуля или добавляя новые пункты в существующие, старайтесь придерживаться стандартного способа группировки, использующегося в остальных меню.

- ***Обычный*** – содержит наиболее часто используемые формы в модуле. Группа Обычный, как правило, содержит ссылки на страницы списков.
- ***Периодические операции*** – содержит ссылки на формы с вторичными данными.
- ***Запросы*** – содержит ссылки на формы, предоставляющие данные модуля только для просмотра.
- ***Отчеты*** – содержит ссылки на отчеты.
- **Настройка** – содержит ссылки на формы настройки, в том числе и формы параметров. Иногда эта группа также содержит формы с вторичными данными.

Основные страницы списков, перечисленные в группе Обычный, должны дополняться вторичными страницами списков, добавляющими к главным страницам диапазоны (фильтры). Такие вторичные страницы можно реализовать как пункт меню, указывающий на главную страницу списка, но дополнительно задающий запрос с указанным на нем фильтром.

## Изменение кода экранных форм

В формах следует использовать код только в том случае, если нужного результата невозможно достичь с помощью изменения метаданных. Если вы изменяете форму, используя метаданные, то легче делать обновления. Конфликты метаданных разрешить легче, тогда как конфликты кода требуют более глубокого изучения. К тому же, это иногда приводит к созданию нового объединенного метода, пытающегося объединить в себе поведение двух конфликтующих методов.

При разработке в Microsoft Dynamics AX воспользуйтесь нижеследующими идеями как базой для дальнейшего изучения возможностей для разработки.

- Пользуясь командой *Find* (Ctrl+F) на узле AOT *Forms*, поищите примеры в базовом приложении Microsoft Dynamics AX 2012.
- За информацией о системных классах, таблицах, функциях, перечислениях и других системных объектах, реализованных в ядре AX, обратитесь к документации по системе в узле AOT\System Documentation.
- Пытаясь найти подходящее место для размещения своего кода, поставьте точку останова в методе *init* формы. Проследите в отладчике выполнение перекрытых методов. Помните, что в событиях элементов управления (таких как *Clicked*) точки останова не срабатывают. Если же вам нужно остановить отладчик в этих методах, добавьте ключевое слово *breakpoint* в их код X++.

Для упрощения сопровождения кода, придерживайтесь следующих рекомендаций.

- При работе с источниками данных форм используйте встроенные функции для работы с полями – *fieldNum*, вот так: *fieldNum(SalesTable, SalesId)*; и таблицами – *tableNum*, вот так: *tableNum(SalesTable)*.
- Избегайте жестко заданных в коде строк-констант. Вместо этого используйте метки, например так: *throw error("@SYS88659")*, и встроенные функции, такие как *fieldStr* и *tableStr*, возвращающие имена указанного поля или таблицы, соответственно.
- Перекрывайте как можно меньше методов. Каждое переопределение метода несет в себе риск создания конфликта слияния во время будущих обновлений, установки заплаток или других добавлений кода.

## Перекрытие методов

Переопределяя методы форм, вы можете влиять на способ функционирования формы и управлять откликом формы на действия пользователя. В табл. 6-5 описаны наиболее важные методы форм, которые разработчик может перекрыть. Наиболее часто подвергаются перекрытию методы *init* и *run*.

