

# **OpenXML**

Кратко и доступно



## Предисловие к русскому изданию

Информационные технологии стремительно меняют жизнь человека, погружая общество в новую среду существования — цифровой мир, материальной основой которого является информация и связанные с ней информационные процессы.

В таком обществе, называемом в литературе обществом знаний, именно информация и знания представляют собой основные активы социума, цель его деятельности.

Ввиду центральной роли информации в деятельности человека средства и инструменты представления и обработки информации должны соответствовать уровню информатизации общества, автоматизации информационных процессов.

Естественно, что в этом контексте важная роль принадлежит средствам описания информации и метainформации, хранящихся в базах данных и знаний компьютеров и компьютерных сетей. Такие средства должны обеспечивать полноту описательных возможностей типов информации, открытость спецификаций и технологичность их обработки, переносимость и повторную используемость информационных ресурсов, самоописываемость и интероперабельность информации, поддержку агрегирования функциональностей систем ИТ.

Для развития технологии в области средств описания и представления информации в настоящее время характерны две основные тенденции. Первая связана с необходимостью перехода от двоичных данных и соответствующих им форматов к самоописывающей информации и соответствующим самоописывающим эту информацию языкам. Важную основу для продвижения в этом направлении заложили язык XML и поддерживающие его технологии.

Вторая тенденция заключается в переходе к комплексной полномасштабной ИТ-стандартизации. В частности, для развития технологий и приложений семантического WEB на принципах открытых систем чрезвычайно актуальным является разработка семейства международных стандартов описания прикладных данных.

Именно для решения проблем в рамках указанных выше тенденций компанией Майкрософт совместно с организацией стандартизации ECMA разработан проект международного стандарта Open XML, построенный на базе стандартов XML, ZIP и Xml-Schema и включающий набор языков разметки для текстовых документов, электронных таблиц и презентаций (данный проект стандарта получил название Ecma 376).

Наличие такого стандарта обеспечивает совершенно новые возможности как в связи с реализацией на практике концепции активного документа, так и благодаря применению форматов документов наиболее распространенной офисной платформы Microsoft Office при создании пользовательских офисных приложений. При этом обеспечивается легкая возможность создания приложений, интегрирующих функциональность стандартных приложений Microsoft Office.

Настоящая книга описывает часть спецификации проекта стандарта Ecma 376, включающую наиболее полезные в практическом применении аспекты Open XML.

В.А. Сухомлин,  
профессор МГУ им. М.В. Ломоносова

## К разработчикам

Уважаемые коллеги! Я рад рекомендовать вашему вниманию первую книгу Воутера Ван Вугта.

Это действительно знаменательный момент — вы держите в руках первую книгу на русском языке (а может, вообще первую в мире книгу) полностью посвященную новому поколению форматов «офисных» документов. Термин «офисный» не вполне отражает значение рассматриваемого в книге формата. Он описывает самый широкий круг документов, и позволяет представить самые разные сущности — от произведений художественной литературы до технической документации, от семейных бюджетов до бизнес-аналитики, от налоговых деклараций до красочных презентаций.

Формат Open XML, утвержденный Европейской ассоциацией стандартизации в области информационных и коммуникационных систем (ECMA International) как отраслевой и представлен в ISO как проект международного стандарта (ISO/IEC DIS 29500), обладает рядом достоинств, которые обеспечили ему признание практиков. Вот только некоторые:

он основан на передовых достижениях компьютерных теории и практики и разработан специалистами таких компаний, как Apple, Barclays Capital, British Petroleum, Британская Библиотека, Библиотека Конгресса США, Essilor, Intel, Microsoft, NextPage, Novell, Statoil, Toshiba;

его использование не связано с искусственными ограничениями юридического характера, не требует лицензионных отчислений и не сопряжено с риском нарушения прав интеллектуальной собственности разработчиков;

это наиболее полный и подробный стандарт форматирования документов из всех существующих, руководствуясь им, можно создавать собственные универсальные и специализированные приложения самого разного назначения — недаром даже самые «заклятые противники» Microsoft уже взяли его на вооружение;

это стандарт, позволяющий управлять обработкой как простых, так и очень сложных и объемных документов;

это стандарт, обеспечивающий высокую степень однозначности при воспроизведении документов различными приложениями;

он предусматривает средства, облегчающие создание инструментов конвертирования файлов из «старых» бинарных форматов Microsoft Office в Open XML, без чего любой формат рискует остаться оторванной от реальных потребностей теоретическим построением;

стандарт Open XML создан с учетом традиций, норм и практик разных стран и наций, с должной мерой внимания к потребностям людей с ограниченными возможностями, с прицелом на интероперабельность приложений независимых разработчиков;

это расширяемый формат и вы легко можете добавить средства, необходимые вашему уникальному специализированному приложению, будь то ERP, CAD/CAM, система синтеза потокового видео или онлайн-сервис сбора отчетности.

Используя эту книгу и спецификацию стандарта, вы можете создать собственное полноценное и интероперабельное офисное приложение «с нуля». Я искренне надеюсь, что ее чтение окажется для вас захватывающе интересным и практически полезным. Успехов вам в новом мире открытых офисных стандартов.

Владислав Шершульский  
Менеджер по стратегическим инициативам Microsoft в России

## Содержание

Предисловие к русскому изданию .....	iii
К разработчикам .....	iv
Содержание .....	v
Благодарности .....	vii
Предисловие .....	viii
Введение .....	viii
Кому адресована эта книга .....	viii
ECMA Office Open XML .....	1
Стандарт Open XML .....	1
Глава 1 WordprocessingML .....	2
Создание цифровых документов .....	2
Формирование основной структуры .....	3
Добавление текста в документ .....	7
Форматирование текста .....	10
Таблицы .....	13
Применение стилей в документе .....	16
Добавление изображений .....	24
Макет страницы .....	26
Пользовательский XML-код в документах .....	28
Завершение документа .....	33
Дополнительные вопросы .....	35
Резюме .....	44
Глава 2 SpreadsheetML .....	45
Введение .....	45
Элементы простой электронной таблицы .....	46
Создание листов .....	47
Формулы .....	48
Оптимизация электронных таблиц .....	49
Таблицы .....	51
Сводные таблицы .....	54
Создание и размещение диаграммы .....	58
Применение стилей .....	60
Условное форматирование .....	64
Листы диаграмм .....	67
Дополнительные возможности .....	67
Резюме .....	68

Глава 3 PresentationML.....	69
Введение .....	69
Структура документа PresentationML .....	69
Фигуры .....	70
Элементы простой презентации .....	73
Объекты-рамки .....	76
Рисунки .....	78
Таблицы, диаграммы и схемы.....	79
Глава 4 DrawingML.....	81
Введение .....	81
Текст .....	81
Графика .....	84
Таблицы.....	90
Диаграммы .....	93
Темы .....	99
Единицы измерения.....	101
EMU.....	101
Twip .....	101

## Благодарности

Свои соображения по поводу технологий я излагаю в блогах и написание книги для меня дело новое. Чтобы ее текст воспринимался легко, а содержание было технически корректным, я прибег к помощи Дага Мэхью (Doug Mahugh) и Маурисио Ордонеса (Mauricio Ordonez), без которых на эту книгу ушло бы гораздо больше времени. Их совместные усилия позволили заметно ее улучшить. Спасибо вам обоим за потраченное время.

## Предисловие

Впервые я узнал о Воутере Ван Вугте в апреле 2006-го, когда он стал отвечать на вопросы разработчиков на сайте OpenXmlDeveloper.org. За несколько месяцев Воутер опубликовал много полезного на OpenXmlDeveloper, выложил примеры программ на языках Open XML в своем блоге и разработал утилиту Package Explorer, которая будет полезна разработчикам, ориентирующимся на Open XML. Эта утилита в качестве проекта с открытым кодом находится на сайте Codeplex.

В конце 2006-го я встретился с Воутером — мы вместе проводили первый семинар по Open XML в Париже. В начале 2007-го мы оба провели множество аналогичных семинаров по всему миру. Воутер должен был лишь проводить семинары, но он этим не ограничивался: он дополнял занятия новым материалом, включая примеры программ и образцы документов. Я использую его примеры на своих семинарах и опубликовал один из них в своем блоге, предварив его заголовком: «Эй, Даг! Ты украл мои примеры!».

Да, это так. Но ведь это можно считать комплиментом!

Воутер всегда с энтузиазмом помогал разработчикам осваивать Open XML. Ближе к окончанию первой серии семинаров, когда была выпущена предварительная версия (CTP) Microsoft SDK для форматов Open XML, я был сильно занят и какое-то время не общался с Воутером. Через два дня после выхода CTP я зашел на форум поддержки MSDN и обнаружил, что Воутер уже отвечает на вопросы по поводу Open XML-разработки. Стоит какому-нибудь разработчику спросить что-то про Open XML — Воутер тут как тут с готовым ответом.

Свой богатый опыт работы с Open XML Воутер обобщил в этой книге, которая позволит разработчикам быстро и легко освоить новую технологию. Участники его семинаров на каждой странице узнают его стиль: доходчивый, энергичный и полный энтузиазма.

Open XML открывает новую эру форматов документов. Впервые в истории ИТ в самом распространенном ПО для работы с документами — Microsoft Office — применяется в качестве формата файлов по умолчанию открытый, документированный стандарт. Это значит, что разработчики могут писать программы для создания и чтения таких документов на любой платформе и языке программирования. Подобно тому, как HTML, HTTP и другие стандарты перенесли интерактивные услуги в открытый мир Интернета, основанные на XML стандарты форматов документов перенесут обработку деловой документации из закрытого, ориентированного на внутренние регламенты прошлого — в будущее с его открытыми универсальными стандартами.

Движение в этом направлении началось в конце 2005-го, когда представители Apple, Barclays Capital, BP, Британской библиотеки, Essilor, Intel, Microsoft, NextPage, Novell, Statoil, Toshiba и Библиотеки Конгресса США сформировали рабочую группу Технического комитета 45 (TC45) ассоциации Ecma International. А уже в декабре 2006-го эта группа выработала стандарт Ecma 376, который в настоящее время является официальным материалом стандарта Open XML.

В данной книге отражена малая часть спецификации Ecma 376 — лишь те моменты, которые такой опытный пользователь Open XML, как Воутер Ван Вугт, счел полезными в практическом применении Open XML. Усвоив материалы этой книги, разработчики смогут задействовать преимущества Open XML и начнут разрушать исторически сложившиеся барьеры между документами, процессами и данными.

Если вы хотите быть одним из лидеров в разработке с применением Open XML, эта книга для вас. К тому же из нее можно позаимствовать классные примеры... Спасибо, Воутер!

Даг Мэхью  
Open XML Technical Evangelist, Microsoft  
23 июня 2007

## Введение

Среди множества новых технологий, реализованных в платформе Microsoft Office 2007, на одну нужно обратить особое внимание. Тем более, что ее значение выходит далеко за рамки пусть самого распространенного, но вполне конкретного программного продукта. Это соответствующие стандарту Open XML языки разметки для текстовых документов, электронных таблиц и презентаций, призванные избавить нас от проблем, обусловленных применением старых двоичных форматов содержимого документов. Open XML предоставляет открытую стандартизованную среду, построенную на таких признанных стандартах, как XML, ZIP и Xml-Schema. Поскольку в наши дни эти технологии применяются практически на всех платформах, документ уже не назовешь черным ящиком с данными неизвестного формата. Теперь документ — это и есть данные! Его можно легко интегрировать в ваш бизнес-процесс. Open XML предоставляет ряд новых технологий, позволяющих отображать содержащиеся в документе данные вне самого документа, упрощающих доступ к нужным частям документа и обеспечивающих развитые возможности многократного использования его компонентов.

Назначение этой книги — дать в ваше распоряжение строительные блоки для создания решений, ориентированных на работу с документами. Вы познакомитесь с основами языков WordprocessingML, SpreadsheetML, PresentationML и DrawingML, а также узнаете, как применять их возможности для создания собственных конструкций при обработке текстов, электронных таблиц и визуальных эффектов.

### Кому адресована эта книга?

---

В этой книге содержится подробный обзор трех основных языков разметки стандарта Open XML. Книга рассчитана на читателей, знакомых с XML или HTML. Архитекторы и разработчики ПО, создающие ориентированные на документы решения, узнают, как построить эти решения на платформе Open XML. Эта книга будет полезна как новичкам в языках разметки, так и специалистам в этой области, не знакомым с Open XML.



# ECMA Office Open XML

## Стандарт Open XML

Новый стандарт разметки офисных документов Open XML был предложен, чтобы преодолеть ограничения традиционных двоичных форматов хранения документов, длительное время использовавшихся с такими платформами, как Microsoft Office. Основанный на XML формат стандартизован и базируется на открытых технологиях, что обеспечивает его применимость на различных платформах и во многих ОС. Первая версия данного стандарта включает три основных языка разметки: WordprocessingML — для описания текстовых документов, SpreadsheetML — для электронных таблиц и PresentationML — для презентаций. Есть также множество вспомогательных языков, таких как DrawingML, который поддерживает работу с изображениями, диаграммами, схемами и таблицами. Документ Open XML представляет собой контейнер, содержащий несколько компонентов. В текущей версии этот контейнер является сжатой ZIP-папкой, а его компонентами — хранящиеся в ней файлы. Однако хранение компонентов документа в базе данных обеспечит более эффективное их повторное использование. Стандарт определяет не только разметку документов, но и структуру контейнера. Структура описана во втором из пяти документов, составляющих данный стандарт, — в Открытом соглашении по правилам упаковки (Open Packaging Convention). Другой важный раздел спецификации — его пятая часть, где описаны требования к совместимости кода разметки (Markup Compatibility). Здесь описаны такие важные для разметки моменты, как правила обработки информации о версиях и прочие тонкости.

Различные уровни спецификации представлены на рис. 1. ZIP, XML и Unicode не являются частью стандарта Open XML.

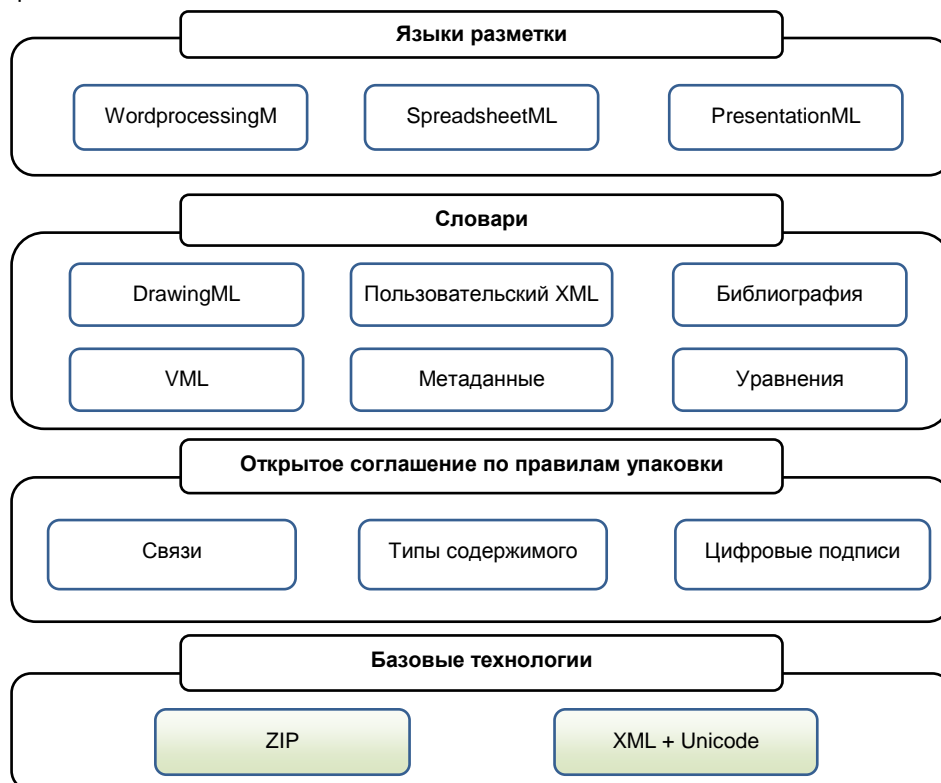


Рис. 1. Компоненты Open XML

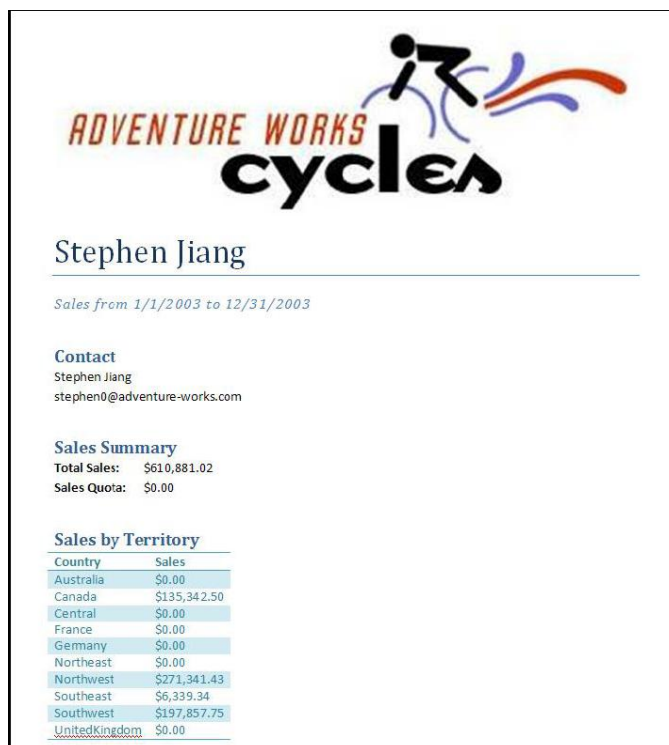
# Глава 1

## WordprocessingML

- Знакомство со структурой документов Open XML
- Основы разметки документов, обеспечиваемые языком WordprocessingML: абзацы, области и таблицы
- Добавление рисунков и схем с помощью DrawingML
- Внесение бизнес-данных в контейнер WordprocessingML
- Удаление комментариев и данных о версиях в окончательном варианте документа

### Создание цифровых документов

С документами мы работали задолго до появления цифровых электронных таблиц и компьютерных презентаций. Создавали их разными способами: раньше печатали на машинках, а в наши дни электронные документы могут генерироваться автоматически. Процесс работы с документами тоже изменился. Цифровые документы имеют массу преимуществ в сравнении с бумажными предшественниками. Цифровые подписи, любые формы встраиваемого в документ содержимого, классификация документов в соответствии с потребностями бизнеса — теперь все это в порядке вещей. Мне нравится определение, подчеркивающее важность технологий обработки документов: «Документ — движущая сила информационного обмена». WordprocessingML и связанные технологии позволяют задействовать все эти преимущества, если вы используете поддерживающее их ПО, такое как Microsoft Office System 2007. В этой главе вы узнаете о структуре документов WordprocessingML и способах их форматирования с применением стилей. Затем вы увидите, как «оживить» документ, добавив в него описываемые пользовательским кодом бизнес-данные, что существенно упрощает использования документа как контейнера информации. В конце главы мы рассмотрим некоторые способы заключительной доводки документа перед его отправкой коллегам или клиентам.



**ADVENTURE WORKS**  
**cycles**

**Stephen Jiang**

*Sales from 1/1/2003 to 12/31/2003*

**Contact**  
Stephen Jiang  
stephen0@adventure-works.com

**Sales Summary**  
Total Sales: \$610,881.02  
Sales Quota: \$0.00

**Sales by Territory**

Country	Sales
Australia	\$0.00
Canada	\$135,342.50
Central	\$0.00
France	\$0.00
Germany	\$0.00
Northeast	\$0.00
Northwest	\$271,341.43
Southeast	\$6,339.34
Southwest	\$197,857.75
UnitedKingdom	\$0.00

Рис. 2. Простой отчет

На рис. 2 показан отчет, который будет использоваться для примеров кода в этой главе. В этом документе несколько интересных элементов. Прежде всего это основные текстовые элементы — главные строительные блоки документа. Таблицу внизу страницы мы разберем очень подробно, в том числе рассмотрим такие полезные оформительские эффекты, как чередование строк. В завершение в заголовке отчета мы добавим рисунок. Мы рассмотрим и другие элементы WordprocessingML. Перемещение сведений о формате в описатели стилей расширяет возможности многократного использования. Мы обсудим применение пользовательских XML-тегов и таких сложных элементов, как оглавления. Но прежде чем добавлять усложненные возможности, построим основу документа.

## Формирование основной структуры

Перед изучением всех элементов, составляющих образец документа, рассмотрим его базовую структуру. Если вы переименуете документ WordprocessingML, заменив расширение файла *docx* на *zip*, вы обнаружите много разных элементов, особенно если это большой документ. Компоненты (части) документа WordprocessingML представлены отдельными файлами в ZIP-пакете. Кроме компонентов, в которых хранится разметка документа, ZIP-контейнер содержит и вспомогательные части с параметрами, шрифтами и стилями. Наиболее распространенные элементы документов показаны на рис. 3. Большинство из них обязательными не являются.

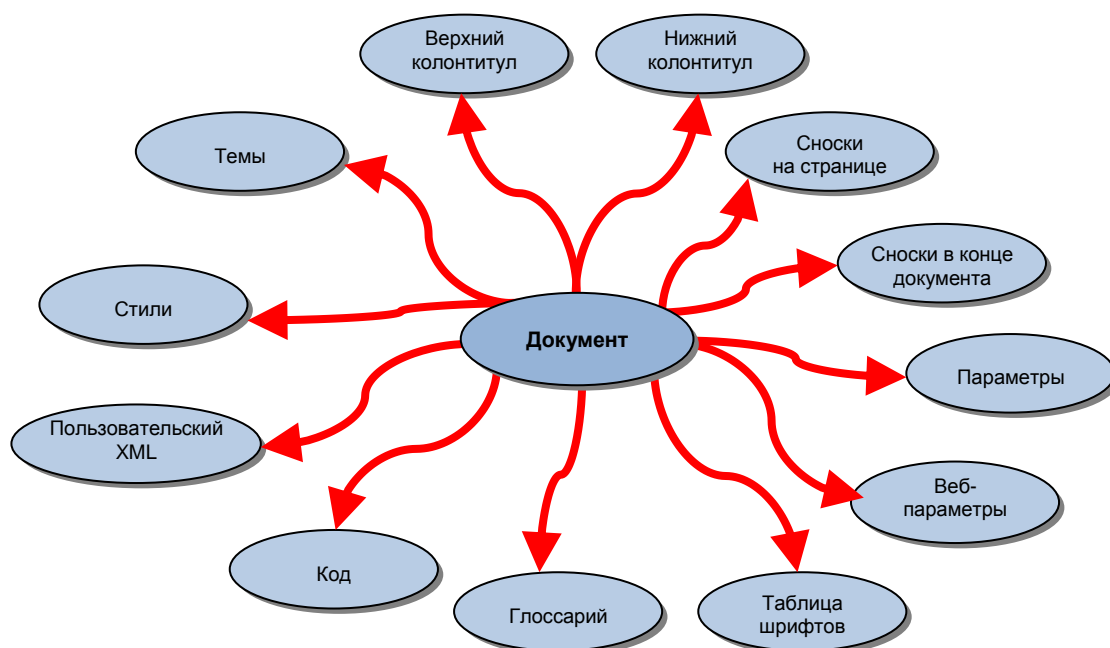


Рис. 3. Структура документа WordprocessingML

В корневом каталоге zip-контейнера вы обнаружите компонент *[Content\_Types].xml*. В нем хранится словарь, описывающий типы содержимого всех остальных компонентов пакета. По типу содержимого потребитель пакета (клиент) может определить вид информационного наполнения пакета. Необходимость различать двоичные и XML-данные очевидна, но и XML-данные следует разделять по типам, поскольку XML содержится почти во всех компонентах zip-контейнеров. При дальнейшем исследовании контейнера вы встретите XML-файлы с расширением *rels*, которые всегда хранятся в папке *\_rels*. Это файлы связей, объединяющих в единое целое различные части документа. Связи между файлами хранятся не в самих связываемых файлах, а в отдельных, специальных файлах. Такая модель позволяет облегчить работу клиентских приложений, анализирующих структуру пакетов в поисках конкретных элементов. При работе с пакетами Open XML очень важно помнить следующее правило: никогда не полагайтесь на пути файлов — всегда анализируйте связи.

---

*При просмотре пакета всегда используйте связи,  
не пытайтесь найти компоненты по «известным» путям.*

---

Документ WordprocessingML должен состоять минимум из трех компонентов. В одном — он обычно называется *document.xml* — определяется основное тело документа. Тип содержимого этой части должен храниться в компоненте типов содержимого. В каждом пакете имеется единственный компонент типов содержимого (content-types part). Расположение частей основного тела документа должно определяться посредством компонента связей (relationship part) — третьей обязательной составляющей любого пакета. Создание нового чистого документа следует начинать с создания пустого каталога. В этом пустом каталоге создайте вложенный каталог *\_rels* (не пропустите знак подчеркивания в названии!). В корневом каталоге будут храниться два файла: со списком типов содержимого и тело документа. Во вложенном каталоге *\_rels* хранится третий компонент, описывающий связи. Вообще-то основной компонент документа может храниться в любом каталоге — важно лишь, чтобы указатель связи корректно на него ссылался.

Корневой каталог в данном примере используется для наглядности. В Microsoft Office Word 2007 для хранения содержимого служит вложенная папка *word*. В других приложениях могут использоваться другие места.

Первым примером в любой книге должен быть, конечно же, «Hello World!». Создадим такой документ. Ниже показан пример кода в основной части документа и его внешний вид, отображаемый клиентской программой. Пока не пытайтесь разобраться в структуре разметки — ее мы подробно рассмотрим ниже.

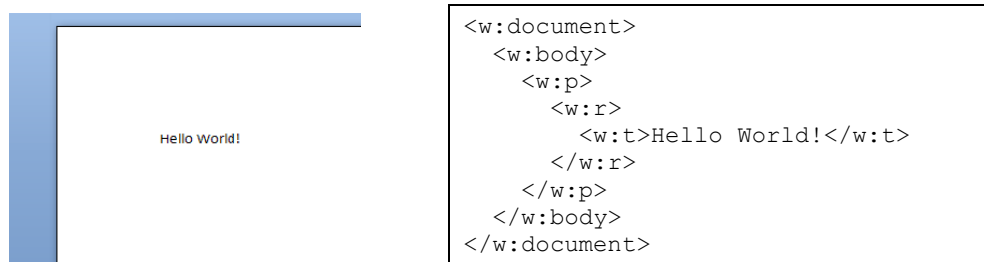


Рис. 4. Документ «Hello World!»

Кроме этого текста, нужны еще два компонента: список типов содержимого и компонент связей. Просто поместить этот XML в какой-нибудь ZIP-контейнер недостаточно, очень важна правильная структура. Прежде всего этот XML с текстом «Hello World!» нужно поместить в специальный компонент пакета, который называется начальным (*start-part*), а затем нужно создать другие компоненты пакета.

## Начальный компонент, *document.xml*

Создание любого документа Open XML начинается с определения начального компонента. Это то место, с которого клиентская программа начинает просмотр содержимого документа. Для всех трех основных языков Open XML всегда существует единственный компонент внутри ZIP-пакета, который интерпретируется как начальный. При этом в каждом из трех языков этот начальный компонент применяется по-своему. В *WordprocessingML* начальный компонент используется для хранения текста основного тела документа, в нашем примере — «Hello World!». Как и большинство других документов, начальный компонент содержит XML-разметку.

Для создания чистого документа много кода не требуется. Элемент *document* — единственный, который должен присутствовать в этом компоненте. Открыв такой документ программой-клиентом Open XML, например редактором Microsoft Word, вы увидите, что он пустой.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
</w:document>
```

### Пример 1. Документ *WordprocessingML* с минимальным содержимым

Внутри элемента *document* можно поместить различные блоки, составляющие документ, скажем, таблицы или абзацы текста. Большинство этих элементов имеет одинаковый идентификатор пространства имен XML. В Microsoft Office Word 2007 используется префикс *w*. Вы можете выбрать любой другой, но пространство имен XML должно быть единым.

*Основное пространство имен WordprocessingML:*  
*http://schemas.openxmlformats.org/wordprocessingml/2006/main*

*В большинстве примеров этой книги написание пространства имен будет сокращено для экономии места.*

*Предложение «schemas.openxmlformats.org» заменено на многоточие (...).*

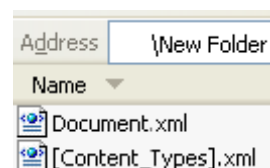
Добавив в тег *document* всего несколько элементов, мы увидим сакральное «Hello World!». В следующем примере показана вся разметка для этого начального варианта. На XML-содержимое внимания пока не обращайте — это всего лишь иллюстрация простейшего документа. Чтобы получить законченный пакет, нам надо добавить определение типов содержимого и основную связь.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<w:document
xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t>Hello World!</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

### Пример 2. Документ *WordprocessingML* с минимальным содержимым

## Список типов содержимого, [Content\_Types].xml

Теперь, сформировав основной компонент, определим тип его содержимого, чтобы клиентские приложения Open XML могли понять, каково информационное наполнение основного компонента. «Стандартные» файлы для этих целей не используются. Сам пакет содержит сведения о типах содержимого. Компонент типов содержимого хранится в корневом каталоге вместе с основным компонентом документа. Это месторасположение неизменно. Имя файла тоже должно быть в точности таким — *[Content\_Types].xml*. Не пропустите квадратные скобки!



Из названия следует, что в компоненте типов содержимого содержатся описания типов всех компонентов пакета (в виде строк). Для хранения используется два подхода. В первом случае определяется соответствие между стандартным типом содержимого (default) и расширением имен файлов с компонентами пакета. Во втором — тип переопределяется (override) для конкретного компонента пакета; при этом указывается его местоположение.

Начальный компонент пакета WordprocessingML определяется следующим типом содержимого.

*Тип содержимого основного документа:*

*application/vnd.openxmlformats-officedocument.wordprocessingml.document.main+xml*

Кроме типа содержимого основного документа, надо указать тип содержимого файла связей и установить некоторые стандартные значения для компонентов, которые будут добавляться в пакет позднее.

Для документа с минимальным содержимым может указываться описание типов содержимого, как в примере 3.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <Default Extension="rels"
    ContentType="application/vnd.openxmlformats-package.relationships+xml" />
  <Default Extension="xml" ContentType="application/xml" />
  <Override PartName="/document.xml"
    ContentType="application/vnd.openxmlformats-officedocument.wordprocessingml.document.main+xml" />
</Types>
```

### Пример 3. Компонент типов содержимого

В разделе типов содержимого для описания XML-наполнения служит специальное пространство имен XML. В список *Types* можно включать элементы двух типов: *Default* (стандартный) и *Override* (переопределенный). В нашем примере для всех файлов с расширением *rels* устанавливается стандартный тип содержимого. В дальнейшем в файле с таким расширением будет сохранен описатель связи между пакетом и основным документом. Второй стандартный тип определен для XML-компонентов пакета. Поскольку входящие в пакет XML-файлы могут иметь разное информационное наполнение, единственным подходящим типом содержимого по умолчанию резонно считать *application/xml*. Для каждого компонента с уникальным типом содержимого, отличным от стандартного, будет указан свой описатель типа. Чтобы создать корректный пакет, нужно добавить одно переопределение. С его помощью описывается содержащий основную часть документа компонент *document.xml*. Здесь для определения типа содержимого применяется не атрибут *Extension*, указывающий расширение имени файла, а *PartName* — позволяющий сослаться на конкретный компонент пакета. В *PartName* допускается прописывать только абсолютный путь, вычисляемый от корневого каталога пакета. Основной компонент документа будет называться *document.xml* и храниться в корневом каталоге вместе с компонентом типов содержимого, который мы сейчас обсуждаем.

*Приложения Open XML должны проверять соответствие информационного наполнения компонентов пакета заявленным типам содержимого. Документы, компоненты которых не соответствуют типам содержимого, считаются поврежденными.*

*Распространенной ошибкой при ручном редактировании документа Open XML является добавление в пакет новых компонентов без обновления списка типов содержимого. Если вы забудете добавить новые записи *Override* в список типов содержимого, документ не откроется, причем будет выдаваться ошибка с неинформативным сообщением*

## Компонент связей

Пакет обычно содержит несколько компонентов связей, но только в одном из них хранятся связи начальных компонентов. Начальные компоненты — это то, с чем вы начинаете работать, открывая документ. Начальный компонент документа WordprocessingML — это созданный на предыдущем этапе компонент *document.xml*.

Связи начальных компонентов хранятся в особом файле связей, *.rels*, который всегда находится в определенном вложенном каталоге. Прежде чем создавать компонент связей для нашего примера, надо создать правильный вложенный каталог. Файл связей для всех начальных компонентов всегда хранится в каталоге *\_rels*, вложенном в корневой каталог пакета. В файле *.rels*, хранящемся в каталоге *\_rels*, и содержатся связи начальных компонентов (рис. 5).

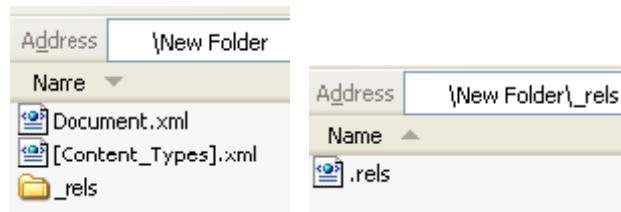


Рис. 5. Основной компонент связей

Вот содержимое компонента связей для образца отчета (пример 4):

```
<?xml version="1.0" encoding="UTF-16" standalone="yes"?>
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship
    Id="rId1"
    Type="http://schemas.openxmlformats.org/officeDocument/...
...2006/relationships/officeDocument"
    Target="document.xml" />
</Relationships>
```

Пример 4. Основной компонент связей

Для организации связей применяется список внутри элемента *Relationships*. Связь описывается тремя составляющими. Это идентификатор связи (ID), определяющий каждую из них. Он должен быть уникален в пределах отдельного файла связей. Тип связи определяется атрибутом *Type*. Атрибут *Target* определяет объект, с которым устанавливается связь. Заметьте: в файле связей нет сведений об их источнике, т. е. объекте, для которого устанавливается связь. Источник определяется самим компонентом связей. Поскольку рассматриваемый файл называется *.rels* и хранится в папке *\_rels*, источником является сам пакет. Значение атрибута *Target* определяется с учетом расположения источника. Поскольку связь устанавливается для самого пакета, источник идентифицируется знаком корневого каталога (*/*). Объединение знака */* со значением атрибута *Target* (*document.xml*) дает в результате путь */document.xml*, который и является точным местоположением основного компонента документа. Для основного компонента документа используется следующий тип связи.

---

*Тип связи для основного документа:*

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument>

---

Далее в этой главе мы создадим новые связи и файлы для них. Не забывайте: данный файл — единственный, который применяется для начальных компонентов. Другие используются аналогично, но немного иначе.

## Окончательный документ

Последний этап создания простого документа WordprocessingML — объединение всех трех частей в сжатую папку. Важно правильно указать исходное расположение. Выделите файлы *[Content\_Types].xml*, *document.xml* и папку *\_rels*, а затем выберите в контекстном меню **Отправить | Сжатая ZIP-папка** (Send-To | Compressed Folder). Если переместиться на уровень выше и сжать саму папку, а не содержащиеся в ней файлы, структура пакета будет некорректной.





Рис. 6. Создание документа в Проводнике

### API упаковки в .NET 3.0

Нормальная разработка с использованием Open XML подразумевает создание пакетов без помощи Проводника. Существуют различные интерфейсы прикладного программирования (API) для таких распространенных платформ, как Java, .NET и PHP. Приведенный ниже фрагмент кода демонстрирует создание компонентов пакета программными средствами. На момент написания книги наиболее развитый API был для .NET Framework, но в ближайшем будущем можно ожидать появления в Интернете и других средств как с открытым исходным кодом, так и коммерческих. Если вы выполните эту написанную на С# программу, вы получите тот же результат, что и на предыдущем этапе.

```
static void Main()
{
    using (Package package = Package.Open("HelloWorld.docx"))
    {
        // Создание основного компонента
        PackagePart mainPart = package.CreatePart(
            new Uri("/document.xml", UriKind.Relative),
            "application/vnd.openxmlformats-officedocument.
            wordprocessingml.document.main+xml");
        // Добавление связей
        package.CreateRelationship(
            mainPart.Uri, TargetMode.Internal,
            "http://.../officeDocument/2006/relationships/officeDocument");
        // Создание XML-кода пустого документа
        using (XmlWriter writer = XmlWriter.Create(
            mainPart.GetStream(FileMode.CreateNew, FileAccess.ReadWrite)))
        {
            writer.WriteStartElement(
                "w", "document",
                "http://.../wordprocessingml/2006/main");
            writer.WriteEndElement();
        }
    }
}
```

### Добавление текста в документ

Пожалуй, первое, что вы захотите сделать с пустым документом, — это добавить в него форматированный текст. Почти весь добавляемый вами текст будет храниться в основном компоненте документа, создание которого мы уже рассмотрели. Некоторые разделы документа, скажем, нижний и верхний колонтитулы, будут храниться в других местах.

В основном компоненте у вас уже есть корневой элемент *document*, с которого начинается формирование документа. В этом элементе может содержаться элемент *body*, в котором хранится тело документа. Различают две разновидности содержимого тела документа: блочное (block-level) и встроенное (inline) содержимое. Блочное содержимое определяет основную структуру документа, распространенные примеры — абзацы и таблицы. Блочное содержимое включает встроенное содержимое, к которому относятся области текста (runs) и рисунки.

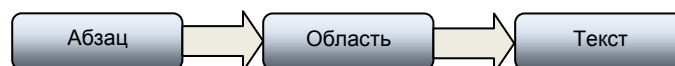


Рис.7. Иерархия текста в WordprocessingML

Абзац делится на области. Элемент, определяющий область, — это элемент самого нижнего уровня, к которому может применяться форматирование. Область в свою очередь делится на элементы текста.

Существуют элементы текста для определения информации, пригодной для печати, а также элементы, в которых хранятся непечатаемые коды, например, возврат каретки и перевод строки. Помните: документ не следует форматировать с помощью символов возвратов каретки и переводов строк. Базовой единицей структуры являются абзацы, и, применяя для них подходящие параметры границ и отступов, документ можно сформатировать гораздо лучше.

