Глава 21

Модели приложения

В этой главе

- Введение
- Слои
- Модели
- Идентификаторы элементов
- Создание модели
- Подготовка модели к публикации
- Обновление модели
- Перенос модели из тестового окружения в рабочее
- Программный интерфейс хранилища моделей

Введение

Microsoft Dynamics AX 2012 начинает новую эру в управлении артефактами метаданных.

В предыдущих версиях продукта артефакты метаданных хранились в файлах данных объектов приложения (Application Object Data, .AOD). Они служили двум целям. Во-первых, они работали как средство развертывания метаданных, например, вы могли скопировать AOD-файл из одной системы в другую. Во-вторых, они были местом хранения элементов приложения во время выполнения. Файл AOD предоставлял место физического хранения для базы данных с индексно-последовательным доступом доступом (ISAM), хранившей метаданные, и из этого хранилища среда выполнения считывала элементы модели.

Данный метод управления артефактами метаданных был не оптимален. С точки зрения развертывания, файлы AOD не допускали параллельной установки метаданных на одном и том же слое, они не содержали никакой структурированной информации об их содержимом и не могли быть подписаны цифровой подписью. С точки зрения времени выполне-

ния, формат AOD поддерживал только одну таблицу и не предоставлял возможности добавлять или изменять ее столбцы. Таким образом, для поддержки возможностей развития среды выполнения, продукту необходимо было перейти к использованию корректной реляционной схемы.

В Microsoft Dynamics AX 2012 метаданные хранятся в базе данных Microsoft SQL Server вместе с бизнес-данными. Таблицы, содержащие метаданные, называются *хранилищем моделей*. Это изменение устраняет все препятствия к использованию реляционной схемы. Элементы, такие как классы, таблицы, формы, методы и элементы управления, группируются в хранилище моделей по моделям. Каждая модель может быть экспортирована в файловый формат с расширением *.axmodel.* Эти файлы являются управляемыми сборками и, следовательно, поддерживают цифровую подпись, что делает их защищенными от внешних манипуляций. Файлы моделей – это основной механизм развертывания для элементов модели в Microsoft Dynamics AX 2012.

В предыдущих версиях Microsoft Dynamics AX независимые разработчики ПО иногда распространяли исходные коды в виде текстовых файлов. Распространение в исходных кодах является нежелательной практикой, но она использовалась для преодоления ограничений на идентификаторы элементов при совмещении нескольких решений на одном и том же слое. В Microsoft Dynamics AX 2012 вам больше не нужно распространять исходные коды клиентам. Файлы модели не содержат идентификаторов элементов. Вы можете устанавливать несколько элементов на один и тот же слой, и каждый файл модели содержит манифест, описывающий эту модель.

Примечание. Поддержка файлов .XPO все так же полностью присутствует в Microsoft Dynamics AX 2012. Разработчики используют их для обмена исходным кодом и для хранения в системе контроля версий.

В предыдущих версиях Microsoft Dynamics AX весь набор файлов AOD мог быть развернут одной операцией, обычно когда решения переносились из тестовой системы на рабочую. Главная задача в этом сценарии – сокращение времени простоя. Копирование всех AOD-файлов одной операцией сокращало это время из-за отсутствия необходимости пересоздания файлов индекса объектов (.AOI), а также ненужности перекомпиляции кода приложения. Для удовлетворения тех же требований в Microsoft Dynamics AX 2012 вы можете экспортировать все хранилище мо-

дели в двоичный файл с расширением *.axmodelstore*. Этот файл может быть импортирован в целевую систему, и результирующее время простоя будет равно времени, требуемому на перезапуск сервера приложения (AOS).

Слои

Среда выполнения Microsoft Dynamics AX 2012 выполняет программу, заданную в среде разработки MorphX. Сама программа определяется в виде элементов.

В отличие от большинства систем, Microsoft Dynamics AX может содержать несколько определений для одного и того же элемента, например несколько реализаций одного и того же метода. Эти определения элемента складываются в слои. Среда выполнения Microsoft Dynamics AX использует определение элемента из наивысшего слоя, который его содержит. Например, метод, определенный на слое SYS (наинизший слой), не будет использоваться, если другое определение того же метода есть на любом другом слое.

Такой послойный подход к организации разработки имеет несколько преимуществ, наиболее важным из которых является возможность кастомизации программы, поставляемой Microsoft, ее партнерами и независимыми разработчиками ПО, без необходимости изменения оригинального исходного кода.

Примечание. Метафора слоя также используется в программах редактирования графики. В них, используя слои, вы можете рисовать поверх существующего изображения, не затрагивая исходное. В Microsoft Dynamics AX слои работают точно таким же образом, но не для пикселов, фигур и заливок, а для кода и свойств

Когда вы запускаете Microsoft Dynamics AX 2012, то указываете, в каком слое хотите ее запустить. По умолчанию вы начинаете в слое *USR*. Любой элемент, который вы создадите или измените, будет сохранен в этом слое. Если вы измените элемент, уже существующий на слое ниже, то его копия с вашими правками будет сохранена в текущий слой. Этот процесс известен как *перекрытие* (over-layering).