**Табл. 6–5.** Список перекрываемых методов формы

| Метод | Описание |
|---|---|
| *init* | Вызывается при инициализации формы. До вызова *super* большая часть формы (объекта *FormRun*) не проинициализирована, в том числе элементы управления и запросы. Этот метод обычно перекрывается тогда, когда доступ к форме нужен на как можно более ранней стадии |
| *run* | Вызывается, когда форма полностью проинициализирована. До вызова *super* сама форма остается готовой к показу пользователя, но еще невидима для него. В переопределении этого метода обычно делают изменения в элементах управления формы, ее внешнем виде и позиции фокуса |
| *createRecord* | Вызывается при создании записи в форме. Этот метод используется для перехвата события создания любой записи в форме. Кроме того, он также используется для задания типа записи при ее создании в наследуемой таблице. Дополнительную информацию см. в разделе «Наследование таблиц» ранее в этой главе |
| *close* | Вызывается при закрытии формы. Перекрывается для корректного освобождения ресурсов и сохранения пользовательских настроек и установок |
| *task* | Вызывается при выполнении пользователем системной задачи или команды на форме. В нем обрабатываются многие из типичных задач в форме |
| *activate* | Вызывается при активации формы (получении фокуса). Обычно используется для задания текущей компании при активации формы |
| *closeOk* | Вызывается при закрытии формы пользователем путем выбора команды OK, например, при нажатии командной кнопки (*CommandButton)* со свойством *Command,* установленным в *OK*. Обычно перекрывается в диалоговых окнах для осуществления необходимых действий, инициируемых пользователем |

**Табл. 6–5.** Список перекрываемых методов формы (*окончание*)

| Метод | Описание |
|---|---|
| *closeCancel* | Вызывается при закрытии формы пользователем путем выбора команды Отмена, например, при нажатии командной кнопки (*CommandButton*) со свойством *Command*, установленным в Cancel. Обычно перекрывается в диалоговых окнах, чтобы проделать необходимые действия по очистке при отмене пользователем запроса на совершение действия |
| *canClose* | Вызывается при попытке закрыть форму перед, собственно, закрытием. Обычно перекрывается, чтобы убедиться перед закрытием формы, что данные останутся в корректном состоянии. Если возвращаемое значение установить в *false*, то закрытие формы будет отменено, и она останется открытой |

Перекрывая методы на источниках данных формы и полях источников данных, вы можете влиять на то, как форма считывает, сохраняет данные и как откликается на действия пользователя над ними. В табл. 6-6 описаны наиболее важные методы источников данных формы, которые разработчик может перекрыть. Наиболее часто подвергаются переопределению методы *init*, *active*, *executeQuery*, *write* и *linkActive*.

**Табл. 6–6.** Список перекрываемых методов источника данных формы

| Метод | Описание |
|---|---|
| *active* | Вызывается при смене текущей активной записи источника, к примеру, когда пользователь щелкает на другой записи. Обычно перекрывается для включения/отключения доступа к кнопкам в зависимости от текущей записи |
| *create* | Вызывается при создании новой записи источника, к примеру, при нажатии пользователем комбинации клавиш Ctrl+N. Обычно перекрывается для изменения интерфейса пользователя с целью упрощения создания новой записи |
| *delete* | Вызывается при удалении выбранной записи источника, к примеру, при нажатии пользователем комбинации Alt+F9. Обычно перекрывается для изменения интерфейса пользователя с целью упрощения удаления записи |

**Табл. 6–6.** Список перекрываемых методов источника данных формы (*продолжение*)

| Метод | Описание |
|---|---|
| *deleting* | Вызывается перед началом удаления выбранной записи источника. Этот метод вызывается, только если свойству *ChangeGroupMode* на источнике данных задано значение *ImplicitInnerOuter*. Обычно перекрывается для изменения интерфейса пользователя с целью упрощения удаления записи |
| *deleted* | Вызывается после завершения удаления выбранной записи источника. Этот метод вызывается, только если свойству *ChangeGroupMode* на источнике данных задано значение *ImplicitInnerOuter*. Обычно перекрывается для изменения интерфейса пользователя в ответ на удаление записи |
| *executeQuery* | Вызывается при выполнении запроса источника данных, к примеру, при запуске формы (при вызове *super* в методе *run*) или по желанию пользователя обновить данные нажатием клавиши F5 (на самом деле, нажатие F5 вызывает метод *research*, который уже затем вызывает *executeQuery*). Обычно перекрывается при реализации пользовательской фильтрации на форме |
| *init* | Вызывается при инициализации источника данных (при вызове *super в методе init ф*ормы). Обычно перекрывается для добавления или удаления фильтров или изменения динамических связей источников данных формы |
| *initValue* | Вызывается сразу после создания новой записи источника. Заданные в этом методе значения каких-либо полей принимаются их значениями по умолчанию. Обычно перекрывается для задания значений по умолчанию. |
| *leaveRecord* | Вызывается при смене пользователем фокуса с одного источника данных (точнее, иерархии объединения источников данных) на другой. Иногда перекрывается для контроля за сохранением данных, которое при этом произойдет, однако рекомендуется вместо этого использовать методы *validateWrite* и *write*. Методы *validateWrite* и *write* как раз и вызываются в *super* метода *leaveRecord* |
| *linkActive* | Вызывается на дочернем источнике данных при вызове метода *active* динамически связанного источника данных родительской формы. Обычно перекрывается для обработки в пользовательском интерфейсе новой выбранной родительской записи (*element.args.record*) |