Давайте усложним пример для знакомства с элементами абзацев, областей и текста. На рис. 8 показан текст «Lorem Ipsum». Дизайнеры-типографы традиционно используют его в качестве образца текста.

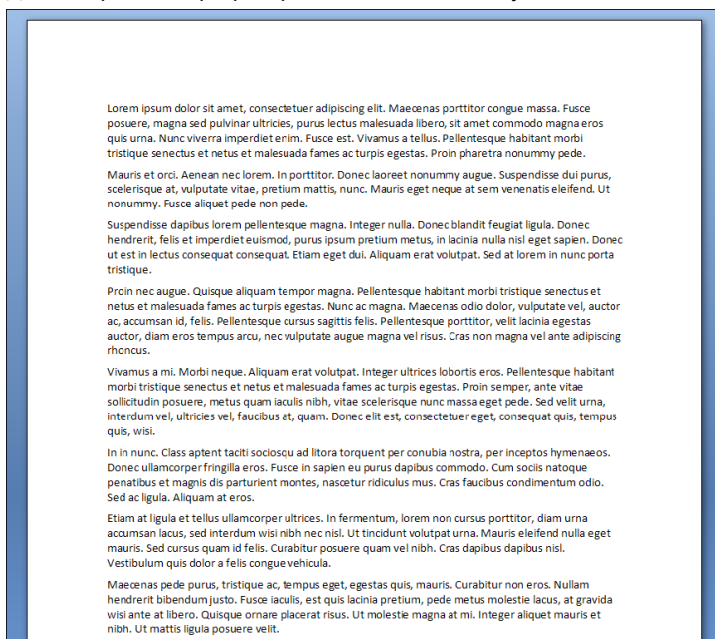


Рис. 8. Пример текста

---

*Вы сами можете сгенерировать страницу с таким текстом, создав новый документ Microsoft Office Word 2007 и введя =lorem(8,8).*

*Макрос «lorem» — специальная функция Word, позволяющая генерировать текст для демонстрационных и проверочных целей. Вы также можете использовать макрос «rand» для генерации псевдослучайного текста.*

---

Разметка первых двух абзацев показана в примере 5.

```
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t>
          Lorem ipsum dolor sit amet, consectetur adipiscing
          elit. Maecenas porttitor congue massa. Fusce posuere,
          magna sed pulvinar ultricies, purus lectus malesuada libero,
          sit amet commodo magna eros quis urna. Nunc viverra imperdiet
          enim. Fusce est. Vivamus a tellus. Pellentesque habitant
          morbi tristique senectus et netus et malesuada fames ac
          turpis egestas. Proin pharetra nonummy pede.
        </w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:r>
        <w:t>
          Mauris et orci. Aenean nec lorem. In porttitor. Donec
          laoreet nonummy augue. Suspendisse dui purus, scelerisque
          at, vulputate vitae, pretium mattis, nunc. Mauris eget
          neque at sem venenatis eleifend. Ut nonummy. Fusce aliquet
          pede non pede.
        </w:t>
      </w:r>
    </w:p>
    <!--Другие абзацы пропущены -->
  </w:body>
</w:document>
```



### Пример 5. Пример абзацев текста

Текст внутри элементов *t* не должен содержать разрывов строк — это повлияет на отображение текста в программе-клиенте. Копируя текст внутрь этого элемента, убедитесь, что он является одной строкой.

Структура абзаца достаточно проста, но с ней вполне можно поэкспериментировать. Например, текст первого абзаца можно по-разному расположить в нескольких элементах областей и текста, а выводиться он будет так же.

Сначала разобьем один элемент текста на несколько. Конечный результат не изменится. Пример 6 демонстрирует тот же абзац, но разбитый на два элемента *t*.

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">
      Lorem ipsum dolor sit amet, consectetur adipiscing
      elit. Maecenas porttitor congue massa. Fusce posuere,
      magna sed pulvinar ultricies, purus lectus malesuada libero,
    </w:t>
    <w:t>
      sit amet commodo magna eros quis urna. Nunc viverra imperdiet
      enim. Fusce est. Vivamus a tellus. Pellentesque habitant
      morbi tristique senectus et netus et malesuada fames ac
      turpis egestas. Proin pharetra nonummy pede.
    </w:t>
  </w:r>
</w:p>
```

### Пример 6. Первый абзац, разбитый на два элемента текста

Поскольку содержимое первого текстового фрагмента заканчивается пробелом, применяется атрибут *xml:space*. Если вы упустили этот атрибут, пробел в конце фрагмента будет удален отображающей программой.

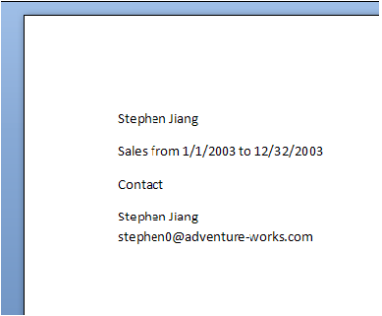
Можно сделать еще одну похожую вещь — разбить текст не на элементы *t*, а на области (пример 7).

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">
      Lorem ipsum dolor sit amet, consectetur adipiscing
      elit. Maecenas porttitor congue massa. Fusce posuere,
      magna sed pulvinar ultricies, purus lectus malesuada libero,
    </w:t>
  </w:r>
  <w:r>
    <w:t>
      sit amet commodo magna eros quis urna. Nunc viverra imperdiet
      enim. Fusce est. Vivamus a tellus. Pellentesque habitant
      morbi tristique senectus et netus et malesuada fames ac
      turpis egestas. Proin pharetra nonummy pede.
    </w:t>
  </w:r>
</w:p>
```

### Пример 7. Абзац, разбитый на два элемента области

Разбиение на уровне областей и текста можно комбинировать в одном абзаце. В частности, это делается для форматирования текста, о чем мы вскоре поговорим. Поскольку самый низкий уровень, на котором можно применять форматирование текста, — область, то, скажем, для выделения одного слова из области полужирным шрифтом нужно добавить новые области. Добавление же элементов текста в элементы области требуется, в частности, для хранения непечатаемых символов, таких как знаки табуляции или перевода строки.

В нашем демонстрационном отчете тоже будет несколько абзацев (рис. 9). Заметьте: пока они не сформатированы. Мы добавим форматирование в следующем разделе.



```
Stephen Jiang
Sales from 1/1/2003 to 12/31/2003
Contact
Stephen Jiang
stephen0@adventure-works.com
```

Рис. 9. Неформатированные абзацы

Чтобы создать пример отчета, который мы будем рассматривать в качестве примера, надо добавить несколько абзацев текста в пустой документ. Хотя и можно попрактиковаться разбивать абзацы на области и текстовые элементы, удобней использовать одну область и текстовый элемент. К модели, которую мы уже рассмотрели, добавилась одна деталь: в последнем абзаце, содержащем имя «Stephen Jiang» и адрес электронной почты, имеется новый элемент — *cr*. В распечатанном документе вы видите имя и адрес в разных строках. Хотя это и выглядит, как два абзаца, для этого текста используется по одному элементу абзаца и области, а также два элемента текста с элементом возврата каретки (*cr*) между ними. Вот что нужно добавить в пустой документ (пример 8):

```
<w:document xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:body>
    <w:p>
      <w:r>
        <w:t>Stephen Jiang</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:r>
        <w:t xml:space="preserve">Sales from 1/1/2003 </w:t>
        <w:t>to 12/32/2003</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:r>
        <w:t>Cont</w:t>
      </w:r>
      <w:r>
        <w:t>act</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:r>
        <w:t>Stephen Jiang</w:t>
        <w:cr />
        <w:t>stephen0@adventure-works.com</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

Пример 8. Абзацы отчета-примера

## Форматирование текста

Следующий этап в создании примера отчета — форматирование текста. Это делается несколькими способами. Простейший — применение формата к элементам абзацев и областей, созданным в предыдущем разделе. Чтобы иметь возможность многократно применять формат, можно создать стиль. Об этом мы поговорим ниже в этой главе.

Прямое форматирование можно применять на уровнях абзаца и области. На обоих уровнях есть весьма широкие возможности. Проще всего воспользоваться диалоговыми окнами

**Абзац** (Paragraph) и **Шрифт** (Font) в Microsoft Office Word. Форматирование абзаца сказывается на нем в целом, например, определяется межстрочный интервал, поля и границы. На уровне областей можно задать вид отдельных фрагментов текста: изменить шрифт, применить полужирное начертание или курсив.

Контейнер для определения форматирования на уровне абзаца также позволяет хранить параметры форматирования уровня области, но применяться они будут не ко всему тексту абзаца, а лишь к знаку абзаца.

## Форматирование областей

В окончательном варианте отчета-примера применяются различные шрифты разных размеров. Этот эффект достигается установкой свойств элементов областей.



Stephen Jiang

На предыдущем рисунке показано название отчета. Кроме форматирования уровня абзаца, которое мы обсудим в следующем разделе, здесь также применено форматирование области, изменяющее семейство и размер шрифта. Все эти параметры форматирования уровня области хранятся внутри элемента-контейнера *rPr*, который называется элементом свойств области (*run-properties*).

---

*Определение свойств на уровне элемента — общая модель для всего стандарта Open XML. Для любого элемента  $x$  допустимо свойство  $xPr$ , которое хранится как первый дочерний элемент.*

---

В нашем примере для всех областей первого абзаца нужно добавить набор свойств, которые определяют семейство шрифтов и размер шрифта для текста, хранящегося внутри элементов текста.

Начать применять свойства области можно с простейшего приема — сделать шрифт, которым набран текст полужирным, добавив элемент *b*. Затем определяется размер шрифта с помощью элемента *sz*. Значения указываются в единицах, равных половине пункта. То есть значение 32 соответствует размеру в 16 пунктов. В примере 9 показано, как определять эти параметры. Размер шрифта здесь устанавливается в 26 пунктов, что соответствует заглавному тексту в отчете-примере.

```
<w:p>
  <w:r>
    <w:rPr>
      <w:b />
      <w:sz w:val="52" />
      <w:rFonts w:ascii="Cambria" />
    </w:rPr>
    <w:t>Stephen Jiang</w:t>
  </w:r>
</w:p>
```

#### Пример 9. Форматирование первого абзаца

Последний интересный параметр, используемый в примере, — определение шрифта с помощью элемента *rFonts*. Заметьте: в названии элемента употребляется множественное число — *rFonts* (шрифты области). Элемент *rFonts* — особенный, он позволяет указывать различные семейства шрифтов в одной области в зависимости от набора символов. Подробнее об этом написано в пункте 2.3.2.24 части 4 спецификации ECMA.

Рассмотрим еще пару элементов, чтобы окончательно преобразить наш отчет. Цвет (по умолчанию — черный) определяется элементом *color*. Курсив задается элементом *i*, а интервал — элементом *spacing* (пример 10):

```
<w:r>
  <w:rPr>
    <w:rFonts w:ascii="Cambria"/>
    <w:i />
    <w:color w:val="4F81BD" />
    <w:spacing w:val="15" />
    <w:sz w:val="24" />
  </w:rPr>
  <w:t>Sales from 1/1/2003 to
12/32/2003</w:t>
</w:r>
```



*Sales from 1/1/2003 to 12/31/2003*

#### Пример 10. Форматирование подзаголовка отчета

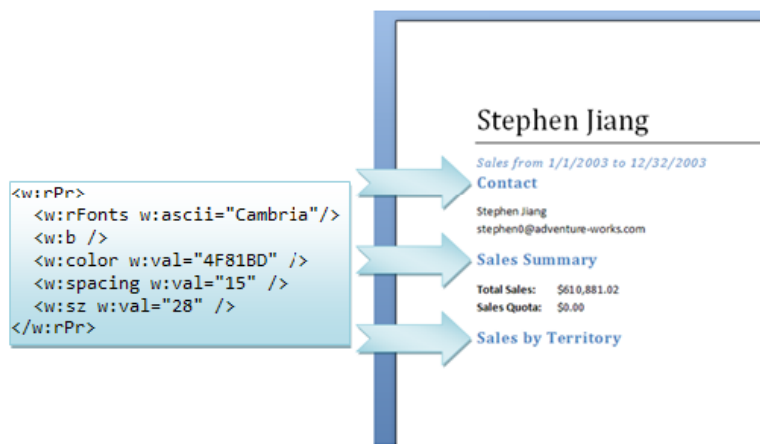
```
<w:r>
  <w:rPr>
    <w:rFonts w:ascii="Cambria"/>
    <w:b />
    <w:color w:val="4F81BD" />
    <w:spacing w:val="15" />
    <w:sz w:val="28" />
  </w:rPr>
  <w:t>Contact</w:t>
</w:r>
```



**Contact**

#### Пример 11. Форматирование заголовков

Форматирование каждой отдельной области может показаться довольно однообразным занятием. Конечно, существует более совершенный механизм, ведь формат некоторых его фрагментов повторяется.



Далее в этой главе мы рассмотрим концепцию иерархии стилей, которая позволяет решить эти задачи.

## Форматирование абзаца

Форматирование на уровне абзаца в нашем примере применяется для задания рамки абзаца. Абзац — это блочный элемент, ширина которого обычно ограничивается размером страницы. Его рамка имеет те же размеры.

Stephen Jiang

Параметры уровня абзаца хранятся в элементе *pPr*. Узел свойств хранится внутри элемента абзаца, как и в случае свойств *rPr*. Среди свойств абзаца есть границы, отступы, варианты выравнивания, позиции табуляции и др. В нашем примере первый абзац имеет нижнюю границу. При необходимости можно добавить границы со всех сторон.

В примере 12 демонстрируется объявление этой границы. Как и в HTML, нужно указать тип границы, ее размер (толщину) и цвет. В отличие от размера шрифта толщина границы указывается в единицах, равных одной восьмой пункта. Например, 24 означает границу толщиной 3 пункта. Причина различий в том, что обеспечивается использование только целых чисел. Таким образом устанавливается нижний предел толщины границы, а верхний также ограничен спецификацией и равен 12 пунктам, что соответствует значению 96.

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:bottom w:val="single" w:sz="4" w:color="auto" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Stephen Jiang</w:t>
  </w:r>
</w:p>
```

### Пример 12. Применение свойств абзаца

Рассмотрим еще некоторые параметры форматирования. Абзац на следующем рисунке имеет отступ с обеих сторон, выровнен по центру и имеет нижнюю границу.

#### Текст с отступом

Здесь используется три параметра форматирования, один из которых, границу, мы уже обсудили. Выравнивание по центру описывается элементом выравнивания *jc*, а для формирования отступа служит элемент *ind*.

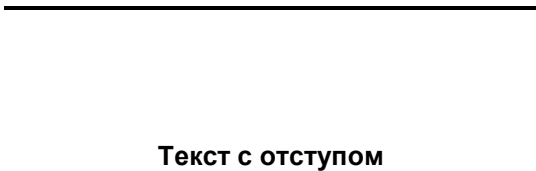
Следующий рисунок демонстрирует изменения внешнего вида абзаца при последовательном применении атрибутов. Сначала показан неформатированный текст, затем появляется нижняя граница, производится выравнивание и добавляются отступы.

#### Текст с отступом

#### Текст с отступом

```
<w:pPr>
</w:pPr>

<w:pPr>
  <w:pBdr>
```



Текст с отступом

Текст с отступом

```

<w:bottom w:val="single" w:sz="12"
w:color="auto" />
</w:pPr>

<w:pPr>
  <w:pBdr>
    <w:bottom w:val="single" w:sz="12"
w:color="auto" />
    <w:jc w:val="center" />
  </w:pPr>

<w:pPr>
  <w:pBdr>
    <w:bottom w:val="single" w:sz="12"
w:color="auto" />
    <w:jc w:val="center" />
    <w:ind w:left="2835" w:right="2835" />
  </w:pPr>

```

### Различные типы разрывов

В нашем документе в двух местах есть разрывы. Абзац, содержащий имя и адрес электронной почты, сформатирован с применением «мягкого» разрыва, разделяющего строку на две. Кроме того, можно использовать разрыв страницы. Если хотите начать следующий отчет о продажах на чистой странице, добавьте внутрь области элемент *br*. Текст, следующий за элементом *br*, начнется на новой странице. Атрибут *type* этого элемента позволяет разбивать содержимое на столбцы. При разбивке документа на разделы элемент *br* не применяется. Создание разделов обсуждается далее в этой главе.

Тип разрыва	Код
Строка	<pre> &lt;w:r&gt;   &lt;w:cr /&gt; &lt;/w:r&gt;  &lt;w:r&gt;   &lt;w:br w:type="page" /&gt; </pre>
Страница	<pre> &lt;/w:r&gt; </pre>

Таблица 1. Типы разрывов

Мы научились создавать базовый текстовый документ — наполним его более сложным содержимым. В документах WordprocessingML могут быть различные блочные и встроенные элементы, скажем, таблицы, в которых используются объекты модели WordprocessingML и различные типы содержимого, описываемого языком DrawingML, например графики и диаграммы.

### Таблицы

Таблицы, как и абзацы — важные строительные блоки документов. Таблицы — это блочные элементы (*tbl*), состоящие из строк и ячеек, подобно таблицам HTML. Строки определяются элементами *tr*, которые содержат элементы ячеек *tc*. Ячейки таблиц являются контейнерами для блочного содержимого, в первую очередь для абзацев.

Ниже вы видите таблицу из двух строк с тремя ячейками и пример кода, которым она описывается. Разметка требует некоторой доработки: прежде всего в ней нужно определить сетку.


Рис. 10. Таблица 3 × 2

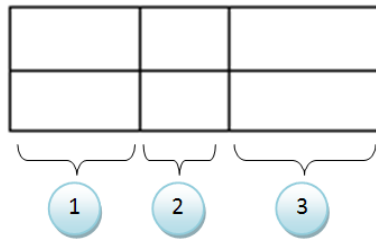
```

<w:tbl>
  <w:tblGrid />
  <w:tr>
    <w:tc>...</w:tc>
    <w:tc>...</w:tc>
    <w:tc>...</w:tc>
  </w:tr>
  <w:tr>
    <w:tc>...</w:tc>
    <w:tc>...</w:tc>
    <w:tc>...</w:tc>
  </w:tr>
</w:tbl>

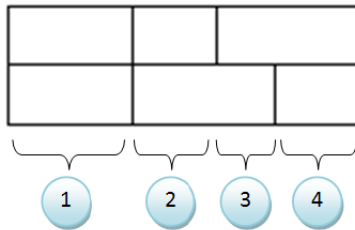
```

### Пример 13. Структура таблицы

При создании таблицы прежде всего надо создать определение ее сетки, которое содержит параметры столбцов таблицы. Каждый столбец описывается элементом внутри определения сетки. Демонстрационная таблица состоит из трех столбцов:



Интересный эффект будет при смещении границы ячеек: размеры этих ячеек в соответствующем столбце изменятся:



Теперь в определении сетки не три, а четыре столбца, т. е. не только «видимые» столбцы. Чтобы создать определение сетки, нужно продлить границы всех ячеек. Каждая из этих линий определяет границу столбца; накладывающиеся линии не учитываются. Таким образом, в нашем примере четыре столбца, хотя в каждой строке видны только три ячейки.

В примерах 14 и 15 показаны определения сетки до и после смещения ячеек. Обратите внимание на суммарную ширину — она осталась неизменной.

```
<w:tbl>
  <w:tblGrid>
    <w:gridCol w:w="5000" />
    <w:gridCol w:w="3000" />
    <w:gridCol w:w="7000" />
  </w:tblGrid>
  <!-- Определения прочих параметров
таблицы -->
</w:tbl>
```

Пример 14. Сетка таблицы до смещения

```
<w:tbl>
  <w:tblGrid>
    <w:gridCol w:w="5000" />
    <w:gridCol w:w="3000" />
    <w:gridCol w:w="2500" />
    <w:gridCol w:w="4500" />
  </w:tblGrid>
  <!-- Определения прочих параметров
таблицы -->
</w:tbl>
```

Пример 15. Сетка таблицы после смещения

Сетку описывает элемент *tblGrid*. Кроме определения столбцов таблицы, он служит для хранения стандартных значений ширины ячеек. Реальные значения ширины ячеек указываются в описывающих их элементах.

В нашем отчете есть таблица из двух столбцов, ширина которых меняется автоматически в зависимости от содержимого (рис. 11 и пример 16).

Country	Sales
Australia	\$0.00
Canada	\$135,342.50
Central	\$0.00
France	\$0.00
Germany	\$0.00
Northeast	\$0.00
Northwest	\$271,341.43
Southeast	\$6,339.34
Southwest	\$197,857.75
United Kingdom	\$0.00

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="0" w:type="auto" />
  </w:tblPr>
  <w:tblGrid>
    <w:gridCol w:w="1614" />
    <w:gridCol w:w="1330" />
  </w:tblGrid>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:w="1614" w:type="dxa" />
      </w:tcPr>
    <w:p>
      <w:r>
        <w:t>Country</w:t>
      </w:r>
    </w:tc>
  </w:tr>
```

Рис. 11. Таблица из демонстрационного отчета

```

</w:p>
</w:tc>
<w:tc>
  <w:tcPr>
    <w:tcW w:w="0" w:type="auto" />
  </w:tcPr>
  <w:p>
    <w:r>
      <w:t>Sales</w:t>
    </w:r>
  </w:p>
</w:tc>
</w:tr>
<w:tr>
  <!--Строки с данными опущены -->
</w:tr>
</w:tbl>

```

Пример 16. Структура демонстрационной таблицы

Ширина таблицы определяется внутри узла свойств таблицы — *tblPr*. В примере задается автоматическое определение размера в соответствии с шириной ячеек таблицы, которая определяется в свойствах ячеек, представленных элементами *tcPr*. Единицей измерения размера является одна двенадцатая часть пункта (*twip*), а на то, что измеряется именно ширина, указывает значение *dx*. Ширина второй ячейки определяется автоматически, на что указывает значение типа *auto*. Существует несколько режимов определения размера. Возможные конфликты между параметрами уровня таблицы и ячейки разрешаются отображающей программой, которая должна вывести все содержимое.

### Границы и заливка ячеек

Открыв документ с таблицей, которую мы привели в примере, вы не поймете, что это таблица. Поскольку границы не отображаются автоматически, вы их не увидите — их нужно определить или на уровне свойств таблицы, или ячеек. Можно определить до восьми типов границ: верхнюю, нижнюю, правую и левую, горизонтальную и вертикальную внутренние границы, а также две диагональных. Для каждой границы указываются такие атрибуты, как тип (одинарная, двойная), цвет и толщина. Эти определения хранятся в контейнере границы: *tblBorders* (для таблицы) или *tcBorders* (для ячейки). Толщина границы указывается в восьмых долях пункта. Допустимый диапазон значений — от 2 до 96.

В демонстрационной таблице заданы верхняя и нижняя границы всего содержимого, а также нижняя граница первой строки. Поскольку границу нельзя установить для строки, граница первой строки относится к двум ячейкам. Добавьте в описание таблицы определения границ в свойствах таблицы и ячейки, показанные в примере 17.

```

<w:tblPr>
  <w:tblBorders>
    <w:top w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
    <w:bottom w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
  </w:tblBorders>
</w:tblPr>
<w:tcPr>
  <w:tcBorders>
    <w:top w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
    <w:left w:val="nil" />
    <w:bottom w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
    <w:right w:val="nil" />
    <w:insideH w:val="nil" />
    <w:insideV w:val="nil" />
  </w:tcBorders>
</w:tcPr>

```

Пример 17. Границы таблицы и ячейки

Рассмотрим одну интересную особенность определения границ в элементе *tcBorders*. На уровне таблицы могут быть определены границы для всех ячеек, но они переопределяются на уровне ячеек, например, значением *nil*. Это общий для Open XML подход к переопределению; он применяется, в частности, при форматировании с использованием стилей.

Еще один прием, позволяющий улучшить внешний вид отчета, — эффект чередования, который достигается при заливке нечетных строк (без учета заголовка). Заливка определяется на уровне ячейки с помощью элемента *shd*, который надо добавить к свойствам всех ячеек в нечетных строках (пример 18). Приготовьтесь к многочисленным операциям копирования/вставки, без которых мы в дальнейшем обойдемся благодаря применению стилей таблиц, имеющих встроенную поддержку этого эффекта.

```
<w:tcPr>
  <w:shd w:val="clear" w:color="auto" w:fill="D2EAF1" />
</w:tcPr>
```

Пример 18. Заливка ячеек таблицы

## Применение стилей в документе

Следующий этап в создании профессионально выглядящего документа — применение стилей. До сих пор формат нашего демонстрационного отчета определялся параметрами непосредственных форматирующих элементов в различных узлах свойств, таких как *rPr*, *pPr*, *tblPr* и *tcPr*. Если нам нужно было применить к одному фрагменту формат другого, мы могли просто скопировать некоторые параметры из соответствующего элемента. Непосредственное форматирование не обеспечивает повторного применения форматов и простого их изменения. Так, для изменения эффекта чередования строк таблицы нужно вручную изменить добрый десяток ячеек. От подобной неэффективной работы вас избавят стили.



В демонстрационном отчете некоторые параметры форматирования применяются многократно. В этом разделе мы их воссоздадим, применив стили.

## Компонент стилей

Стиль является набором определенных значений параметров форматирования, который можно применить как единое целое к абзацу, области или таблице. Стили хранятся в отдельной части пакета WordprocessingML, которая называется компонентом стилей. Он содержит XML-код, соответствующий стандарту WordprocessingML, и имеет особый тип содержимого. Чтобы создать демонстрационный документ, описываемый в этом разделе, мы применим следующий тип содержимого для хранения нового компонента пакета.

*Тип содержимого для стилей:*

*application/vnd.openxmlformats-officedocument.wordprocessingml.styles+xml*

Откройте пакет и добавьте в любой каталог новый файл *styles.xml* и добавьте в компонент типов содержимого новый тип, переопределив расширение XML-файла. Не допускайте разрывов в строке со значением атрибута *ContentType*.

```
<Override PartName="/styles.xml"
  ContentType="application/vnd.openxmlformats-...
  ...officedocument.wordprocessingml.styles+xml"/>
```

Пример 19. Изменение компонента типов содержимого

Компонент стилей связан с основным компонентом документа. Тип этой связи также специфичен для компонента стилей.

*Тип связи для стилей:*

*http://schemas.openxmlformats.org/wordprocessingml/2006/styles*

Вам нужно определить связь между основным компонентом документа и компонентом стилей. Связи, устанавливаемые определенным компонентом, можно хранить в файле связей, относящемся к данному компоненту. Эти файлы нужно хранить в подкаталоге *\_rels* каталога, в котором хранится компонент. Имя файла связей совпадает с именем данного компонента, и к нему добавляется расширение *.rels*. Если основной компонент документа называется *document.xml* и хранится в корневом каталоге пакета, его файл связей — *\_rels\document.xml.rels*. Поскольку основной компонент документа Microsoft Office Word хранится в папке *word*, соответствующие ему связи хранятся в *word\\_rels\document.xml.rels*.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
  xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  <Relationship Id="rId1"
    Type="http://schemas.openxmlformats.org/officeDocument/...
    ...2006/relationships/styles" Target="styles.xml" />
</Relationships>
```

Пример 20. Файл связей документа

## Типы стилей

В компоненте стилей хранятся три разновидности элементов: стили, скрытые стили (*latent-styles*) и стандартные стили документа (*document defaults*). Все они могут определять параметры форматирования содержимого документа. Стили определяют параметры форматирования, применяемые в документе в данный момент. Скрытые стили не используются и не видны в документе и формируют своего рода кэш



стилей: к ним, например, относятся стили, скопированные из шаблона документа. Обычно разработчики могут игнорировать эти стили. Стандартные стили документа — это стили, используемые в документе по умолчанию.



Стили позволяют задавать параметры форматирования для элементов трех типов: областей текста, абзацев и таблиц. Стили абзаца могут определять формат как абзацев, так и областей, т. е. применяться в элементах *pPr* и *rPr*. Стил символа может быть применен только на уровне свойств области (*rPr*). Стили таблиц могут быть указаны в свойствах таблиц и ячеек (*tblPr* и *tcPr*), а также в свойствах абзацев и областей.

```
<w:styles
  xmlns:w="http://.../wordprocessingml/2006/main">
  <w:docDefaults />
  <w:latentStyles />
  <w:style>...</w:style>
  <w:style>...</w:style>
  <w:style>...</w:style>
</w:styles>
```

Пример 21. Содержимое компонента стилей

Прежде чем перейти к обсуждению иерархии стилей и способов форматирования демонстрационного отчета, рассмотрим еще один аспект стилей. Некоторые из них можно применять как к абзацам, так и к областям текста. Вы можете проверить это, открыв новый документ и введя какой-либо текст. Если вы ничего не выделите, а просто установите курсор внутри текста и выберете какой-нибудь стиль в Microsoft Office Word 2007, изменится стиль всего абзаца. Если же вы выделите некоторый фрагмент и выберете другой стиль, он будет применен только к выделенному фрагменту. Хотя может показаться, что и к абзацу, и к области применяется один и тот же стиль, на самом деле стили абзаца и области — разные сущности.

Для применения к элементу стиля используются узлы свойств. Для определения стиля абзаца используется *pStyle*. В примере 22 к абзацу применен стиль Title (Название).

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Title" />
  </w:pPr>
  <w:r>
    <w:t>Stephen Jiang</w:t>
  </w:r>
</w:p>
```

Пример 22. Абзац со стилем Title

## Иерархия стилей

Несколько сложнее стили таблиц, которые могут включать параметры форматирования символов и абзацев. Стили также могут наследовать другие стили, формируя иерархии. Каждый уровень такой иерархии определяет одну из частей конечного представления документа и может переопределять параметры нижних уровней иерархии. Далее вы узнаете, в каком порядке определенные параметры форматирования применяются к таблицам, областям, абзацам и нумерованным спискам. Верхние уровни (табл. 2) применяются первыми (стандартные стили документа), а нижние — последними (непосредственное форматирование). Каждый последующий уровень может расширять и переопределять параметры предыдущего (верхнего) уровня.

Таблицы	Символы	Абзацы	Элементы списка
	Стандартные параметры документа		
Таблица			
		Нумерация	
	Абзац		
	Символ		
Прямое форматирование			

## Таблица 2. Иерархия стилей содержимого документа

Рассмотрим второй столбец этой таблицы — применение стилей к областям символов (рис. 12).

```
<tbl style="MyTable" >
  <row>
    <cell style="MyCell">
      <paragraph style="MyParagraph">
        <run style="MyRun">
```

Рис. 12. Применение стилей к области

Отображающая программа должна решить, как отобразить эту область символов. Прежде всего ей известны стили, используемые по умолчанию. Поскольку область — это часть таблицы, к которой применены стили, программа ищет в определении стиля *MyTable* другие свойства уровня области, соответствующие ячейке таблицы (*MyCell*). Область также находится в абзаце, к которому применены стили. Стиль этого абзаца в свою очередь может иметь свойства уровня области, применяемые к нашей области. И, наконец, для самой области определен стиль. Непосредственное форматирование, применяемое последним, в примере не представлено.

Возможные конфликты разрешаются двумя способами. Любое свойство, определенное на родительском (более высоком) уровне, переопределяется дочерним уровнем (более низким). На дочернем свойство может быть или переопределено (скажем, изменен размер шрифта), или сброшено. В случае сброса значение свойства наследуется от предка. Если в родительском стиле свойство тоже не определено, поиск возможного значения производится вверх по иерархии — вплоть до корневого стиля, которым является стандартное значение стиля документа. Если в документе не заданы никакие параметры, приложение применяет собственные параметры стилей по умолчанию.

---

*Стили также помогают переносить форматирование из одного документа в другой. Вы можете скопировать многие параметры форматирования, полностью изменяя внешний вид документа. Так, можно полностью изменить устаревший корпоративный стиль.*

*Решая подобные задачи, всегда используйте встроенные имена стилей и стройте иерархии со стилем Normal (Обычный) в качестве корневого. Это будет страховкой от потери стилей при изменении их параметров.*

*Пользовательские стили также должны порождаться из встроенных, чтобы обеспечить переносимость.*

---

## Стандартные стили документа

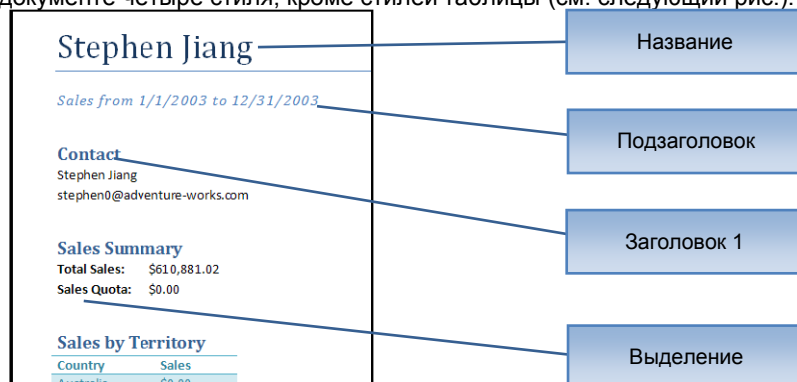
Прежде всего вполне очевидно использовать некоторые стандартные параметры для шрифта. В компоненте стилей можно хранить стандартные значения двух типов — для символов и абзацев. Эти стандартные значения применяются в тех же элементах свойств, что и прямое форматирование: *pPr* (для абзацев) и *rPr* (для символов). Они содержатся в специальных контейнерах: *pPrDefault* и *rPrDefault*. В примере 23 показано определение шрифта по умолчанию для демонстрационного отчета. Здесь используется уже знакомый элемент *rFonts*, который позволяет задать четыре шрифта в соответствии с диапазонами Unicode-кодов символов, а также элемент *sz*, определяющий размер в половинных долях пункта. Кроме того, в этом примере задается интервал между абзацами. Это значение хранится в свойстве уровня абзаца, в элементе *spacing*. Его атрибуты *before* и *after* позволяют уточнять, какой именно интервал определяется — до или после абзаца. Интервал измеряется в двенадцатых долях пункта, которые называют *twip*.

```
<w:styles xmlns:w="http://.../wordprocessingml/2006/main">
  <w:docDefaults>
    <w:rPrDefault>
      <w:rPr>
        <w:rFonts w:ascii="Calibri" />
        <w:sz w:val="22" />
      </w:rPr>
    </w:rPrDefault>
    <w:pPrDefault>
      <w:pPr>
        <w:spacing w:after="120" />
      </w:pPr>
    </w:pPrDefault>
  </w:docDefaults>
</w:styles>
```

Пример 23. Стандартные параметры стиля

## Создание стилей

Несколько параметров форматирования нашего отчета мы применяли многократно, копируя фрагменты разметки. Более эффективный способ многократного применения форматирования — использование стилей. В нашем документе четыре стиля, кроме стилей таблицы (см. следующий рис.).



Форматирование документа с применением стилей осуществляется так: в компонент стилей нужно добавить описание стилей, которые вы хотите использовать, а затем ссылаться на эти стили в теле документа. Распространенными являются такие общеизвестные стили, как *Заголовок 1 (Heading 1)*, *Заголовок 2 (Heading 2)* и пр. Допустимы и пользовательские стили. Преимущество употребления встроенных имен в том, что они могут применяться для таких типов содержимого, как оглавления, которые формируются на основе абзацев определенного стиля.

Стиль определяется с помощью элемента *style* прямо в корневом элементе компонента стилей. Здесь можно хранить множество стилей. Каждый из них состоит из трех частей: общие свойства, тип и свойства, связанные с типом. К общим свойствам относятся имя и родительский стиль. Обычно в качестве корня иерархии используется стиль *Обычный (Normal)*.

## Стили абзаца

В нашем демонстрационном документе применяются три стиля абзацев: для названия документа, его подзаголовка и заголовков разделов. Используемые для них встроенные имена обеспечивают интерфейс пользователя, согласованный с отображающей программой, и возможность создания оглавления.

Сначала создадим стиль для названия. Проще всего для этого открыть компонент стилей, добавить в него глобальное определение стиля, а затем перенести свойства абзаца и области из основного документа в этот компонент. При этом текст названия отчета останется неформатированным, пока мы не применим к нему созданный стиль. Код примера 24 можно скопировать в компонент стилей, чтобы получить стиль названия.

```
<w:style w:type="paragraph" w:styleId="Title">
  <w:name w:val="Title" />
  <w:next w:val="Normal" />
  <w:basedOn w:val="Normal" />
  <w:qFormat />
</w:style>
```

### Пример 24. Основные свойства стиля

Из кода видно, что это стиль абзаца. Свойства абзаца и области будут применяться ко всему тексту абзаца. Атрибут *styleId* определяет идентификатор стиля, по которому на него будут ссылаться. В элементе *name* определяется удобное для восприятия пользователем название стиля. В стилях абзацев могут быть специфические элементы *next*. В них указываются стили, которые должны быть у абзацев, вставляемых за абзацами данного стиля. Далее в описании стиля указывается, что он унаследован от обычного стиля, а элемент *qFormat* указывает, что данный стиль нужно отображать в окне быстрого форматирования клиентской программы.

В стиле абзаца могут быть свойства уровня абзаца (пример 25).

```
<w:pPr>
  <w:pBdr>
    <w:bottom w:val="single" w:sz="4"
      w:space="1" w:color="auto" />
  </w:pBdr>
  <w:spacing w:after="200" />
  <w:keepNext />
  <w:keepLines />
  <w:spacing w:before="480" w:after="0" />
  <w:outlineLvl w:val="0" />
</w:pPr>
```

### Пример 25. Свойства уровня абзаца в стиле абзаца

Свойства уровня области также могут быть в стиле абзаца (пример 26).

```
<w:rPr>
  <w:rFonts w:ascii="Cambria" />
  <w:spacing w:val="5" />
  <w:sz w:val="52" />
</w:rPr>
```

Пример 26. Свойства уровня области в стиле абзаца

Полностью определение стиля таково (пример 27):

```
<w:style w:type="paragraph" w:styleId="Title">
  <w:name w:val="Title" />
  <w:next w:val="Normal" />
  <w:basedOn w:val="Normal" />
  <w:qFormat />
  <w:pPr>
    <w:pBdr>
      <w:bottom w:val="single" w:sz="4"
        w:space="1" w:color="auto" />
    </w:pBdr>
    <w:spacing w:after="200" />
    <w:keepLines />
    <w:spacing w:before="480" w:after="0" />
    <w:outlineLvl w:val="0" />
  </w:pPr>
  <w:rPr>
    <w:rFonts w:ascii="Cambria" />
    <w:spacing w:val="5" />
    <w:sz w:val="52" />
  </w:rPr>
</w:style>
```

Пример 27. Стиль абзаца для названия документа

Осталось создать еще три стиля абзаца. На обычный стиль ссылается элемент *next*. Определение стиля для подзаголовка и заголовка первого уровня можно скопировать из основного документа.

## Стили символов

Текст «Sales Summary» и «шапка» таблицы выделяются полужирным шрифтом. Для этого можно применить стиль символов. В отличие от стилей абзаца стили символа содержат только свойства уровня области, определяются же они аналогично стилям абзаца — с указанием типа, идентификатора и имени. Остальное можно скопировать из основного документа.