Другие преимущества слоев включают возможность возврата к исходному определению элемента путем простого удаления его перекрытой версии. Вы также можете сравнивать версии какого-либо элемента, например,

для того чтобы увидеть, какие строки кода вы добавили. Это особенно полезно во время обновления.

Внимание. Разрабатывайте ваши решения послойно, переходя от слоя к слою, снизу вверх. Работа в нескольких слоях одновременно на одном и том же AOS приводит к путанице и неразберихе, даже если это делается разными пользователями. Подробнее рассказано в техническом описании к Microsoft Dynamics AX 2012 «Developing Solutions in a Shared AOS Development Environment» по адресу: *http://www.microsoft.com/download/en/details.aspx?id=26919*.

Процесс редактирования элемента со слоя выше, чем текущий слой, называется *подкладывание* (under-layering). Подобные правки будут сохранены в том же слое выше текущего.

Microsoft Dynamics AX 2012 имеет 16 слоев метаданных, каждый со своим назначением. В табл. 21-1 описаны все эти слои в алфавитном порядке.

Табл. 21–1. Слои метаданных

Название	Описание
USP (верхний)	Слой исправлений для слоя *USR*
USR	Пользовательский слой. Этот слой предназначен для мелких окончательных изменений опытными пользователями системы. Сюда могут включаться простые изменения размещения элементов на форме, новые роли безопасности и обработки для конкретной компании
CUP	Слой исправлений для слоя *CUS*
CUS	Слой клиентов. Он позволяет реализовать специфичные для конкретного клиента доработки расширения решения. Обычно содержит доработки, выполненные партнером Microsoft или внутренней командой разработчиков
VAP	Слой исправлений для слоя *VAR*
VAR	Слой дополнительных возможностей. Партнеры Microsoft будут использовать его для поставки доработок и расширений, устанавливаемых нескольким клиентам
ISP	Слой исправлений для слоя *ISV*

Табл. 21–1. Слои метаданных (*окончание*)

Название	Описание
ISV	Слой независимых поставщиков ПО. Зарегистрированные в Microsoft независимые поставщики ПО могут поставлять решения в этом слое
SLP	Слой исправлений для слоя *SLN*
SLN	Слой решений. Он содержит вертикальные решения, предлагаемые партнерами Microsoft
FPP	Слой исправлений для слоя *FPK*
FPK	Слой пакетов возможностей (feature pack). Он содержит отраслевые решения, предлагаемые Microsoft. В нем доступны решения Public sector, Process industries, Retail и Service industries
GLP	Слой исправлений для слоя *GLS*
GLS	Слой глобальных решений. Он содержит региональные расширения к горизонтальному решению из слоя *SYS*
SYP	Слой исправлений для слоя *SYS*
SYS (Нижний)	Системный слой. Платформа Microsoft и слой базового основания. Он содержит горизонтальное приложение, разработанное Microsoft

Внутри слоя элементы метаданных группируются по моделям, которые описаны в разделе ниже.

Модели

Модель – это логическое хранилище метаданных, таких как формы, таблицы, отчеты и методы. Больше информации о типах элементов можно найти в главе 1.

Хранилище моделей может содержать так много моделей, сколько вам может понадобиться. На рис. 21-1 показаны отношения между слоями, моделями и элементами.

Примечание. Термин *модель* был выбран по нескольким причинам. Во-первых, любое решение, построенное в Microsoft Dynamics AX – это модель реального предприятия. Во-вторых, понятие модели не сокращаемо. Даже часть модели является моделью. Соответственно, этот термин покрывает как отдельные решения,

так и различные расширения, доработки, исправления и т. д. И, наконец, этот термин прост и легок для понимания. Он должен быстро стать частью вашего словаря Microsoft Dynamics AX .

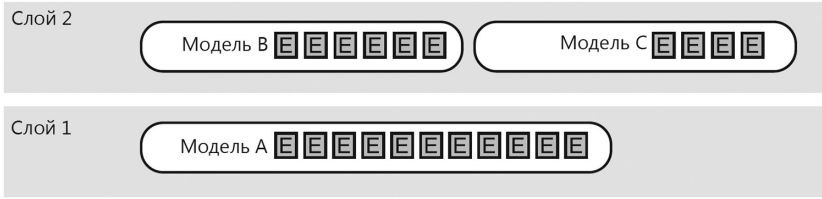

Рис. 21–1. Слои, модели и элементы

В каждом слое может быть так много моделей, как вы хотите. Это означает, что можно сегментировать слой на модели по вашему предпочтению. Ниже приведены несколько сценариев разработки, в которых это может быть использовано.