**Табл. 6–6.** Список перекрываемых методов источника данных формы (*продолжение*)

| Метод | Описание |
|---|---|
| *markChanged* | Вызывается при изменении набора маркированных записей, к примеру, при выборе пользователем нескольких строк источника данных. Обычно перекрывается для включения/отключения кнопок, которые могут обрабатывать несколько строк одновременно |
| *validateDelete* | Вызывается при попытке удаления пользователем записи в форме. Обычно перекрывается для проверки корректности удаления записи в зависимости от текущего состояния формы. Если возвращаемое значение установить в *false*, то запись удалена не будет. Используйте метод *validateDelete* на таблице для выполнения проверок возможности удаления записи в контексте таблицы. Это будет общим правилом для всех форм, использующих эту таблицу |
| *validateWrite* | Вызывается при попытке сохранения пользователем записи в форме, к примеру, при нажатии кнопки Сохранить, попытке закрытия формы или переходе фокуса на поле другого источника данных. Обычно перекрывается для выполнения проверок возможности сохранения записи в контексте формы. Если возвращаемое значение установить в *false*, то запись сохранена не будет. Используйте метод *validateWrite* на таблице для выполнения проверок возможности сохранения записи в контексте таблицы. Это будет общим правилом для всех форм, использующих эту таблицу |
| *write* | Вызывается при сохранении записи после успешной проверки возможности ее сохранения. Обычно перекрывается для выполнения дополнительных действий в контексте формы, таких как обновление интерфейса пользователя. Используйте метод *write* на таблице для выполнения действий при событиях сохранения записи во всех формах, использующих эту таблицу. |
| *writing* | Вызывается перед сохранением записи в БД. Этот метод вызывается, только если свойству *ChangeGroupMode* на источнике данных задано значение *ImplicitInnerOuter*. Используйте метод *writing* на таблице для выполнения действий при событиях сохранения записи во всех формах, использующих эту таблицу |

**Табл. 6–6.** Список перекрываемых методов источника данных формы (*окончание*)

| Метод | Описание |
|---|---|
| *written* | Вызывается после того, как запись сохранена в БД. Этот метод вызывается, только если свойству *ChangeGroupMode* на источнике данных задано значение *ImplicitInnerOuter*. Используйте метод *written* на таблице для выполнения действий при событиях сохранения записи во всех формах, использующих эту таблицу |

В табл. 6-7 описаны наиболее важные методы полей источников данных, которые разработчик может перекрыть. Чаще всех подвергается перекрытию метод *modified*.