```
<w:style w:type="character" w:styleId="Emphasis">
  <w:name w:val="Emphasis" />
  <w:qFormat />
  <w:rPr>
    <w:b />
  </w:rPr>
</w:style>
```

Пример 28 Стиль символов

Применяются стили символов аналогично стилям абзацев: элемент *rStyle* в свойстве уровня области указывает на используемый в ней стиль символов.

---

*Параметры форматирования текста содержатся только в свойствах,  
применяемых непосредственно с помощью элемента rPr.  
Свойства области внутри свойств абзаца определяют формат знака абзаца.*

---

В примере 29 показан абзац с примененным стилем. Абзац состоит из двух областей, к первой из которых также применен стиль. На рисунке рядом с примером показан один из возможных результатов такой разметки.

```

<w:p>
  <w:pPr>
    <w:pStyle w:val="Normal" />
  </w:pPr>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="MyCharacterStyle" />
    </w:rPr>
    <w:t>Hello</w:t>
  </w:r>
  <w:r xml:space="preserve">
    <w:t> World</w:t>
  </w:r>
</w:p>

```



Пример 29. Применение стилей абзаца и символов

## Стили таблиц

Прежде чем переходить к обзору разметки страницы, остановимся подробнее на стилях таблиц. Эти стили обеспечивают хранение не только набора информации о свойствах таблицы и ее ячеек. WordprocessingML позволяет выделять такие структурные элементы, как верхняя и нижняя строки, первый и последний столбец, а также чередующиеся строки. Эффект чередования строк заключается в том, что четные и нечетные строки имеют разный формат, что облегчает чтение таблицы (см. следующий рисунок).

	Price	Cost
Item 1	14,95	5,65
Item 2	13,92	3,14
Item 3	12,65	9,30
Item 4	16,00	10,00

В таблице демонстрационного отчета выделяется заглавная строка, а также задан эффект чередования строк данных. Создается стиль таблицы аналогично прочим: добавляется элемент `style` с типом `table`, внутрь которого помещаются описания свойств уровня таблицы, ячейки, абзаца и области.

Сначала определите структуру таблицы, а затем добавьте те же элементы, что и в определениях прочих типов стилей.

	Price	Cost
Item 1	14,95	5,65
Item 2	13,92	3,14
Item 3	12,65	9,30
Item 4	16,00	10,00

```

<w:style w:type="table"
w:styleId="ReportTable">
  <w:name w:val="My Table Style" />
  <w:qFormat />
  <!-- Прочие параметры -->
</w:style>

```

Пример 30. Стиль таблицы с эффектом чередования строк

Внутри элемента стиля таблицы можно добавлять стили для отдельных ее фрагментов. В нашем примере свои стили имеют первая строка, первый столбец и каждая четная строка. Сначала зададим стандартные значения для абзацев и областей внутри таблицы с помощью элементов `pPr` и `rPr`. На следующем рисунке показан результат добавления этих свойств. Код примера 31 добавляется внутрь элемента `style` вслед за комментарием в примере 30.

	Price	Cost
Item 1	14,95	5,65
Item 2	13,92	3,14
Item 3	12,65	9,30
Item 4	16,00	10,00

```

<w:pPr>
  <w:spacing w:after="0" />
</w:pPr>
<w:rPr>
  <w:color w:val="31849B" />
</w:rPr>

```

Пример 31. Стиль таблицы с эффектом чередования строк

Теперь добавим границы. Поскольку они обрамляют всю таблицу сверху и снизу, их можно определить в узле свойств уровня таблицы, `tblPr`.

	Price	Cost
Item 1	14.95	5.65

```

<w:tblPr>
  <w:tblBorders>

```

```

<w:top w:val="single" w:sz="8"
      w:color="4BACC6" />
<w:bottom w:val="single" w:sz="8"
         w:color="4BACC6" />
</w:tblBorders>
</w:tblPr>

```

Пример 32. Стиль таблицы с эффектом чередования строк

Верхняя строка имеет особый стиль: у нее есть нижняя граница, задан другой цвет фона и полужирный текст белого цвета. Для хранения этих параметров создается новый контейнер внутри определения стиля. Внутри этого контейнера будут все те же узлы *pPr*, *rPr* и *tcPr* (пример 33).

	Price	Cost
Item 1	14,95	5,65
Item 2	13,92	3,14
Item 3	12,65	9,30
Item 4	16,00	10,00

```

<w:tblStylePr w:type="firstRow">
  <w:rPr>
    <w:b />
    <w:color w:val="FFFFFF" />
  </w:rPr>
  <w:tcPr>
    <w:tblBorders>
      <w:bottom w:val="single"
                w:sz="8"
                w:space="0"
                w:color="4BACC6" />
    </w:tblBorders>
    <w:shd w:val="clear"
           w:color="auto"
           w:fill="D2EAF1" />
  </w:tcPr>
</w:tblStylePr>

```

Пример 33. Стиль таблицы с эффектом чередования строк

Контейнерный элемент для первого столбца напоминает контейнер для первой строки. Лишь атрибут *type* элемента *tblStylePr* указывает другой фрагмент таблицы и не заданы границы. Их параметры наследуются от параметров всей таблицы и первой строки.

	Price	Cost
Item 1	14,95	5,65
Item 2	13,92	3,14
Item 3	12,65	9,30
Item 4	16,00	10,00

```

<w:tblStylePr w:type="firstColumn">
  <w:rPr>
    <w:b />
    <w:color w:val="FFFFFF" />
  </w:rPr>
  <w:tcPr>
    <w:shd w:val="clear"
           w:color="auto"
           w:fill="D2EAF1" />
  </w:tcPr>
</w:tblStylePr>

```

Пример 34. Свойства первого столбца

В завершение реализуем эффект чередования строк (пример 35).

	Price	Cost
Item 1	14,95	5,65
Item 2	13,92	3,14
Item 3	12,65	9,30
Item 4	16,00	10,00

```

<w:tblStylePr w:type="band1Horz">
  <w:tcPr>
    <w:shd w:val="clear" w:color="auto"
           w:fill="D2EAF1" />
  </w:tcPr>
</w:tblStylePr>

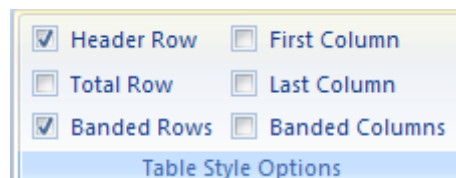
```

Вот окончательное описание стиля (пример 36):

```
<w:style w:type="table"
  w:styleId="ReportTable">
  <w:name w:val="My Table Style" />
  <w:qFormat />
  <w:pPr>
    <w:spacing w:after="0" />
  </w:pPr>
  <w:rPr>
    <w:color w:val="31849B" />
  </w:rPr>
  <w:tblPr>
    <w:tblBorders>
      <w:top w:val="single" w:sz="8" w:color="4BACC6" />
      <w:bottom w:val="single" w:sz="8" w:color="4BACC6" />
    </w:tblBorders>
  </w:tblPr>
  <w:tblStylePr w:type="firstRow">
    <w:rPr>
      <w:b />
      <w:color w:val="FFFFFF" />
    </w:rPr>
    <w:tcPr>
      <w:tcBorders>
        <w:bottom w:val="single" w:sz="8" w:space="0" w:color="4BACC6" />
      </w:tcBorders>
      <w:shd w:val="clear" w:color="auto" w:fill="D2EAF1"/>
    </w:tcPr>
  </w:tblStylePr>
  <w:tblStylePr w:type="firstColumn">
    <w:rPr>
      <w:b />
      <w:color w:val="FFFFFF" />
    </w:rPr>
    <w:tcPr>
      <w:shd w:val="clear" w:color="auto" w:fill="D2EAF1"/>
    </w:tcPr>
  </w:tblStylePr>
  <w:tblStylePr w:type="band1Horz">
    <w:tcPr>
      <w:shd w:val="clear" w:color="auto" w:fill="D2EAF1"/>
    </w:tcPr>
  </w:tblStylePr>
</w:style>
```

#### Пример 36. Окончательный стиль таблицы

Стиль к таблице задается элементом *tblStyle*. При этом могут применяться стили не всех разделов. В определении стиля могут быть заданы параметры всех разделов таблицы, но отображающая программа должна быть проинформирована, какие из них применять к текущей таблице. Это делается с помощью элемента *tblLook*, в котором указывается битовая маска, определяющая к каким разделам применять стили. Например, значение *0420* соответствует первой строке и чередованию строк (пример 37).



```
<w:tbl>
  <w:tblPr>
    <w:tblStyle w:val="ReportTable" />
    <w:tblLook w:val="0420" />
    <w:tblW w:w="0" w:type="auto" />
  </w:tblPr>
  <!-- Прочий код описания таблицы пропущен -->
</w:tbl>
```

#### Пример 37. Применение стиля таблицы

## Добавление изображений

В документ WordprocessingML можно встраивать содержимое разных типов. Картинки и другие графические элементы можно создавать с помощью языка DrawingML или по-прежнему применяемого языка векторной разметки (Vector Markup Language, VML). Предпочтительнее DrawingML, поскольку он предлагает больше возможностей и не привязан к пространствам имен, относящимся к платформам Microsoft. Впрочем, окончательный выбор будет зависеть от решаемой задачи. Основное достоинство VML для разработчиков — это минимальный объем кода, необходимый для хранения изображений в документах. Код DrawingML гораздо объемнее; этому языку посвящена отдельная глава этой книги.

Чтобы изображение можно было увидеть в документе, его надо сохранить в пакете. Как и в случае добавления стилей, вам потребуется создать файл в пакете, изменить список типов содержимого (добавить обычный MIME-тип, например, *image/png*) и создать связь между основным документом и изображением.



Тип связи для изображений:

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/image>

## Вставка рисунка DrawingML

Вставив картинку в документ WordprocessingML с помощью такой программы, как Microsoft Word, вы обнаружите, что к ней можно применить различные эффекты. Это же относится к схемам и диаграммам. На самом деле это содержимое документов описывается не языком WordprocessingML, а с помощью DrawingML. Для вставки подобного информационного наполнения в WordprocessingML предусмотрен специальный контейнер — графическая область (graphic frame).

В WordprocessingML для создания графической области служит элемент *drawing*. Этот позиционирующий элемент позволяет рисовать внутри документа. Содержимое графической области определяется в элементе *graphic*. Структура *graphicData* описывает графические данные. Эта структура во всех языках Open XML служит для хранения содержимого различного типа. Тип содержимого задается атрибутом *uri* структуры *graphicData*. Реальное содержимое должно соответствовать этому типу, чтобы клиент Open XML мог нормально его визуализировать.

Особенности рисунков, схем и диаграмм обсуждаются в главе, посвященной DrawingML. Вот некоторый минимально необходимый код для отображения рисунка в демонстрационном отчете (пример 38):



```
<w:drawing>
  <wp:inline distT="0" distB="0" distL="0" distR="0">
    <wp:extent cx="5943600" cy="2003354" />
    <wp:docPr id="1" name="Picture 1" />
    <a:graphic xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main">
      <a:graphicData
        uri="http://schemas.openxmlformats.org/drawingml/2006/picture">
        <pic:pic
          xmlns:pic="http://schemas.openxmlformats.org/drawingml/2006/picture">
          <pic:nvPicPr>
            <pic:cNvPr id="0" name="Picture 1" />
            <pic:cNvPicPr />
          </pic:nvPicPr>
          <pic:blipFill>
            <a:blip r:embed="rId2" />
            <a:stretch>
              <a:fillRect />
            </a:stretch>
          </pic:blipFill>
          <pic:spPr>
            <a:xfrm>
              <a:off x="0" y="0" />
              <a:ext cx="5943600" cy="2003354" />
            </a:xfrm>
            <a:prstGeom prst="rect">
              <a:avLst />
            </a:prstGeom>
          </pic:spPr>
        </a:graphicData>
      </a:graphic>
    </wp:inline>
  </w:drawing>
```



```

    </pic:pic>
  </a:graphicData>
</a:graphic>
</wp:inline>
</w:drawing>

```

### Пример 38. Применение DrawingML для добавления рисунка в отчет

Хотя в целом работа с изображениями довольно сложна, многие вещи реализовать очень просто — хватает стандартных параметров. Особенно развиты возможности DrawingML по усовершенствованию изображений.



Если ваш рисунок не слишком выразителен, достаточно добавить пару элементов — скажем, изменяющий форму и создающий отражение, — и он станет гораздо интересней.



```

<a:prstGeom prst="star5">
  <a:avLst />
</a:prstGeom>
<a:effectLst>
  <a:reflection blurRad="6350"
    stA="50000"
    endA="300"
    endPos="55000"
    dir="5400000"
    sy="-100000"
    algn="bl"
    rotWithShape="0" />
</a:effectLst>

```

Подробнее возможные эффекты описаны в главе 4.

### Вставка рисунка VML

Хотя VML не связан со стандартом Open XML, VML-изображения применяются документах WordprocessingML (наряду со многими другими форматами). Набор возможностей этого языка сравним с DrawingML, но последний предоставляет больше эффектов. Как и в языке DrawingML, можно изменить геометрию фигуры, добавить тени или перейти в трехмерное пространство. Можно управлять линиями и заливкой. В примере для заливки применяется картинка, но вы можете задать один цвет или градиент, а также текстуру. У VML есть одно явное преимущество: описание в документе VML-изображения требует гораздо меньше кода, чем для изображения DrawingML. В примере 39 показано, как добавить в документ фигуру, отображаемую в документе в виде стандартного рисунка. Для этого выбирается прямоугольная форма изображения (элемент *rect*), а в качестве заливки указывается картинка, встроенная в пакет. На картинку указывает связь с соответствующим идентификатором.



```

<w:p>
  <w:r>
    <w:pict>
      <v:rect style="width:16.56cm;height:5.36cm;"
        stroked="f">
        <v:fill r:id="rId2" type="frame" />
      </v:rect>
    </w:pict>
  </w:r>
</w:p>

```

### Пример 39. Добавление в отчет VML-изображения

Вы видите, что объем VML-кода гораздо меньше эквивалента на DrawingML. И все же я предпочитаю работать с последним, так как он имеет более мощные изобразительные возможности и эффективней обрабатывается анализаторами. Синтаксический анализ VML-кода требует разборки текстовых строк, а в DrawingML применяется стандартный XML, который обрабатывается гораздо эффективней.

Мы закончили создание демонстрационного отчета, добавив в него картинку. Теперь у нас есть все необходимые элементы: текст, таблица и рисунок. Следующий этап — разметка страницы и добавление дополнительного содержимого.

## Макет страницы

Итак, мы познакомились с азами форматирования документов WordprocessingML — можно перейти к изучению структуры страницы. Прежде всего рассмотрим разделы и колонтитулы, а затем поля, которые позволяют связывать с колонтитулами данные, например формировать номера страниц.

## Создание разделов

Разделы — это высокоуровневые строительные блоки вашего документа. Обычно титульную страницу документа, его тело, а также приложения помещают в отдельные разделы. Делается это потому, что на уровне раздела можно, например, нумеровать страницы или определять колонтитулы. Ориентация страниц также определяется для раздела.

Раздел может быть создан на двух уровнях: внутри абзаца и в теле документа. В первом случае сам абзац и все предыдущие становятся частью данного раздела вплоть до следующего абзаца, в котором определяется другой раздел. Определение раздела на уровне документа должно описывать последний раздел в документе и быть последним элементом в теле документа (контейнере *body*).

Ниже демонстрируется простейшее определение двух разделов со стандартными параметрами (поскольку у элемента *sectPr* нет дочерних):

```
<w:body>
  <w:p>
    <w:r>
      <w:t>First</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:pPr>
      <w:sectPr />
    </w:pPr>
  </w:p>
  <w:p>
    <w:r>
      <w:t>Second</w:t>
    </w:r>
  </w:p>
  <w:sectPr />
</w:body>
```

### Пример 40. Два раздела

Существует несколько видов разрывов разделов. По умолчанию применяется разрыв, после которого начинается новая страница (как в примере 40). Кроме типа разрыва в элементе свойств раздела определяются и другие параметры. Например, можно задать ориентацию страницы (рис. 13).

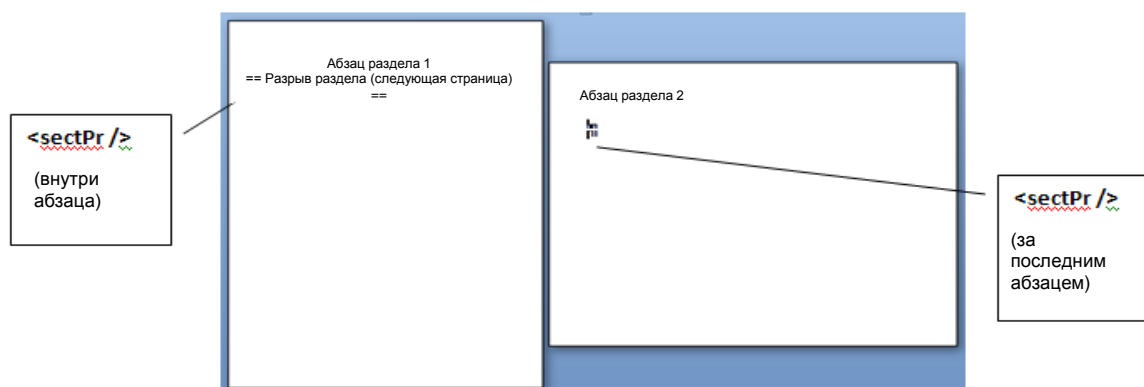


Рис. 13. Документ с двумя разделами

Для получения такого эффекта нужно сделать две вещи. Во-первых, в свойствах второго раздела — задать альбомную ориентацию через элемент размера страницы *pgSz*. Одного указания ориентации недостаточно — надо изменить высоту и ширину страницы. Единицей измерения высоты и ширины является одна двенадцатая пункта. В следующих примерах показано определение разделов с книжной и альбомной ориентацией для страниц размера A4.

```
<w:sectPr>
  <w:pgSz w:w="12240"
    w:h="15840" />
</w:sectPr>
```

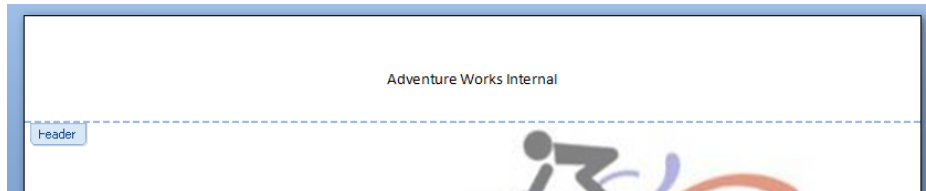
Пример 41. Книжный макет страницы

```
<w:sectPr>
  <w:pgSz w:w="15840" w:h="12240"
    w:orient="landscape" />
</w:sectPr>
```

Пример 42. Альбомный макет страницы

## Добавление верхних и нижних колонтитулов

На каждой странице есть определенные поля между текстом и краем страницы. Верхнее и нижнее поля называются колонтитулами. В WordprocessingML их заполнить определенным содержимым, которое будет повторяться на всех страницах текущего раздела. Обычно в колонтитулах содержится название главы, номер страницы или какое-либо изображение.



Колонтитулы хранятся в специальном компоненте пакета. Чтобы создать собственные колонтитулы, надо добавить в пакет XML-файл и добавить в список новые типы содержимого. Затем нужно установить связь между основным документом и компонентом верхнего или нижнего колонтитула. Поскольку в документе может быть произвольное число разделов, компонентов колонтитулов может быть много. Однако можно применять одинаковые колонтитулы во всех или многих разделах одного документа.

Ссылаются на колонтитулы из элементов свойств разделов с помощью элементов *headerReference* и *footerReference*. В следующем примере показано, как сослаться на верхний и нижний колонтитул раздела. С помощью атрибута типа можно также различать колонтитулы для четных и нечетных страниц. Это позволяет выводить номера страниц на одной стороне страницы при двусторонней печати.

```
<w:sectPr>
  <w:headerReference w:type="default" r:id="rId3" />
  <w:footerReference w:type="default" r:id="rId4" />
</w:sectPr>
```

Пример 43. Верхний и нижний колонтитулы

*Тип связи для верхних и нижних колонтитулов:*  
<http://schemas.openxmlformats.org/officeDocument/2006/relationships/header>  
<http://schemas.openxmlformats.org/officeDocument/2006/relationships/footer>

Содержимое колонтитулов аналогично содержимому тела документа. Корневым является один из двух элементов: *hdr* или *ft* в зависимости от типа колонтитула. Внутри этих элементов можно помещать обычное блочное содержимое: абзацы или таблицы. Можно вставить рисунки и другие изображения.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<w:hdr xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:p>
    <w:pPr>
      <w:jc w:val="center" />
    </w:pPr>
    <w:r>
      <w:t>AdventureWorks Report</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Пример 44. Содержимое верхнего колонтитула

Тип содержимого для данного компонента показан ниже. Способ добавления новых компонентов пакета и обновления типов содержимого нами уже рассматривался.

*Тип содержимого для верхних и нижних колонтитулов:*  
*application/vnd.openxmlformats-officedocument.wordprocessingml.header+xml*  
*application/vnd.openxmlformats-officedocument.wordprocessingml.footer+xml*

## Пользовательский XML-код в документах

Если проанализировать динамику использования различных документов за последнее время, несложно заметить, что они становятся основным средством обмена данными. Только теперь документы — это не страницы форматированного текста, документы и есть данные. Это стало возможно благодаря внедрению ряда технологий в WordprocessingML. Рассмотрим, как с помощью Open XML форматировать обычный деловой документ — счет. Данные в типичном бизнес-документе можно отнести к двум уровням: форма (внешний вид) и содержание (смысл). До сих пор мы рассматривали код, определяющий отображение текста. Теперь предположим, что требуется сохранить смысл или бизнес-контекст сведений, содержащихся в документе. Можно пытаться ввести правила форматирования элементов данных, например «полужирным выделяются записи клиентов», но такой подход неуклюж и чреват ошибками. Документы с дополнительной семантикой, которую невозможно передать одними лишь средствами визуального форматирования, создаются с помощью двух технологий, и у каждой собственные области применения, преимущества и недостатки.

Для добавления пользовательской бизнес-семантики к элементам разметки используют пользовательский XML-код либо структурный тег документа. Структурные теги документа появились в Open XML, они устраняют ряд сложностей работы с пользовательским XML-кодом. Хотя они обладают массой полезных свойств, в их функциональности есть неожиданные прорехи. Обе технологии позволяют добавлять в документ код с бизнес-семантикой, основанную на пользовательских XML-сообщениях. Последние можно проверять на соответствие XML-схеме, чтобы гарантировать корректность созданного пользователем документа.

## Пользовательский XML-код

Первый подход к добавлению бизнес-семантики — ровесник применению XML с документами. Необходимую для этого разметку изменили для улучшения поддержки проверки XML, но концептуально все осталось по-прежнему. В качестве примера применения пользовательского XML-кода рассмотрим демонстрационный отчет. В отчете собрано множество элементов данных. В заголовке находится имя таблицы, а также даты начала и конца отчетного периода. В теле содержатся контактная информация, суммы продаж и удельные продажи по регионам. Большинство элементов данных можно представить простыми значениями, число их заранее не известно. Пользовательская XML-разметка позволяет работать и с ними. На рисунке ниже показан фрагмент отчета и составляющих его элементов данных (обведены), не выделен единственный элемент данных — сам отчет. В целом отчет можно считать контейнером данных. Он представлен корневым узлом пользовательского XML-сообщения, конструируемого в документе.

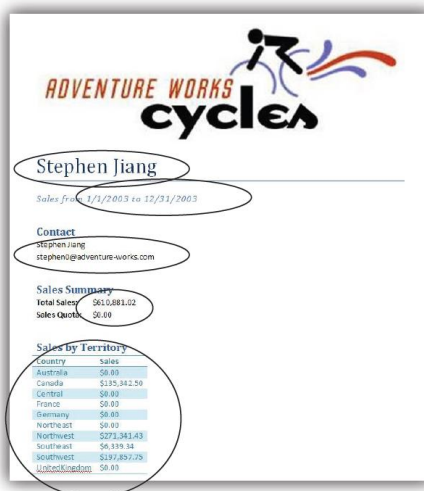


Рис. 14. Данные по регионам в отчете

Содержимое отчета можно представить как отдельные XML-сообщения, содержащие исключительно элементы из пространств имен, не связанных с Open XML.

```
<salesReport xmlns="http://adventureworks.com/salesReport">
  <name>Stephen Jiang</name>
  <salesPeriod>
    <startDate>1-1-2003</startDate>
    <endDate>12-31-2003</endDate>
  </salesPeriod>
  <salesPerson>
    <name>Stephen Jiang</name>
    <email>stephen0@adventure-works.com</email>
  </salesPerson>
  <salesSummary>
    <totalSales>610881.02</totalSales>
    <salesQuota>0.00</salesQuota>
  </salesSummary>
</salesReport>
```

```

</salesSummary>
<salesByTerritory>
  <territory>
    <name>Australia</name>
    <sales>0.00</sales>
  </territory>
  <territory>
    <name>Canada</name>
    <sales>135342.50</sales>
  </territory>
  <!-- остальные регионы опущены -->
</salesByTerritory>
</salesReport>

```

#### Пример 45. Пользовательские данные в формате XML

Для представления этих данных в WordprocessingML-разметке пользовательские данные сочетаются с обычной разметкой документа. Ниже показано, как выглядит такая конструкция внутри Open XML-приемника (приводится только начало отчета).

Рис. 15. Пользовательский XML-код в разметке интерфейса

В документе данные заменены пользовательскими заполнителями. Пользователь подставляет вместо них значения, просто вводя их как текст с клавиатуры. Такое представление ориентировано на разработчика, но существует и более понятная пользователю форма отображения тех же элементов. Эти параметры сохраняются в соответствующем компоненте пакета.

Рис. 16. Заполнители

Для представления пользовательских бизнес-данных в документе нужно изменить главный компонент и применить пользовательские элементы разметки к форматированному тексту. Специальный тег *customXml* служит для определения узлов пользовательских бизнес-сообщений. Для определения имен узлов и пространства имен применяются атрибуты. Это позволяет проверять документ на соответствие схеме. Если бы разрешалось сочетать разметку WordprocessingML с действительно произвольным XML-кодом, такая проверка оказалась бы невозможной. Следующий пример демонстрирует включение имени в пользовательскую разметку. Заметьте: по-прежнему можно форматировать текст ранее рассмотренными методами.

```

<w:customXml w:uri="http://adventureworks.com/salesReport" w:element="name">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="Title" />
    </w:pPr>
    <w:r>
      <w:t>Stephen Jiang</w:t>
    </w:r>
  </w:p>
</w:customXml>

```

Из этого кода видно, что имя отчета не является корневым элементом пользовательского бизнес-сообщения. Как создается XML-иерархия документов? Вложением тегов *customXml* в другие такие же теги. Первый способ — заключить таблицу в теги *customXml*, определяющие узел *salesByTerritory*, а каждую строку — в теги *customXml*, определяющие узел *territory*. Имеется также элемент, в который обычно заключают весь документ, — корневой узел бизнес-сообщения. Он может иметь такой вид:

```

<w:document>
  <w:body>
    <w:customXml w:uri="http://adventureworks.com/salesReport"
w:element="salesReport">
      <w:customXml w:uri="http://adventureworks.com/salesReport" w:element="name">
        <w:p>
          <w:r>

```

```

        <w:t>Stephen Jiang</w:t>
      </w:r>
    </w:p>
  </w:customXml>
</w:customXml>
</w:body>
</w:document>

```

Пример 46. Пример сопоставления пользовательской XML-разметки

В элемент *customXml* можно заключать следующие элементы (рис. 17):

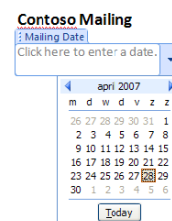


Рис. 17. Элементы, поддерживающие пользовательскую XML-разметку

Работа с пользовательским XML-кодом на платформе Microsoft Office имеет существенное преимущество. При вставке строк к таблице в приложении Microsoft Word новые строки автоматически связываются с добавлением тегов *customXml*. Это позволяет воспроизводить повторяющееся содержимое бизнес-сообщения в приемнике (consumer). Эта прикладная функция не поддерживается другой моделью хранения бизнес-семантики — структурными тегами документа, значительные преимущества которой заслуживают более подробного рассмотрения.

## Структурный тег документа

Вторая модель названа разработчиками «структурный тег документа» (structured document tag, SDT), в Microsoft ее также называют «элемент управления content». Она позволяет выделять в документе при помощи разметки области, которые могут быть как элементами данных, так и группами элементов форматированного текста. Для простых данных возможно отображение некоторых элементов интерфейса, например, календаря (см. рис. справа). Помимо разметки документа, можно связать содержимое элемента управления content к файлу с пользовательскими бизнес-данными из пакета. Подобная изоляция данных невозможна с применением пользовательского XML-кода. Пользовательские XML-данные хранятся в отдельном компоненте внутри пакета и поддерживают двустороннее связывание. Так, при редактировании SDT в приемнике компонент с пользовательским XML-кодом также обновляется. И наоборот при программном обновлении компонента, содержащего пользовательский XML, внесенные изменения будут видны после открытия документа. Можно (но не обязательно) предоставить и схему для этих данных.



Одно из основных преимуществ структурных тегов документов — возможность запретить изменение и удаление тегов документа. А в модели, основанной на пользовательском XML-коде, не исключено случайное удаление привязки. Используя же структурные теги, такие события легко предотвратить. С помощью структурных тегов можно также создать документ, по сути являющийся заполнителем для подстановки содержимого, чтобы обезопасить последнее от случайного удаления при редактировании разметки.

Структурные теги документа могут применяться на уровне блоков, а также как встроенное содержимое, это зависит от содержимого тега. Структурные теги документа позволяют создавать следующие типы содержимого:



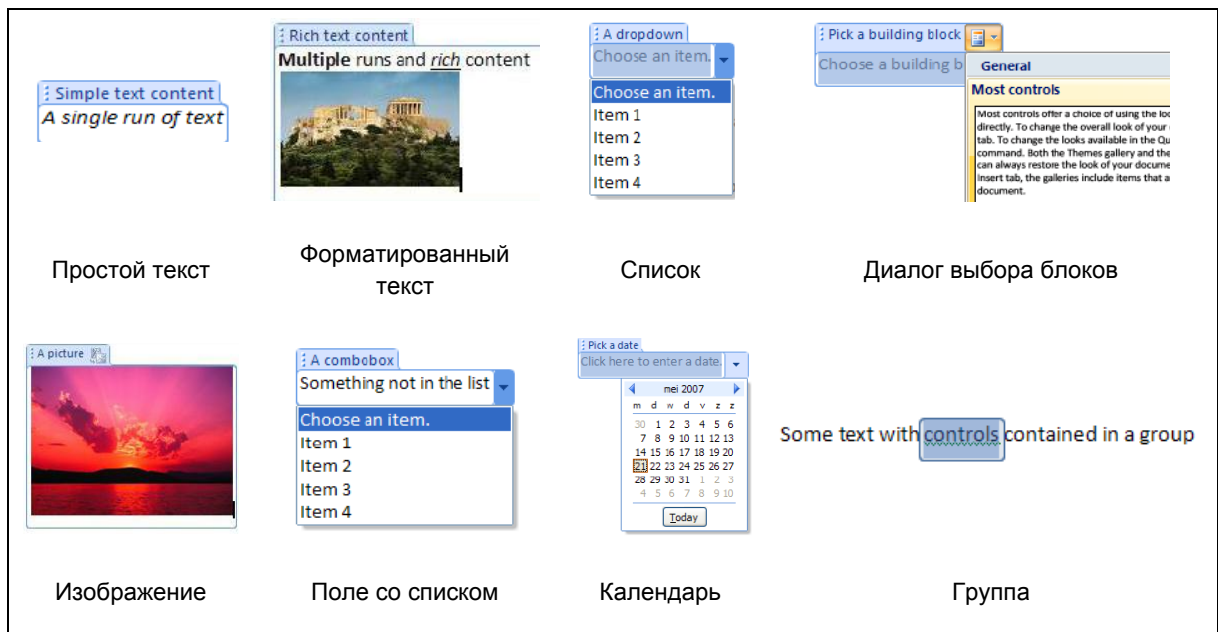


Рис. 18. Типы структурных тегов документа

Большинство структурных тегов поддерживает содержимое в виде простых документов. Особыми тегами являются Group и Building Block. Group служит для группировки содержимого, в том числе других тегов документов. Обычно с его помощью блокируют редактирование определенных частей документа, кроме тегов документа, вложенных в Group. Так удается создать заполнители, содержимое которых можно изменять. Не менее интересен тег Building Block, содержимым которого может быть готовый строительный блок, определенный в шаблоне документа.

Для создания структурного тега документа служит элемент *sdt*. Структурный тег документа включает два раздела: свойства и содержимое.

В свойствах задают такие данные, как тип тега документа. Тег документа также включает элементы *alias* и *tag*. Первый содержит сведения для пользователя, второй — имя для ссылки в коде. Псевдоним (содержимое *alias*) отображается в заголовках UI. Возможна дополнительная настройка любого тега документа в элементе, определяющем его тип. Примером может служить включение сведений о формате даты в соответствующий тег.

```

<w:sdt>
  <w:sdtPr>
    <w:alias w:val="Day of birth" />
    <w:tag w:val="Birthday" />
    <w:showingPlcHdr />
    <w:date>
      <w:dateFormat w:val="d-M-yyyy" />
      <w:calendar w:val="gregorian" />
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:t>Click here to enter a date.</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>

```

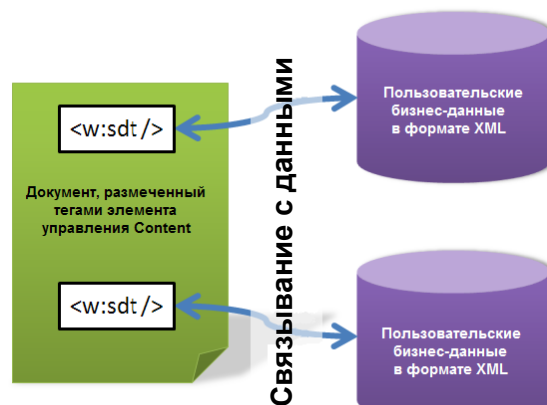
Пример 47. Структурный тег документа Date

Содержимое тега документа задается элементом *sdtContent*. Этот элемент отражает один из двух элементов данных: текст-заполнитель для пустых элементов или реальные данные. Среди свойств имеется флаг *showingPlcHdr*, указывающий, что вместо реального содержимого отображается заполнитель. В случае связывания с данными последние не сохраняются в пользовательских бизнес-компонентах.

В нашем примере с помощью заполнителей текста и даты можно разметить все места, где будет храниться содержимое. Связывание с таблицей осуществляется путем заключения в теги ее строк либо значений отдельных ячеек. Связывание с данными (о нем пойдет речь ниже) настраивается на уровне тега документа, при этом в отличие от подхода с пользовательским XML иерархия значения не имеет.

## Связывание с данными

Большим подспорьем в работе с тегами документов является связывание с данными. Основа этой методики — хранение данных внутри тега документа, отдельно от основной разметки. Когда данные хранятся отдельно, осуществлять к ним доступ намного проще, чем когда они хранятся вместе с разметкой. Модель связывания с данными, описанная в Open XML, поддерживает двустороннее связывание. Если вы изменяете содержимое тега документа как пользователь (через компонент-приемник), модифицированные данные будут храниться в отдельном контейнере. Если же вы измените контейнер как разработчик, а затем откроете документ, в нем будут уже модифицированные данные. Такой подход с двусторонним связыванием весьма удобен при создании бизнес-содержимого со сложным оформлением и структурой.



Связывание структурных тегов документа с данными требует добавления к каждому элементу *sd t* выражения X-Path. При обработке этого выражения извлекаются данные из особого пользовательского бизнес-компонента пакета. В пакете может храниться несколько компонентов, на которые можно ссылаться по отдельности. Чтобы добавить к нашему примеру отчета поддержку связывания с данными, прежде всего следует создать в пакете пользовательский бизнес-компонент. Он идентифицируется MIME-типами. По умолчанию используется MIME-тип для XML. В пакете следует создать и второй компонент для хранения пользовательских свойств для бизнес-данных. В дальнейшем они послужат для определения компонента с пользовательскими данными. Здесь применяется тот же тип содержимого. Возможно, вам не потребуется обновлять список *[Content\_Types].xml* list, поскольку MIME-тип, связанный с расширением XML-файлов по умолчанию, обычно является верным.

---

*Тип содержимого для пользовательских бизнес-данных в формате XML: application/xml*

---

Второй этап — связывание основного тела документа с пользовательским бизнес-компонентом, содержащим данные, и этого компонента — с компонентом свойств. Для этого применяются следующие типы связей:

---

*Тип связи для пользовательских бизнес-данных в формате XML:*  
<http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXml>

*Тип связи для пользовательских свойств XML:*  
<http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXmlProps>

---

Поскольку в пакете разрешено хранить несколько файлов с пользовательскими данными, определить, с каким из них связать тег документа, бывает непросто. Данные в разных компонентах могут быть идентично структурированы. По умолчанию компонент-приемник использует первый пользовательский XML-компонент, соответствующий выражению X-Path, заданному для тега документа. В отсутствие таковых никаких действий не выполняется. Указать компонент, в котором должны храниться данные, можно через идентификатор элемента хранения (store item ID). Этот идентификатор присоединен к компоненту с пользовательским XML-кодом через компонент, содержащий пользовательские свойства. В компоненте свойств определены два элемента данных: значение идентификатора и схемы для XML-данных пользовательского сообщения.