- **Когда вы поставляете более чем одно решение.** Вы можете иметь отдельную модель для каждого решения, над которым работаете. Таким образом, можно работать над ними одновременно.
- **Когда ваше решение становится слишком большим.** Вы можете сегментировать ваше решение на несколько моделей и работать над каждой частью различными командами или членами команд. Модель может либо быть самодостаточной, либо зависеть от других моделей. Это дает возможность четко разделить сферы ответственности между моделями, строго задать интерфейсы программного взаимодействия (API) между моделями, проводить раздельное построение моделей и т. д.
- **Когда вы пишете юнит-тесты.** У вас может быть модель для проверенного рабочего кода и модель для юнит-тестов. Это даст возможность импортировать все ваши юнит-тесты, выполнить их и легко убрать из системы.

Вы можете получить модель двумя путями: либо создав свою, либо взяв ее от кого-либо еще. Поскольку на любом слое вы можете иметь много моделей, то можно развернуть на одном слое модели из нескольких источников.

Предположим, что вы являетесь клиентом и хотите установить два решения независимых поставщиков, доступных на слое *ISV*. В предыдущих версиях вы становились перед тяжелым выбором. Вам пришлось бы взять одно из решений и учиться обходиться без другого или потратить деньги на слияние двух решений на одном слое. Это слияние было технически сложным и затратным, особенно если продолжали выходить обновления этих решений. В Microsoft Dynamics AX 2012 вы просто берете обе модели и используете для их импорта AXUtil – утилиту командной строки, устанавливающуюся вместе с Microsoft Dynamics AX. При выходе новой версии любой из моделей вы просто снова используете AXUtil для обновления модели.

Иерархия послойного хранения элементов моделей имеет одно ограничение: элемент может быть определен в слое только однажды. Другими словами, вы не можете установить две модели, содержащие определение одного и того же элемента на том же самом уровне в одном приложении. Приведем пару примеров.

- Две модели, содержащие класс *MyClass,* не могут быть одновременно установлены в один и тот же слой.
- Два элемента одного и того же типа, подчиненные одному и тому же родительскому элементу (или без него), не могут существовать на одном слое, если они имеют одинаковое имя или одинаковый идентификатор. Например, таблица не может иметь два поля с одинаковым именем или два поля с одинаковым идентификатором. Вы также не можете иметь два пункта меню с одинаковым именем на одном слое.

Это ограничение позволяет среде выполнения Microsoft Dynamics AX выбрать правильную версию элемента в процессе работы, основываясь на слое, в котором он хранится.

Вы можете столкнуться с этим ограничением в двух случаях.

- Вы создали элемент и случайно дали ему имя, использующееся другим элементом в другой модели. Хорошим способом избежать этого является именование новых элементов с использованием короткого префикса, уникального для вас, вашей компании или вашего решения. Это широко используемая практика.
- Вы изменили существующий элемент, который также был изменен кем-либо еще в другой модели. Есть много путей ограничить число

изменяемых элементов, например, используя события, но в некоторых ситуациях этого невозможно избежать.

Примечание. Поскольку идентификаторы элементов назначаются во время развертывания, система автоматически избегает дублирования идентификаторов. Об идентификаторах элементов рассказывается ниже.

Идентификаторы элементов

Все типы элементов имеют названия, некоторые из них также имеют целочисленный идентификатор. Он представляет собой 32-разрядное целое и присваивается во время установки. Это означает, что один и тот же элемент может иметь различные идентификаторы на двух разных системах.

Примечание. В предыдущих версиях Microsoft Dynamics AX идентификаторы были 16-разрядными целыми, которые присваивались элементам во время их создания из диапазона идентификаторов, выделенного для каждого слоя. Это могло привести к исчерпанию идентификаторов и конфликтам между ними при установке независимо разработанных решений.

Для поддержки сценариев, в которых элементы обновляются или переименовываются, были введены два новых свойства.

- Свойство *LegacyID* добавлено к типам элементов, имеющих идентификатор. Это свойство позволяет элементам сохранять свой идентификатор (созданный еще во времена AOD-файлов) при импорте их как файлов модели в хранилище моделей.
- Свойство *Origin* представляет собой глобально-уникальный идентификатор (GUID), идентифицирующий элемент и предотвращающий риск конфликта. Это свойство добавлено ко всем корневым типам элементов и к типам элементов с идентификаторами. Оно позволяет утилите AXUtil (и другим компонентам системы) распознать переименованные элементы во время импорта.

AXUtil назначает идентификаторы элементам при импорте, руководствуясь следующими правилами.

1. Если элемент уже существует с тем же *Origin,* то он заменяется и его идентификатор остается тем же. Иначе выполняется шаг 2.
2. Если элемент уже существует с тем же *Type*, *Name* и *ParentID*, то он также заменяется и его идентификатор остается тем же. Иначе выполняется шаг 3.
3. Если импортируемый элемент имеет *LegacyID* и идентификатор, равный *LegacyID* доступен на целевой системе, то добавляется новый элемент с идентификатором, равным *LegacyID*. Иначе выполняется шаг 4.
4. Назначается новый для приложения идентификатор, не совпадающий ни с одним *LegacyID* (со значением, большим чем 60000 для полей, и большим чем 1000000 для всех остальных типов элементов).