**Табл. 6–7.** Список перекрываемых методов полей источника данных формы

| Метод | Описание |
|---|---|
| *modified* | Вызывается при изменении значения в поле источника данных. Обычно перекрывается для выполнения соответствующих изменений пользовательского интерфейса и изменения значений других полей источника |
| *lookup* | Вызывается при открытии выпадающего списка на связанном элементе управления. Обычно перекрывается для построения нестандартной формы выпадающего списка. Старайтесь умеренно использовать этот метод – только для случаев, специфичных для конкретной формы выпадающих списков. Используйте свойство *EDT.FormHelp* расширенного типа данных для реализации выпадающего списка, одинакового для всех форм, отображающих поля этого типа данных |
| *validate* | Вызывается перед изменением значения в поле источника данных. Обычно перекрывается для выполнения соответствующих проверок сохраняемого значения в контексте формы, изменений пользовательского интерфейса и изменения значений других полей источника. Если возвращаемое значение установить в *false*, то значение сохранено не будет. Используйте метод *validate* на таблице для выполнения проверок сохраняемого в поле значения в контексте таблицы и тем самым во всех формах, использующих эту таблицу |

## Встроенные (*Auto*) переменные формы

Во время выполнения кода формы для разработчика доступен набор встроенных переменных для доступа к важным внутренним элементам

формы. Эти переменные доступны только для чтения. Их описание представлено в табл. 6-8.

**Табл. 6–8.** Встроенные переменные формы

| Переменная | Описание |
|---|---|
| *element* | Переменная, дающая доступ к объекту *FormRun* в текущем контексте выполнения. Обычно используется для вызова методов формы или изменения ее внешнего вида. Пример:<br>`if (element.args().record().TableId == tablenum(SalesTable))`<br>`{`<br>`name = element.design().addControl(FormControlType::String, "Dynalink from SalesTable");`<br>`}` |
| *DataSourceName* (например, *SalesTable*) | Переменная, позволяющая быстро обратиться к текущей активной записи и таблице в каждом источнике данных. Обычно используется для вызова методов таблицы либо свойств *get u set* текущей записи.<br>Пример:<br>`if (SalesTable.type().canHaveCreditCard())` |
| *DataSourceName_DS* (например, *SalesTable_DS*) | Переменная, позволяющая быстро обратиться к любому источнику данных. Обычно используется для вызова методов источника данных либо его свойств *get u set*.<br>Пример:<br>`SalesTable_DS.research();` |
| *DataSourceName_Q* (например, *SalesTable_Q*) | Переменная, позволяющая быстро обратиться к объекту *Query* любого источника данных. Обычно используется для работы с запросом источника и добавления на него фильтров до начала его выполнения. Результат работы аналогичен вызову *SalesTable_DS.query()*.<br>Пример:<br>`rangeSalesLineProjId = salesLine1_q.dataSourceTable(tablenum(SalesLine)).addRange(fieldnum(SalesLine, ProjId)); rangeSalesLineProjId.value(ProjTable.ProjId);` |

**Табл. 6–8.** Встроенные переменные формы (*окончание*)

| Переменная | Описание |
|---|---|
| *DataSourceName_QR* (например, *SalesTable_QR*) | Переменная, позволяющая быстро обратиться к объекту *QueryRun* любого источника данных. Он содержит копию последнего выполненного запроса. Запрос внутри объекта*QueryRun* создается во время вызова метода *FormDataSource.ExecuteQuery*. Обычно используется для доступа к последнему использованному запросу для анализа значений установленных фильтров. Результат работы аналогичен вызову *SalesTable_DS.queryRun()*.<br>Пример:<br>`SalesTableQueryBuildDataSource = SalesTable_QR.query().dataSourceTable(tablenum(SalesTable));` |
| *ControlName* (например, *SalesTable_SalesId*) | Переменная, создающаяся для любого элемента управления с установленным в *Yes* значением свойства *AutoDeclaration*. Обычно используется для доступа к элементам управления, не привязанным к полям источника данных, таким как поля, задействованные для реализации пользовательских фильтров.<br>Пример:<br>`backorderDate.dateValue(systemdateget());` |

## Бизнес-логика

После того как проектирование структуры формы завершено, необходимо добавить в нее вызовы бизнес-логики, используя элемент управления Пункт меню (*MenuItem)* либо явные вызовы из кода перекрытых методов. Пытайтесь сократить количество кода на форме, так как он не может быть повторно использован в других формах, отчетах, службах или классах форм. Если вам нужно реализовать новую функциональность бизнес-логики, помещайте ее в отдельный класс или классы. Тогда вы сможете использовать ее из многих форм системы.