```
<ds:datastoreItem
  xmlns:ds="http://schemas.openxmlformats.org/officedocument/2006/2/customXml"
  ds:itemID="{C65DD089-1234-4A84-8443-BC4CB07DEB45}">
  <ds:schemaRefs>
    <ds:schemaRef ds:uri="http://adventureworks.com/sampleReport" />
  </ds:schemaRefs>
</ds:datastoreItem>
```

### Пример 48. Компонент с пользовательскими свойствами XML

Теперь пользовательский бизнес-компонент можно определить по идентификатору, так что пора добавить к нему содержимое. Важный момент — отсутствие определенных пространств имен Open XML. Допускается хранение любых XML-данных и связывание с ними.

```
<a:report xmlns:a="http://adventureworks.com/sampleReport">
  <a:salesMan>Stephen Jiang</a:salesMan>
  <a:contactEmail>stephen0@adventure-works.com</a:contactEmail>
  <a:salesPeriod>
    <a:start>1-1-2003</a:start>
    <a:end>31-12-2003</a:end>
  </a:salesPeriod>
```



```

<a:salesSummary>
  <a:totalSales>610,881.02</a:totalSales>
  <a:salesQuota>0</a:salesQuota>
</a:salesSummary>
</a:report>

```

#### Пример 49. Компонент с пользовательским XML-кодом

В завершение структурному тегу документа требуются ссылки на данные, хранимые в компоненте с пользовательским XML. Для этого вместе с X-Path-запросом, извлекающим нужные данные, надо добавить атрибут *dataBinding*. Разрешается применять префиксы пространств имен путем добавления к элементу *dataBinding* сопоставлений для отдельных префиксов. Это нужно для использования префиксов пространств имен XML в X-Path-запросах. Атрибут *storeId* не обязателен. Если он не определен, приемник попытается связаться с первым подходящим компонентом с пользовательским XML, как описано выше.

```

<w:sdt>
  <w:sdtPr>
    <w:alias w:val="Sales-man" />
    <w:tag w:val="salesManName" />
    <w:dataBinding
      w:prefixMappings="xmlns:a='http://adventureworks.com/sampleReport'"
      w:xpath="/a:report/a:salesMan" />
    <w:text />
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>Click here to enter text</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>

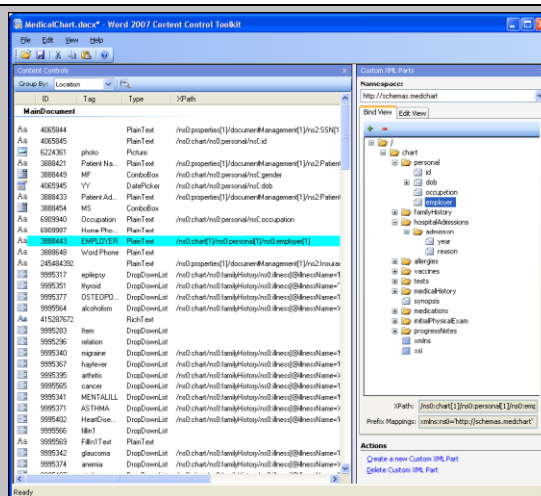
```

#### Пример 50. Структурированный тег документа, связанный с данными

### Content Control Toolkit

Связывание структурных тегов документа с пользовательскими файлами XML-данных в пакете облегчает инструмент Content Control Toolkit. Эта Open Source-программа генерирует компоненты, связи и разметку для связывания с пользовательскими XML-данными.

Загрузить инструмент можно с сайта CodePlex по адресу <http://www.codeplex.com/Wiki/View.aspx?ProjectName=dbe>



### Завершение документа

Наверное, многие из вас отправляли заказчику документы, из которых по ошибке не были удалены черновые данные и примечания рецензентов. Ситуация может быть весьма неприятной, если эти примечания не предназначались для посторонних. WordprocessingML хранит различные сведения о внесенных изменениях в документе. Если включить режим рецензирования, клиент сохранит все внесенные в документ изменения. Для каждого из модифицированных фрагментов сохраняется исходная версия. Кроме того, в документе можно сохранять комментарии рецензента. Зная структуру хранения этих данных, их можно без труда удалить из документа. Указать, что документ является окончательной версией, можно, установив специальный флаг (вероятно, после удаления черновых элементов и примечаний). Если этот флаг установлен, клиент получит рекомендацию открыть документ только для чтения.

### Удаление данных рецензирования

Если включить режим рецензирования, все внесенные в документ изменения будут сохранены со специализированной разметкой в компоненте параметров. В дальнейшем пользователь сможет выбрать, какие изменения следует сохранить, приняв изменения, либо вернуться к исходной версии. При этом довольно часто в окончательной версии документа по недосмотру остаются следы рецензирования.

Связанная с рецензированием разметка хранится в специализированных элементах. Большинство из них легко удаляются из документа вместе с объявленными в них вложенными элементами. Другие элементы служат контейнерами для текущего содержимого документа, при удалении таких элементов необходимо сохранять их вложенные элементы. Вот список элементов, с которыми приходится работать во время очистки документов от следов рецензирования:

Элемент	Содержит	Действие по очистке
rPrChange	Исходные свойства области	Удаляется
tblPrChange	Исходные свойства таблицы	Удаляется
trPrChange	Исходные свойства строки	Удаляется
tcPrChange	Исходные свойства ячеек	Удаляется
moveFrom	Контейнер перемещенного содержимого	Удаляется
moveFromRangeStart	Маркер исходного положения перемещенного содержимого	Удаляется
moveFromRangeEnd	Маркер исходного положения перемещенного содержимого	Удаляется
moveTo	Контейнер перемещенного содержимого	Удаляется, сохраняются вложенные элементы
moveToRangeStart	Маркер целевого положения перемещенного содержимого	Удаляется
moveToRangeEnd	Маркер целевого положения перемещенного содержимого	Удаляется
ins	Контейнер со вставленным содержимым	Удаляется, сохраняются вложенные элементы
del	Маркер удаленного содержимого	Удаляется
delText	Контейнер удаленного текста	Удаляется

Таблица 3. Список элементов, используемых для рецензирования

Большинство действий по удалению не представляет сложности. От любых элементов, помеченных в таблице как удаляемые, можно избавиться, просто отредактировав XML-код. Для обработки контейнеров `moveTo` и `ins` требуются дополнительные усилия. Поскольку они содержат, собственно, модифицированное и вставленное содержимое, надо сохранить все элементы, вложенные в эти контейнеры.

## Удаление примечаний

Примечание — это комментарий, привязанный к определенной части документа. Обычно на этапе завершения приходится не создавать, а удалять их из документа. В любом случае знать структуру примечаний необходимо.

Примечание состоит из двух частей. Текст примечания хранится в отдельном компоненте пакета, он привязывается к определенному месту документа связью стандартного типа.

*Тип связи для примечаний:*

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/comments>

Внутри компонента, определенного этой связью, находятся отдельные примечания. В корневом узле `comments` хранятся элементы `comment` для каждого примечания, определенного в документе. Примечания различаются по идентификатору (ID) и сведениям об авторе. Внутри узла `comment` разрешается применять обычные блочные конструкции, такие как абзацы и таблицы.

```
<w:comments
  xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main">
  <w:comment w:id="0" w:author="Wouter van Vugt"
    w:date="2007-04-28T19:35:00Z" w:initials="WvV">
    <w:p>
      <w:r>
        <w:t>А comment</w:t>
      </w:r>
    </w:p>
  </w:comment>
</w:comments>
```

### Пример 51. Компонент, хранящий примечания

Ссылаться на примечания из документа просто. В предполагаемом месте привязки примечания находится узел `commentReference`. Указав значение ID, назначенное примечанию в соответствующем компоненте, можно привязать к нему ссылку в документе. Поскольку `commentReference` — одиночный элемент, он не

позволяет размечать примечания, относящиеся к нескольким фрагментам содержимого. Возможна привязка к единственному месту. Чтобы определить область, к которой относится примечание, в дополнение к *commentReference* используют элементы *commentRangeStart* и *commentRangeEnd*.

```
<w:p>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>This text is commented on</w:t>
  </w:r>
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
  <w:commentRangeEnd w:id="0" />
</w:p>
```



Пример 52. Ссылка на примечание

Преимущество такого подхода в простоте удаления примечаний из пакета WordprocessingML. Для этого нужно лишь удалить из пакета компонент с примечаниями, а затем — удалить из разметки узлы примечаний. Поскольку примечания могут быть только в теле основного документа, достаточно обработать один XML-документ. Структура узлов ссылок также облегчает их удаление. Все узлы примечаний пусты, поэтому перемещать содержимое при их удалении не требуется.

## Маркировка документа как завершенного

Если документ помечен как завершенный, UI для редактирования в приложении-клиенте будет деактивирован. Эта функция не является защитной — она позволяет пользователям определить тип документа, с которым им приходится иметь дело.

Эта функция не определена в Open XML и освещается здесь ввиду ее востребованности. Чтобы пометить документ как завершенный, в пакете нужно создать пользовательский XML-компонент с определенной разметкой Open XML. В частности, здесь применяется разметка, задающая пользовательские свойства и обеспечивающая доступ к ним для клиента.

Итак, чтобы пометить документ как завершенный, сначала создайте в пакете новый компонент и корректно установите тип содержимого. Затем создайте связь в главном файле связей. Связь надо добавить в файл `_rels\rels`. Ниже показаны необходимые для этого типы содержимого и связи.

---

*Тип содержимого (для компонента с пользовательскими свойствами):*  
*application/vnd.openxmlformats-officedocument.custom-properties+xml*

*Тип связи (для компонента с пользовательскими свойствами):*  
*http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties*

---

В созданный компонент нужно поместить определенный XML-код, который подскажет Word, что данный документ является завершенным.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<Properties
  xmlns="http://schemas.openxmlformats.org/officeDocument/2006/custom-properties"
  xmlns:vt="http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes">
  <property fmtid="{D5CDD505-2E9C-101B-9397-08002B2CF9AE}" pid="2"
    name="_MarkAsFinal">
    <vt:bool>true</vt:bool>
  </property>
</Properties>
```

Пример 53. Свойства, помечающие документ как завершенный

Эта разметка, целиком и полностью специфичная для платформы Microsoft Office, демонстрирует один из многочисленных способов расширения клиентом функциональности документа и настройки его под свои нужды. Вы убедитесь, что для реализации большинства корпоративных требований, включая поддержку функций доступности, пользовательская разметка не нужна и удастся обойтись возможностями «чистого» Open XML.

## Дополнительные вопросы

---

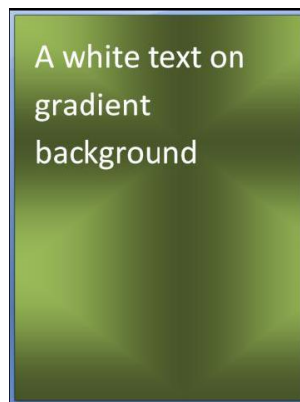
После вставки в документ изображения наш отчет выглядит как на иллюстрации в этой книге. Однако есть интересные моменты, которые не относятся непосредственно к этому примеру, но все же заслуживают внимания. Мы рассмотрим такие распространенные вещи, как объединение ячеек таблиц, вставка полей и списков.

## Фон страницы

На уровне документа разрешается задавать фоновые изображения для документа. Фоновое изображение заполняет страницу целиком, задавать его можно только на уровне документа — на уровне раздела сделать этого невозможно. Фоновое изображение объявляют с помощью тега *document* и элемента *background*. Существует два способа определения фона документа. Задать в качестве фона сплошную одноцветную заливку можно в самом теге *background*. Более сложный фон можно задать через VML-разметку. Этим способом можно использовать в качестве фона градиентную заливку либо изображение. Вот как создать градиентный фон с помощью VML-разметки:

```
<w:document
xmlns:w="http://.../wordprocessingml/2006/main">
  <w:background w:color="9BBB59">
    <v:background fillcolor="#9bbb59 [3206]">
      <v:fill color2="fill darken(118) "
method="linear sigma"
          focus="100%" type="gradientRadial">
        <o:fill v:ext="view" type="gradientCenter" />
      </v:fill>
    </v:background>
  </w:background>
</w:document>
```

Пример 54. Фон для документа



Заметьте: этот фон не отображается по умолчанию в программе-клиенте. Чтобы увидеть его, нужно переключиться в режим полноэкранного просмотра (например, в Word). Чтобы фон отображался и в режиме разметки страницы, в компоненте параметров необходимо установить свойство *displayBackgroundShape*.

## Таблицы с объединенными ячейками

В таблицах часто объединяют ячейки по горизонтали либо по вертикали. Эта функция поддерживается и в WordprocessingML. Для каждого типа объединения ячеек имеется отдельная модель. Обе они основаны на свойствах узлов ячеек таблицы. При этом ячейки должны быть прямоугольными — объединение сразу по двум направлениям не допускается. На практике это означает, что ячейка может быть частью ячейки, объединенной либо по вертикали, либо по горизонтали, но не по вертикали и горизонтали одновременно.

Каждая строка, объявленная элементом *tr*, должна соответствовать определению сетки таблицы, заданному элементом *tblGrid*. Обычно это делается путем создания элементов *tc* для каждого *gridCol* в определении сетки таблицы. Однако иногда это правило может нарушаться. К ячейке таблицы может применяться элемент *gridSpan* со значением более 1. В этом случае число ячеек в строке будет на одну меньше, чем задано определением, и т. д. Например, если определением сетки задано три элемента *gridCol*, по умолчанию в этой таблице будет три ячейки в строке, если значение атрибута *gridSpan* равно 2, останется две ячейки, а если *gridSpan* = 3, то одна. Если в строке несколько групп ячеек, объединяемых по горизонтали, этот принцип просто применяется повторно.

Вот пример для таблицы размером два на два (объединяются ячейки верхней строки).

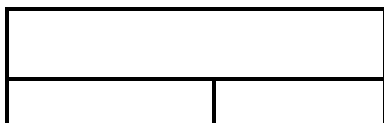



Рис. 19. Объединение ячеек по горизонтали

```
<w:tbl>
  <w:tblGrid>
    <w:gridCol w:w="4788" />
    <w:gridCol w:w="4788" />
  </w:tblGrid>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:gridSpan w:val="2" />
      </w:tcPr>
      <w:p />
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:p />
    </w:tc>
    <w:tc>
      <w:p />
    </w:tc>
  </w:tr>
</w:tbl>
```

Пример 55. Объединение ячеек по горизонтали

По вертикали ячейки объединяются иначе. При объединении по горизонтали число элементов *tc* уменьшается на один для каждой из объединенных ячеек. При объединении же по вертикали имеется два значения. В свойствах ячейки можно задать свойство *vMerge*, чтобы указать, что ячейка входит в группу, объединенную по вертикали. Значение *restart* указывает, что объединение вертикали начинается с данной ячейки. Для других ячеек той же группы задается значение *continue*. Группа заканчивается на первой ячейке, у которой нет элемента *vMerge*. Начать следующее объединение следует с нового *restart*.

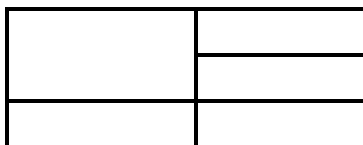



Рис. 20. Объединение ячеек по вертикали

```
<w:tbl>
  <w:tblGrid>
    <w:gridCol w:w="4788" />
    <w:gridCol w:w="4788" />
  </w:tblGrid>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:vMerge w:val="restart" />
      </w:tcPr>
      <w:p />
    </w:tc>
    <w:tc>
      <w:p />
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
```

```

<w:tcPr>
  <w:vMerge w:val="continue" />
</w:tcPr>
<w:p />
</w:tc>
<w:tc>
  <w:p />
</w:tc>
</w:tr>
<w:tr>
  <w:tc>
    <w:p />
  </w:tc>
  <w:tc>
    <w:p />
  </w:tc>
</w:tr>
</w:tbl>

```

Пример 56. Объединение ячеек по вертикали

Заметим, что при объединении по вертикали в отличие от объединения по горизонтали элементы *tc* не удаляются, поэтому возможно произвольное сокращение содержимого. В ячейках таблицы, помеченных как *continue*, можно хранить содержимое, но пользователю это содержимое видно не будет.

## Нумерованные списки

WordprocessingML позволяет создавать списки двух видов: нумерованные и маркированные. Существует и третий вид, по сути гибридный, в котором первый уровень — нумерованный, а второй — маркированный. Вот примеры разных видов списков:

- |                      |                       |                      |
|----------------------|-----------------------|----------------------|
| • <i>First item</i>  | 1. <i>First item</i>  | 1) <i>First item</i> |
| ○ <i>Second item</i> | a. <i>Second item</i> | • <i>Second item</i> |
| • <i>Third item</i>  | 2. <i>Third item</i>  | 2) <i>Third item</i> |

**Маркированный список**

**Нумерованный список**

**Гибридный список**

Для создания списка используются обычные абзацы. Каждый абзац образует элемент списка. Чтобы объявить абзац элементом списка, в его свойствах *pPr* следует указать два значения, ID нумерации и уровень элемента. Элемент *pPr* в составе свойств абзаца указывает клиенту, какую нумерацию применить к нему. Элемент *numId* указывает на определение нумерации. У абзацев, включенных в один список, идентичны значения *numId*. Значение *ilvl* или уровень абзацного отступа, указывает абзацные отступы в многоуровневых списках. Вместо вложения собственно абзацев для указания уровня вложенности используют их свойства.

```

<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="1" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>First item</w:t>
  </w:r>
</w:p>

```

Пример 57. Нумерованный абзац

Элемент абзаца не предоставляет данных о нумерации — только идентификатор нумерации и уровень отступа. Основная информация об отступах хранится вне тела главного документа, в отдельном компоненте со сведениями о нумерации. Для поиска этого компонента используется особый тип связи.

Тип связи для нумерации:

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/numbering>

Помимо связи, данный компонент имеет собственный тип содержимого:

*Тип содержимого для нумерации:*  
*application/vnd.openxmlformats-officedocument.wordprocessingml.numbering+xml*

В пакете эта связь используется минимум одним компонентом. Он содержит определения нумерации. Если требуется поддержка нумерации, прежде всего нужно создать в пакете WordprocessingML новый компонент, а затем связать компоненты с главным документом и сведениями о нумерации.

Компонент, содержащий сведения о нумерации, состоит из двух частей, в которых хранятся абстрактные и реальные определения нумерации. И те и другие определения содержат сведения о типе списка (нумерованный, маркированный или гибридный) и параметры для каждого из уровней списка, такие как используемый шрифт. Абзацы никогда не ссылаются прямо на абстрактные определения нумерации, а только на реальные определения нумерации. Реальные же определения нумерации ссылаются на абстрактные определения. Смысл применения такой модели состоит в возможности многократного использования параметров нумерации. Если два списка идентичны во всем, кроме шрифта на некотором уровне, имеет смысл указать только параметр, которым списки различаются. Глобальные (идентичные) параметры хранятся в абстрактном определении, а различающиеся параметры — в реальном определении. Следующий пример демонстрирует начало содержимого компонента со сведениями о нумерации. После него следует добавить абстрактное и реальные определения нумерации.

```
<w:numbering xmlns:w="http://.../wordprocessingml/2006/main">
</w:numbering>
```

#### Пример 58. Компонент со сведениями о нумерации

Изучая компонент со сведениями о нумерации, нетрудно заметить, что большинство параметров находится в абстрактных определениях. Первым важным параметром является ID абстрактного определения. Этот ID используется в реальном определении. Элемент *multiLevelType*, вложенный в *abstractNum*, определяет тип нумерации. Возможные значения этого параметра включают одноуровневые, многоуровневые и гибридные списки.

Каждый из уровней нумерации форматируется с помощью элемента *lvl*. Следующий пример разметки демонстрирует свойства, необходимые для создания одноуровневого маркированного списка. Текстовое значение этого элемента хранит символ, используемый в качестве маркера, а номер формата указывает на то, что используется обычный маркер. Параметры нумерованного списка отличаются. Как правило, списки выделяются отступами. Сведения об отступах хранятся в узлах свойств уровня абзаца и области. В дальнейшем они объединяются с форматированием, явно назначенным абзацу или области, определяя окончательный облик списка.

Чтобы создать маркированный список следующего вида:

- Пункт 1
- Пункт 2
- Пункт 3

— необходим такой код:

```
<w:abstractNum w:abstractNumId="0">
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    <w:numFmt w:val="bullet" />
    <w:pPr />
    <w:rPr />
  </w:lvl>
</w:abstractNum>
<w:num>
  <w:abstractNumId w:val="0" />
</w:num>
```

#### Пример 59. Маркированный список

Чтобы создать нумерованный список, нужно изменить несколько элементов. Формат нумерации, определяемый элементом *numFmt*, надо изменить в соответствии с желаемым стилем нумерации. Значение этого элемента выбирают из предопределенного списка. Базовый стиль задается значением *decimal*. Необходимо изменить и значение элемента, определяющего символ нумерации. У маркированного списка свойство *lvlText* определяет символ-маркер. Теперь в это свойство нужно поместить строку с описанием формата, содержащую символ-заполнитель вместо номера. Заполнитель *%1* задает нумерацию первого уровня, которая имеет вид «1) 2) 3)...». Заполнитель для нумерации второго уровня имеет вид *%1.%2* и генерирует номера вида «1.1, 1.2, 2.1, 2.2, 2.3». По умолчанию нумерация начинается с 0. Чтобы начать с другого значения, нужно добавить к определению уровня элемент *start*. Существует множество и других параметров, позволяющих изменять стиль нумерации.

Так, создать нумерацию вида:

- 101.Пункт 1
- 102.Пункт 2

### 103. Пункт 3

— позволит код:

```
<w:abstractNum w:abstractNumId="0">
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    <w:numFmt w:val="decimal" />
    <w:start w:val="101" />
    <w:lvlText w:val="%1)" />
  </w:lvl>
</w:abstractNum>
```

#### Пример 60. Нумерованный список

Реальное определение нумерации содержит как минимум ID нумерации и ссылку на абстрактное определение. Разрешается переопределять любые параметры родительского абстрактного определения независимо от его уровня. Для переопределения параметров нужно явно создавать определение уровня целиком. В результате соответствующие параметры абстрактного определения действовать не будут. Для переопределения параметров служит элемент *lvlOverride*. В нем создается определение уровня с использованием тех же элементов *lvl*, что и в абстрактных определениях.

```
<w:num w:numId="1">
  <w:abstractNumId w:val="0" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:numFmt w:val="bullet" />
      <w:lvlText w:val=">>" />
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

#### Пример 61. Переопределение уровня нумерации

## Закладки и гиперссылки

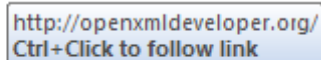
Общий способ связывания содержимого — использование гиперссылок. В WordprocessingML гиперссылки применяют для создания двух типов ссылок: на внешнее содержимое и на другие области того же документа. Для ссылки на внешнее содержимое нужно указать URL в соответствии с правилами Open Packaging Convention. Ссылки на внутреннее содержимое документа осуществляются с помощью закладок.

Гиперссылка на внешнее содержимое хранится в виде связи в пакете. Используется особый тип связи, идентифицирующий ее как гиперссылку. Поскольку ресурс, на который указывает гиперссылка, располагается вне пакета, связь описывает не компонент пакета и не тип содержимого. Вот этот тип:

*Тип связи для внешних гиперссылок:*

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink>

Если гиперссылка ведет из тела главного документа, эту связь надо сохранить в файле связей внутри компонента с главным документом.



[Open XML Developer](http://openxmldeveloper.org/)

Гиперссылки, используемые в верхних и нижних колонтитулах, хранятся в соответствующих файлах.

```
<Relationships xmlns="http://.../package/2006/relationships">
  <Relationship
    Id="rId2"
    Type="http://.../officeDocument/2006/relationships/hyperlink"
    Target="http://www.openxmldeveloper.org"
    TargetMode="External"/>
</Relationships>
```

#### Пример 62. Гиперссылка с внешней связью

Гиперссылки создаются с помощью элемента *hyperlink*, в который заключают текстовые области, составляющие текст гиперссылки. Ресурс, на который указывает гиперссылка, задают через ID связи.

```
<w:p>
  <w:hyperlink r:id="rId2">
    <w:r>
      <w:t>Open XML Developer</w:t>
    </w:r>
  </w:hyperlink>
</w:p>
```



```
</w:hyperlink>
</w:p>
```

### Пример 63. Гиперссылка на внешний ресурс

В гиперссылках, указывающих на другие области того же документа, действует аналогичный принцип. Но, поскольку гиперссылка ведет в другое место того же документа, прежде всего нужно создать маркер, на который будет указывать ссылка. Маркер определяется с помощью закладки. Элемент гиперссылки включает в себе ее текст, а начало и конец закладки определяются отдельными элементами, *bookmarkStart* и *bookmarkEnd*. Дело в том, что начало и конец закладки могут располагаться в разных абзацах. Начало и конец закладки задаются отдельными ID, для удобства пользователя закладке также присваивается имя.

```
<w:p>
  <w:bookmarkStart w:id="0"
w:name="MyMark" />
  <w:r>
    <w:t>Bookmarked Text</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
</w:p>
```

Пример 64. Закладка

```
<w:p>
  <w:hyperlink w:anchor="MyMark">
    <w:r>
      <w:t>Hyperlink Text</w:t>
    </w:r>
  </w:hyperlink>
</w:p>
```

Пример 65. Внутренняя гиперссылка

Ссылки на закладку в документе также создаются с помощью элемента *hyperlink*. Вместо атрибута с ID связи для ссылки на содержимое используется атрибут *anchor*, в котором закладка задается по имени.

## Применение полей

Поля — это динамические элементы содержимого в остальном почти статического документа. Большинство частей документа не меняется, скажем, при изменении оформления, однако некоторые части являются динамическими. Примером могут быть номера страниц и скользящие колонтитулы, отображающие название главы на каждой странице. Если жестко прописать эти элементы в документе, его будет очень сложно поддерживать в актуальном состоянии. В этой ситуации выручают поля — они позволяют создавать заполнители, вместо которых клиент подставляет актуальные данные во время открытия документа.

Поля бывают простыми и сложными. Простые умещаются в одной текстовой области (run). В ней хранится кэшированная версия данных поля, записанная туда при последнем обновлении. Сложные включают несколько областей, но их сложность этим не исчерпывается. И в простых, и в сложных полях динамическое содержимое определяется с помощью функций. Хороший пример сложного поля — оглавление документа. Другим примером являются поля с вложенными функциями, такими как вложенная конструкция IF.

Вот примеры общих функций:

Функция	Возвращаемое значение
PAGE	Номер текущей страницы
STYLEREF "Heading 1"	Текст текущего заголовка первого уровня
TITLE	Заголовок документа
AUTHOR	Имя автора документа
SAVEDATE \@ "dddd d MMMM уууу"	Дата последнего сохранения в заданном формате
TOC	Таблица оглавления

Таблица 4. Общие функции полей

Простые поля создаются с помощью элементов *fldSimple*, как правило, вложенных в абзац. Элемент *fldSimple* принимает единственную функцию; ее значение кэшируется в области, расположенной внутри поля.

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Page </w:t>
  </w:r>
  <w:fldSimple w:instr="PAGE">
    <w:r>
      <w:t>32</w:t>
    </w:r>
  </w:fldSimple>
</w:p>
```

```
</w:fldSimple>
</w:p>
```

### Пример 66. Простые поля

Сложные поля более замысловаты. Они могут включать несколько абзацев и областей. Как сказано выше, примером может служить таблица оглавления. Хотя оглавление генерируется единственной функцией, оно состоит из множества абзацев, составляющих сложное поле. Ниже приводится соответствующий пример оглавления на основе сложных полей.

## Оглавления

Множество документов содержит список содержимого в виде таблицы, известной как оглавление документа. Эта часть документа постоянно меняется, и обновлять ее вручную нежелательно. WordprocessingML поддерживает простой механизм для автоматической генерации таких таблиц. Определяют их с помощью полей.

```
<w:p>
  <w:fldSimple w:instr="TOC" />
</w:p>
```

### Пример 67. Незатейливое оглавление

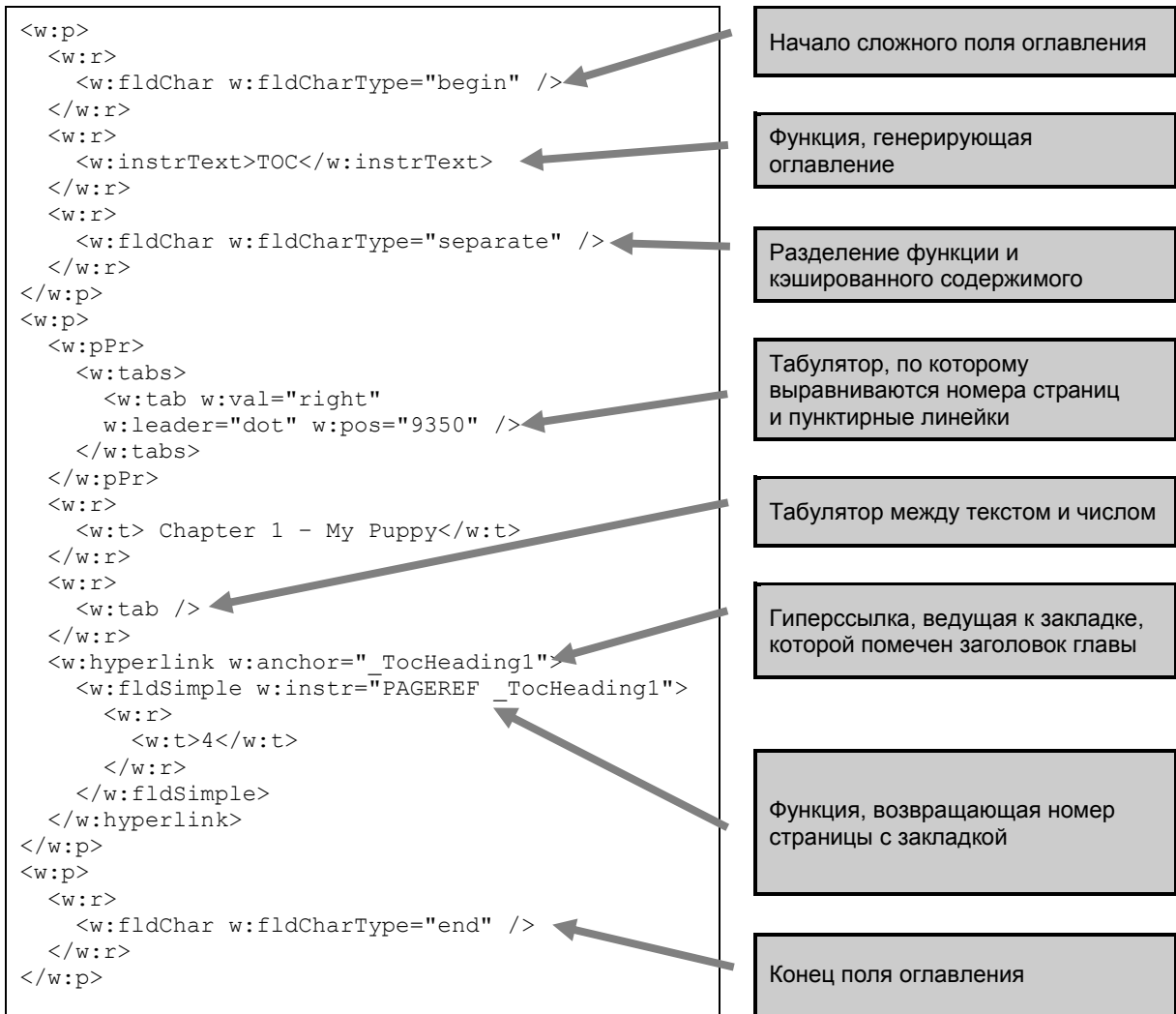
О том, что поле определяет оглавление, клиентской программе сообщает функция *TOC*. В этом поле можно использовать ранее описанную простую либо, при необходимости, более сложную нотацию. Преимущество сложной нотации — в возможности добавления элементов, формирующих оглавление до открытия документа. В случае же простой нотации пользователю придется вручную обновлять поле оглавления, чтобы оно заполнилось содержимым.

Для вставки поля оглавления служит код из примера 67. Как видно из названия примера, это минимальный вариант оглавления, и появляется оно только после ручного обновления поля в открытом документе. Однако пользователи ожидают, что оглавление будет доступно сразу после открытия документа. Чтобы этого достичь, следует применять сложную нотацию поля и добавлять элементы оглавления с разметкой WordprocessingML. Другой вариант — заставить клиент обновлять все поля после загрузки документа.

Как сказано выше, преимущество создания оглавления с применением поля сложной конструкции в том, что оно генерируется автоматически, без участия пользователя. Ниже показано простое оглавление с текстом, номерами страниц и соединяющими их точками.

Chapter 1 — My Puppy .....	4
Chapter 2 — My Cat.....	9

Более сложное оглавление включает три части, каждая со своим набором абзацев. Такое оглавление состоит из маркера начала поля, содержимого поля и маркера конца поля.



Пример 68. Усложненное оглавление

Обычно в оглавлении программа-клиент отображает гиперссылки, которые пользователь может щелкать. реализовать это позволяют внутренние гиперссылки: текст глав заключается в разметку, объявляющую закладки (пример 69).

```

<w:p>
  <w:pPr>
    <w:pStyle w:val="Heading1" />
  </w:pPr>
  <w:bookmarkStart w:id="0" w:name="_TocHeading1" />
  <w:r>
    <w:t>Heading 1</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
</w:p>

```

Пример 69. Абзац — заголовок 1 уровня

Чтобы автоматизировать обновление полей после загрузки документа, добавьте элемент *updateFields* в компонент параметров *WordprocessingML* и установите его в *true*. Когда пользователь откроет документ в программе-клиенте, все поля в документе будут обновлены, и оглавление станет видимым. Недостаток этого подхода в том, что для обновления оглавления участие пользователя все же требуется.

## Подписи

Вместе с изображениями, таблицами и другими типами особого содержимого обычно вставляют подписи, содержащие краткое описание добавляемого элемента. Обычно подпись включает порядковый номер и текст подписи. Для создания подписи не потребуется много кода. И в этом случае все основано на применении полей и функций в разметке документа.

Специальная функция для подсчета подписей, *SEQ* принимает имена подписей, которые требуется сосчитать. При вставке каждой подписи с заданным для функции именем ее счет увеличивается на 1. В этом случае поле также кэширует текущий номер подписи. Вот пример для подписи «Рис. 1»:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Рис. </w:t>
  </w:r>
  <w:fldSimple w:instr="SEQ Figure">
    <w:r>
      <w:t>1</w:t>
    </w:r>
  </w:fldSimple>
</w:p>
```

Пример 70. Подписи

## Резюме

---

Здесь удалось осветить далеко не все функции WordprocessingML, например, мощный механизм поддержки формул, а также использование словаря для хранения блоков. Рассказом об этих и других достойных рассмотрения возможностях пришлось пожертвовать, чтобы представить вам другие языки Open XML — их осталось три. Первый в списке — SpreadsheetML.

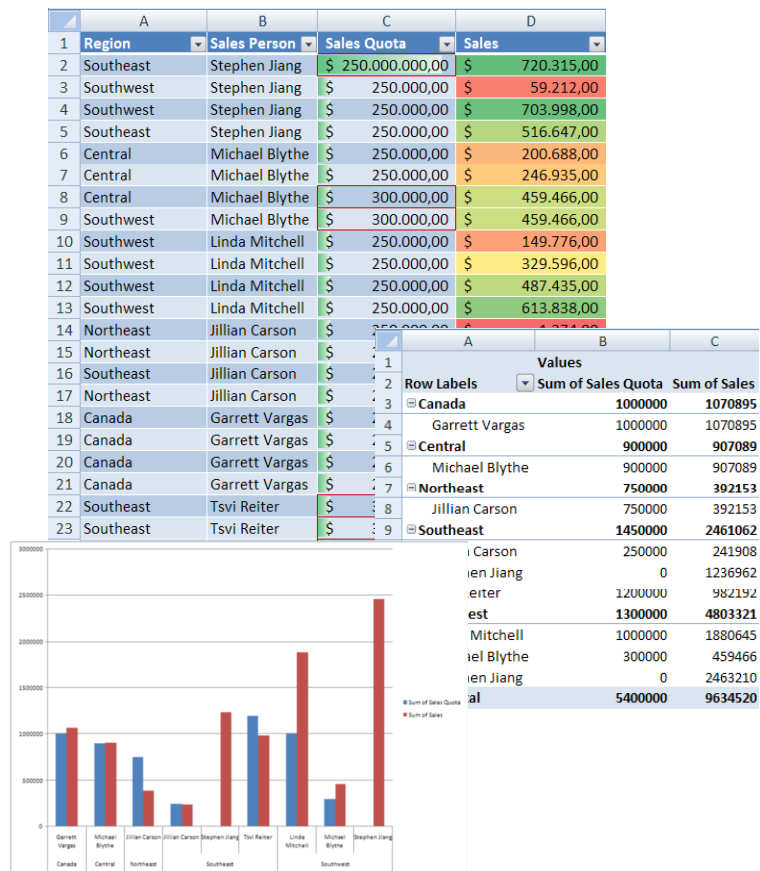
## Глава 2 SpreadsheetML

- Элементы простых электронных таблиц
- Работа с формулами
- Создание таблиц и сводных таблиц
- Оформление таблиц стилями и условное форматирование
- Вывод диаграмм

### Введение

Как и в главе, посвященной WordprocessingML, мы воспользуемся здесь демонстрационным отчетом. Наш пример электронной таблицы включает три основных элемента данных на языке SpreadsheetML: данные, сводную таблицу и построенную на их основе диаграмму. Сначала мы рассмотрим создание простейшей книги с минимальным количеством кода. Затем на первый лист книги мы добавим данные, структурированные в виде таблицы для упрощения их фильтрации. На основе этой таблицы будет создана сводная таблица, а по ее данным мы построим диаграмму. Поскольку диаграмма будет создана средствами DrawingML, о ней пойдет речь в соответствующей главе.

Ниже показан пример документа на языке SpreadsheetML.



## Элементы простой электронной таблицы

По минимальному объему кода, необходимого для создания электронной таблицы (по числу компонентов в пакете и количеству разметки внутри этих компонентов), SpreadsheetML находится между WordprocessingML и PresentationML. Документ SpreadsheetML включает главный компонент с книгой и компоненты, соответствующие отдельным листам.



Рис. 21. Минимально необходимые компоненты электронной таблицы

Для создания корректного пакета SpreadsheetML нужен компонент с книгой и минимум один компонент с листом. Пакет состоит из пяти элементов, которые надо собрать, как в случае документа на WordprocessingML. Сначала создаются компоненты с книгой и листом. Их можно хранить в любом каталоге, в нашем примере они размещены в корневого каталоге пакета. Для книги и листа используется особый тип содержимого, хранимый в соответствующем компоненте пакета. Компонент, хранящий сведения о типах содержимого, находится в файле *[Content\_Types].xml*, также расположенном в корне пакета. Вот эти типы:

---

*application/vnd.openxmlformats-officedocument.spreadsheetml.sheet.main+xml*  
*application/vnd.openxmlformats-officedocument.spreadsheetml.worksheet+xml*

---

Завершающий этап — создание связей между пакетом и книгой, а также книгой и листом. Поскольку первая связь направлена от пакета к первому компоненту, ее следует поместить в файл *.rels* в папке *\_rels* в корне пакета. Следуя этому принципу, вторую связь следует создать в файле *workbook.xml.rels* в том же каталоге *\_rels*. Для создания этих связей в пакете используются следующие типы:

---

*http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument*  
*http://schemas.openxmlformats.org/officeDocument/2006/relationships/worksheet*

---

### Компонент, содержащий книгу

Первая и главная функция компонента, содержащего книгу, — мониторинг листов, глобальных параметров и прочих общих составляющих книги. В содержащем книгу компоненте ведется список всех листов, составляющих электронную таблицу. Список требуется для именования листов и их перечисления приложениями, поддерживаемыми Open XML. В книге хранится три вида сведений о каждом листе. Во-первых, у каждого листа есть имя, отображаемое UI программы-клиента; во-вторых, у листов есть ID, необходимый для их сортировки, и в-третьих, ID связи, соединяющий книгу и компонент пакета, в котором хранится лист. Кроме того, в книге хранятся сведения о представлениях, вычислениях и версиях, и другие параметры.