Такой алгоритм обеспечивает сохранение идентификаторов как в простых, так и в сложных сценариях импорта. Представьте себе, что вам нужно распространить несколько вариантов одного и того же решения некоторому множеству клиентов в виде AOD- или XPO-файлов в Microsoft Dynamics AX 2009. Это означает, что вам, возможно, придется вести свою копию исходного кода для поддержки каждого из клиентов. По мере роста количества клиентов вы все более будете стремиться к объединению всех вариантов в одно общее решение. Шаг 2 этого алгоритма обеспечит сохранение идентификаторов при обновлении клиента со специализированного решения на общее.

Во время обычной разработки система сохраняет идентификаторы, так что вам не придется заботиться об этом. Однако будет необходимо обратить на них внимание в двух ситуациях: при обновлении модели и при переносе кода с тестового приложения на рабочее. Обе **эти ситуации разбираются** в следующих разделах.

Примечание. Функциональность импорта/экспорта, доступная в Администрировании системы, автоматически корректирует ссылки на идентификаторы элементов в импортируемых бизнес-данных для обеспечения их соответствия идентификаторам целевой системы. Однако эта коррекция пропускает любые неструктурированные данные. Если вам нужно иметь ссылку на элемент, например, в хранимом контейнере, то наилучшим вариантом будет хранить ссылку на элемент в виде его имени.

Создание модели

Перед началом реализации своего решения, вам потребуется создать новую модель. Это можно сделать несколькими путями. Вы можете сделать это в MorphX, зайдя в Сервис > Управление моделью > Создать модель, используя Windows PowerShell из Microsoft Dynamics AX 2012 Management Shell или же путем использования AXUtil. Примеры в этой главе используют последний способ.

```
AXUtil create /model:"My Model" /Layer:USR
```

Обратите внимание, что вам нужно указать, к какому слою будет принадлежать модель. Она не может пересечь границу слоя.

Примечание. Каждый слой имеет системную модель. Если вы не создадите свою модель, то для ваших элементов будет использована системная модель. Эта модель имеет определенные ограничения при развертывании, так как ее манифест доступен только для чтения. Так что настоятельно советуем создавать свои модели.

После создания модели вам необходимо выбрать ее. В строке состояния MorphX вы можете видеть текущую модель, а также изменить ее, щелкнув по ней. Все создаваемые вами элементы будут помещены в текущую модель.

Вы можете легко увидеть, к какой модели принадлежит элемент, открыв его свойства. Можно также включить отображение моделей в AOT на каждом элементе в меню Сервис > Параметры > Разработка. Можно переместить элемент в другие модели того же слоя, щелкнув правой кнопкой на нем и выбрав Переместить в модель.

Теперь у вас есть своя модель, и вы готовы начать разработку своего решения.

Примечание. Вы можете удалить модель, используя команду удаления AXUtil. Это относится к созданным или установленным вами моделям. Указав опцию */layer:<слой>*, вы даже можете удалить со слоя все модели.

Подготовка модели к публикации

Когда разработка завершена, наступает время подготовки решения к публикации. Но перед тем, как начать, возможно, вы заходите создать MorphX-проект, содержащий все элементы вашей модели. Это делается через Сервис > Управление моделью > Создать проект на основе модели. Это позволит вам перед публикацией убедиться, что в модели содержится все необходимое. Можно также использовать AXUtil для получения списка элементов в модели.

```
AXUtil view /model:"My Model" /verbose
```

Подготовка модели к публикации состоит из следующих шагов.

1. Задание манифеста модели.
2. Экспорт модели на диск.
3. Добавление цифровой подписи.

Задание манифеста модели

Модель является вместилищем вашего решения. В манифесте модели можно описать модель. Табл. 21-2 содержит описание свойств манифеста модели. При экспорте модели полученный файл экспорта будет содержать в себе манифест. Он поможет вашим клиентам перед установкой модели понять, что содержится в ней.

Простейший путь редактирования манифеста состоит в редактировании сохраненной в файл копии XML.

```
AXUtil manifest /model:"My Model" /xml >MyManifest.xml
notepad MyManifest.xml
AXUtil edit /model: "My Model" @MyManifest.xml
```

Табл. 21–2. Свойства манифеста модели

Тип	Описание
Name	Полное название модели. Обычно содержит несколько слов и используется также в маркетинговых материалах и других общедоступных документах
Publisher	Издатель модели, например, Microsoft Corp. Свойства *Name* и *Publisher* должны содержать уникальный ключ. Другими словами, вы не сможете установить одновременно две модели с одинаковыми *Name* и *Publisher* в одно хранилище моделей

Табл. 21–2. Свойства манифеста модели (*окончание*)