Для реализации бизнес-логики в классе выполните следующие действия.

1. Создайте в классе статический метод *main*.
2. В метод main добавьте код, запускающий собственно нужную бизнес-логику.
3. Создайте пункт меню Действия (Action), ссылающийся на класс.

4. Создайте элемент управления Кнопка пункта меню на форме, ссылающийся на этот пункт меню Действия (Action).

   Например, в методе *main* может быть следующий код:

```
static void main(Args args)
{
    print "Hello World";
    pause;
}
```

Как только управление будет передано классу, метод *args* может сможет предоставить всю необходимую контекстную информацию, которая будет полезна, если он вызывается из многих форм.

## Нестандартные ниспадающие списки (Lookups)

Выпадающие списки для ссылок на таблицы создаются клиентской средой автоматически и в подавляющем большинстве случаев их вполне достаточно. Такие автоматические списки создаются на основе метаданных целевой таблицы. Чтобы определить поля, которые войдут в ниспадающий список, среда выполнения вначале анализирует группу полей *AutoLookup* на таблице. Если эта группа пуста или не существует, далее проверяется группа полей *AutoIdentification*. Если и эта группа пуста, то для списка берутся поля *TitleField1* и *TitleField2* из свойств таблицы.

Автоматически сгенерированные ниспадающие списки практически идеально удобны для использования. Если же вы собрались реализовать собственную форму ниспадающего списка для какой-либо таблицы, то должны использовать тот же способ построения, чтобы его поведение было аналогичным.

Создать простой ниспадающий список довольно легко, особенно если вы хотите, чтобы он использовался для всех ниспадающих списков этого табличного типа. Создайте простую форму с элементом управления *Grid* для отображения записей, а затем укажите наименование формы в качестве значения свойства *FormHelp* расширенного типа данных внешнего ключа таблицы. Это сработает как для обычных внешних ключей, так и для суррогатных. С установленным значением свойства *FormHelp*, вместо автоматической генерации ниспадающего списка, будет использована ваша созданная форма.

В некоторых сценариях, таких как изменение запроса, использующегося формой ниспадающего списка, или потребности в отборе показыва-

емых записей, вам понадобится выполнить некоторый код перед загрузкой формы ниспадающего списка или после нее. В этом случае вы можете использовать класс *FormAutoLookupFactory*, который реализован в ядре системы и выполняет большую часть функций ниспадающего списка, например начальное позиционирование на нужной записи и фильтрацию для облегчения реализации стандартного поведения списка. В качестве примера в приложении можно привести класс и форму *HCMWorkerLookup*. Разбирая код класса, обратите внимание, что эта форма может быть вызвана из многих мест системы в различных вариантах использования. В каждом случае будут вызваны различные методы класса *FormAutoLookupFactory*. На форме также есть код, обрабатывающий эти случаи, к примеру, код по заданию режима *SelectMode в зависимости от* типа вызвавшего ниспадающий список элемента управления.

# Интеграция с клиентом Microsoft Office

С помощью дополнений к Office Microsoft Dynamics AX 2012 пользователи могут извлекать данные из Microsoft Dynamics AX в Microsoft Excel для дальнейшей обработки или формирования какой-либо отчетности, переносить данные из Excel, вводя их в Microsoft Dynamics AX, и генерировать документы Microsoft Word для последующего обмена данными с третьими сторонами.

В этом разделе описываются настройка источников данных для доступа из дополнений к Office, создание шаблонов документов Excel и Word (общий обзор) и их размещение для общего использования.

## Настройка источников данных для доступа из дополнений к Office

Перед тем как дополнения к Office смогут работать с данными из Microsoft Dynamics AX, необходимо создать соответствующие службы и запросы, выступающие в роли источников данных.