```
<workbook
  xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
  xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships">
  <sheets>
    <sheet name="Sheet1" sheetId="1" r:id="rId1" />
  </sheets>
</workbook>
```

#### Пример 71. Простейшая книга

В этом примере у имени пространства имен SpreadsheetML по умолчанию отсутствует префикс пространства имен XML. Вы будете постоянно сталкиваться с этим при работе с SpreadsheetML. Префикс не используется опять же ради оптимизации: чем короче имена в разметке, тем меньше данных приходится разбирать и загружать в память. В нашем случае отказ от префикса дает экономию минимум в четыре символа на узел XML: префикс (один символ) и двоеточие, отделяющие его от имени узла, есть в каждом открывающем и замыкающем узле. Недостаток этого подхода — небольшое снижение удобочитаемости, но оно минимально и не имеет большого значения при работе с файлами SpreadsheetML.

Эта оптимизация не описана в стандарте Open XML, но используется в Excel. Возможно, другие приложения не последуют этой стратегии.

### Компонент, содержащий лист

Документ SpreadsheetML должен содержать не менее одного листа. Это может быть лист с таблицей, диалоговым окном либо диаграммой. Обычно электронная таблица состоит минимум из одного листа, а его связывают с электронной таблицей. Он имеет табличную структуру для определения данных. Чтобы создать пустой лист, достаточно определить лишь несколько его элементов. Как видно из следующего примера, корневым элементом листа является элемент *worksheet*. Все данные внутри листа можно разбить на три раздела. Первый содержит свойства листа, второй — данные (в обязательном элементе *sheetData*). Сразу за *sheetData* идет вспомогательная информация, включая сведения о защите и

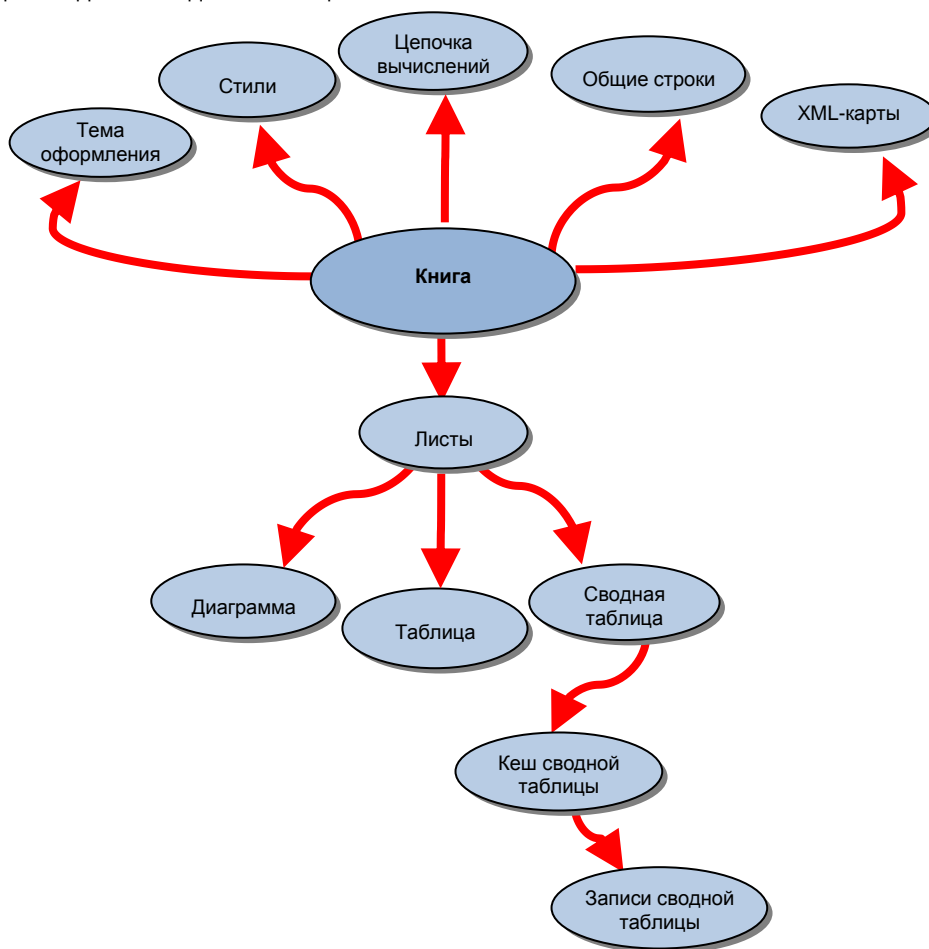
фильтрации. Чтобы определить пустой лист, достаточно создать элементы *worksheet* и *sheetData*. Элемент данных листа может быть пустым.

```
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main" >
  <sheetData>
    <row>
      <c>
        <v>1234</v>
      </c>
    </row>
  </sheetData>
</worksheet>
```

Пример 72. Простейший лист, содержащий значение 1234

## Общая структура

В пакет SpreadsheetML можно поместить массу других элементов. На рисунке показано большинство элементов листа корпоративных электронных таблиц. Большинство из них служит для специфичных нужд, например, для создания сводных таблиц и стилей.



## Создание листов

Лист — «рабочая лошадка» SpreadsheetML. Листы хранят данные в табличной форме. Чтобы создать новые значения на листе, необходимо объявить строки в элементе *sheetData*. Строки содержат ячейки, а те — значения. Объявление строк и ячеек может быть оптимизировано различными способами, но сначала рассмотрим структуру простого листа.

Элемент *row* определяет новую строку. Обычно первая строка, объявленная в *sheetData*, является первой строкой видимой части листа. Некоторые оптимизации изменяют такой порядок, но о них позже. В строке при помощи элемента *c* объявляют ячейки. Для сохранения числового значения в ячейке электронной таблицы достаточно внести эти значения в элемент *v*.

	A	B	C	D
1	Region	Sales Person	Sales Quota	Sales
2	Southeast	Stephen Jiang	0	720315
3	Southwest	Stephen Jiang	0	1759212
4	Southwest	Stephen Jiang	0	703998
5	Southeast	Stephen Jiang	0	516647
6	Central	Michael Blythe	300000	200688
7	Central	Michael Blythe	300000	246935
8	Central	Michael Blythe	300000	459466
9	Southwest	Michael Blythe	300000	459466

```

<worksheet
  xmlns="http://.../spreadsheetml/2006/main" >
  <sheetData>
    <row>
      <c>
        <v>42</v>
      </c>
    </row>
  </sheetData>
</worksheet>

```

	A
1	42
2	

Пример 73. Простой лист

Нетрудно заметить, что код ячеек в таблицах Excel отличается от приведенного выше примера. В Excel хранение текста оптимизировано с помощью общих строк (о них см. ниже). Подобно большинству оптимизаций Open XML, такой вариант вполне допустим, но является компромиссом между размером, производительностью и сложностью. Для создания текстовой ячейки служит элемент *is* (от *inline-string*). Чтобы сохранять текст в ячейке листа, сначала ее нужно объявить контейнером встроенных строк, для чего атрибут *t* устанавливается в *inlineStr*. Собственно, текст добавляется в элемент *is*, вложенный в *t*. Текст не хранится просто внутри элемента *is*, так как появление текста возможно и в других элементах разметки. В этом случае его заключают в тот же элемент *t*. В примере ниже показана вторая строка листа.

```

<worksheet
  xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main" >
  <sheetData>
    <row>
      <c t="inlineStr">
        <is>
          <t>Southeast</t>
        </is>
      </c>
      <c t="inlineStr">
        <is>
          <t>Stephen Jiang</t>
        </is>
      </c>
      <c>
        <v>0</v>
      </c>
      <c>
        <v>720315</v>
      </c>
    </row>
  </sheetData>
</worksheet>

```

Пример 74. Простой лист

Текст ячеек в таблицах Excel отличается от приведенного выше примера. В Excel хранение текста оптимизировано с использованием общих строк (о них см. ниже). Подобно большинству оптимизаций Open XML, такой вариант вполне допустим, но является компромиссом между размером, производительностью и сложностью.

## Формулы

Мощь электронных таблиц во многом базируется на возможности использования формул для создания математических моделей. Формулы обеспечивают автоматическое вычисление значений на основе внутренних и внешних по отношению к листу данных, а также генерацию вычисляемых ячеек.

Формулы хранятся в использующих их ячейках. Текст формул определяется элементом *f*. Формулы содержат математические выражения, включающие самые разные предопределенные функции.

```

<c>
  <f>(A3*B3) * (1-C3)</f>
  <v>3168</v>
</c>

```

	A	B	C	D
1				
2	Unit	Price	Discount	Total
3	32	110	10%	3168
4				

fx = (A3\*B3) \* (1-C3)

Пример 75. Формулы

Элемент *v*, в предыдущих примерах содержащий конкретные значения, также служит для хранения кэшированных результатов вычисления по формулам, полученных при последнем открытии таблицы.



Клиент может отложить пересчет формул, чтобы избежать задержки при открытии листов. Указывать значение формул не обязательно; если опустить его, ответственность за вычисление значений формул ложится на клиент Open XML.

При использовании функций в многоязычной среде, скажем, в системах совместной работы, возникают интересные проблемы. Так, Excel допускает вводить названия функций на родном языке пользователя. В результате функция *SUM()* в русскоязычной версии Excel примет вид *СУММ()*. Важно понять, как функции хранятся в Open XML. В документе хранятся нелокализованные функции: когда русскоязычный пользователь вводит функцию *СУММ()* в программе-клиенте, в XML-коде документа сохраняется функция *SUM()*. То есть локализован лишь пользовательский интерфейс, а не код. Это сильно упрощает программирование функций, позволяя разработчикам применять стандартный набор функций. Формулы используются не только в ячейках листов. Другие элементы, такие как столбцы таблиц, также способны хранить формулы, при этом действуют вышеописанные принципы. Формулы всегда хранятся в виде текста.

## Оптимизация электронных таблиц

Модель SpreadsheetML поддерживает ряд оптимизаций. Существует особый параметр строк и ячеек, определяющий код т. н. разреженных таблиц (sparse table). Он позволяет исключать пустые ячейки и работать только с нужными. Общие строки и формулы снижают нагрузку на клиент и размер файла документа.

### Разметка разреженных таблиц

В предыдущих примерах координаты строк и ячеек отсчитывались от верхнего левого угла таблицы, координаты первой ячейки — A1. Такая модель требует большого объема разметки даже для таких простых задач, как запись значения, скажем, в ячейку E5. Столбец E — пятый по счету, строка 5 — также пятая, поэтому придется перебрать 25 ячеек, чтобы добраться до нужной. Ясно, что это далеко не оптимальный способ по количеству кода, подлежащего разбору. Решить проблему помогает включение в строки и ячейки идентификатора позиции. Так, чтобы создать лист с единственной строкой под номером 5, можно добавить к элементу строки атрибут *r* (см. пример 76). Аналогичная модель указания положения используется и при работе с ячейками. Атрибут *r* задает номер строки и столбца. Ясно, что номер строки, заданный в координатах ячейки, должен совпадать с таковым, заданным на уровне строки.

```
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
  <sheetData>
    <row r="5">
      <c r="E5">
        <v>1234</v>
      </c>
    </row>
  </sheetData>
</worksheet>
```

#### Пример 76. Оптимизация по координатам

В таблице-примере (см. выше) не используется оптимизация по координатам строк и ячеек. Лист заполнен начиная с ячейки A1, поэтому данная информация не обязательна. Можно произвольно перемещать данные по листу, но для этого придется заняться редактированием. В фрагменте кода 76 оптимизация по координатам показана на примере единственной ячейки.

Нумерация строк и столбцов должна идти сверху вниз и слева направо. Нельзя задать строку 5 перед строкой 4, а столбец E — перед столбцом D: это приведет к ошибке.

## Общие строки

Общие строки (shared strings) оптимизируют использование памяти, если таблица содержит множество экземпляров некоторой строки. Такая ситуация имеет место и в нашем примере — она характерна и для современных бизнес-документов и аналитических расчетов. Если хранить эти строки прямо в коде, они будут многократно повторяться. Такой подход имеет ряд недостатков. Прежде всего избыточное содержимое увеличивает размер файла на диске. Кроме того, чем больше файл, тем больше времени занимает его запись и загрузка.

Для оптимизации хранения строк SpreadsheetML предусматривает хранение в документе единственных экземпляров строк — т. н. таблицы общих строк. Вместо повторяющихся строк в ячейках хранятся ссылки в виде индекса в таблице общих строк. Таблица общих строк хранится в отдельном компоненте пакета. В каждой книге имеется только один такой компонент независимо от того, повторяются ли строки на одном или на разных листах (подробнее о структуре пакета см. в главе о WordprocessingML). Чтобы переместить повторяющиеся строки из кода листа в таблицу общих строк, сначала надо создать отдельный компонент для хранения общих строк.

	A	B
1	Region	Sales Person
2	Southeast	Stephen Jiang
3	Southwest	Stephen Jiang
4	Southwest	Stephen Jiang
5	Southeast	Stephen Jiang
6	Central	Michael Blythe
7	Central	Michael Blythe
8	Central	Michael Blythe

Этот компонент, подобно другим, можно сохранить в любом месте пакета. Тип содержимого должен соответствовать таблице со строками. Необходимо указать значение:

---

*Тип содержимого для общих строк:*  
*application/vnd.openxmlformats-officedocument.spreadsheetml.sharedStrings+xml*

---

Второй этап — связывание таблицы общих строк с книгой. Электронная таблица может содержать не более одной таблицы общих строк. Поскольку все листы используют одну и ту же таблицу, ссылка на нее устанавливается в книге, а не в листе. Для этого служит связь следующего вида:

---

*Тип связи для общих строк:*  
*http://schemas.openxmlformats.org/officeDocument/2006/relationships/sharedStrings*

---

Таблицу, связанную с книгой, надо заполнить данными. Таблицу общих строк объявляют с помощью элемента *sst*. В нем имеется два атрибута для мониторинга содержимого и использования общих строк. Для добавления общей строки служит элемент *si* (от *string-item*) — «антипод» элемента встроенной строки *is*.

```
<sst xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
  <si>
    <t>Region</t>
  </si>
  <si>
    <t>Sales Person</t>
  </si>
  <si>
    <t>Sales Quota</t>
  </si>
  ...еще 12 элементов ...
</sst>
```

#### Пример 77. Таблица общих строк

Последний этап создания поддержки общих строк — обновление данных листа. В ячейках, содержащих встроенные строки, имеется указатель типа *t*. Теперь встроенные строки нужно заменить общими. Для этого ячейкам назначают тип *shared string*, указав вместо *t* значение *s*. Хороший пример — строка с «шапкой» таблицы. В ней ссылку на общие строки обеспечивает такой код:

```
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
  <sheetData>
    <row r="1" spans="1:4">
      <c r="A1" t="s">
        <v>0</v>
      </c>
      <c r="B1" t="s">
        <v>1</v>
      </c>
      <c r="C1" t="s">
        <v>2</v>
      </c>
      <c r="D1" t="s">
        <v>3</v>
      </c>
    </row>
    ...еще 24 строки...
  </sheetData>
</worksheet>
```

#### Пример 78. Ссылки на таблицу общих строк

## Общие формулы

Для формул существует оптимизация, аналогичная общим строкам. Подобно строковым значениям, формулы часто повторяются в диапазонах ячеек, например, при расчетах содержимого столбцов. При этом формулы различаются лишь незначительно (например, номерами строк, содержащих исходные значения). Клиенту проще загрузить в память первую формулу и вывести из нее остальные, чем по отдельности загружать повторяющиеся формулы. В отличие от общих строк общие формулы хранятся не в отдельном компоненте пакета, а вместе с данными листа. Обычно формулу объявляют, помещая в ячейку элемент *f*, заключая выражение формулы между открывающим и замыкающим тегами этого элемента. Элемент общей формулы имеет дополнительный атрибут, который объявляет формулу общей подобно объявлению типа ячейки. В первой ячейке с общей формулой хранятся три значения. Это обычный текст формулы и ссылка на диапазон ячеек, содержащий подобные формулы. Собственно,

формулу объявляют с помощью атрибута *si*. В следующей ячейке, содержащей эту формулу, данный атрибут имеет то же значение, но текста формулы там нет — достаточно указать его в первой ячейке.

```
<row>
  <c>
    <v>1</v>
  </c>
  <c>
    <f t="shared" ref="C2:C4" si="0">A2 + 1</f>
    <v>2</v>
  </c>
</row>
<row>
  <c>
    <v>2</v>
  </c>
  <c>
    <f t="shared" si="0" />
    <v>3</v>
  </c>
</row>
```

Пример 79. Общие формулы

## Таблицы

Таблица в SpreadsheetML — логическая конструкция, определяющая принадлежность некоторых диапазонов данных к одному набору. В SpreadsheetML и так используется модель таблицы для определения данных в виде строк и столбцов, однако можно дополнительно объявить подмножество содержимого листа как таблицу (*table*) и определить ряд его свойств, удобных для анализа этой таблицы. Таблицы SpreadsheetML позволяют выполнять над данными действия, невозможные при иных способах форматирования, рассматривавшихся ранее. Примерами могут служить фильтрация, форматирование и связывание с данными. Подобно иным конструкциям SpreadsheetML, каждая таблица, объявленная на листе, хранится в отдельном компоненте пакета. Компонент, содержащий таблицу, не хранит табличных данных. Эти данные всегда находятся в ячейках листа (о способе их хранения см. выше). Можно сказать, что лист, отображающий данные, не «подозревает» о существовании таблицы. Чтобы создать таблицу SpreadsheetML, в пакете нужно создать отдельный компонент. На этот компонент должен ссылаться лист, отображающий таблицу. Для этого служат такие типы:

	A	B	C	D
1	Region	Sales Person	Sales Quota	Sales
2	Southeast	Stephen Jiang	0	720315
3	Southwest	Stephen Jiang	0	1759212

Тип содержимого для таблицы:

*application/vnd.openxmlformats-officedocument.spreadsheetml.table+xml*

Тип связи для таблицы:

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/table>

Этот компонент содержит определение единственной таблицы. Если на листе объявлено несколько таблиц, для каждой из них существует собственный компонент. Корневым элементом этого компонента является элемент *table*. В простейшем варианте в нем указывают имена столбцов, составляющих таблицу и, возможно, пустой автофильтр (подробнее об автофильтрах — чуть ниже). Если опустить элемент автофильтра, кнопки фильтрации столбцов в программе-клиенте будут деактивированы.

```
<table xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
  id="1" name="Table1" displayName="Table1" ref="A1:D25">
  <autoFilter ref="A1:D25" />
  <tableColumns count="4">
    <tableColumn id="1" name="Region" />
    <tableColumn id="2" name="Sales Person" />
    <tableColumn id="3" name="Sales Quota" />
    <tableColumn id="4" name="Sales" />
  </tableColumns>
</table>
```

Пример 80. Простая таблица

У элемента *table* имеется несколько атрибутов, идентифицирующих таблицу и охватываемый ею диапазон данных. Атрибут *id* должен быть уникальным среди всех компонентов с таблицами; то же верно для атрибутов *name* и *displayName*. У *displayName* имеется дополнительное ограничение: он должен быть уникальным среди всех имен, объявленных в книге. Дело в том, что, как будет сказано ниже, имена можно назначать самым разным элементам, включая ячейки и формулы. Значение атрибута *name* применяется в объектной модели Excel. Значение *displayName* используется для ссылки в формулах. Атрибут *ref*

идентифицирует диапазон ячеек, охватываемый таблицей. К нему относятся не только ячейки с данными, но и «шапка».

Чтобы добавить столбцы к таблице, надо объявить дополнительные элементы *tableColumn* в контейнере *tableColumns*. Как и в таблице общих строк, последний поддерживает счетчик числа столбцов.

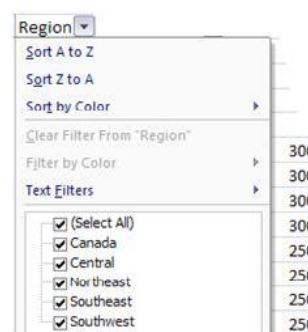
Помимо определения таблицы в хранящем ее компоненте, в компоненте пакета, хранящем лист, надо указать, какие таблицы отображаются на этом листе. Для этого компонент с листом поддерживает особый элемент *tableParts*. Ссылки на компоненты с таблицами осуществляются по ID связи, поддерживается и счетчик этих компонентов. Ниже приводится выдержка из прилагаемого листинга, в котором для экономии места опущены элементы данных. Для ссылки на таблицу достаточно добавить элемент *tableParts* (естественно, после создания и сохранения компонента с таблицей).

```
<worksheet xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main">
  ...
  <tableParts count="1">
    <tablePart r:id="rId1" />
  </tableParts>
</worksheet>
```

Пример 81. Ссылка в листе на компонент, содержащий таблицу

## Автофильтры

Если таблица велика, имеет смысл выборочно отображать ее содержимое, отбирая его по разным критериям. В модели таблиц SpreadsheetML для этого предусмотрена поддержка автофильтров. Эта функция позволяет назначать столбцам критерии для фильтрации по значению, диапазону значений и даже по цвету ячеек.



Всего поддерживается шесть типов фильтров, и любой из них можно применить к каждому столбцу таблицы.

Тип фильтра	Элемент	Критерий
<b>value</b>	filters	Список ячеек с значениями
<b>color</b>	colorFilter	Цвет шрифта или заливки ячейки
<b>icon</b>	iconFilter	Набор значков или значок из набора
<b>top items</b>	top10	Число или процентная доля элементов
<b>dynamic</b>	dynamicFilter	Динамическое значение, например, текущая дата
<b>custom</b>	customFilters	Диапазон, заданный логическим оператором

Рис. 22. Типы фильтров

Как правило, автофильтры объявляют в компоненте с таблицей. В нем определяются фильтры для каждого из столбцов, но требуется внести изменения и в данные листа. Рассмотрим сначала определение простого автофильтра. Чтобы создать автофильтр, достаточно добавить контейнер *autoFilter* к элементу *table* в компоненте, хранящем таблицу. В этом контейнере хранятся параметры фильтров для всех столбцов. Каждому столбцу разрешается назначать единственный фильтр — столбцы, не имеющие фильтров, здесь можно не объявлять. Атрибут *ref* указывает диапазон ячеек, на который действует фильтр, объявленный в *autoFilter*. Обычно это вся таблица, включая скрытые ячейки. Для каждого столбца с фильтром создают элемент *filterColumn*, в котором столбец указывают с помощью атрибута *colId*. У первого столбца ID равен 0, у второго — 1 и т. д. Последнее действие — определение типа фильтра с помощью элемента *filterColumn*.

```

<autoFilter ref="A1:D25">
  <filterColumn colId="0">
    <filters>
      <filter val="Southeast" />
      <filter val="Northeast" />
    </filters>
  </filterColumn>
</autoFilter>

```

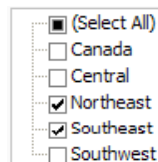


Рис. 23. Фильтр по значению

Чтобы фильтр заработал, просто объявить автофильтр в компоненте с таблицей недостаточно. Данные листа должны отражать действие фильтра, выборочно скрывая заданные строки. Для этого у элемента *row* есть атрибут *hidden*. Это важно для программной работы с фильтрами. При модификации параметров фильтра нужно пересчитать и обновить атрибуты строк. Поддерживается сложная разметка, включая формулы.

```

<row hidden="1">
  ...
</row>

```

### Пример 82. Скрытие строк

Если ваше приложение генерирует Open XML-код на основе источников бизнес-данных, скорее всего можно включить автофильтры, чтобы не заботиться о применении фильтров и не скрывать строки.

Ниже показан пример реализации простых фильтров. Фильтры применяются к первому столбцу таблицы.

```

<filterColumn colId="0">
  <top10 percent="1"
    val="20" filterVal="5"/>
</filterColumn>

```

Критерий «top10» позволяет вывести соответствующие элементы, заданные числом либо процентной долей. Атрибут *percent* — это булев переключатель, определяющий режим. Атрибут *val* указывает процентную долю, например 'top 20%'; значение *filterVal* используется для сравнения.

```

<filterColumn colId="0">
  <filters>
    <filter val="3" />
    <filter val="4" />
    <filter val="5" />
  </filters>
</filterColumn>

```

Это жестко прописанные критерии фильтрации. Согласно им отображаются только ячейки первого столбца со значениями 3, 4 или 5.

```

<filterColumn colId="0">
  <customFilters and="1">
    <customFilter
      operator="greaterThan"
      val="2" />
    <customFilter
      operator="lessThan"
      val="4" />
  </customFilters>
</filterColumn>

```

Пользовательский фильтр позволяет соединять пары логических операторов. Для атрибута *and* можно указать режим *and* («И») либо *or* («ИЛИ»). Этот пример выводит все строки со значением больше 2 и меньше 4.

```

<filterColumn colId="0">
  <dynamicFilter
    type="today"
    val="39206"
    maxVal="39207" />
</filterColumn>

```

Динамический критерий, например *today* (текущая дата), составляет динамический фильтр. Атрибут *type* содержит название встроенного типа, а элемент значения служит для кэширования.

## Вычисляемые столбцы

Значения столбцов электронных таблиц часто рассчитывают по формулам. Примером может быть столбец с разностью фактического и планового объемов продаж, по которой легко увидеть отставание или перевыполнение плана продаж. Вычисляемые столбцы объявляют в определении таблицы, подобно обычным столбцам.

Чтобы создать столбец с формулой, к определению столбца надо добавить элемент *calculatedColumnFormula*. Формулу задают как текстовое содержимое.

```
<table xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
  id="1" name="Table1" displayName="Table1" ref="A1:D25">
  <autoFilter ref="A1:D25" />
  <tableColumns count="5">
    <tableColumn id="1" name="Region" />
    <tableColumn id="2" name="Sales Person" />
    <tableColumn id="3" name="Sales Quota" />
    <tableColumn id="4" name="Sales" />
    <tableColumn id="5" name="Exceeding Sales">
      <calculatedColumnFormula>[Sales] - [Sales Quota]</calculatedColumnFormula>
    </tableColumn>
  </tableColumns>
</table>
```

Пример 83. Вычисляемый столбец

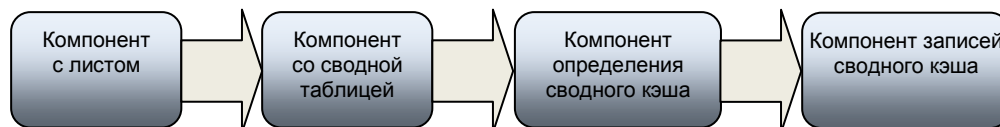
## Сводные таблицы

Теперь данные электронной таблицы определены, так что можно приступить к их анализу при помощи сводной таблицы. В демонстрационный документ включена простая сводная таблица, в строках которой стоят регион и имя торгового представителя, а в столбцах — суммарные объемы продаж и суммарные нормы. Завершает сводную таблицу строка с общим итогом. В этом разделе мы создадим сводную таблицу «с нуля».

	A	B	C
1		Values	
2	Row Labels	Sum of Sales Quota	Sum of Sales
3	Canada	1000000	1070895
4	Garrett Vargas	1000000	1070895
5	Central	900000	907089
6	Michael Blythe	900000	907089
7	Northeast	750000	392153
8	Jillian Carson	750000	392153
9	Southeast	1450000	2461062
10	Jillian Carson	250000	241908
11	Stephen Jiang	0	1236962
12	Tsvi Reiter	1200000	982192
13	Southwest	1300000	4803321
14	Linda Mitchell	1000000	1880645
15	Michael Blythe	300000	459466
16	Stephen Jiang	0	2463210
17	Grand Total	5400000	9634520

## Структура сводной таблицы

Сводная таблица — один из самых сложных элементов пакета SpreadsheetML. Чтобы ее построить, нужно создать три компонента и связать их друг с другом. Также понадобится дополнительная связь для рабочей книги. Иерархия связей показана на рисунке:



Изучение иерархии проще начать с определения сводного кэша (pivot cache definition). В этом компоненте содержатся определения всех полей сводной таблицы. При создании сводной таблицы на основе обычной таблицы в определении сводного кэша каждый столбец становится полем. В сводном кэше содержатся определения полей и информация о типе содержимого конкретного поля. В нем также хранится ссылка на источник данных, чтобы сводный кэш можно было обновлять одновременно с кэшированными данными из компонента записей сводного кэша.

Данные, отображаемые в сводной таблице, хранятся в двух местах. В компоненте записей сводного кэша содержатся реальные данные для сводной таблицы. В ячейках рабочего листа также хранится кэшированная версия данных, но она служит исключительно для отображения. Есть два способа хранения данных в компоненте записей сводного кэша. Уникальные значения для области данных сводной таблицы кэшируются на месте (inline). Для повторяющихся элементов строк и столбцов сохраняются ссылки. Сами

общие данные хранятся в определении сводного кэша. Каждая запись компонента записей сводного кэша состоит из N значений, где N — число полей, заданных в определении сводного кэша.

Наконец, в единое целое все эти компоненты связывает сама сводная таблица. В этом компоненте хранится информация о том, какие поля должны стоять в разных частях сводной таблицы. Есть четыре варианта размещения поля: строка, столбец, область данных и область фильтра. Поля выбираются из кэшированных полей определения сводного кэша.

Чтобы построить настоящую сводную таблицу, которая, будучи открытой, уже готова к использованию, вы должны создать и разметку ячеек таблицы. Сводная таблица отображается в обычных ячейках рабочего листа, и об этом вы тоже должны позаботиться. Можно сделать и так, чтобы клиентское ПО обновляло сводную таблицу при открытии документа.

## Создание основы для сводной таблицы

Для создания даже простой сводной таблицы требуется много компонентов. В предыдущем разделе описаны три новых компонента, но и уже существующие компоненты тоже должны быть обновлены.

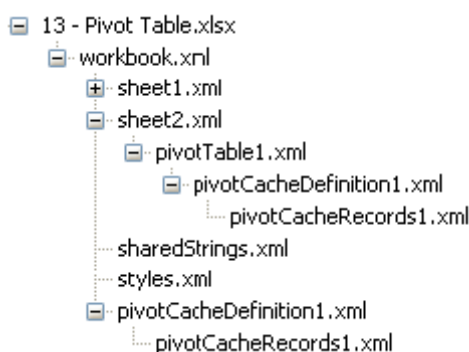
Первый шаг — создание базовой файловой структуры. Сохраняйте три новых компонента внутри пакета. Далее приведен список типов содержимого, которые понадобятся для создания структуры.

Компонент	Тип содержимого
Сводная таблица	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotTable+xml
Определение кэша	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheDefinition+xml
Записи кэша	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheRecords+xml

Следующий шаг перед добавлением содержимого — связывание компонентов. Настраиваются четыре связи. Вы должны создать описанную выше иерархию, а также связать рабочую книгу с определением сводного кэша.

Из	В	Тип связи
Рабочая книга	Определение кэша	http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotCacheDefinition
Рабочий лист	Сводная таблица	http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotTable
Сводная таблица	Определение кэша	http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotCacheDefinition
Определение кэша	Записи кэша	http://schemas.openxmlformats.org/officeDocument/2006/relationships/pivotCacheRecords

Это как раз тот случай, когда картинка стоит многих слов в таблице. Итоговая структура такова:



## Создание определения сводного кэша

Структура сводной таблицы построена, и теперь можно создать простой сводный кэш. На этом этапе создается много элементов, но мы для простоты ограничимся двумя. В сводном кэше будет определен источник данных для сводной таблицы, чтобы ее можно было обновлять, а также список полей данных. Данные в учебной электронной таблице уже есть. Важно отметить, что в кэше определяются все доступные поля для сводной таблицы, а не только те, что действительно используются. Эта информация будет позже определена в компоненте со сводной таблицей. Сейчас будет показано, как построить определение сводного кэша.

Просматривая приведенное ниже определение сводного кэша, прежде всего обратите внимание на корневой элемент. Идентификатор связи, использованный в списке атрибутов, указывает на компонент записей сводного кэша.



```
<pivotCacheDefinition
xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships
r:id="rId1">
<!-- между этими тегами вставляется остальная часть определения кэша... -->
</pivotCacheDefinition>
```

Внутри самого корневого элемента содержится определение источника данных — ссылка на данные, отображаемые в сводной таблице. Эта же информация хранится и в компоненте записей кэша, чтобы таблицу можно было обновлять даже при отсутствии подключения к источнику данных. Хранить данные в ячейках сводной таблицы не следует, так как содержимое этих ячеек по природе своей преходяще и изменяется при любом построении сводной таблицы. Доступные источники данных весьма разнообразны: рабочие листы, базы данных, кубы OLAP, другие сводные таблицы и пр. Наша простая сводная таблица будет извлекать данные из первого листа учебной электронной таблицы. Сама сводная таблица будет храниться на отдельном листе.

```
<cacheSource type="worksheet">
<worksheetSource name="Table3" />
</cacheSource>
```

В последней части определения кэша задаются поля источника данных. Наша таблица содержит четыре столбца, которым соответствуют четыре элемента *cacheField*. Элемент *cacheField* применяется в двух целях: задает тип данных и форматирование поля, а также используется в качестве кэша для общих строк. Помните, как работает таблица с общими строками? Здесь все происходит так же. Сводные значения хранятся в компоненте записей сводного кэша. Если в качестве значения используется повторяющаяся строка, в запись кэша подставляется ссылка на коллекцию общих элементов *cacheField*. Поскольку в первом и втором полях хранятся метки строк — повторяющиеся элементы, — в двух этих полях используется коллекция *sharedItems*. Оставшиеся два поля — суммарные нормы и объемы продаж. Эти значения уникальны для каждой ячейки, и потому использование общих элементов не принесет никакой выгоды.

```
<cacheFields count="4">
<cacheField name="Region">
<sharedItems count="5">
<s v="Southeast" />
<s v="Southwest" />
<s v="Central" />
<s v="Northeast" />
<s v="Canada" />
</sharedItems>
</cacheField>
<cacheField name="Sales Person">
<sharedItems count="6">
<s v="Stephen Jiang" />
<s v="Michael Blythe" />
<s v="Linda Mitchell" />
<s v="Jillian Carson" />
<s v="Garrett Vargas" />
<s v="Tsvi Reiter" />
</sharedItems>
</cacheField>
<cacheField name="Sales Quota">
<sharedItems
containsSemiMixedTypes="0" containsString="0"
containsNumber="1" containsInteger="1"
minValue="0" maxValue="300000" />
</cacheField>
<cacheField name="Sales">
<sharedItems
containsSemiMixedTypes="0" containsString="0"
containsNumber="1" containsInteger="1"
minValue="1374" maxValue="1759212" />
</cacheField>
</cacheFields>
```

Понятно, что здесь мы рассматриваем очень простую сводную таблицу. В определение сводного кэша можно включить много других параметров, описание которых выходит за рамки этой книги.

## Определение записей кэша

Второй компонент сводной таблицы применяется исключительно в качестве кэша данных. В компоненте записей кэша разрешается хранить любое число кэшированных записей. В каждой записи определено столько же значений, сколько полей есть в определении кэша. Записи кэша построены просто: каждая



запись определяется элементом *r*. Элементы данных в записи хранятся как дочерние элементы. Вы можете использовать несколько «типизированных» значений, например числовые, логические, «дата-время» или ссылки на общие элементы.

В первую запись кэша включены две ссылки на общие элементы (с помощью дочернего элемента *x*) и два численных значения (с помощью элемента *v*). Значения совпадают со значениями из первой строкой таблицы, используемой в качестве источника данных. Остальная часть компонента заполнена записями того же типа. Чтобы сэкономить место, 22 записи опущены.

```
<pivotCacheRecords
  xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
  xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
  count="24">
  <r>
    <x v="0" />
    <x v="0" />
    <n v="0" />
    <n v="720315" />
  </r>
  <r>
    <x v="1" />
    <x v="0" />
    <n v="0" />
    <n v="1759212" />
  </r>
  <!-- еще 22 записи -->
</pivotCacheRecords>
```

Пример 84. Записи сводного кэша

## Ссылка на кэш из рабочей книги

Прежде чем мы посредством сводной таблицы свяжем воедино все кэшированные поля, нужно создать в компоненте рабочей книги ссылку на определение сводного кэша. Это довольно просто. Имеется идентификатор связи, соответствующий компоненту, содержащему кэш, и уникальный номер, идентифицирующий сам кэш. В нашем документе сводная таблица отображается на отдельном листе, на который также имеется ссылка из компонента с рабочей книгой.