Тип	Описание
Description	Более подробное текстовое описание модели
Display Name	Удобное имя, отображаемое в интерфейсе разработчика, в статусной строке MorphX и в AOT. *Display Name* обычно значительно короче *Name* и часто имеет лишь мнемоническое значение
Version	Номер версии из четырех частей, например, 1.0.0.0
Category	Категория модели. Модели разделяются на четыре категории. ■ *Standard.* Обычная модель. ■ *Hotfix.* Модель, содержащая исправление для проблемы в другой модели. Заплатки обычно распространяются на слое исправлений. ■ *Virtual.* Модель, автоматически созданная в результате разрешения конфликта элементов во время импорта модели, при использовании опции */conflict:push*. ■ *Temporary.* Модель, использующаяся во время процесса импорта. В конце импорта она будет удалена. Это свойство обычно устанавливается в значение *Standard*
Details	Точка для добавления дополнительной информации в манифест. Если вам нужно указать в манифесте еще больше информации, то можете поместить ее сюда. Это свойство должно содержать корректно оформленный XML-текст. Хранилище моделей и среда выполнения Microsoft Dynamics AX не используют это свойство. В моделях *Standard* от Microsoft оно оставлено пустым; модели *Hotfix* включают в нем информацию о номере статьи базы знаний (KB)

Примечание. В манифесте невозможно задать зависимости между моделями. Однако если вы воспользуетесь механизмом совместной (slipstream) установки Microsoft Dynamics AX Setup, то сможете управлять последовательностью инсталляции.

Экспорт модели

Когда манифест уже заполнен, пришло время экспорта модели на диск для дальнейшего распространения за пределы вашей организации.

```
AXUtil export /model:"My Model" /file:MyModel.axmodel
```

Для файлов моделей используется расширение *.axmodel.* Файл модели содержит все ее элементы плюс манифест модели. В файлах модели отсутствуют идентификаторы элементов. При импорте элементов модели в целевую систему им будут присвоены новые, специфичные для нее идентификаторы. Файлы XPO работают с идентификаторами аналогичным образом.

Совет. Внутри файлы моделей представляют из себя управляемую сборку. Это означает, что вы можете использовать инструменты отражения, такие как *ildasm,* для изучения ее содержимого.

Можно запустить проверку содержимого файла модели, используя AXUtil:

```
AXUtil view /file:MyModel.axmodel /verbose
```

Примечание. AXUtil это мощный инструмент, и она настолько дружественна к пользователю, насколько это возможно для утилиты командной строки. Заметьте, что некоторые ее команды, такие как *view* и *manifest*, могут быть применены как для хранилища моделей, так и для файла модели на диске. Наиболее часто используемым параметром является */model.* В примерах данной главы при использовании этого параметра везде указывается имя модели, но вы также можете использовать ее идентификатор (который обычно гораздо короче и удобнее записывается) или указывать XML-файл манифеста модели. Последняя возможность особенно полезна при написании командных скриптов, вроде управляющих сборкой для системы контроля версий. Все команды также поддерживают параметр */verbose*, который показывает дополнительные сведения о процессе выполнения команды. Чтобы получить полный список команд, введите ***AXUtil /?***.

Добавление цифровой подписи

Файл модели готов к распространению. Однако вам следует подумать еще об одной вещи перед тем, как сделать его общедоступным. Файл модели содержит двоичный код, и потенциально он может навредить системе, особенно если будет изменен кем-либо уже после того, как покинет вас. Чтобы обеспечить получение вашими клиентами файла модели, которому

можно доверять, или хотя бы сказать, что он был получен из заслуживающего доверия источника, вы можете добавить к файлу модели цифровую подпись.

При импорте подписанной модели вы гарантировано защищены от умышленных или неумышленных изменений в файле модели с тех пор, как он был экспортирован. Если же файл был изменен, процесс импорта будет отменен. Microsoft Dynamics AX 2012 поддерживает два способа подписывания модели: с использованием строгого имени и подписи Authenticode.

Подпись с использованием строгого имени

Чтобы подписать модель с использованием строгого имени, вам нужно использовать инструмент .NET Framework Strong Name Tool, SN.exe, для генерации файла с парой ключей. При экспорте модели в файл *.axmodel* вы указываете ключ, которым должна быть подписана сборка.

```
SN -k mykey.snk
AXUtil export /model:"My Model" /file:MyModel.axmodel /key:mykey.snk
```

Подпись Authenticode

Если вы публикуете модели, например, в качестве независимого поставщика ПО, предлагающего их для скачивания, рассмотрите возможность подписывания моделей с помощью Authenticode. В этом случае вашим клиентам гарантируется отсутствие нежелательных изменений в файле.

При импорте модели, подписанной Authenticode, происходит аутентификация ее издателя. Это означает, что происхождение модели может быть прослежено прямо к вам.

Чтобы подписать файл Authenticode, сначала экспортируйте его с помощью AXUtil. Затем используйте *SignTool* для проведения, собственно, подписи.

```
signtool sign /f mycertprivate.pfx /p password MyModel.axmodel
```

Импорт файлов моделей

Если вы получили или загрузили из Интернета файл модели, то можете импортировать его с помощью AXUtil. Файл модели всегда импортируется в слой, из которого он был экспортирован. Хорошей рекомендацией будет остановить сервер приложения перед импортом.

```
AXUtil import /file:SomeModel.axmodel
```

> **Примечание.** Вам не нужно указывать расширение файлов при использовании AXUtil. **Утилита сама добавит нужное расширение**, если оно было опущено. В этой книге расширения указываются только для ясности.