### Предоставление доступа к службе

Для того чтобы предоставить доступ к службе, выполните следующие действия.

1. В AOT, в развернутом узле *Services* щелкните правой кнопкой мыши на службе, которую вы хотите сделать доступной, и затем выберите Add-ins > Register Service.

2. В клиенте зайдите в меню Администрирование > Настройка > Службы и Application Integration Framework > Входящие порты и сделайте следующее:
   - создайте новый входящий порт;
   - выберите службы, которые будут работать на этом порту;
   - активируйте порт.
3. В клиенте зайдите в меню Управление организацией > Настройка > Управление документами > Источники данных документа.
4. В форме Источники данных документа сделайте следующее:
   - создайте новый источник данных документа;
   - выберите модуль, который будет связан с источником данных;
   - установите значение поля Тип в Служба;
   - выберите входящий порт, который вы только что создали перед добавлением источника данных;
   - активируйте новый источник данных документа.

#### Настройка доступного запроса

Чтобы сделать запрос доступным для использования с дополнениями для Office, выполните следующие действия.

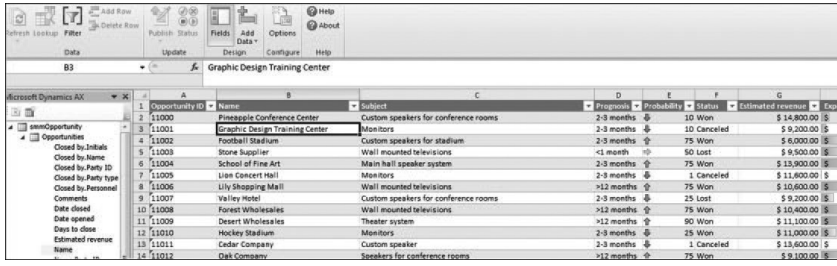
1. Если нужно, создайте новый запрос в AOT.
2. В клиенте зайдите в меню Управление организацией > Настройка > Управление документами > Источники данных документа.
3. В форме Источники данных документа сделайте следующее:
   - создайте новый источник данных документа;
   - выберите модуль, который будет связан с источником данных;
   - установите значение поля Тип в Запрос;
   - выберите запрос, который вы хотите использовать как источник данных;
   - активируйте новый источник данных документа.

## Создание шаблона Excel

После того как вы настроили нужные запросы и службы как источники данных для дополнений к Office, вы можете создавать шаблоны Excel, по-

средством которых работать с данными. С их помощью пользователи могут просматривать и анализировать данные Microsoft Dynamics AX в Excel, используя все возможности Excel, включая условное форматирование, сводные таблицы и вычисляемые поля. Если книга Excel использует службу в качестве источника данных, то пользователи смогут, изменив данные в книге, осуществить перенос измененных данных обратно в Microsoft Dynamics AX.

Шаблон может быть простым, таким как список последних заказов на продажу, или же настолько сложным, как панель мониторинга менеджера-управленца. На рис. 6-12 показан пример шаблона Excel.