```
<workbook
  xmlns="http://schemas.openxmlformats.org/spreadsheetml/2006/main"
  xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships">
  <sheets>
    <sheet name="Data" sheetId="1" r:id="rId1" />
    <sheet name="Pivot" sheetId="2" r:id="rId2" />
  </sheets>
  <pivotCaches>
    <pivotCache cacheId="1" r:id="rId5" />
  </pivotCaches>
</workbook>
```

Пример 85. Лист рабочей книги со сводными кэшами

## Создание сводной таблицы

Последний этап построения сводной таблицы — создание компонента сводной таблицы. Этот компонент был добавлен ранее с помощью специальных типов содержимого и связи, на которые имеются ссылки с листа, содержащего сводную таблицу.

Главное назначение компонента со сводной таблицей — хранить сведения о том, какие поля на какой оси сводной таблицы отображаются и в каком порядке. В определении сводной таблицы можно включить еще много параметров, но здесь мы для простоты обсуждаем лишь самую простую структуру. Сводная таблица создается в несколько этапов.

Тут ничего сложного, если пока забыть о дополнительных элементах. Первым делом обратим внимание на корневой элемент. В нем задается имя сводной таблицы, которое позволит использовать ее в качестве источника данных. Кроме того, в корневом элементе содержится ссылка на сводный кэш по идентификатору, добавленному в компонент рабочей книги, и определяется заголовок, который будет отображаться над областью данных сводной таблицы. Необходимы все элементы.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<pivotTableDefinition
  xmlns=http://schemas.openxmlformats.org/spreadsheetml/2006/main
  name="PivotTable1" cacheId="1" dataCaption="Values">
  <!-- между этими тегами вставляется остальная часть
```

```
определения сводной таблицы... -->
</pivotTableDefinition >
```

Определение *pivotTableDefinition* включает в себя три основных компонента: сведения о расположении таблицы, об отображении кэшированных полей и, наконец, о расположении кэшированных полей.

Размещение сводной таблицы нужно рассматривать с двух точек зрения. Во-первых, она отображается в конкретной позиции некоего рабочего листа. В пакете этот рабочий лист связан с соответствующим компонентом сводной таблицы. Для указания диапазона ячеек, занимаемых сводной таблицей, применяется тег *location*. Во-вторых, вы должны задать еще три параметра, чтобы указать, в какой строке, считая от верхней границы таблицы, отображается ее заголовок, а также в какой строке и в каком столбце располагается первый элемент данных. Эта информация требуется для корректного обновления сводной таблицы. Без нее клиентская программа не сможет различить ячейки сводной таблицы.

```
<location ref="A1:C17" firstHeaderRow="1" firstDataRow="2" firstDataCol="1" />
```

Когда расположение сводной таблицы задано, можно приступить к ее построению, используя поля из определения сводного кэша. Прежде всего нужно через элемент *pivotFields* определить коллекцию полей, которые будут отображены в сводной таблице. Каждое поле служит кэшем для информации из соответствующего поля источника данных. Определять этот кэш вам не нужно. Воспользуйтесь вместо этого элементом *default*, чтобы клиентское ПО обновляло таблицу при открытии документа. Атрибут *showAll* служит для сокрытия некоторых элементов конкретного измерения. Рассмотрим в качестве примера сводную таблицу, в которой итоги подводятся по критериям *продажи / продажи-представитель / продажа-регион*. Что делать, если данный торговый представитель не работал в конкретном регионе? Отменять ли его отображение? Этот выбор производится посредством параметра *showAll*.

```
<pivotFields count="4">
  <pivotField axis="axisRow" showAll="0">
    <items count="1">
      <item t="default" />
    </items>
  </pivotField>
  <pivotField axis="axisRow" showAll="0">
    <items count="1">
      <item t="default" />
    </items>
  </pivotField>
  <pivotField dataField="1" numFmtId="164" showAll="0" />
  <pivotField dataField="1" numFmtId="164" showAll="0" />
</pivotFields>
```

Задав поля, входящие в таблицу, их нужно распределить по четырем областям таблицы.

```
<rowFields count="2">
  <field x="0" />
  <field x="1" />
</rowFields>
<colFields count="1">
  <field x="-2" />
</colFields>
<dataFields count="2">
  <dataField name="Sum of Sales Quota" fld="2" />
  <dataField name="Sum of Sales" fld="3" />
</dataFields>
```

## Подводя итоги

Более подробное обсуждение возможностей сводной таблицы придется отложить до другого раза и другой книги. Есть множество параметров, позволяющих значительно расширить функциональность сводной таблицы. К вашим услугам стили и условное форматирование, позволяющие разнообразить отображение данных. Фильтруйте и сортируйте данные с помощью определений полей, работайте с иерархиями данных, уже имеющимися в источниках, подобных кубам OLAP. За дополнительной информацией для начала обратитесь к третьей части спецификаций Open XML, а также к примерам из Markup Language Reference.

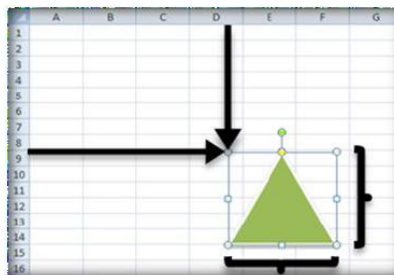
## Создание и размещение диаграммы

Компонент контейнера закрепляет диаграмму на листе и определяет место, где она будет отображаться, а также как диаграмма будет реагировать на изменение размеров строк и столбцов. Имеется три метода привязки: абсолютный, привязка по одной ячейке и привязка по двум ячейкам. Все три опираются на систему значений DrawingML. Абсолютные координаты и смещения измеряются в EMU (English Metric Unit). Преобразование EMU в другие единицы, например дюймы или сантиметры, описано в главе, посвященной DrawingML.

Проще всего работать с абсолютной привязкой (absolute anchor). Она заключается в указании положения верхнего левого угла графического элемента, в частности диаграммы. Кроме того, вы задаете ширину и высоту графической области. Внутри элемента *absoluteAnchor* имеется элемент *graphicFrame*, содержащий ссылку на отдельный компонент с данными диаграммы.

```
<xdr:absoluteAnchor>
  <xdr:pos x="2276475" y="1552575" />
  <xdr:ext cx="1238250" cy="1238250" />
  <xdr:graphicFrame>
    ...
  </xdr:graphicFrame>
  <xdr:clientData />
</xdr:absoluteAnchor>
```

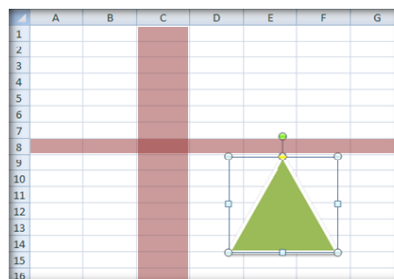
Пример 86. Абсолютная привязка



Второй метод — привязка по одной ячейке (one cell anchor). В этом случае вы задаете расположение, размер и содержимое привязки. Расположение задается путем указания номеров строки и столбца (начиная с 0). Задав строку и столбец, можно при желании сместить верхний левый угол графического элемента на заданное расстояние относительно позиции привязки. Кроме того, нужно задать размер и содержимое графической области. О содержимом мы поговорим попозже, когда рассмотрим третий метод привязки (по двум ячейкам).

```
<xdr:oneCellAnchor>
  <xdr:from>
    <xdr:col>3</xdr:col>
    <xdr:colOff>457200</xdr:colOff>
    <xdr:row>7</xdr:row>
    <xdr:rowOff>104775</xdr:rowOff>
  </xdr:from>
  <xdr:ext cx="1234567" cy="1234567" />
  <xdr:graphicFrame>
    ...
  </xdr:graphicFrame>
  <xdr:clientData />
</xdr:oneCellAnchor>
```

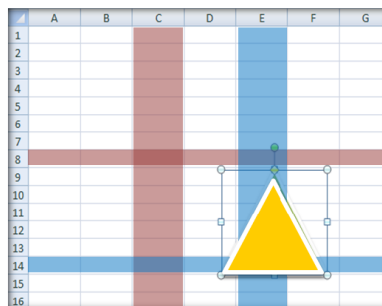
Пример 87. Привязка по одной ячейке



Привязка по двум ячейкам (two cell anchor) похожа на привязку по одной ячейке, но в данном случае вы задаете не абсолютный размер элемента, а нижнюю правую ячейку, к которой он также привязан. Размер графической области будет меняться в зависимости от положения и размера верхней левой и нижней правой ячеек привязки. Помимо самих ячеек, вы можете указать и смещение элемента относительно них.

```
<xdr:twoCellAnchor>
  <xdr:from>
    <xdr:col>5</xdr:col>
    <xdr:colOff>438150</xdr:colOff>
    <xdr:row>11</xdr:row>
    <xdr:rowOff>9525</xdr:rowOff>
  </xdr:from>
  <xdr:to>
    <xdr:col>13</xdr:col>
    <xdr:colOff>133350</xdr:colOff>
    <xdr:row>25</xdr:row>
    <xdr:rowOff>85725</xdr:rowOff>
  </xdr:to>
  <xdr:graphicFrame macro="">
    ...
  </xdr:graphicFrame>
  <xdr:clientData />
</xdr:twoCellAnchor>
```

Пример 88. Привязка по двум ячейкам



Во всех трех методах объект, для которого задается привязка, хранится внутри содержимого элемента. Элемент *graphicFrame* практически одинаков во всех трех языках Open XML. Применение графических областей описано главным образом в главе, посвященной PresentationML. Здесь же достаточно сказать, что вы задаете очевидные значения: идентификатор, размер и положение. Графический элемент — многоцелевой контейнер, используемый здесь для хранения диаграммы. Элемент *chart* идентифицирует отображаемую диаграмму по конкретному идентификатору связи. Содержимым компонента, ссылка на

который помещена в *graphicFrame*, является диаграмма DrawingML, подробно обсуждаемая далее в этой главе.

```
<xdr:graphicFrame>
  <xdr:nvGraphicFramePr>
    <xdr:cNvPr id="2" name="Chart 1" />
    <xdr:cNvGraphicFramePr />
  </xdr:nvGraphicFramePr>
  <xdr:xfrm>
    <a:off x="0" y="0" />
    <a:ext cx="0" cy="0" />
  </xdr:xfrm>
  <a:graphic>
    <a:graphicData uri="http://.../drawingml/2006/chart">
      <c:chart xmlns:c="http://.../drawingml/2006/chart"
        xmlns:r="http://.../officeDocument/2006/relationships"
        r:id="rId1" />
    </a:graphicData>
  </a:graphic>
</xdr:graphicFrame>
```

Пример 89. Графическая область SpreadsheetML

## Применение стилей

Как и во всех остальных языках Open XML, вы можете обеспечить единообразие форматирования ячеек с помощью стилей. В отличие от WordprocessingML прямое форматирование в разметке рабочего листа не допускается. Информация о форматировании всегда хранится отдельно. Стили SpreadsheetML позволяют форматировать ячейки и таблицы. Они хранятся внутри пакета в отдельном компоненте. Вы, наверное, уже освоились с процессом сохранения в пакете новых компонентов. Прежде всего создайте файл XML для хранения информации о стилях. Обновите компонент типов содержимого, чтобы в нем содержалась корректная информация о типе, и, наконец, добавьте связь между рабочей книгой и компонентом стилей. Одни и те же стили применяются на всех рабочих листах. Во всем пакете электронной таблицы имеется только один компонент стилей. Необходимо применять следующие типы содержимого и связи:

*Тип содержимого для стилей:*

*<application/vnd.openxmlformats-officedocument.spreadsheetml.styles+xml>*

*Тип связи для стилей:*

*<http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles>*

## Структура компонента стилей

Разнообразие разделов в компоненте стилей поначалу может несколько обескуражить. Такая структура призвана по максимуму обеспечить возможность повторного использования стилизованных параметров. Если сократить запись всех элементов и просто взглянуть на общую структуру, то вы увидите нечто вроде:

```
<styleSheet xmlns="http://.../spreadsheetml/2006/main">
  <fonts />
  <fills />
  <borders />
  <numFmts />
  <cellStyleXfs />
  <cellXfs />
  <cellStyles />
  <dxfs />
  <tableStyles />
</styleSheet>
```

Пример 90. Компонент стилей SpreadsheetML

Собственно стиливые параметры определяются в четырех элементах внутри элемента *styleSheet* компонента стилей: *font*, *fills*, *borders* и *numFmts*. По их именам можно понять, что они содержат параметры шрифта, заливки, границ и нумерации. Каждый из них представляет собой контейнер для перечисленных свойств и может содержать несколько определений.

Прямых ссылок на первые четыре элемента из ячеек или другого содержимого, к которому применяются стили, не бывает. Вместо этого поддерживаются различные типы записей форматирования (*formatting records*). Когда вы применяете стиль к ячейке, создается ссылка на элемент коллекции *cellXfs*. Эта коллекция содержит форматирование, применяемое непосредственно к ячейкам. Коллекция *cellXfs* в свою очередь ссылается на элемент коллекции *cellStyleXfs*. В этой второй коллекции хранятся параметры форматирования для глобальных стилей ячеек, например «normal». Можно считать *cellStyleXfs*

глобальным определением, а *cellXfs* — коррективами к нему. Ячейка ссылается на коррективы, а через них — также на глобально определенные параметры. И *cellStyleXfs*, и *cellXfs* обращаются к элементам коллекций шрифтов, заливок, границ и нумерации через записи форматирования.

## Применение стиля ячейки

Когда стиль создан, осталось самое простое — применить его к ячейкам. Для этого нужно лишь добавить еще один атрибут в элемент ячейки компонента рабочего листа.

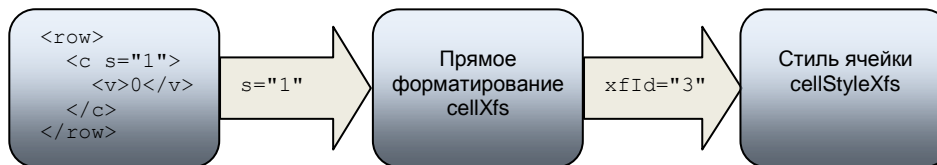
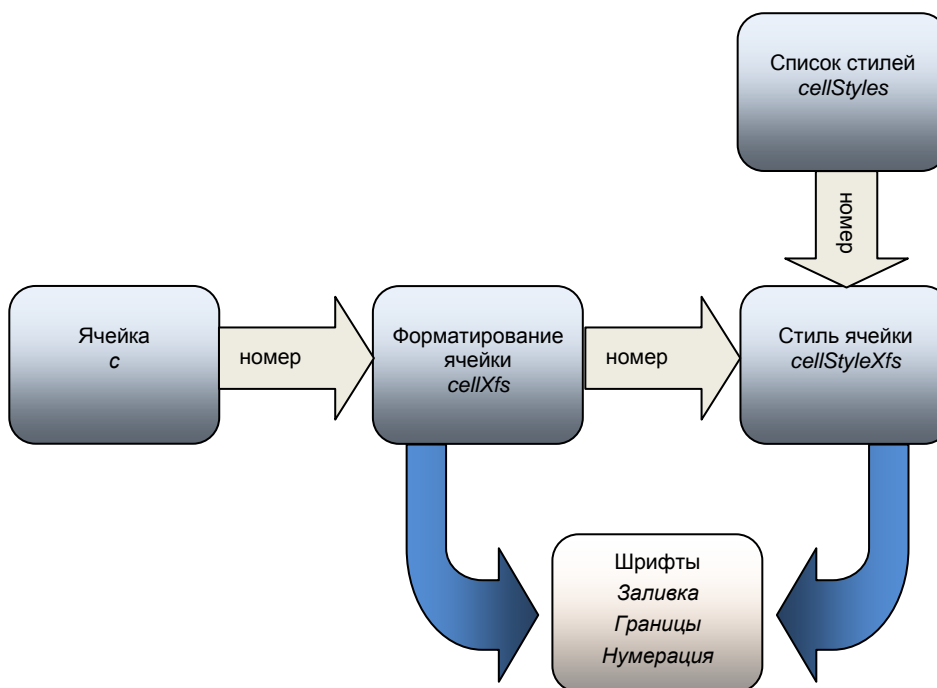


Рис. 24. Иерархия стилей ячеек

Работая с клиентским ПО, например Microsoft Office Excel, пользователь волен создавать новые стили для применения к ячейкам. Параметры форматирования, выбранные им, хранятся в стилевой коллекции *cellStyleXfs*. Данные об именах хранятся в отдельном списке *cellStyles*, который можно считать списком стилей, доступных пользователю. Для хранения параметров форматирования в этой коллекции поддерживается указатель на глобальную коллекцию стилей. Когда вы применяете стиль к ячейке, ссылка на *cellStyles* не создается. Вместо этого ссылка на стиль копируется в запись прямого форматирования внутри *cellXfs*.



## Шрифты, заливки, границы и нумерация

Все записи форматирования, идентифицируемые элементами *...Xfs*, ссылаются на определения форматирования, хранимые глобально в компоненте стилей. Обращение к шрифту, границе и заливке осуществляется по номеру, а к методу нумерации — по идентификатору. О шрифтовом разделе нужно сделать важное замечание. Пока не создан компонент стилей, используются параметры шрифта по умолчанию. Как только в компоненте стилей создан раздел шрифтов, во всей рабочей книге в качестве шрифта по умолчанию начинает применяться первый включенный в него шрифт.

Для форматирования ячеек демонстрационной электронной таблицы вам понадобятся два шрифта. Первый будет применяться по умолчанию, а второй — для текста в заголовке и примечаниях, набранного белым полужирным шрифтом. Еще у нас будет четыре типа заливки: пустая и три заливки с цветом для оформления таблицы. Наконец, мы создадим три определения границ: одно опять же пустое, второе для границы строки заголовков и третье для таблицы в целом.

## Определения границ

Список элементов границ поддерживается в элементе *borders*. Любая граница создается при помощи элемента *border*. Внутри тега *border* вы можете создать шесть границ, используя специализированные теги, например *left* или *bottom*. У каждой границы имеется список возможных типов, например *thin* или *double*. Задать точную толщину границы невозможно — она определяется типом. Цвет границы задается значением ARGB, состоящим из параметров красного, зеленого и синего цветов, а также альфа-канала (параметра прозрачности).

```
<borders>
  <border>
    <left style="thin">
      <color rgb="FFFFFFF" />
    </left>
  </border>
  <!--остальные границы пропущены-->
</borders>
```

Пример 91. Определение стиля границы

## Шрифты

Шрифты определяются примерно по тому же образцу, что и в WordprocessingML. Свойства каждого шрифта задаются при помощи элементов, подобных *sz* (размер), *name* (семейство) и *color* (цвет текста).

```
<font>
  <sz val="11" />
  <name val="Calibri" />
  <color rgb="00000000" />
</font>
<!--остальные шрифты пропущены-->
</font>
```

Пример 92. Определение стиля шрифта

## Заливки

Заливка состоит из фоновых цвета, основного цвета и узора. Цвета опять же задаются значениями ARGB. Список узоров вы найдете в спецификациях Open XML.

```
<fills>
  <fill>
    <patternFill patternType="solid">
      <fgColor rgb="FF4F81BD" />
      <bgColor rgb="FF1281F3" />
    </patternFill>
  </fill>
  <!--остальные заливки пропущены-->
</fills>
```

Пример 93. Определение стиля заливки

## Записи форматирования

Второй этап в форматировании ячейки с помощью стиля — создание записей форматирования со ссылками на определенные выше форматы. Поскольку наша электронная таблица проста, много записей форматирования нам создавать не придется, но в реальной жизни их будет гораздо больше. В нашем примере мы используем одну главную запись форматирования (*master formatting record*) и четыре записи прямого форматирования (*direct formatting record*). Последние четыре будут применяться во всех ячейках таблицы. Отдельные записи форматирования понадобятся для строки заголовков, для чередующихся строк и для пустой записи форматирования по умолчанию.

Чтобы создать записи форматирования, в компонент стилей нужно добавить две новые коллекции. Как вы помните, в коллекции *cellStyleXfs* содержится форматирование, связанное со стилем, а в коллекции *cellXfs* — дополнительное форматирование ячейки, произведенное после применения стиля.

	A
1	Region
2	Southeast
3	Southwest
4	Southwest
5	Southeast
6	Central
7	Central

Заголовок

Первая строка с чередованием

Вторая строка с чередованием

*Не думайте, что эти стили можно применять только в контексте таблицы. Стилль применяется непосредственно к ячейке, которая понятия не имеет о том, что является частью таблицы.*

В следующем примере разметки показано, как использовать две эти коллекции. Все значения являются указателями на элементы коллекций форматов, обсуждавшихся выше. Запись прямого форматирования в `cellXfs` указывает на коллекцию `cellStyleXfs` с помощью атрибута `xfld`.

```
<cellStyleXfs count="1">
  <xf fontId="0" fillId="0" borderId="0" />
</cellStyleXfs>
<cellXfs count="1">
  <xf fontId="0" fillId="0" borderId="0" xfId="0" />
  <xf fontId="1" fillId="4" borderId="1" xfId="0" />
  <xf fontId="0" fillId="3" borderId="2" xfId="0" />
  <xf fontId="0" fillId="2" borderId="2" xfId="0" />
</cellXfs>
```

Пример 94. Записи форматирования

## Дифференциальные записи форматирования

Текущая ситуация с форматированием ячеек не идеальна. Для применения корректного стиля нужно обойти все ячейки по отдельности. В SpreadsheetML имеется конструкция для оптимизации стилей — дифференциальная запись форматирования (differential formatting record, *dxf*), существенно повышающая эффективность применения стилей в таблице. Как и другие записи форматирования, *dxf* определяет некий формат. Главное отличие в том, что дифференциальная запись применяется в дополнение к форматированию ячейки, заданному примененным к ней стилем. Кроме того, она не содержит ссылок на первые четыре контейнера форматов, а определяет заливку, шрифт, нумерацию и границу прямо на месте.

```
<dxfs count="1">
  <dxf>
    <fill />
    <font />
    <numFmt />
    <border />
  </dxf>
</dxfs>
```

Пример 95. Дифференциальное форматирование

Чтобы создать дифференциальные записи форматирования, надо добавить элемент-контейнер *dxfs*. Потом включите в него по одной записи *dxf* для каждой из трех областей таблицы с различным оформлением — для заголовка и для двух видов чередующихся строк. Затем задайте нужные шрифты, заливки и границы, сгруппировав их в записях *dxf*.

Дифференциальные записи форматирования для стиля таблицы готовы. Теперь применение стилей в таблице значительно упростится.

## Стилль таблицы

Чтобы приблизиться к созданию стиля таблицы, нужно сделать еще два шага. Во-первых, создайте новый стиль в компоненте стилей, во-вторых, обновите компонент таблицы, добавив в него ссылку на новый стиль. Создание стиля таблицы похоже на создание записей форматирования. К вашим услугам коллекция `tableStyles` с отдельными элементами `tableStyle` для каждого определенного вами стиля таблицы. Как и в WordprocessingML, вы вольны разработать индивидуальное оформление для различных областей таблицы, например, строк заголовков, чередующихся строк и столбцов и угловых ячеек. В этом вам помогут дифференциальные записи форматирования. Каждая область таблицы ссылается на дифференциальную запись форматирования, в которой описано соответствующее оформление. Вот пример разметки стиля таблицы для нашей таблицы:

```

<tableStyles count="1">
  <tableStyle name="ReportTable" pivot="0" count="5">
    <tableStyleElement type="wholeTable" dxfid="0" />
    <tableStyleElement type="headerRow" dxfid="1" />
    <tableStyleElement type="totalRow" dxfid="2" />
    <tableStyleElement type="firstRowStripe" dxfid="3" />
    <tableStyleElement type="secondRowStripe" dxfid="4" />
  </tableStyle>
</tableStyles>

```

### Пример 96. Стили таблицы

Второй шаг — обновление компонента таблицы. В определение таблицы нужно вставить ссылку на табличный стиль. Ссылка осуществляется по имени. В компонент таблицы добавляется элемент *tableStyleInfo*, в котором определяется, какие стили и как использовать. Если в таблице применен стиль, это не означает, что обязательно нужно использовать все определение целиком. Если вам, например, не хочется применять особенное форматирование к строке с итогом, вы и не должны этого делать. Имеется несколько логических атрибутов, позволяющих указать, к каким разделам таблицы надо применить специальное форматирование. Взгляните на обновленное определение таблицы (пример 97):

```

<table xmlns="http://.../spreadsheetml/2006/main">
  <tableColumns count="4">
    <tableColumn id="1" name="Region" />
    <tableColumn id="2" name="Sales Person" />
    <tableColumn id="3" name="Sales Quota" />
    <tableColumn id="4" name="Sales" />
  </tableColumns>
  <tableStyleInfo name="My Table Style"
    showFirstColumn="0" showLastColumn="0"
    showRowStripes="1" showColumnStripes="0" />
</table>

```

### Пример 97. Применение стиля таблицы

## Условное форматирование

Условное форматирование данных в ячейках позволяет нагляднее структурировать информацию на рабочем листе. Гораздо проще оценить относительную важность показателей в таблице по их цвету, чем просто по числовым значениям. Есть несколько способов форматирования ячеек в зависимости от их содержимого. Можно выделить минимальное и максимальное значения, добавить в ячейку столбик, размер которого определяется ее значением, выделить цветом значения в различных диапазонах. Условное форматирование применяется непосредственно к ячейке рабочего листа. Она не должна быть частью таблицы, хотя чаще всего условное форматирование применяется именно к ячейкам таблиц. В примере на рис. 25 использовано цветовое форматирование: текущее минимальное значение отмечено красным цветом, текущее максимальное значение — зеленым. Цвета промежуточных значений определяются интерполяцией.

	A
1	1
2	2
3	3
4	4
5	5

Рис. 25. Применение условного форматирования

Параметры условного форматирования хранятся на уровне рабочего листа. Для каждого формата, примененного к набору ячеек, в листе имеется один элемент *conditionalFormatting*. Диапазон ячеек, к которым применяется формат, задается атрибутом *sqref*. Границы диапазона указываются в виде «от:до», например «A1:A10». Пример задания условного форматирования приведен ниже.

```

<worksheet>
  <sheetData>
    ...
  </sheetData>
  <conditionalFormatting sqref="A1:A5">
    <cfRule type="dataBar" priority="1">
      ...
    </cfRule>
    <cfRule type="colorScale" priority="2">
      ...
    </cfRule>
  </conditionalFormatting>
</worksheet>

```

### Пример 98. Правила условного форматирования

Для любого условного форматирования диапазона ячеек можно задать несколько правил, например, применить одновременно цветовой масштаб и столбики. Каждое правило представлено элементом *cfRule*. Приоритет правила задается атрибутом *priority*. Поскольку элемент *conditionalFormatting* может



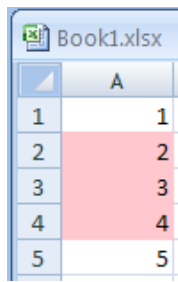
перекрывать другое форматирование, приоритет является глобальным для всех заданных условных форматов. Элемент *cfRule* позволяет применять довольно много различных типов форматирования, для определения параметров которых, однако, применяются общие элементы. Далее приведен список доступных типов условного форматирования.

Типы условного форматирования			
aboveAverage	beginWith	cells	colorScale
containsBlanks	containsErrors	containsText	dataBar
duplicateValues	endsWith	iconSet	notContainsBlanks
notContainsErrors	notContainsText	timePeriod	top10
uniqueValues			

Таблица 5. Типы условного форматирования

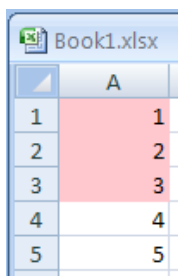
## Правила выделения

Первое правило условного форматирования — *cells*. Применяя его в зависимости от оператора и заданных значений, можно добиться разных результатов. Количество формул, необходимых для задания диапазона, определяется конкретным оператором, скажем, *between*, *greaterThan* или *notEqual*. Оператору, использованному в примере разметки, требуется две формулы. В данном случае в качестве формул использованы жестко заданные константы, но в более сложных сценариях можно применять полноценные формулы со ссылками на ячейки и пр. Оформление значений, попавших в заданный диапазон, осуществляется посредством дифференциальных записей форматирования, указанных в атрибуте *dxfld*. В данном примере числа, попавшие в диапазон, будут окрашены в красный цвет.

<pre>&lt;cfRule type="cellIs"   dxflId="0"   priority="1"   operator="between"&gt;   &lt;formula&gt;2&lt;/formula&gt;   &lt;formula&gt;4&lt;/formula&gt; &lt;/cfRule&gt;</pre>	<pre>&lt;dxfl&gt;   &lt;fill&gt;     &lt;patternFill&gt;       &lt;bgColor         rgb="FFFC7CE" /&gt;     &lt;/patternFill&gt;   &lt;/fill&gt; &lt;/dxfl&gt;</pre>	
--	---	--

## Правила для максимальных и минимальных значений

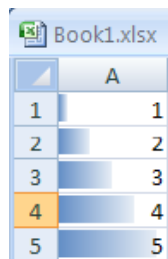
Для выделения N максимальных или минимальных значений применяется правило *top10*. Оформление и в этом случае задается при помощи дифференциальной записи форматирования. В приведенном ниже примере выделяются три минимальных значения. Атрибут *bottom* — логический переключатель. В атрибуте *rank* указывается количество значений. Чтобы выделить N процентов максимальных значений, применяйте логический атрибут *percent*. В этом случае в аргументе *rank* задается необходимое число процентов.

<pre>&lt;cfRule type="top10"   dxflId="0"   priority="1"   bottom="1"   rank="3" /&gt;</pre>	<pre>&lt;dxfl&gt;   &lt;fill&gt;     &lt;patternFill&gt;       &lt;bgColor         rgb="FFFC7CE" /&gt;     &lt;/patternFill&gt;   &lt;/fill&gt; &lt;/dxfl&gt;</pre>	
--	---	---

## Столбики

Относительный вклад различных значений можно отобразить при помощи столбиков (data-bar). У столбиков в разных ячейках один и тот же цвет, а длина их определяется значением в ячейке. Цветовая шкала, которую мы обсудим далее, также иллюстрирует разницу в значениях, но через изменения цвета. Для определения параметров столбика внутри соответствующего правила применяется отдельная модель. Все нужные данные хранятся в элементе *dataBar*. Отображение столбиков регулируется тремя параметрами: минимальным и максимальным значениями и цветом. Первый элемент *cfvo* (conditional format value object) определяет нижний предел, второй — верхний. Задавать предельное значение можно несколькими способами, например, непосредственно или по формуле. Другой распространенный способ — использование типов *cfvo* «min» и «max», задающих минимальное и максимальное значения в диапазоне, к которому применяется форматирование. В этом случае обеспечивается четкое спадание длины столбиков от максимального к минимальному значению.

```
<cfRule type="dataBar" priority="1">
  <dataBar>
    <cfvo type="formula" val="$D$2" />
    <cfvo type="number" val="5" />
    <color rgb="FF638EC6" />
  </dataBar>
</cfRule>
```

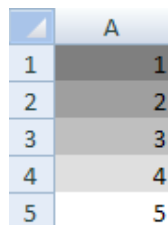


	A	D
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

## Цветовые шкалы

Цветовые шкалы применяются примерно так же, как и столбики. Они позволяют наглядно отобразить относительный перепад значений в ячейках. Подобно столбику, цветовой шкале также соответствует конкретный дочерний элемент в контейнере *cfRule*. Вы можете задать три параметра: начало, середину и конец шкалы. Возможен также двухпараметрический подход, в котором среднее значение шкалы не используется. В приведенном ниже примере шкала начинается минимальным значением диапазона ячеек. Вторая точка задается процентной долей — 50%. Заканчивается шкала максимальным значением диапазона. Каждой точке соответствует определенный цвет. Первый элемент *color* относится к первому *cfvo* и т. д.

```
<cfRule type="colorScale" priority="1">
  <colorScale>
    <cfvo type="min" val="0" />
    <cfvo type="percentile" val="50" />
    <cfvo type="max" val="0" />
    <color tint="-0.499984740745262" />
    <color tint="-0.249977111117893" />
    <color />
  </colorScale>
</cfRule>
```

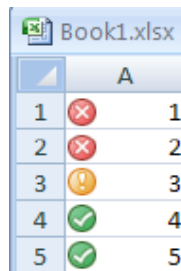


	A	D
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5

## Набор значков

Последний способ форматирования — использование набора значков. К элементам можно применять разные наборы значков. Чтобы определить, к каким ячейкам относится правило форматирования, используется примерно тот же диапазон значений. Первым элементом *cfvo* задается наименьшее значение, вторым — промежуточное, последним — максимальное. Элемент *iconSet* определяет применяемый набор значков. Доступные наборы значков описаны в спецификации Open XML.

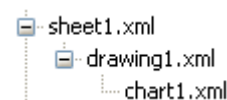
```
<cfRule type="iconSet" priority="1">
  <iconSet iconSet="3Symbols">
    <cfvo type="percent" val="0" />
    <cfvo type="percent" val="33" />
    <cfvo type="percent" val="67" />
  </iconSet>
</cfRule>
```



	A	D
1	✘	1
2	✘	2
3	⚠	3
4	✔	4
5	✔	5

## Листы диаграмм

Помимо обычного рабочего, есть еще два вида листов. На листах окна диалога создаются диалоговые окна, а на листах диаграмм отображаются диаграммы. Окна диалога выходят за рамки этой книги, а вот о листах диаграмм мы поговорим.



Разметка DrawingML, применяемая в диаграммах, позволяет использовать одни и те же диаграммы во всем спектре языков Open XML. Для отображения диаграмм в разных языках разметки используются специфические контейнеры. В языке SpreadsheetML таким контейнером является отдельный компонент пакета, на который ссылается лист диаграммы. Вы вольны разместить диаграмму и на обычном листе. В этом случае на контейнер в отдельном компоненте будет ссылаться лист. Сама диаграмма также размещается в отдельном компоненте DrawingML внутри пакета. Таким образом, иерархия состоит из трех уровней: лист, контейнер и диаграмма. Простейший элемент этой иерархии — уровень листа:

```
<chartsheet
  xmlns="http://.../spreadsheetml/2006/main"
  xmlns:r="http://.../officeDocument/2006/relationships">
  <sheetViews>
    <sheetView workbookViewId="0" />
  </sheetViews>
  <drawing r:id="rId1" />
</chartsheet>
```

### Пример 99. Лист диаграммы

В нем вы видите две ссылки. Первая — ссылка на список элементов *workbookView* компонента рабочей книги. Она представляет собой просто номер (нумерация начинается с 0). Вторая ссылка — на контейнер SpreadsheetML для диаграмм — представляет собой идентификатор связи. Если вы вставляете диаграмму не на лист диаграммы, а на обычный рабочий лист, вставьте еще элемент *drawing* и добавьте ссылку на отдельный компонент контейнера.

## Дополнительные возможности

### Определение имен

Когда вы обращаетесь к элементам электронной таблицы, ссылка, скажем, на диапазон ячеек наподобие «*\$D\$2*», не особенно информативна. Для использования в формулах удобнее было бы писать «*TotalSales – TotalCost*» вместо «*D2 – E5*». Чтобы обеспечить эту возможность, вам разрешается определять имена для ячеек и других элементов. Область действия этих имен может распространяться на всю рабочую книгу или ограничиваться одним листом. Имена, заданные на уровне книги, очевидно, можно использовать на всех листах. Будучи вставленными в формулы, имена существенно улучшают их читабельность.

Заданное имя — на самом деле просто формула с привязанным к ней именем. Вы создаете имена и определяете их параметры в элементе *definedNames* внутри рабочей книги. Текстовая часть элемента *definedName* содержит формулу, к которой относится имя. В число его параметров входят имя, отображаемое в клиентской программе, комментарии, текст меню и описание. Чтобы имя было определено локально на данном листе, используйте атрибут *localSheetId*, указав в нем номер листа в списке.

```
<definedNames>
  <definedName name="Formula"
    comment="A Named Formula"
    localSheetId="0">SUM(1,2,3)</definedName>
  <definedName name="MyFirstCell">Sheet1!$B$1</definedName>
  <definedName name="MySecondCell">Sheet1!$B$2</definedName>
</definedNames>
```

### Пример 100. Определение имен

### Свойства рабочей книги

Мы не обсудили еще нескольких свойств уровня книги и листа. Часть их относится к версиям, другая определяет размеры окон и страниц.

На уровне книги задаются некоторые общие свойства, связанные с сохранением документа из приложения Microsoft Office Excel. В свойстве *fileVersion* указано клиентское ПО, сохранившее электронную таблицу, в свойстве *appName* — имя приложения, в свойстве *lastEdited* — версия приложения. Свойство *lowestEdited* содержит самую раннюю версию приложения, использованную для сохранения файла, и, наконец, свойство *rupBuild* в совокупности с *appName* и *lastEdited* указывает номер сборки.

Для задания параметров уровня рабочей книги служит элемент *workbookPr*. В частности, вы можете указать, какую систему дат применять: на базе 1900 или на базе 1904. SpreadsheetML позволяет использовать и ту, и другую, поскольку должен обеспечить совместимость с миллиардами документов,

уже созданных на персональных компьютерах. В одной из первых электронных таблиц, Lotus 1-2-3, имелась хорошо известная ошибка, связанная с вычислением «високосности» 1900 года. Это привело к тому, что в SpreadsheetML было включено две системы дат. Система 1900 года обеспечивает совместимость с упомянутыми ранними электронными таблицами. Система 1904 года лишена совместимости, но всегда корректно учитывает високосные годы. Есть много и других параметров, включая информацию о резервном копировании и об обновлении данных при открытии электронной таблицы.

```
<workbook>
  <fileVersion appName="xl" lastEdited="4"
    lowestEdited="4" rupBuild="4505" />
  <workbookPr date1904="0" refreshAllConnections="1" updateLinks="always" />
  <sheets>
    ...
  </sheets>
  <calcPr fullCalcOnLoad="1" />
</workbook>
```

#### Пример 101. Свойства рабочей книги

В элементе *calcPr* задаются свойства, связанные с обработчиком вычислений (calculation engine). Здесь интересен параметр *fullCalcOnLoad*, позволяющий пересчитывать всю рабочую книгу при открытии электронной таблицы.