На рис. 21-2 показана модель, которая была успешно импортирована в слой, на котором уже существовала другая модель.

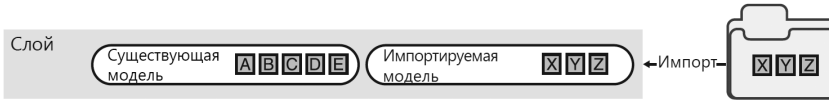

Рис. 21–2. Совместное размещение двух моделей

Операция импорта может быть прекращена, если один или несколько элементов из файла модели уже определены на слое, в который импортируется модель. Если вы перезапустите импорт с параметром */rerun*, то получите список конфликтующих элементов.

```
AXUtil import /file:SomeModel.axmodel /verbose
```

У вас есть две возможности, что делать дальше: *overwrite* (перезаписать) или *push* (вытолкнуть).

Импорт файлов модели с параметром *overwrite*

Вы можете решить перезаписать существующие конфликтующие элементы их новыми определениями из файла модели. Чтобы сделать это, нужно указать параметр */conflict:overwrite* в команде импорта:

```
AXUtil import /file:SomeModel.axmodel /conflict:overwrite
```

Рис. 21-3 показывает результат успешного импорта с использованием параметра */conflict:overwrite*. Импортировавшаяся модель и существующая модель, содержавшие конфликтующие элементы, после выполнения этой операции теперь связаны. Они являются связанными, поскольку ранее существовавшая модель теперь содержит лишь часть необходимых ей элементов. Связь между ними приводит к невозможности удаления импортированной модели, кроме случая ее удаления вместе с ранее существовавшей моделью. Этот параметр, как правило, применяется при установке кумулятивных обновлений или пакетов исправлений.

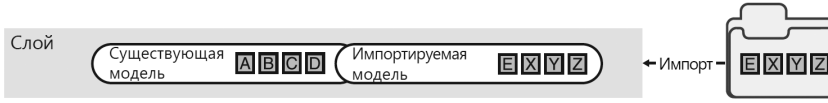

Рис. 21–3. Совместное размещение двух моделей с использованием параметра */conflict:overwrite*

Импорт файлов модели с параметром *push*

Наиболее типичным решением для разрешения конфликтов является использование параметра */conflict:push*. Этот параметр создает новую виртуальную модель на слое выше, содержащую конфликтующие элементы:

```
AXUtil import /file:SomeModel.axmodel /conflict:push
```

Рис. 21-4 показывает результат успешного импорта с использованием параметра */conflict:push.* Элементы в виртуальной модели идентичны импортировавшимся. Другими словами, существующие модели были не затронуты. После импорта модели войдите в слой, содержащий виртуальную модель, чтобы вручную разрешить конфликты. Для этого вы можете использовать в AOT **функциональность сравнения для поиска конфликтующих** версий каждого из элементов.

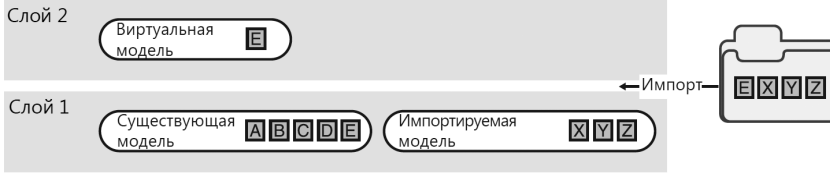

Рис. 21–4. Результат операции импорта с использованием параметра */conflict:push*

Если вы разрешили все конфликты на одном слое, то риск исчерпать доступные слои при использовании параметра */conflict:push* отсутствует. Однако вам может потребоваться вручную перенести элементы с разрешенными конфликтами на один и тот же слой. Например, если вы импортируете третью модель, конфликтующую с элементами в виртуальной модели на рис. 21-4, полученная виртуальная модель будет создана на слое 3. После того как вы разрешите конфликты на слое 3, перенесите элементы на слой 2. Простейший путь для этого – выгрузить элементы из слоя 3 в файл XPO, удалить их и импортировать в слой 2.

По умолчанию виртуальная модель создается на один слой выше того, в который импортируется модель из файла. Если у вас нет доступа на разработку в этом слое, вы можете заставить AXUtil создать виртуальную модель на другом слое (например, слое *USR*), использовав параметр */targetlayer*, как показано в следующем примере.

```
AXUtil import /file:SomeModel.axmodel /conflict:push /targetlayer:USR
```

Обновление модели

Когда вы получаете новую версию модели и желаете обновить ее старую версию в хранилище моделей, важно помнить, что импорт новой модели производится поверх существующей. AXUtil автоматически обнаруживает существование этой модели в хранилище и производит действия, требующиеся для обеспечения согласованности хранилища моделей.

```
AXUtil import /file:NewerModel.axmodel
```