| | Opportunity ID | Name | Subject | Prognosis | Probability | Status | Estimated revenue |
|---|---|---|---|---|---|---|---|
| 2 | 11000 | Pineapple Conference Center | Custom speakers for conference rooms | 2-3 months | | 10 Won | $ 14,800.00 |
| 3 | 11001 | Graphic Design Training Center | Monitors | 2-3 months | | 10 Canceled | $ 9,200.00 |
| 4 | 11002 | Football Stadium | Custom speakers for stadium | 2-3 months | | 75 Won | $ 6,000.00 |
| 5 | 11003 | Stone Supplier | Wall mounted televisions | <1 month | | 50 Lost | $ 9,500.00 |
| 6 | 11004 | School of Fine Art | Main hall speaker system | 2-3 months | | 75 Won | $ 13,900.00 |
| 7 | 11005 | Lion Concert Hall | Monitors | 2-3 months | | 1 Canceled | $ 11,800.00 |
| 8 | 11006 | Lily Shopping Mall | Wall mounted televisions | >12 months | | 75 Won | $ 10,600.00 |
| 9 | 11007 | Valley Hotel | Custom speakers for conference rooms | 2-3 months | | 25 Lost | $ 9,200.00 |
| 10 | 11008 | Forest Wholesales | Wall mounted televisions | >12 months | | 75 Won | $ 10,400.00 |
| 11 | 11009 | Desert Wholesales | Theater system | >12 months | | 90 Won | $ 11,100.00 |
| 12 | 11010 | Hockey Stadium | Monitors | 2-3 months | | 25 Won | $ 11,000.00 |
| 13 | 11011 | Cedar Company | Custom speaker | 2-3 months | | 1 Canceled | $ 13,600.00 |
| 14 | 11012 | Oak Company | Speakers for conference rooms | >12 months | | 75 Won | $ 9,100.00 |

**Рис. 6–12.** Шаблон Excel с данными из Microsoft Dynamics AX

Чтобы получить доступ к данным Microsoft Dynamics AX из книги Excel, выполните следующие действия.

1. Откройте диалоговое окно Options с вкладки Microsoft Dynamics AX на ленте меню, чтобы убедиться, что для соединения выбран правильный сервер и порт.
2. Щелкните Add Data и выберите нужный источник данных – запрос или службу.
3. Щелкните дважды на нужных вам полях или перетащите их из общего списка полей.
4. Обновите лист книги и убедитесь, что данные из Microsoft Dynamics AX попали в книгу. Если выборка оказалась очень велика, воспользуйтесь кнопкой Filter на ленте, чтобы ограничить ее.

Перед тем как отдать книгу другим пользователям, выполните следующее.

1. При необходимости сделайте добавочную фильтрацию для ограничения получаемой выборки.

2. Откройте диалоговое окно Connection Options и уберите из настроек соединения имеющуюся там информацию, чтобы нужные для конкретного пользователя параметры были автоматически подставлены туда клиентскими средствами разработки (используя информацию из настроек Microsoft Dynamics AX Client Configuration Utility).
3. Сохраните книгу с очищенными настройками соединения.

## Создание шаблона Word

Вы можете создавать шаблоны Word, позволяющие пользователям генерировать документы Word, содержащие данные из Microsoft Dynamics AX. На рис. 6-13 показан пример шаблона Word.

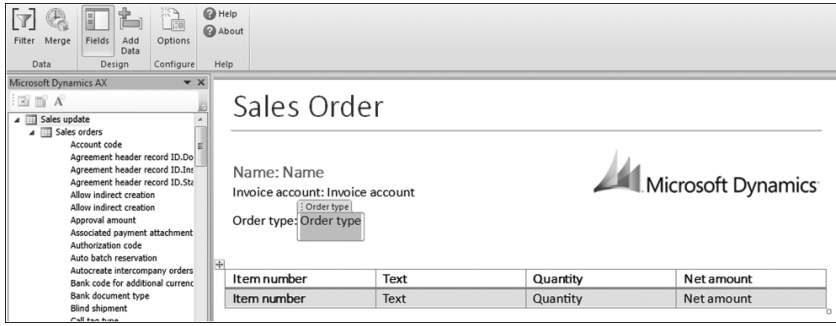



**Рис. 6–13.** Шаблон Word с данными из Dynamics AX

Чтобы получить доступ к данным Microsoft Dynamics AX из документа Word, выполните следующие действия.

1. Откройте диалоговое окно Options с вкладки Microsoft Dynamics AX на ленте меню, чтобы убедиться, что для соединения выбран правильный сервер и порт.
2. Щелкните Add Data и выберите нужный источник данных – запрос или службу.
3. Щелкните дважды на нужных вам полях или перетащите их из общего списка полей. Если хотите выбрать какое-либо вычисляемое поле (display-метод) источника данных, щелкните правой кнопкой мыши в общем списке полей и выберите Show Calculated Fields.