## Резюме

---

Мы давно уже миновали середину книги, а нам нужно обсудить еще два языка разметки. Не то чтобы SpreadsheetML не заслуживал большего внимания, но нам приходится рассматривать все возможности Open XML, а объем книги ограничен. Вы узнаете много интересного, прочитав спецификации и разобрав образцы документов. Среди вопросов, не рассмотренных здесь, стоит упомянуть, например, более сложные сводные таблицы и интеграцию бизнес-данных с помощью разметки Custom XML.

## Глава 3 PresentationML

- Элементы презентации
- Различные фигуры
- Применение объектов-рамок и образцов слайдов для разработки единого стиля презентации
- Вставка рисунков, таблиц и диаграмм

### Введение

Как легко догадаться по названию, язык разметки PresentationML предназначен для создания презентаций. В модели PresentationML учтен опыт Microsoft PowerPoint и поддерживается множество различных элементов. Несколько углубившись в PresentationML, вы узнаете, что он в значительной степени опирается на возможности языка DrawingML. Тут, конечно, можно немного запутаться, поскольку в третьей части спецификации на иллюстрациях о DrawingML упоминаются пространства имен PresentationML и наоборот. В большинстве элементов, связанных с параметрами отображения, применяется пространство имен DrawingML, но сам по себе DrawingML обсуждается в следующей главе. В этой главе мы рассмотрим презентации только с точки зрения PresentationML.

### Структура документа PresentationML

Подобно другим языкам разметки, определенным в стандарте Open XML, PresentationML следует правилам Open Packaging Convention, чтобы разделить различные элементы, составляющие набор слайдов. Это приводит к тому, что структура PresentationML довольно сложна (рис. 26).

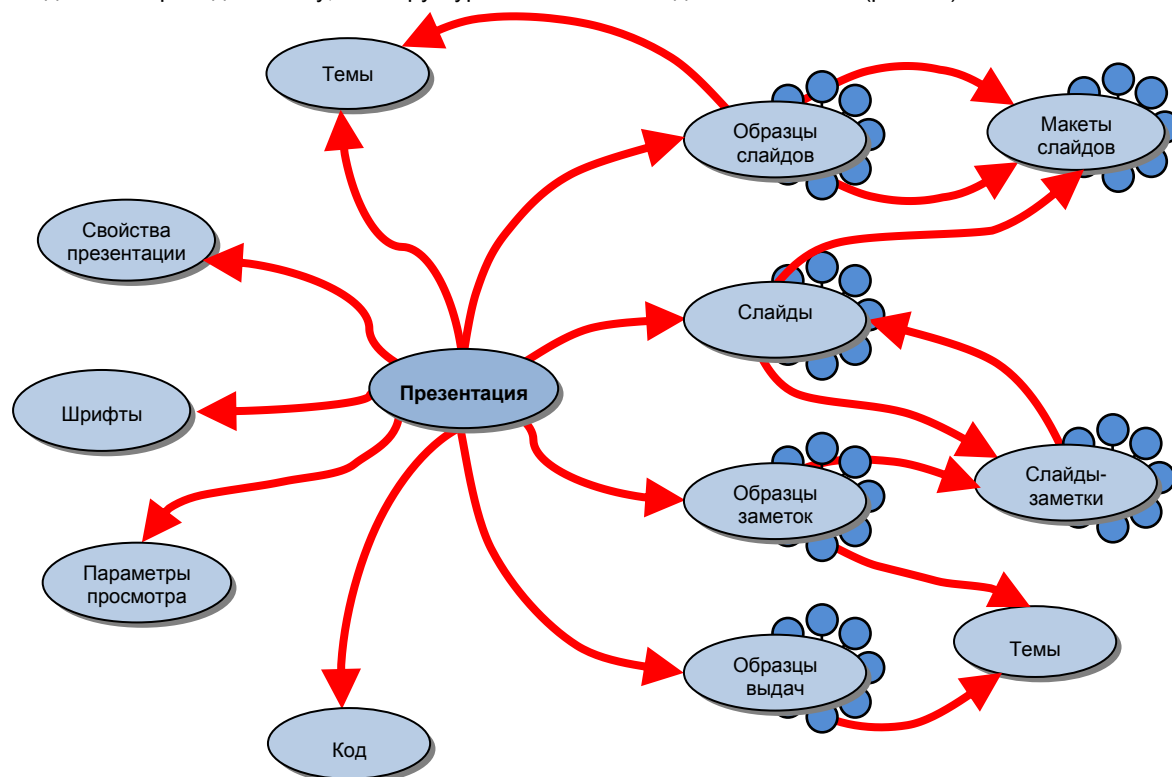


Рис. 26. Структура PresentationML

Как видите, структура пакета состоит из множества элементов и циклических связей. Средоточием всего является компонент презентации, и он же представляет собой отправную точку для работы с пакетом. Презентация связана с несколькими типами образцов слайдов, а также с другими типами содержимого. Если кто-то не знает, образец слайда — это шаблон общего назначения, применяемый к самим слайдам. В PresentationML иерархия еще сложнее: слайд ссылается на конкретный макет слайдов, а макет в свою

очередь — на образец слайдов. Три этих элемента в совокупности формируют облик слайда на экране. Есть несколько типов образцов для различных типов содержимого: для слайдов, заметок и выдач. В пакете каждый образец, каждый макет и каждый слайд презентации хранится в отдельном компоненте, что облегчает доступ к ним. Та же модель используется и для заметок: каждая заметка хранится в отдельном компоненте. На него ссылается слайд, к которому относится заметка.

Важно обратить внимание и на темы. Подобно двум другим основным форматам Open XML, темы DrawingML могут применяться для задания шрифтов, макета и других элементов оформления по умолчанию, отделяя их от основного массива данных презентации. Тему можно использовать в любом языке разметки Open XML. В пакете PresentationML задействованы различные темы. По умолчанию любой образец слайдов, заметок и выдач ссылается на отдельный компонент темы. Сама презентация ссылается на тему по умолчанию, применяемую в этой презентации. Эта же тема по умолчанию используется и образцами слайдов, ссылающимися на тот же компонент в пакете.

Структура пакета содержит две циклические ссылки: одну пару ссылок между образцом слайда и макетом и вторую между слайдом и заметками. Оставшаяся часть структуры вполне очевидна.

Большая часть элементов, включаемых в презентацию, сохраненную в PowerPoint, на самом деле не нужна. В следующей главе обсуждается макет очень простого пустого набора слайдов. Он может послужить хорошей основой для построения демонстрационной презентации. Далее мы обсудим элементы, которые можно размещать на слайдах, в частности, фигуры и диаграммы. Но сначала мы поговорим об элементах пустой презентации, а для этого нам придется познакомиться с самым важным общим элементом презентации — фигурой (shape).

## Фигуры

С помощью фигуры определяется множество объектов: текст презентации, иллюстрации, объекты-рамки на уровне макета или образца. Фигуры применяются на всех трех уровнях — слайд, макет, образец. Можно, например, на уровне образца определить объект-рамку для текста, применить к нему стиль на уровне макета и заполнить содержимым на уровне слайда.

В определении фигуры есть немало зависимостей от DrawingML. В разметке фигуры PresentationML определены специфические контейнеры, например контейнер для свойств фигуры, стиля и текста.

Эти контейнеры обычно заполняются свойствами DrawingML, позволяющими задать эффекты, например, заливку определенного вида, отражение или вращение, а также добавить к фигуре текст. Разница между надписью и запрещающим знаком на рис. 27 минимальна. Одно из расширений состоит в том, что эффекты DrawingML можно применять не только к обычным фигурам, но и собственно к тексту, что позволяет использовать фигуры в качестве кирпичиков для создания вполне профессиональных слайдов.

Фигура создается при помощи элемента *sp*, входящего в дерево фигур, — мы поговорим об этом далее. Большая часть контейнеров будет описана в главе, посвященной DrawingML, поскольку их разметка в основном опирается на эти пространства имен.



Рис. 27. Различные фигуры

```
<p:sp>
  <!--остальное вставляется здесь-->
</p:sp>
```

### Пример 102. Обзор элементов фигуры

Первый контейнер внутри фигуры предназначен для свойств фигуры, не связанных с ее отображением (non-visual), т. е. не влияющих на ее внешний облик — например, определяющих привязку. Обязательны здесь лишь атрибуты, задающие общую структуру, имя и идентификатор.

```
...
<p:nvSpPr>
  <p:cNvPr id="1" name="TextShape" />
  <p:cNvSpPr />
  <p:nvPr />
</p:nvSpPr>
...
```

### Пример 103. Свойства фигуры, не связанные с отображением

За свойствами *nvSpPr*, не влияющими на внешний облик фигуры, естественно, следуют свойства, влияющие на ее внешний облик. Эта информация сохраняется в узле *spPr* с помощью разметки DrawingML. Никакое содержимое в узле *spPr* вы задавать не обязаны. В приведенном ниже примере показано, как задавать положение и размер фигуры.

```
...
<p:spPr>
  <a:xfrm>
```

```

    <a:off x="285720" y="428604" />
    <a:ext cx="1928826" cy="369332" />
  </a:xfrm>
</p:spPr>
...

```

#### Пример 104. Свойства фигуры

Последняя часть определения фигуры — ее текст. Задавать его необязательно, но на практике текст в фигурах используется довольно часто, поэтому в примере мы покажем, как вставить в фигуру простой текстовый фрагмент (примерно так же, как в WordprocessingML).

```

<p:txBody>
  <a:bodyPr />
  <a:p>
    <a:r>
      <a:t>Shape Text</a:t>
    </a:r>
  </a:p>
</p:txBody>

```

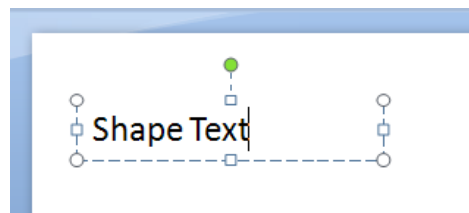
#### Пример 105. Текст фигуры

Объединив все элементы, вы получите в итоге довольно простой код (пример 106). Его можно усложнить добавлением нестандартных геометрий и эффектов DrawingML.

```

<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="1" name="TextShape" />
    <p:cNvSpPr />
    <p:nvPr />
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="285720" y="428604" />
      <a:ext cx="1928826" cy="369332" />
    </a:xfrm>
  </p:spPr>
  <p:txBody>
    <a:bodyPr />
    <a:p>
      <a:r>
        <a:t>Shape Text</a:t>
      </a:r>
    </a:p>
  </p:txBody>
</p:sp>

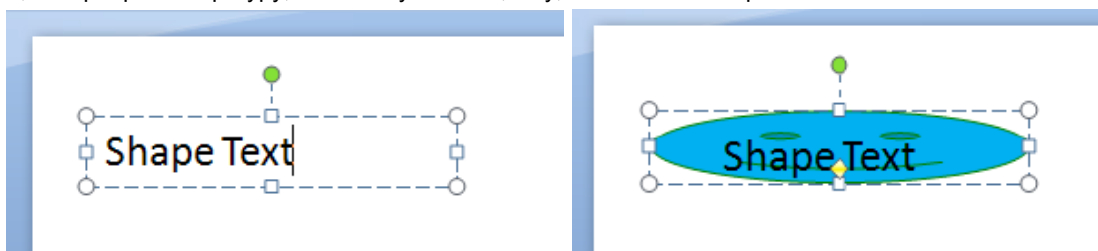
```



#### Пример 106. Фигура с текстом

### Фигура с рисунком

Чтобы превратить обычную фигуру с текстом в рисунок, достаточно добавить в раздел свойств фигуры, связанных с отображением, несколько дополнительных свойств. Задав вид заливки, цвет заливки и цвет линий, вы превратите фигуру, показанную слева, в ту, что показана справа.



Конечно, получилось простенько, но позже вы сможете существенно улучшить изображение, добавив к нему трехмерные эффекты.

Все перечисленные далее элементы нужно определять в свойствах фигуры *spPr*, связанных с отображением, после размера фигуры, заданного элементом *xfrm*.

Для начала применим нестандартную геометрию. По умолчанию используется прямоугольная геометрия, но к вашим услугам масса других вариантов. Выбор predetermined варианта осуществляется при помощи элемента *prstGeom*. Вы также можете создать собственную геометрию.

```
<a:prstGeom prst="smileyFace">
  <a:avLst />
</a:prstGeom>
```

#### Пример 107. Применение другой геометрии

Далее надо задать для структуры геометрии размер и цвет. Ассортимент заливок и цветов весьма разнообразен. Возможны, например, градиентные заливки. Линии и заливки линий подробно обсуждаются в следующей главе.

```
<a:ln>
  <a:solidFill>
    <a:prstClr val="green" />
  </a:solidFill>
</a:ln>
```

#### Пример 108. Структура геометрии

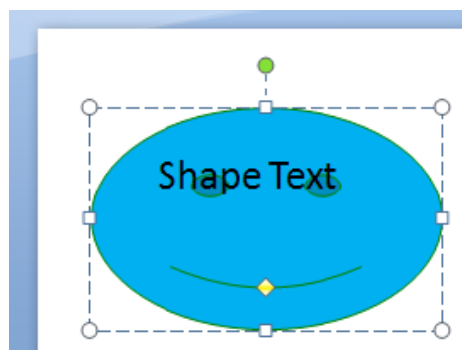
Наконец, примените заливку фигуры. Геометрию можно заполнить цветом, градиентом, изображением или узором.

```
<a:solidFill>
  <a:srgbClr val="00B0F0" />
</a:solidFill>
```

#### Пример 109. Применение заливки

Все перечисленные свойства составляют целостную разметку для фигуры с улыбающимся лицом. Мы немного увеличили высоту фигуры, чтобы ее было лучше видно.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="1" name="TextShape" />
    <p:cNvSpPr />
    <p:nvPr />
  </p:nvSpPr>
  <p:spPr>
    <a:xfrm>
      <a:off x="285720" y="428604" />
      <a:ext cx="1928826" cy="1209332" />
    </a:xfrm>
    <a:prstGeom prst="smileyFace">
      <a:avLst />
    </a:prstGeom>
    <a:solidFill>
      <a:srgbClr val="00B0F0" />
    </a:solidFill>
    <a:ln>
      <a:solidFill>
        <a:prstClr val="green" />
      </a:solidFill>
    </a:ln>
  </p:spPr>
  <p:txBody>
    <a:bodyPr />
    <a:p>
      <a:r>
        <a:t>Shape Text</a:t>
      </a:r>
    </a:p>
  </p:txBody>
</p:sp>
```



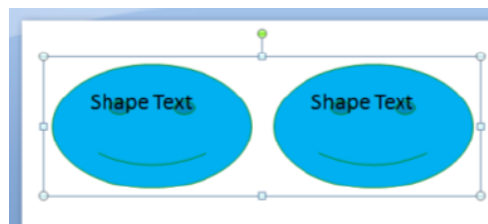
#### Пример 110. Простая фигура



## Группирование фигур

Последний вид фигур, который мы рассмотрим, — это фигура-группа. Концепция группирования фигур, возможно, знакома вам по Microsoft PowerPoint. Фигура-группа содержит несколько фигур и образует контейнер для них. Группа позволяет редактировать набор фигур как единое целое.

Для определения фигуры-группы вместо обычного элемента *sp* применяется элемент *grpSp*. Остальное в принципе мало отличается от обычной фигуры. Сначала вы определяете контейнер.



```
<p:grpSp>
  <!-- Остальные элементы вставляются в этот контейнер -->
</p:grpSp>
```

### Пример 111. Фигура-группа

Затем задаете свойства, не связанные с отображением.

```
<p:nvGrpSpPr>
  <p:cNvPr id="4" name="Group 3" />
  <p:cNvGrpSpPr />
  <p:nvPr />
</p:nvGrpSpPr>
```

### Пример 112. Свойства фигуры-группы, не связанные с отображением

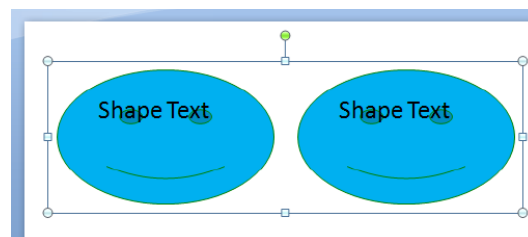
Затем задаются свойства, связанные с внешними свойствами фигуры: положение и размер группы, а также параметры прямоугольника, охватывающего все дочерние фигуры. Если не определить эти элементы, группа не будет корректно отображаться.

```
<p:grpSpPr>
  <a:xfrm>
    <a:off x="285720" y="428604" />
    <a:ext cx="4071966" cy="1209332" />
    <a:chOff x="285720" y="428604" />
    <a:chExt cx="4071966" cy="1209332" />
  </a:xfrm>
</p:grpSpPr>
```

### Пример 113. Свойства фигуры-группы, связанные с отображением

Наконец, заключительные элементы группы — ее фигуры. Они ничем не отличаются от тех фигур, о которых мы говорили в начале раздела. Группа может содержать другие группы, формируя дерево фигур.

```
<p:grpSp>
  <p:nvGrpSpPr>...</p:nvGrpSpPr>
  <p:grpSpPr>
    ...</a:xfrm>
  </p:grpSpPr>
  <p:sp>
    <!-- разметка пропущена -->
  </p:sp>
  <p:sp>
    <!-- разметка пропущена -->
  </p:sp>
</p:grpSp>
```



### Пример 114. Фигура-группа (с сокращениями)

## Элементы простой презентации

Теперь, когда мы обсудили основной элемент презентации — фигуру, поговорим о том, как составить из фигур презентацию. Простейшая презентация состоит из пяти компонентов: основного компонента презентации, образца слайдов, макета слайдов, слайда и темы. Тема — обязательный элемент, ссылка на который присутствует как в компоненте презентации, так и в образце слайдов.

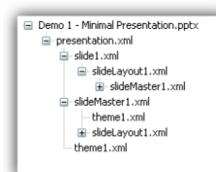


Рис. 28. Простая презентация

На каждом из трех структурных уровней слайда вы при помощи элемента *spTree* создаете дерево фигур. Можете считать *spTree* группой, компонентами которой являются все фигуры слайда, макета или образца.

## Компонент слайда

На конечном уровне иерархии находится слайд. В нем разрешается создавать фигуры не всех типов. Так, на уровне слайда не разрешается создавать фигуры-рамки, о которых мы поговорим позже.

Для создания слайда в пакете PresentationML нужны следующие типы содержимого и связи.

---

*Тип содержимого для слайда:*

*application/vnd.openxmlformats-officedocument.presentationml.slide+xml*

*Тип связи для слайда:*

*http://schemas.openxmlformats.org/officeDocument/2006/relationships/slide*

---

Содержимое слайда начинается с элемента *sld*. Здесь от вас требуется не так много. Во-первых, вы создаете в общих данных слайда *cSld* дерево фигур. В элементе дерева фигур применяются те же два узла свойств (связанных и не связанных с отображением), что и в фигуре-группе. Для краткости фигуры в дереве пропущены.

```
<p:sld>
  <p:cSld>
    <p:spTree>
      <p:nvGrpSpPr>
        <p:cNvPr id="1" name="" />
        <p:cNvGrpSpPr />
        <p:nvPr />
      </p:nvGrpSpPr>
      <p:grpSpPr />
      <p:sp>
        <!-- Первая фигура слайда -->
      </p:sp>'
      <p:sp>
        <!-- Вторая фигура слайда -->
      </p:sp>
    </p:spTree>
  </p:cSld>
</p:sld>
```

Пример 115. Слайд

## Компонент макета слайдов

В компоненте макета слайдов, как и в компоненте слайда, определяется дерево фигур, содержащее элементы слайда. На экране будет отображаться сочетание этих элементов с элементами, определенными на уровне слайда и образца. Поскольку здесь вы определяете совершенно новое дерево фигур, доступа к фигурам других уровней у вас нет. Нельзя, например, группировать фигуры на слайде с фигурами на макете. Но способ связать фигуры разных уровней все же есть: вы можете создать фигуру-рамку на уровне макета или мастера, а затем связать с ней фигуру на уровне слайда.

В следующем примере разметки изображен простой пустой макет слайда. О рамках мы поговорим ниже.

Как и в случае образца, здесь вы вольны задавать различные параметры, например, параметры фона, и добавлять элементы в дерево фигур. В отличие от образца на макете слайдов можно располагать нестандартные рамки, чтобы задать области содержимого на слайде. Формирование дерева фигур от рамок на образце до содержимого слайда мы обсудим в главе 4.

```
<p:sldLayout
  xmlns:a="http://.../drawingml/2006/main"
  xmlns:r="http://.../officeDocument/2006/relationships"
  xmlns:p="http://.../presentationml/2006/main">
  <p:cSld name="Title Slide">
    <p:spTree>
      <p:nvGrpSpPr>
        <p:cNvPr id="1" name="" />
        <p:cNvGrpSpPr />
        <p:nvPr />
      </p:nvGrpSpPr>
      <p:grpSpPr />
    </p:spTree>
  </p:cSld>
</p:sldLayout>
```

Пример 116. Макет слайда

## Компонент образца слайдов

Образец слайдов — корневой элемент иерархии, составляющей слайд. Он также содержит дерево фигур. Помимо элемента дерева фигур, для формирования корректного образца нужно определить еще несколько фрагментов содержимого.

Дерево фигур и здесь ничем не отличается от деревьев, рассмотренных ранее. Общим данным слайда дочерние элементы не нужны, поэтому они здесь сокращены, чтобы можно было сосредоточиться на другом содержимом образца.

Первое отличие — применение элемента *clrMap* для связывания цветов, использованных на слайде, с цветами компонента темы. Ссылка осуществляется по именам цветов темы.

Второе отличие — добавление списка макетов слайдов. Список ведется в образце, чтобы макеты можно было сортировать. Значение идентификатора должно быть уникальным, а идентификатор связи указывает на компонент макета слайдов.

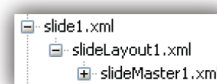


Рис. 29. Иерархия — от слайда к образцу

```
<p:sldMaster
  xmlns:a="http://.../drawingml/2006/main"
  xmlns:r="http://.../officeDocument/2006/relationships"
  xmlns:p="http://.../presentationml/2006/main">
  <p:cSld>
    <!-- Дерево фигуры пропущено -->
  </p:cSld>
  <p:clrMap bg1="lt1" tx1="dk1" bg2="lt2" tx2="dk2" accent1="accent1"
    accent2="accent2" accent3="accent3" accent4="accent4"
    accent5="accent5" accent6="accent6" hlink="hlink"
    folHlink="folHlink" />
  <p:sldLayoutIdLst>
    <p:sldLayoutId id="2147483649" r:id="rId1" />
  </p:sldLayoutIdLst>
</p:sldMaster>
```

Пример 117. Образец слайдов

## Компонент темы

Ссылка на компонент темы имеется в компоненте образца слайдов и в основном компоненте презентации, который обсуждается далее в этой главе, а его содержимое подробно описано в главе 4. В приведенном ниже примере показана сокращенная версия компонента темы.

```
<a:theme xmlns:a="http://.../drawingml/2006/main"
  name="Office Theme">
  <a:themeElements>
    <a:clrScheme/>
    <a:fontScheme/>
    <a:fmtScheme/>
  </a:themeElements>
  <a:objectDefaults />
  <a:extraClrSchemeLst />
</a:theme>
```

Пример 118. Обзор элементов темы

## Компонент презентации

В основном компоненте презентации все ее составные части объединяются в целое. Этот компонент — начало документа. Корневым здесь является элемент *presentation*.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<p:presentation
  xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
  xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
  xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main">
</p:presentation>
```

Пример 119. Корневой элемент компонента презентации

В элементе *presentation* содержатся два списка: слайдов и образцов слайдов. Они применяются для сортировки этих элементов презентации в клиентской программе.

```
<p:sldMasterIdLst>
  <p:sldMasterId id="2147483648" r:id="rId1" />
</p:sldMasterIdLst>
```

#### Пример 120. Список идентификаторов образцов слайдов

Компонент презентации владеет не только образцами, но и всеми ее слайдами. В нем ведется список всех слайдов.

```
<p:sldIdLst>
  <p:sldId id="256" r:id="rId2" />
  <p:sldId id="257" r:id="rId3" />
  <p:sldId id="258" r:id="rId4" />
</p:sldIdLst>
```

#### Пример 121. Список идентификаторов слайдов

В конце компонента презентации нужно задать размеры слайдов и заметок. В заметках вы записываете примечания к каждому отдельному слайду. Размеры задаются в EMU. Эта единица измерения обсуждается в первом приложении.

```
<p:sldSz cx="9144000" cy="6858000" />
<p:notesSz cx="6858000" cy="9144000" />
```

#### Пример 122. Размеры слайда

Полная разметка компонента презентации показана в примере 123.

```
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<p:presentation
  xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
  xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
  xmlns:p="http://schemas.openxmlformats.org/presentationml/2006/main">
  <p:sldMasterIdLst>
    <p:sldMasterId id="2147483648" r:id="rId1" />
  </p:sldMasterIdLst>
  <p:sldIdLst>
    <p:sldId id="256" r:id="rId2" />
    <p:sldId id="257" r:id="rId3" />
    <p:sldId id="258" r:id="rId4" />
  </p:sldIdLst>
  <p:sldSz cx="9144000" cy="6858000" />
  <p:notesSz cx="6858000" cy="9144000" />
</p:presentation>
```

#### Пример 123. Компонент презентации

## Объекты-рамки

Пока что мы добавили на уровне слайда лишь несколько простых фигур. Хотя это и позволяет создать базовый набор слайдов, настоящий набор должен содержать области содержимого, определенные на уровнях образца и макета и заполняемые на уровне слайда. Для этого предназначены рамки (placeholder). Любая обычная фигура *sp* может как сама быть рамкой, так и заполнять содержимым другую рамку. Рамка идентифицируется по использованию специфического встроенного типа (например, рамка для заголовка) или нестандартного значения идентификатора. Применение встроенных типов позволяет обеспечить совместимость элементов презентации с другими наборами слайдов. В следующей таблице перечислены типы рамок и указано, где их можно использовать.

Рамки создаются на уровнях макета или образца. Рамка, созданная на уровне образца, может заполняться содержимым на уровне макета. У слайдов нет прямого доступа к рамкам образца; на них могут использоваться только рамки, определенные на уровне макета. Рамку в компоненте макета можно привязать к подобной рамке в компоненте образца. Таким образом, создается трехуровневая иерархия для определения фигур. Скажем, рамка определяется без форматирования на уровне образца, оформляется рамкой на уровне макета и заполняется текстом на уровне слайда. В этой структуре участвуют три фигуры — по одной на каждом уровне.



Рамка	Образец слайдов	Образец заметок	Образец выдач	Макет слайдов	Слайд	Заметки
Основной текст		x		x	x	x
Диаграмма				x	x	
Объект clipart				x	x	
Центрированный заголовок				x	x	
Схема				x	x	
Дата и время		x	x	x	x	x
Верхний колонтитул		x	x			x
Нижний колонтитул		x	x	x	x	x
Медиаданные				x	x	
Объект				x	x	
Рисунок				x	x	
Образ слайда		x				x
Номер слайда		x	x	x	x	x
Подзаголовок				x	x	
Таблица				x	x	
Заголовок				x	x	

Для создания этой иерархии используются обычные элементы-фигуры с одним дополнительным свойством. Как вы помните, у каждой фигуры имеются свойства, не связанные с ее отображением (см. следующий пример). Определение фигуры сокращено до минимума, чтобы сосредоточиться на разделе свойств. Ни основной текст, ни свойства, связанные с отображением, не определены.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="1" name="TextShape" />
    <p:cNvSpPr />
    <p:nvPr />
  </p:nvSpPr>
  <p:spPr />
</p:sp>
```

#### Пример 124. Структура фигуры

Элемент *nvPr* пуст, но рамки позволяют это изменить. Внутри этого элемента вы с помощью элемента *ph* идентифицируете фигуру в качестве рамки. Последствия его применения зависят от того, на каком уровне иерархии вы работаете.

Если вы задаете свойство рамки для фигуры на уровне слайда, оно сделает рамкой фигуру на уровне макета, к которой привязана данная фигура. Создавать новые рамки на уровне слайда нельзя, поскольку это последний уровень иерархии. Ссылка на рамку уровня макета осуществляется по идентификатору или типу. В следующем примере показано, как вставить в слайд ссылку на рамку с помощью идентификатора. Поскольку тип указывать не нужно, использован общий тип *body*.

Макет	Слайд
<pre>&lt;p:sp&gt;   &lt;p:nvSpPr&gt;     &lt;p:cNvPr id="1"       name="MyTitleShape" /&gt;     &lt;p:cNvSpPr /&gt;     &lt;p:nvPr&gt;</pre>	<pre>&lt;p:sp&gt;   &lt;p:nvSpPr&gt;     &lt;p:cNvPr id="1"       name="MyTitleShape" /&gt;     &lt;p:cNvSpPr /&gt;     &lt;p:nvPr&gt;</pre>

<code>&lt;p:ph type="body" idx="1" /&gt;</code>	<code>&lt;p:ph type="body" idx="1" /&gt;</code>
<code>&lt;/p:nvPr&gt;</code>	<code>&lt;/p:nvPr&gt;</code>
<code>&lt;/p:nvSpPr&gt;</code>	<code>&lt;/p:nvSpPr&gt;</code>
<code>&lt;p:spPr /&gt;</code>	<code>&lt;p:spPr /&gt;</code>
<code>&lt;/p:sp&gt;</code>	<code>&lt;/p:sp&gt;</code>

Таблица 6. Связывание фигуры уровня слайда с рамкой уровня макета

На уровне макета элемент *ph* служит для определения рамок как совсем новых, так и привязанных к рамкам на уровне образца, образующих трехуровневую иерархию.

Образец	Макет
	
<pre> &lt;p:sp&gt;   &lt;p:nvSpPr&gt;     &lt;p:cNvPr id="1" name="MyShape" /&gt;     &lt;p:cNvSpPr /&gt;     &lt;p:nvPr&gt;       &lt;p:ph type="body" idx="1" /&gt;     &lt;/p:nvPr&gt;   &lt;/p:nvSpPr&gt; &lt;/p:spPr /&gt; &lt;/p:sp&gt; </pre>	<pre> &lt;p:sp&gt;   &lt;p:nvSpPr&gt;     &lt;p:cNvPr id="1" name="MyShape" /&gt;     &lt;p:cNvSpPr /&gt;     &lt;p:nvPr&gt;       &lt;p:ph type="body" idx="1" /&gt;     &lt;/p:nvPr&gt;   &lt;/p:nvSpPr&gt; &lt;/p:spPr /&gt; &lt;/p:sp&gt; </pre>

Таблица 7. Связывание фигуры уровня макета с рамкой уровня образца

## Рисунки

В принципе для размещения на слайде рисунка можно использовать фигуру с большим двоичным объектом в качестве заливки, но для объявления рисунков предусмотрен также специальный элемент. К нему, как и к любой фигуре, можно применять границы и эффекты DrawingML, но вам доступны и дополнительные параметры, специально предназначенные для рисунков. С их помощью вы откорректируете яркость и контрастность, а также зададите дополнительный цветовой слой, чтобы модифицировать итоговое отображение рисунка на экране. В интерфейсе PowerPoint для рисунков также предусмотрены специальные команды.

На самом деле рисунок и фигура с рисунком в качестве заливки мало отличаются друг от друга. Главное различие в том, что по умолчанию в клиентской программе изменение пропорций рисунка будет, как правило, заблокировано, а изменение пропорций фигуры — разрешено.

```

<p:pic>
  <p:nvPicPr>...</p:nvPicPr>
  <p:blipFill>
    <a:blip r:embed="rId2" />
  </p:blipFill>
  <p:spPr>
    <a:xfrm>
      <a:off x="762000" y="571500" />
      <a:ext cx="7620000" cy="5715000" />
    </a:xfrm>
    <a:prstGeom prst="rect" />
  </p:spPr>
</p:pic>

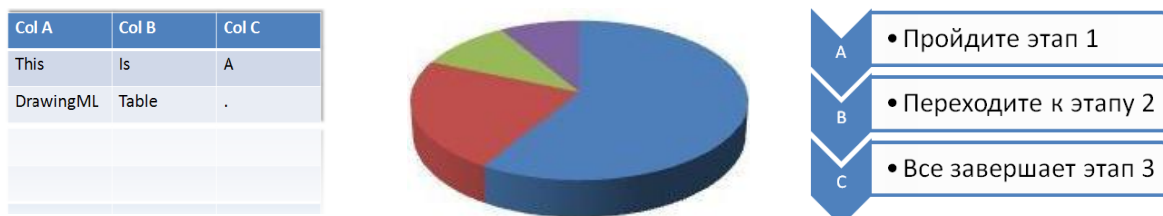
```

Пример 125. Рисунок

Чтобы создать рисунок, добавьте в дерево фигур узел *pic*. В элементе рисунка вы обязательно должны задать обычные параметры, связанные и не связанные с отображением, и ссылку на отображаемый рисунок. Можете также задать стиль. Само изображение при помощи связи хранится в пакете отдельно. Для ссылки на рисунок применяется идентификатор этой связи. Сослаться на изображение можно двумя способами. Оно может храниться как внутри, так и вне пакета. Атрибут *embed* указывает на внутреннее изображение, а атрибут *link* — на внешнюю ссылку. Но в качестве данных ссылки в обоих случаях применяется идентификатор связи. Подробнее об оформлении изображений — в главе, посвященной DrawingML.

## Таблицы, диаграммы и схемы

Помимо фигур и рисунков, о которых мы до сих пор говорили, дерево фигур может содержать элементы и других типов: таблицы, диаграммы и схемы («интеллектуальные» объекты, или «smart-art»). Они обрабатываются несколько иначе, чем фигуры и рисунки. Разметка для этих трех элементов в пространстве имен PresentationML не определяется — вместо этого используется DrawingML, который отчасти также применяется для фигур и рисунков. Подробно структура таблиц, диаграмм и схем обсуждается в главе 4, посвященной DrawingML. Их вставка в дерево фигур осуществляется общим для всех элементов способом, присущим PresentationML, и примерно по тем же правилам, что и при вставке разметки DrawingML в других языках, — с помощью элемента-контейнера общего назначения.



Замечательная особенность разметки DrawingML для вставки диаграмм и схем заключается в том, что их можно повторно использовать во всех языках разметки. То есть вы вольны вставить диаграмму или схему DrawingML в любой нужный вам язык, используя для определения содержимого в точности одну и ту же разметку. Применение DrawingML также открывает доступ к визуальным эффектам, например, к использованию отражения в таблицах и трехмерного вида в диаграммах.

```
<p:graphicFrame>
  <p:nvGraphicFramePr />
  <p:xfrm />
  <a:graphic>
    <a:graphicData uri="http://.../drawingml/2006/table">
      <!-- Элементы таблицы -->
    </a:graphicData>
  </a:graphic>
</p:graphicFrame>
```

### Пример 126. Графическая область таблицы

Контейнер общего назначения для PresentationML строится по модели, показанной в примере 126. Вместо определения содержимого (скажем, таблицы) непосредственно в дереве фигур здесь применяется многоцелевой контейнер *graphicFrame*. Эта графическая область может применяться для определения любого содержимого, в частности, объектов DrawingML, обсуждавшихся в этой главе. Схемы ECMA разрешают использование в *graphicFrame* содержимого XML любого типа, но приложение не должно «понимать» содержимое, чтобы успешно открыть документ. Для идентификации типа содержимого графической области служит атрибут *uri* внутреннего элемента *graphicData*. Свои типы содержимого, специфические для данной платформы, могут определяться в любом редакторе, поддерживающем работу с Open XML. В 2007 Microsoft Office System поддерживается определенный список значений *uri*, все из которых ссылаются на пространство имен DrawingML.

<http://schemas.openxmlformats.org/drawingml/2006/chart>

<http://schemas.openxmlformats.org/drawingml/2006/compatibility>

<http://schemas.openxmlformats.org/drawingml/2006/diagram>

<http://schemas.openxmlformats.org/drawingml/2006/lockedCanvas>

<http://schemas.openxmlformats.org/drawingml/2006/picture>

<http://schemas.openxmlformats.org/drawingml/2006/table>

<http://schemas.openxmlformats.org/drawingml/2006/ole>

Рис. 30. Доступные значения URI для идентификации содержимого

В самой графической области определяются лишь избранные параметры, например, идентификатор и имя области, ее размер и информация о связи с рамкой уровня макета. В примере разметки 127 показаны свойства графической области, связывающие ее с пронумерованной рамкой, а также запрещающие группирование и изменение размера области внутри редактора. Объект-рамка относится к типу *table*.

```
<p:nvGraphicFramePr>
  <p:cNvPr id="6" name="Content Placeholder 5" />
  <p:cNvGraphicFramePr>
    <a:graphicFrameLocks noGrp="1" noResize="1" />
  </p:cNvGraphicFramePr>
  <p:nvPr>
    <p:ph idx="1" />
  </p:nvPr>
</p:nvGraphicFramePr>
```

**Пример 127. Задание свойств графической области**

Различные типы содержимого графических областей обсуждаются в главе, посвященной DrawingML.



## Глава 4

# DrawingML

- Создание текста и списков
- Контуры рисунков, геометрические фигуры и заливки
- Двумерные и трехмерные эффекты
- Таблицы и диаграммы DrawingML
- Определения тем

### Введение

DrawingML — это язык Open XML, предназначенный для определения графических объектов. Хотя в некоторых разделах спецификации упоминается язык векторной разметки (Vector Markup Language), DrawingML обладает таким же набором функциональных возможностей и во многом его превосходит. Для отображения графических объектов DrawingML позволяет использовать трехмерное пространство. Вы можете взять изображение, показанное на рис. 31, добавить к нему рамку, отразить его на виртуальной поверхности и повернуть боком, создавая впечатление, что оно стоит. Подобные эффекты можно применять к графическим объектам, а многие из них и к тексту, что дает простор для творчества. В Microsoft Office для реализации таких преобразований применяется технология DirectX.



Рис. 31. Некоторые возможности DrawingML

При обсуждении DrawingML мы уделим внимание нескольким темам. Прежде всего мы рассмотрим возможности DrawingML по работе с текстом, затем — использование графических объектов и, наконец, добавление эффектов. Эти эффекты можно применять к примитивам DrawingML, о которых будет рассказано в разделах, посвященных таблицам и диаграммам.

### Текст

Создание текста с помощью кода DrawingML и WordprocessingML имеет много общего. Есть элемент, описывающий абзац, который состоит из областей, содержащих текст и символы форматирования. Абзац хранится в других элементах-контейнерах, различных для разных языков Open XML. В качестве примера можно привести фигуру с текстом в PresentationML, описанную в предыдущей главе. В этом языке контейнером для хранения текста DrawingML служит элемент *txBody*.