По умолчанию импорт AXUtil входит в режим обновления в случае, когда находит существующую модель с таким же именем и издателем. Иногда происходят переименования моделей или их расщепление на несколько новых моделей (например, как результат рефакторинга). Или же, напротив, несколько моделей могли быть объединены в единую совокупную модель. AXUtil поддерживает обновление существующих моделей новыми. Вы можете заставить AXUtil перейти в этот режим, указав список файлов и обновляемых моделей, разделенный запятыми:

```
AXUtil import /file:f1,f2,f3 /replace:m1,m2,m3,m4,m5
```

Внимание. Вам может показаться хорошей идеей удалить существующую модель перед импортом новой версии. Но в случае, если вы сделаете это, AXUtil не перейдет в режим обновления и будет назначать новые идентификаторы импортируемым элементам. Это может привести к повреждению данных из-за того, что бизнес-данные содержат исходные идентификаторы элементов. Все ссылки на элементы из удаленной модели будут некорректными. Больше информации об этом есть в разделе «Идентификаторы элементов» ранее в этой главе.

Перенос модели из тестового окружения в рабочее

Хорошей практикой является наличие тестовой или подготовительной среды, в которой проводится подготовка и тестирование изменений в системе перед тем, как они будут развернуты в рабочем окружении.

Хранилище моделей предоставляет возможности, которые позволят вам экспортировать все его метаданные в двоичный файл и затем импортировать этот файл в целевую систему. Такое действие приводит к созданию идентичной для двух систем двоичной копии метаданных, включая идентификаторы элементов. Файлы хранилища моделей имеют расширение *.axmodelstore*. Помимо метаданных, они содержат также скомпилированный P-код и промежуточный код .NET (CIL). Это означает, что вам не придется компилировать целевую систему.

Примечание. Размер файлов хранилища моделей зависит от ее содержимого. Файл хранилища моделей для стандартной инсталляции Microsoft Dynamics AX 2012 имеет размер около 2 Гбайт. Эти файлы хорошо сжимаются, обычно более чем на 80%, так что такой подход может быть использован в качестве простого резервного копирования.

Рис. 21-5 показывает наилучший путь создания и подготовки тестового окружения и развертывания далее на рабочую систему. В этом процессе могут быть вариации и действия, выполняемые после этапа установки. Для более подробной информации обратитесь к техническому описанию Microsoft Dynamics AX 2012 «Deploying Customizations Across Microsoft Dynamics AX 2012 Environments» по адресу: *http://www.microsoft.com/download/en/details.aspx?id=26571*.

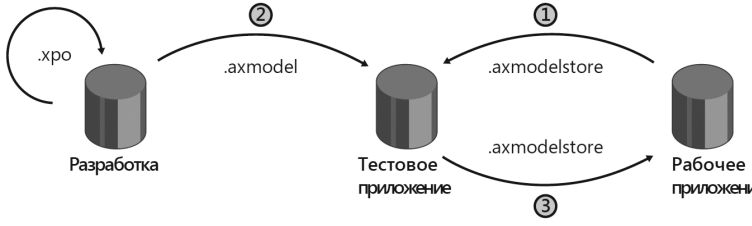

Рис. 21–5. Создание и подготовка тестового окружения, развертывания хранилища моделей в рабочей среде

Примечание. Файлы XPO **не используются в процессе развертывания** системы из тестовой на рабочую. Они упомянуты на рис. 21-5, чтобы показать сценарии, в которых используются эти три формата. Файлы XPO должны использоваться только для обмена кодом между разработчиками.

Создание тестового окружения

Цель создания тестового окружения состоит в обеспечении идентичности метаданных в хранилище моделей с метаданными хранилища в рабочем приложении. Простейший путь к достижению этой цели – установка отдельной инсталляции Microsoft Dynamics AX и перенос на нее метаданных с рабочего приложения. Для этого переноса сначала вам потребуется экспортировать хранилище моделей из рабочего окружения.

```
AXUtil exportstore /file:ProductionStore.axmodelstore
```

На тестовой системе вы останавливаете AOS и затем импортируете файл хранилища моделей.

```
Net stop AOS60$01
AXUtil importstore /file:ProductionStore.axmodelstore
Net start AOS60$01
```

Подготовка тестового окружения

Цель подготовки тестового окружения – обновление системы новыми метаданными путем установки новых или обновления существующих моделей. Произведите импорт или обновление, как рассказывалось ранее в этой главе.

После импорта моделей запустите клиент Microsoft Dynamics AX и завершите контрольный список установки. Наиболее важные шаги – компиляция в P-код и CIL, так как результаты этих шагов являются частью хранилища моделей.

Рекомендуем также произвести тщательную проверку систему. Убедитесь, что вы проверили как правильную работу новой функциональности, так и сохранение работоспособности ранее существовавшей и отсутствие в ней регрессов.

Развертывание модели на рабочее приложение

Цель развертывания на рабочем приложении – обеспечить обновление метаданных на рабочем приложении метаданными из тестовой среды. Для их переноса сначала вам придется экспортировать хранилище моделей из тестового окружения.

```
AXUtil exportstore /file:TestStore.axmodelstore
```

Импортируйте файл хранилища моделей на рабочую систему. Для минимизации времени простоя AXUtil поддерживает двухфазный процесс импорта. Первая фаза импортирует метаданные в новую схему базы данных. Это занимает несколько минут и может делаться без остановки рабочей системы. Вторая фаза заменяет метаданные хранилища моделей импортированными метаданными из этой схемы. Это занимает несколько секунд и должно происходить при остановленном сервере приложений.