4. Ссылки на какое-либо поле могут быть вставлены в произвольное место документа. Они могут перемежаться любым текстом, форматированием, рисунками и другим содержанием.
5. Повторяющиеся значения, такие как строки заказов на продажу, могут быть отображены путем вставки таблицы и размещения в ее первой строке ссылок на нужные поля.
6. Можно добавить фильтр для вывода определенной записи. Однако такая фильтрация будет недоступна при использовании шаблона из функции Создать из шаблона.
7. Сохраните шаблон.
8. Щелкните кнопку Merge для создания документа на основе шаблона.

Перед тем как отдать шаблон документа другим пользователям, выполните следующие действия.

1. Добавьте необходимые фильтры для ограничения получаемой выборки.
2. Откройте диалоговое окно Connection Options и уберите из настроек соединения имеющуюся там информацию, чтобы нужные для конкретного пользователя параметры были автоматически подставлены туда клиентскими средствами разработки (используя информацию из настроек Microsoft Dynamics AX Client Configuration Utility).
3. Сохраните книгу с очищенными настройками соединения.
4. Передайте пользователям копию документа.

## Предоставление шаблонов пользователям

Некоторые формы в Microsoft Dynamics AX в группе Вложения имеют кнопку Создать из шаблона на панели операций. Как пример можно привести форму Клиенты на рис. 6-14.

Чтобы добавить в список доступных групп шаблоны для функции Создать из шаблона, выполните следующие действия.

1. Создайте шаблон документа Word или книги Excel с фильтром, не ограничивающим показываемые записи какой-либо записью.
2. В клиенте выберите Управление организацией > Настройка > Управление документами > Типы документов.

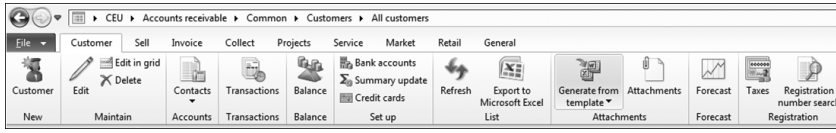



**Рис. 6–14.** Кнопка Создать из шаблона на панели операций

3. Создайте новый тип документов, указав в поле Класс значение Библиотека шаблонов.
4. В поле Библиотека документов укажите папку SharePoint, в которой находятся шаблоны. Убедитесь, что URL указывает именно на папку, а не на страницу; к примеру, *http://myserver/DocumentTemplates/*.
5. Нажмите Синхронизировать, чтобы импортировать список шаблонов и сделать их активными.
6. Проверьте, что шаблоны появились на форме в списке Создать из шаблона. Если там их нет, проверьте, что главный источник данных для шаблонов совпадает с главным источником данных формы.

Если же на форме вообще недоступна кнопка Создать из шаблона, то пользователи все равно **смогут сгенерировать документ по шаблону, используя** форму Работа с документами (Файл > Команда > Работа с документами) и создав новое вложение нужного типа Библиотеки шаблонов.

Чтобы добавить кнопку Создать из шаблона на формы или страницы списков, выполните следующие действия.

1. Найдите кнопку Создать из шаблона на странице списка Клиенты и скопируйте ее на форму, в которую вы хотите добавить эту кнопку. Путь к кнопке выглядит так:

   *AOT\Forms\CustTableListPage.Designs\Design\ActionPane:ActionPane\ActionPaneTab:HomeTab\ButtonGroup:AttachmentsGroup\MenuButton:mbTemplatesButton*.
2. Отредактируйте метод *MouseDown* на кнопке **так, чтобы передать правильный** *TableId* методу *createTemplateOnMenuButton*, **изменив на правильную** таблицу параметр *CustTable.TableId*. Например так: *MyTable.TableId*.