Создание новой схемы:

```
AXUtil schema /schemaname:TransferSchema
```

Импорт файла хранилища моделей в новую схему:

```
AXUtil importstore /file:TestStore.axmodelstore /schema:TransferSchema
```

Когда все пользователи отключились, останавливаем AOS:

```
Net stop AOS60$01
```

Применяем изменения к хранилищу моделей и переносим новую схему в активную схему:

```
AXUtil importstore /apply:TransferSchema /backupschema:dbo_backup
```

Перезапускаем AOS:

```
Net start AOS60$01
```

Примечание. Отметьте использование параметра */backupschema* в примере. С его помощью, если вдруг возникнут неожиданные проблемы, вы сможете быстро вернуться к исходным метаданным. Если же резервная схема вам более не нужна, то можете удалить ее с помощью команды AXUtil *schema* /*drop*:<имя схемы>.

На этом этапе метаданные в рабочем окружении идентичны метаданным из тестового окружения. Но нужно выполнить еще несколько действий перед тем, как система готова будет принять пользователей. Сюда входят синхронизация базы данных, создание ролевых центров, размещение веб-контента, настройка документооборотов, развертывание кубов, импорт портов интеграции и размещение отчетов. Обо всех этих задачах подробнее можно прочитать в техническом описании «Deploying Customizations Across Microsoft Dynamics AX 2012 Environments» по адресу: *http://www.microsoft.com/download/en/details.aspx?id=26571*.

Пара слов об идентификаторах элементов

Бизнес-данные ссылаются на идентификаторы элементов метаданных. Процесс, упомянутый в предыдущих разделах, обеспечивает, что идентификаторы элементов на рабочей системе останутся неизменными, и тем самым гарантирует целостность бизнес-данных.

Это достигается только обменом метаданными между тестовым и рабочим приложением посредством файлов хранилища метаданных, что сохраняет идентификаторы элементов. Но если идентификаторы элементов между тестовым и рабочим окружениями будут рассинхронизованы из-за импортированных XPO-файлов или файлов моделей, вы должны пересоздать тестовое окружение.

Команда *importstore* имеет встроенный механизм безопасности. Она гарантирует соответствие между идентификаторами элементов на целевой системе и в импортируемом файле. Если же возникнут конфликты, операция импорта будет остановлена. Используйте параметр */verbose* для получения списка конфликтов и параметр */idconflict:overwrite* для принудительного продолжения импорта. Используйте последний параметр только на системах, где вам не нужны данные – и никогда на рабочем приложении.

Дополнительная информация имеется в разделе «Идентификаторы элементов» ранее в этой главе.

Программный интерфейс хранилища моделей

Утилита AXUtil, используемая во всех примерах этой главы, предоставляет интерфейс командной строки для команд хранилища моделей, предлагаемых API этого хранилища. Кроме того, из оболочки Microsoft Dynamics AX 2012 Management Shell доступен также и Powershell-интерфейс.

Обе эти реализации программного интерфейса основаны на использовании сборки *AXUtilLib.dll*. Вы также можете использовать эту сборку для автоматизации любых операций с хранилищем моделей. Ссылка на сборку есть в X++, так что к хранилищу моделей можно легко обратиться из X++. Некоторые из наиболее часто используемых команд доступны также в классе *SysModelStore*.

Программный интерфейс также содержит метод для генерации лицензионных ключей для лицензионных кодов в AOT, основываясь на имени владельца лицензии и серийном номере. В примере ниже показано, как вызвать этот метод из управляемого сайта в рамках сценария автоматизированного приобретения лицензии. Дополнительная информация о защите вашего решения с помощью лицензионных кодов есть в главе 11.

```
using Microsoft.Dynamics.MorphX;
using Microsoft.Dynamics.AX.Framework.Tools.ModelManagement;

protected void Submit_Click(object sender, EventArgs e)
{
    string certPath = @"c:\Licenses\MyCertPrivate.pfx";
    string licensePath = @"c:\Licenses\" + Customer.Text + "-license.txt";
    string licenseCodeNameInAot = "MyLicenseCode";
    string certificatePassword = "password"; //TODO: Перенести в безопасное хранилище
    AXUtilContext context = new AXUtilContext();
    AXUtilConfiguration config = new AXUtilConfiguration();

    LicenseInfo licenseInfo = new LicenseInfo(licensePath, certPath,
                                licenseCodeNameInAot, Customer.Text, Serial.Text, null,
                                certificatePassword);
    config.LicenseInfo = licenseInfo;
    AXUtil AXUtil = new AXUtil(context, config);
    if (AXUtil.GenerateLicense())
    {
        Response.AddHeader("Content-Disposition", "attachment;filename=license.txt");
        Response.TransmitFile(licensePath);
        Response.Flush();
        Response.End();
    }
}
```