```
<p:txBody>
  <a:bodyPr />
  <a:p>
    <a:r>
      <a:t>Место для текста</a:t>
    </a:r>
  </a:p>
</p:txBody>
```

Пример 128. Контейнер PresentationML для текста DrawingML

Параметры DrawingML внутри контейнера позволяют определять формат текста. Эти параметры можно применять на уровне контейнера, абзаца или области. На уровне контейнера можно задавать, к примеру, параметры трехмерного изображения (о них чуть позже). На этом уровне также можно применять стили Word-Art, которые сейчас постепенно вытесняются DrawingML. На уровне абзаца задаются такие параметры, как размер полей или выравнивание текста. И, наконец, в свойствах области хранятся такие параметры, как полужирный шрифт или курсив. Область является низшим уровнем, на котором применяется форматирование. Чтобы создать один абзац, в середине которого есть слово, набранное полужирным шрифтом, потребуется минимум три области.

## Форматирование текста

Форматирование текста позволяет создавать замысловатые надписи. Помимо привычных параметров форматирования вроде полужирного или курсивного начертания, можно воспользоваться такими расширениями DrawingML, как трехмерные преобразования, отражение и свечение. Возможности DrawingML по работе с текстом показаны на рис. 32, где для буквы «А» задан шрифт *Brush Script MT*, созданы градиентная заливка, контур и добавлено отражение. Справ — та же буква «А», изогнутая и повернутая в трехмерном пространстве. В сущности заливка текстовых элементов создается по тем же принципам, что перечислены в следующем разделе, посвященном графическим объектам. Трехмерные преобразования рассматриваются далее в этой главе.



Рис. 32. Текст в формате DrawingML

И, наконец, говоря о тексте, описываемом DrawingML, нужно отметить использование обычных возможностей форматирования, например полужирного или курсивного начертания. Эта модель немного отличается от WordprocessingML, где вместо элементов используются атрибуты.

```
<a:p>
  <a:r>
    <a:rPr b="1" />
    <a:t>a</a:t>
  </a:r>
  <a:r>
    <a:rPr sz="2800" i="1">
      <a:latin typeface="Arial" />
    </a:rPr>
    <a:t>b</a:t>
  </a:r>
  <a:r>
    <a:rPr u="sng" />
    <a:t>c</a:t>
  </a:r>
  <a:endParaRPr />
</a:p>
```

abc

Пример 129. Непосредственное форматирование текста

## Маркированные списки

Чтобы создать из простого текста маркированный список, в абзац достаточно добавить одно дополнительное свойство. Как и в WordprocessingML, для описания свойства служит элемент *pPr*. Каждый элемент маркированного списка определяется как отдельный абзац. Для хранения сведений о нумерации в WordprocessingML служит отдельный компонент, а в PresentationML эти данные хранятся в самом контейнере. Тип списка можно задать несколькими способами. В примере 130 показано базовое непосредственное форматирование. Элемент *buChar* служит для определения символа маркера. К другим параметрам форматирования относятся шрифт, размер и положение маркера.

```
<p:txBody>
  <a:bodyPr />
  <a:p>
```

```

<a:pPr lvl="0">
  <a:buChar char="•" />
</a:pPr>
<a:r>
  <a:t>Level 0 text</a:t>
</a:r>
</a:p>
<a:p>
  <a:pPr lvl="1">
    <a:buChar char="•" />
  </a:pPr>
  <a:r>
    <a:t>Level 1 text</a:t>
  </a:r>
  <a:endParaRPr />
</a:p>
</p:txBody>

```

### Пример 130. Маркированный список

Есть несколько способов задания типа маркера. В этом примере показано, как задать символ для маркера с помощью элемента *buChar*. Маркер в виде рисунка можно создать, используя элемент *buBlip*, а нумерованный список — с помощью элемента *buAutoNum*. О хранении и использовании больших двоичных изображений (BLIP) вы узнаете из раздела, посвященного графике DrawingML.

## Нумерованные списки

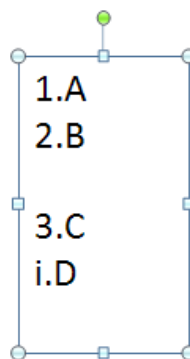
Для списков с автоматической нумерацией нужно добавить элемент *buAutoNum* и указать тип нумерации. Можно выбрать один из разнообразных встроенных стилей нумерации, например «a...z» или «1...9»; доступны и другие форматы нумерации. Принцип работы нумерации состоит в том, что все абзацы с одинаковым стилем нумерации, следующие друг за другом, входят в одну группу нумерации. В пределах группы нумерации каждый абзац получает свой уникальный порядковый номер. Если один из абзацев пуст, он не занимает отдельного номера, но по-прежнему является частью группы нумерации. Абзац, для которого не задан тип нумерации, прерывает группу нумерации, изменяя и стиль нумерации. В следующий раз при использовании того же стиля нумерации отсчет начнется с 1.

В примере 131 создается список с разрывом. Чтобы отделить элементы списка, между ними добавлен абзац без областей, к которому применяется тот же тип нумерации, что и к предыдущему абзацу. Для последнего абзаца используется другой тип нумерации, и нумерация выполняется с самого начала.

```

<a:p>
  <a:pPr>
    <a:buAutoNum type="arabicPeriod" />
  </a:pPr>
  <a:r>...</a:r>
  <a:endParaRPr />
</a:p>
<a:p>
  <a:pPr>
    <a:buAutoNum type="arabicPeriod" />
  </a:pPr>
  <a:r>...</a:r>
</a:p>
<a:p>
  <a:pPr>
    <a:buAutoNum type="arabicPeriod" />
  </a:pPr>
  <!--Пусто -->
  <a:endParaRPr />
</a:p>
<a:p>
  <a:pPr>
    <a:buAutoNum type="arabicPeriod" />
  </a:pPr>
  <a:r>...</a:r>
  <a:endParaRPr />
</a:p>
<a:p>

```



```

<a:pPr>
  <a:buAutoNum type="romanLcPeriod" />
</a:pPr>
<a:r>...</a:r>
<a:endParaRPr />
</a:p>

```

Пример 131. Нумерованный список с разрывом в нумерации

Для создания нескольких уровней нумерации нужно определить каждый уровень с помощью атрибута *lvl* в свойствах абзаца, как показано в примере 130. На каждом новом уровне нумерация начинается с 1, а остальная часть правила также применяется на каждом подуровне. В пределах нумерованных уровней можно создавать нумерованные и наоборот. Используя атрибут *lvl*, можно переместить параметры абзаца заданные параметры можно применять к каждому из девяти возможных уровней вложенности, используя именованные элементы, например *lvl1pPr* и *lvl2pPr*. В случае конфликта между свойствами, определенными в элементе стиля списка, и свойствами самого абзаца, приоритет имеют свойства абзаца. В спецификации Open XML это трактуется как «близость к исходному тексту».

## Графика

Помимо работы с текстовыми объектами, DrawingML предлагает широкие возможности по созданию фигур векторной графики и применению к ним различных эффектов для вывода их в отображающей программе. К таким эффектам относятся перемещение фигуры в трехмерном пространстве или добавление к ней тени и отражения. И что особенно замечательно, все эти эффекты применимы к большинству типов содержимого, определенных с помощью формата DrawingML, в том числе к фигурам, таблицам и диаграммам.

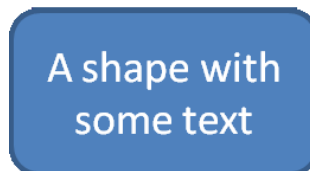


Рис. 33. Пример фигуры

Возможности разметки в DrawingML столь обширны, что рассказать здесь о них подробно невозможно. В этой книге мы сосредоточимся на применении стандартных эффектов к различным фигурам, например таблицам, а потом, возможно, появится и целая книга, посвященная DrawingML.

Мы обсудим элементы DrawingML. Прежде всего нужно рассмотреть геометрическую форму графических объектов DrawingML и различные связанные с ней свойства, например стиль линий и заливки. Затем в этом разделе мы обсудим расположение фигуры в трехмерном пространстве DrawingML, а также эффекты, применяемые в трехмерном пространстве. Все примеры фигур, приведенные в этом разделе, созданы на основе фигуры, показанной в начале этого раздела.

## Форма графических объектов

Первое, что бросается в глаза при взгляде на фигуру на рис. 33, — это ее закругленные углы. При изменении размера фигуры ее углы останутся точно так же закругленными. Для определения ее геометрической формы применяется векторная разметка. Для фигур из примеров этого раздела использован предопределенный шаблон *roundRect*. С помощью DrawingML можно создавать два типа геометрических фигур: предопределенные и настраиваемые. Контур настраиваемой геометрической фигуры задается с помощью векторной разметки, а предопределенные геометрические фигуры избавят вас от этой головной боли. В этой книге не рассматриваются настраиваемые геометрические формы, но код для предопределенных геометрических объектов мы рассмотрим. А поскольку они предопределены, для их описания не требуется много кода.

```

<a:prstGeom prst="noSmoking">
  <a:avLst />
</a:prstGeom>

```

Пример 132. Изменение предопределенной геометрической формы

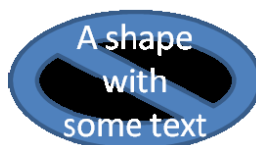


Рис. 34. Изменение предопределенной геометрической формы

Чтобы задать фигуре предопределенную геометрическую форму, используют элемент *prstGeom*. Этот элемент применяется, главным образом, для фигур PresentationML и рисунков Open XML. Для задания предопределенной формы используют ее имя; полный список имен есть в спецификации. В этом списке есть как основные формы (прямоугольник, *rect*), так и более сложные геометрические формы вроде знака запрета, показанного на рис. 34; для его создания использована XML-разметка из примера 132. В этом же

примере показано применение элемента *avLst*. Он задает список переменных значений, которые могут быть впоследствии изменены, чтобы настроить некоторые свойства фигуры. В сущности геометрическая форма вычисляется «на лету», на основе стандартных и изменяемых значений. Пример использования элемента *avLst* — на рис. 35. В этом примере для исходной фигуры (прямоугольника с закругленными углами) задан список изменяемых значений, определяющий другую степень закругления углов прямоугольника.

```
<a:prstGeom prst="roundRect">
  <a:avLst>
    <a:gd name="adj" fmla="val
50000" />
  </a:avLst>
</a:prstGeom>
```

Пример 133. Изменение значений для геометрической формы



Рис. 35. Геометрическая форма с измененными значениями

## Система координат

Графические объекты DrawingML размещаются на плоскости. Чтобы разместить графический объект на плоскости, нужно указать расположение и размер объекта. Для этого служит элемент *xfrm*, или элемент преобразования. Отсчет производится от левого верхнего края графического объекта в сторону правого нижнего края. Расположение и размер объекта указываются в EMU (English Metric Unit). Подробнее об этой единице измерения см. главу, посвященную PresentationML. С помощью элемента *xfrm* также можно указать вертикальное или горизонтальное зеркальное отражение и поворот фигуры на плоскости.

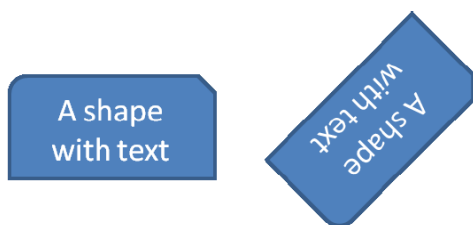


Рис. 36. Фигура с поворотом и зеркальным отражением

```
<a:xfrm flipV="1" rot="-2700000">
  <a:off x="571472" y="1214422" />
  <a:ext cx="1428760" cy="714380" />
</a:xfrm>
```

Пример 134. Задание размера, расположения и поворота фигуры

## Задание цвета в DrawingML

Графический объект занимает некоторую область на плоскости (слайд, таблицу и пр.). Эту область можно настроить, задавая для нее параметры, определяющие цвет и заливку графического объекта. Для графических объектов в формате DrawingXML можно применять параметры цветового оформления и заливки, а также стили линий. Поскольку для стилей заливки и линий применяются цвета, логичнее начать обсуждение с задания цвета в DrawingML.

В DrawingML значение цвета определяется несколькими способами. Для обозначения цвета в DrawingML используются как широко известные стандартные цветовые модели, например RGB и HSL, так и некоторые специальные цвета и возможность ссылки на цвет в теме.

Для цветового пространства RGB есть два способа задания красного, зеленого и синего компонента цвета: либо нужно указать значение в диапазоне от 0 до 255 для всех трех компонентов в шестнадцатеричном формате, либо применить алгоритм, вычисляющий соответствующее процентное отношение. При использовании диапазона значений 0A141E означает, что цвет состоит из 10 единиц красного, 20 единиц зеленого и 30 единиц синего. Этот же цвет можно выразить в процентном отношении. Для этого разделите значения для красного, зеленого и синего компонента на 255 и умножьте на 1000. Для цветов со значениями в процентном отношении данные хранятся в отдельных полях с точностью до 1/1000 процента. Для абсолютных значений используется элемент *srgbClr*, а для процентного отношения — *scrgbClr*.

Цветовая модель RGB стала популярной благодаря применению ЭЛТ-мониторов, в которых каждый пиксел описывается тремя основными цветами. Для обозначения цветов также используют цветовую модель HSL (от «hue, saturation and lightness» — «тон, насыщенность и яркость»). Эта модель более привычна для человека, потому что мы обычно не говорим, что цвет состоит из трех компонентов, а отмечаем оттенки цвета и его яркость. Тон определяет цвет, насыщенность — глубину цвета. При яркости равной 0 изображение будет в тонах серого. Яркость меняется в диапазоне от 0 (темный и черный) до 100 (белый и светлый). Цветовую модель HSL можно перевести в RGB. В модели HSL тон измеряется в 1/60000 градуса, а насыщенность и яркость — в 1/1000 процента. Поэтому тон, равный 250 градусов, в HSL будет представлен как 15 000 000 (15 миллионов). Для задания цветов согласно цветовой модели HSL служит элемент *hsClr*.

```

<!--RGB — процентное отношение -->
<a:scrgbClr r="10000" g="20000" b="30000" />
<!--RGB — абсолютное значение -->
<a:srgbClr val="597C95" />
<!-- HSL — абсолютное значение -->
<a:hslClr hue="12300000" sat="23500" lum="46700" />
<!-- Системные цвета -->
<a:sysClr val="windowText" />
<!-- Стандартные цвета -->
<a:prstClr val="black" />
<!-- Цвета схемы -->
<a:schemeClr val="accent1" />

```

### Пример 135. Цветовые режимы

В разных ОС есть несколько predefined названий цветов, например *WindowText*. Эти цвета можно задать посредством элемента *sysClr*. Элемент *prstClr* служит для predefined цветов, например *black* или *slateGray*. Список имен таких цветов есть в спецификации Open XML.

Для облегчения применения и изменения стилей можно использовать темы. Темы поддерживаются различными языками Open XML и определяют шрифты и цвета. В приведенном примере в последнем типе цвета, применяемого к тексту или фигуре, использованы цвета темы. Элемент *schemeClr* служит для задания имени цвета темы. Есть несколько встроенных имен цветов темы, и этот список нельзя расширить.

В примере 135 первые три цвета, RGB и HSL, представляют один и тот же цвет в клиентском приложении. В последних трех используются произвольные значения. Цвета можно дополнительно настраивать, но об этом мы говорить в этой книге не будем.

## Создание эффектов для фигур

Задание цвета можно использовать при создании заливки фигуры. DrawingML позволяет создавать заливки фигур с применением указанного цвета или цветового градиента, а также изображения. Необычного эффекта можно добиться, используя для заливки изображения. Также имеется обратно совместимый тип заливки по специальному образцу. Такая заливка применялась в Microsoft Office для выделения различных областей до появления новых возможностей по заданию цвета.

Сначала рассмотрим сплошную заливку, при которой вся область, занимаемая фигурой, заполняется одним цветом. Задать цвет такой заливки можно, используя любую из вышеперечисленных цветовых моделей. Для указания этого типа заливки служит элемент *solidFill*, который не принимает других параметров, кроме одного.

```

<a:solidFill>
  <a:scrgbClr
    r="10000"
    g="20000"
    b="30000" />
</a:solidFill>

```

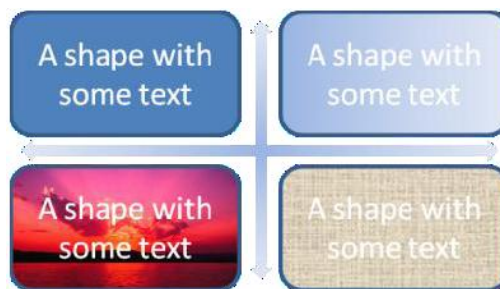


Рис. 37. Различные заливки

A shape with some text

Часто для того, чтобы придать фигуре «менее плоский» вид, используют градиентную заливку, при которой цвет плавно меняется от одного края графического объекта до противоположного, например, из левого верхнего угла в правый нижний. Чтобы создать градиент, нужно указать его положение и используемые в нем цвета. Положение градиента указывается в диапазоне от 0 до 100% и задается в 1/1000-й процентного пункта. Значение в 100 000 — это крайнее возможное положение конца градиента. Для градиента можно задать и другие параметры. Часто указывают способ поворота градиента при повороте фигуры. Для создания градиентной заливки служит элемент *gradFill*.

```

<a:gradFill flip="none"
  rotWithShape="1">
  <a:gsLst>
    <a:gs pos="0">
      <a:prstClr val="white" />
    </a:gs>
    <a:gs pos="100000">
      <a:prstClr val="black" />
    </a:gs>
  </a:gsLst>
</a:gradFill>

```

A shape with some text



```

</a:gs>
</a:gsLst>
<a:lin ang="0" scaled="1" />
<a:tileRect />
</a:gradFill>

```

Наконец, рассмотрим заливку большими двоичными изображениями или рисунками (BLIP, Binary Large Image or Picture). Этот тип заливки применяют для заливки графических объектов изображениями. Изображение можно хранить как в самом пакете, так и вне его, что определяется компонентами связей пакета. Для создания заливки изображением служит элемент *blipFill*. Он состоит из ссылки на рисунок посредством идентификатора связи. Если изображение не является встроенным, вместо него указывают атрибут *link*.

```

<a:blipFill>
  <a:blip r:embed="rId1" />
</a:blipFill>

```



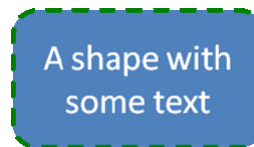
Помимо упоминавшихся типов заливки, есть еще два важных типа. Прежде всего для фигуры без заливки можно использовать элемент *noFill*. Сквозь область без заливки будут видны расположенные за ней фигуры. Элемент *grpFill* означает, что при сохранении фигуры в составе сложной фигуры она наследует тип заливки сложной фигуры.

Создав заливки графического объекта, можно поработать над его контурами. Для многих фигур в формате DrawingML для хранения информации о границах фигуры используют элемент *ln* в свойствах фигуры. С его помощью можно задать толщину, стиль заливки и стиль штриха для контура. Вы можете выбрать любой из стилей заливки, упоминавшихся ранее. Стиль штриха, применяемый для линии, тоже можно изменять. Имеется несколько стандартных стилей штриха, например стиль штриха, используемый на рисунке. Можно создавать и собственные стили, указывая длину каждого штриха и расстояние между штрихами.

```

<a:ln>
  <a:solidFill>
    <a:prstClr val="green" />
  </a:solidFill>
  <a:prstDash val="dash" />
</a:ln>

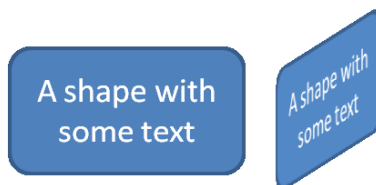
```



## Трехмерное пространство

Плоская поверхность фигур, с которыми мы работали для создания графических объектов DrawingML, на самом деле находится в трехмерном пространстве, что позволяет применять к фигурам разные необычные эффекты, например трехмерное представление диаграмм DrawingML. Для правильного создания трехмерных эффектов с помощью любого 3D-движка нужно задействовать дополнительные настройки.

При открытии объекта в формате DrawingML в клиентском приложении вы видите трехмерное изображение в трехмерном пространстве с позиции, которая определяется направлением камеры на некоторую точку. Готовая картинка получается с помощью этой камеры и выводится в виде плоского изображения на экран. Как и в реальном мире, чтобы что-то увидеть, нужен источник света. Поэтому в первую очередь нужно задать положение камеры и направление света. Направление камеры указывать не нужно, за вас это сделает DrawingML.



```

<a:scene3d>
  <a:camera
    prst="isometricOffAxis2Right" />
  <a:lightRig rig="threePt" dir="t" />
</a:scene3d>

```

Пример 136. Перемещение камеры в трехмерном пространстве

Для настройки положения камеры и освещения в трехмерном пространстве служит элемент *scene3D*. Камеру можно перемещать в одном из двух режимов: в стандартном режиме, например *isometricOffAxis2Right*, или вручную указывая координаты камеры в пространстве. Для камеры также можно настроить поле обзора и масштабирование изображения. Поле обзора определяет угол обзора камеры. У человека угол обзора примерно 80 градусов, и лишь в пределах 5 градусов мы видим все очень отчетливо. Настроить освещение не менее важно, чем положение камеры. Есть несколько стандартных

типов освещения, так же как и при настройке освещения в фото-студии. Вы не можете добавлять новые типы освещения (например, источник света, который светит красным) или задавать точное положение источников света в трехмерном пространстве. Придется выбирать из стандартных возможностей, описанных в спецификации ESMA.

Третий аспект, касающийся определения трехмерного пространства в DrawingML, — это возможность указать способ создания фона. Этот фон используется как слой, на который проецируются такие двумерные эффекты, как тени. Пример использования фона показан на рис. 38. Создается впечатление, что картина отстает от поверхности, потому что фон немного наклонен. На самом деле для достижения такого эффекта фон не используется, но в этом примере видно, как выглядит фон в трехмерном пространстве.



Рис. 38. Изменение фона

## Трехмерные эффекты

Итак, графический объект находится в трехмерном пространстве — пора пустить в дело трехмерный движок. К графическим объектам, созданным с помощью DrawingML, можно, к примеру, применять эффекты, столь привычные для трехмерных игр. Параметры графического объекта включают возможность изменения вещества, из которого сделан объект. Параметры типа вещества и типа освещения «работают вместе»: вещество отражает свет, как и в реальном мире. Можно создавать разные типы поверхностей, по-разному отражающие и поглощающие свет. Также у плоских графических объектов, помещенных в трехмерное пространство, нет глубины. Чтобы преодолеть это ограничение, можно применить к объекту эффект глубины или добавить скошенную границу, имитирующую глубину.



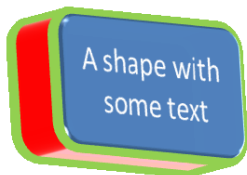
Для задания трехмерных параметров графического объекта служит элемент *sp3D*. В этом элементе можно указать тип материала, а также скошенную границу объекта.

На следующих трех рисунках показано применение экструзии («выдавливания») для придания графическому объекту глубины: в качестве контура фигуры использована «лента» и скошенная граница. За исходный графический объект в этих примерах взят объект, приведенный в начале этой главы. К объектам также применили поворот, чтобы результат преобразований был лучше виден.



Рис. 39. Применение к фигуре трехмерных параметров

Все эти эффекты можно также сочетать. В следующем коде показано, как задавать 3D-параметры фигуры.



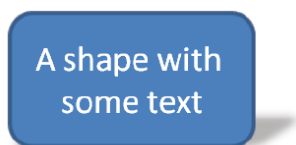
```
<a:sp3d prstMaterial="softEdge"
  extrusionH="412750"
  contourW="82550" >
  <a:bevelT />
  <a:extrusionClr>
    <a:srgbClr val="FF0000" />
  </a:extrusionClr>
  <a:contourClr>
    <a:srgbClr val="92D050" />
  </a:contourClr>
</a:sp3d>
```

Пример 137. 3D-параметры

Второй элемент, хранящий данные о трехмерных преобразованиях, применяют при создании эффектов для фигур. Чтобы применить к фигуре трехмерный эффект, нужно воспользоваться списком эффектов. Список эффектов имеется для таких элементов, как фигуры, рисунки, таблицы и диаграммы. Он содержит фиксированный набор стандартных примитивов эффектов. Настраивая все эти эффекты, можно создать большее количество готовых эффектов. В этой книге подробно рассматриваются эффекты свечения,



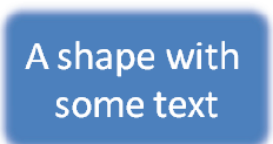
тени, отражения и размытой границы. Все они определяются в элементе контейнера, похожем на *sp3d*, — *effectLst*.



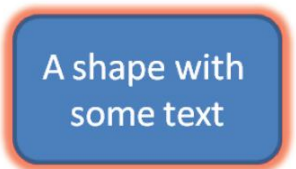
```
<a:effectLst>
  <a:outerShdw blurRad="76200"
    dir="18900000" sy="23000"
    kx="-1200000" algn="b1"
    rotWithShape="0">
    <a:prstClr val="black">
      <a:alpha val="20000" />
    </a:prstClr>
  </a:outerShdw>
</a:effectLst>
```



```
<a:effectLst>
  <a:reflection
    blurRad="6350"
    stA="52000" endA="300"
    endPos="35000" dir="5400000"
    sy="-100000" algn="b1"
    rotWithShape="0" />
</a:effectLst>
```



```
<a:effectLst>
  <a:softEdge rad="63500" />
</a:effectLst>
```



```
<a:effectLst>
  <a:glow rad="101600">
    <a:srgbClr val="FF0000">
      <a:alpha val="60000" />
    </a:srgbClr>
  </a:glow>
</a:effectLst>
```

В следующей таблице показано, какие эффекты можно применять к компонентам PresentationML.

	Фигура <i>sp</i>	Рисунок <i>pic</i>	Текст <i>txBody</i>	Таблица <i>tbl</i>	Диаграмма <i>chart</i>
Двумерное вращение	x				
Преобразование текста			x		
Тень	x		x	x	
Отражение	x		x	x	
Свечение	x		x		
Размытый край	x				
Скошенный край	x		x	x	
Трехмерное вращение	x	x	x		

## Таблицы

С помощью DrawingML можно определять таблицы. Более того, некоторые из эффектов DrawingML для фигур можно применять к таблицам целиком или на уровне отдельных ячеек. Такая модель таблицы, используемая в основном в PresentationML, позволяет применять таблицы со сложным форматированием. В Open XML есть три модели таблиц. О первой говорилось в главе, посвященной WordprocessingML. Эта модель предназначена для хранения таблиц, используемых в документах, и не поддерживает трехмерных эффектов, создаваемых с помощью DrawingML. Модель таблиц SpreadsheetML служит для хранения больших объемов данных, поскольку потенциально в таблице может быть большое количество строк. Модель таблиц DrawingML позволяет создавать впечатляющие графические объекты.

### Элементы таблицы

Для создания таблицы служит элемент *tbl* из пространства имен DrawingML. Модель для объявления структуры таблицы похожа на WordprocessingML. Элемент таблицы содержит определение сетки и набор строк. Определение сетки задает все «виртуальные» столбцы. Рассмотрим таблицу из примера 138.

Хотя в каждой строке всего две ячейки, в определении сетки каждую вертикальную линию нужно рассматривать как столбец. Поэтому в этой таблице четыре столбца сетки: в ней есть четыре вертикальные линии без учета первого столбца (начальный столбец определять не нужно). Таким образом, в определении сетки задано три столбца.

```
<a:tbl>
  <a:tblPr firstRow="1" bandRow="1" />
  <a:tblGrid>
    <a:gridCol w="1325565" />
    <a:gridCol w="1325565" />
    <a:gridCol w="1325565" />
  </a:tblGrid>
  <a:tr h="463019">
    <a:tc>
      <a:txBody />
    </a:tc>
    <a:tc />
    <a:tc />
  </a:tr>
  <a:tr h="463019">
    <a:tc />
    <a:tc />
    <a:tc />
  </a:tr>
</a:tbl>
```

#### Пример 138. Разметка таблицы

После объявления определения сетки таблицы добавляется последовательность строк. Нужно определить хотя бы одну строку, иначе таблица в редакторе не отобразится. Строки (*tr*) содержат отдельные ячейки (*tc*). Эта модель точно такая же, как в WordprocessingML и даже в HTML. В отличие от WordprocessingML, в ячейках таблиц DrawingML можно размещать лишь текст. Текст в формате DrawingML хранится в контейнере *txBody*. Благодаря этому к любому символу в таблице можно применять различные эффекты, например отражение. Подробнее об определении текста в формате DrawingML см. раздел, посвященный тексту. В примере 138 таблица имеет три столбца и две строки. В сетке таблицы хранится ширина каждого столбца. Ее нельзя задать на уровне каждой ячейки.

Как и в WordprocessingML, вы можете выполнять объединение ячеек по горизонтали и по вертикали. Также каждая ячейка таблицы может участвовать не более чем в одном объединении ячеек; нельзя одновременно объединить ячейку с ее соседней ячейкой по горизонтали и с ячейкой под ней по вертикали. Такая модель объединения ячеек очень похожа на объединения ячеек для таблиц в формате WordprocessingML. В отличие от WordprocessingML объединение ячеек не уменьшает общего количества ячеек, определенных в элементе *tr*. В таблице формата PresentationML для строки всегда определено одинаковое число ячеек, потому что в таблице определены столбцы сетки.

### Объединение ячеек

Объединение ячеек таблицы несколько отличается от модели, применяемой в WordprocessingML. В WordprocessingML каждая группа ячеек, объединенных по горизонтали, на единицу уменьшает число элементов *tc*, определяемых в коде соответствующей строки. В таблицах DrawingML при этом общее число ячеек не уменьшается. У первой ячейки в группе объединенных ячеек есть атрибут, показывающий, что эта ячейка является начальной в группе горизонтального объединения ячеек. У остальных ячеек той же группы объединения есть атрибут, показывающий их принадлежность к той же группе. Аналогично выполняется вертикальное объединение, но для его определения служат другие атрибуты. Одновременно объединять одну и ту же ячейку с другими по горизонтали и по вертикали нельзя. В

примере 139 показаны свойства, необходимые для объединения ячеек по горизонтали и по вертикали. Отметим также, что в клиентском приложении будет отображаться содержимое только первой ячейки в группе объединенных ячеек. В других ячейках также можно хранить данные, но пользователю они будут не видны. И все же эти скрытые данные пользователь может обнаружить средствами поиска.

Горизонтальное объединение	Вертикальное объединение
<pre>&lt;a:tr&gt;   &lt;a:tc gridSpan="2" /&gt;   &lt;a:tc hmerge="true" /&gt; &lt;/a:tr&gt;</pre>	<pre>&lt;a:tr&gt;   &lt;a:tc rowspan="2" /&gt; &lt;/a:tr&gt; &lt;a:tr&gt;   &lt;a:tc vmerge="true" /&gt; &lt;/a:tr&gt;</pre>

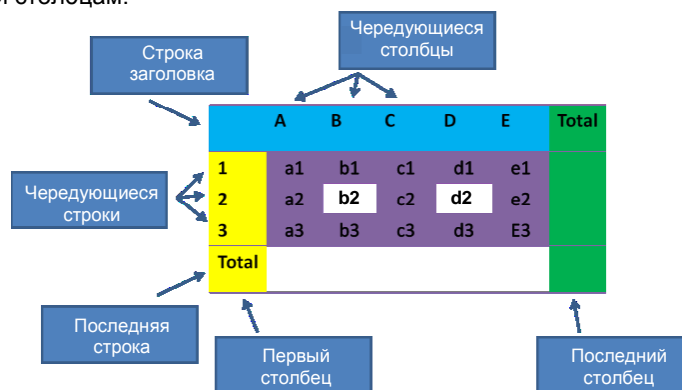
Пример 139. Объединение ячеек таблицы по горизонтали и по вертикали

## Форматирование таблиц

Для создания таблиц для профессиональных презентаций в модели таблиц DrawingML есть параметры, применяемые как на уровне всей таблицы, так и ее отдельных строк и ячеек. Чтобы задать такие свойства строки или ячейки, как заливка (например, градиентная), используют узел графических свойств DrawingML. Элемент таблицы также поддерживает добавление параметров DrawingML, применяемых ко всей таблице, например отражение. При выборе стилей для таблицы появилась замечательная возможность многократно использовать стили таблицы для разных таблиц в группе слайдов. В модели таблицы стили таблицы отделены от основного определения таблицы. Таким образом, тот же стиль можно применить для всех таблиц в презентации, чтобы придать единообразие слайдам.

Для непосредственного форматирования строки или ячейки таблицы служит узел свойств для элемента (*tcPr* для ячеек таблицы), в котором хранятся параметры DrawingML. Эти параметры мы рассмотрим в разделе, посвященном графическим эффектам DrawingML. То же можно сказать и о добавлении эффектов для таблицы. Узел *tblPr* служит в качестве элемента-контейнера для эффектов фигур DrawingML. Также при определении свойств на уровне таблицы можно задать специальный стиль таблицы.

Специальный стиль таблицы содержит сведения о типе линий, заливке, шрифте и цвете для отдельных элементов таблицы. Используя стили таблицы, можно указать разнообразные параметры для строк и столбцов таблицы. Параметры для строки заголовка и последней строки, а также для первого и последнего столбца различны. Еще большего эффекта вы добьетесь, применив к строкам и столбцам чередование. Чередование строк и столбцов позволяет задать стили для четных и нечетных строк. Выбирая разный фон для четных и нечетных элементов, можно визуально выделить ячейки, относящиеся к отдельным строкам и столбцам.



Информация о пользовательских стилях хранится в отдельном компоненте пакета. Есть всего лишь один компонент стиля таблицы, который может содержать несколько стилей таблицы. В определении таблицы ссылка на табличный стиль выполняется по его идентификатору. Для идентификации стиля таблицы применяется глобальный уникальный идентификатор (GUID), отличный от других идентификаторов, как правило, целых чисел.

```
<a:tbl>
  <a:tblPr firstRow="1" bandRow="1">
    <a:tableStyleId>
      {37CE84F3-28C3-443E-9E96-99CF82512B78}
    </a:tableStyleId>
  </a:tblPr>
</a:tbl>
```

Пример 140. Ссылка на стиль таблицы

В свойствах таблицы задают разнообразные параметры для первых и последних элементов и чередования, которые будут применяться к таблице. Данные о стиле хранятся отдельно, в компоненте стиля таблицы. Если параметр *firstRow* в свойствах таблицы равен 1, в клиентском приложении для стиля первой строки таблицы будет применяться стиль, заданный глобальным уникальным идентификатором для первой строки таблицы. Свойство *bandRow* означает, что клиентская программа применит к первой строке стиль *band1H*, ко второй — *band2H*, к третьей — опять *band1H*. Стили таблицы можно использовать вместе с непосредственным форматированием. Непосредственное форматирование имеет больший приоритет, чем заданный стиль, что дает еще больше возможностей для изменения внешнего вида таблицы.

Стили таблицы хранятся в отдельном компоненте стилей таблицы. Ссылка на этот компонент стилей таблицы из компонента презентации выполняется посредством следующего типа связи:

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/tableStyles>

Эта связь указывает на компонент, содержащий список стилей таблицы, в котором хранятся стили, различаемые по их GUID. Имеется один стиль таблицы, который считается стилем по умолчанию. Этот стиль применяется к новым таблицам, добавляемым пользователем в презентацию. Стиль таблицы определяется с помощью двух типов содержимого: информации для его идентификации, состоящей из его GUID и его имени, отображаемого в пользовательском интерфейсе, а также набора данных о стиле для всех компонентов таблицы, например строки заголовка и последнего столбца.

```
<a:tblStyleLst
  xmlns:a="http://schemas.openxmlformats.org/drawingml/2006/main"
  def="{5C22544A-7EE6-4342-B048-85BDC9FD1C3A}">
  <a:tblStyle
    styleId="{5C22544A-7EE6-4342-B048-85BDC9FD1C3A}"
    styleName="Medium Style 2 - Accent 1">
    <!--Описание стиля -->
  </a:tblStyle>
</a:tblStyleLst>
```

#### Пример 141. Стили таблицы

К таблице можно применить тринадцать стилей.

Элемент стиля	Область применения	Требуемые свойства таблицы
<b>wholeTbl</b>	Форматирование всей таблицы	Всегда
<b>band1H</b>	1, 3, 5-я ... строка	bandRow="1"
<b>band2H</b>	2, 4, 6-я ... строка	bandRow="1"
<b>band1V</b>	1, 3, 5-й ... столбец	bandCol="1"
<b>band2V</b>	2, 4, 6-й ... столбец	bandCol="1"
<b>firstCol</b>	Первый столбец	firstCol="1"
<b>lastCol</b>	Последний столбец	lastCol="1"
<b>firstRow</b>	Первая строка	firstRow="1"
<b>lastRow</b>	Последняя строка	lastRow="1"
<b>seCell</b>	Правый нижний угол	lastRow="1", lastCol="1"
<b>swCell</b>	Левый нижний угол	lastRow="1", firstCol="1"
<b>neCell</b>	Правый верхний угол	firstRow="1", lastCol="1"
<b>nwCell</b>	Левый верхний угол	firstRow="1", firstCol="1"

Таблица 8. Типы стилей таблицы

Каждый из этих элементов стиля таблицы позволяет определить два типа данных: стиль ячейки и стиль текста в ячейке.

```
<a:tblStyle
  styleId="{5C22544A-7EE6-4342-B048-85BDC9FD1C3A}"
  styleName="Medium Style 2 - Accent 1">
  <a:wholeTbl>
    <a:tcTxStyle>
      <a:fontRef />
      <a:schemeClr />
    </a:tcTxStyle>
    <a:tcStyle>
```

```

    <a:cell3D />
    <a:fill />
    <a:tcBdr />
  </a:tcStyle>
</a:wholeTbl>
</a:tblStyle>

```

Пример 142. Пример определения стиля для всей таблицы

Стиль ячейки таблицы определяет параметры ячеек, а стиль текста ячеек таблицы — параметры текста в ячейке. Есть несколько параметров форматирования. Стиль текста определяет шрифт и цвет, а стиль ячейки — некоторые трехмерные параметры, границы и заливку ячеек. В целом с помощью этих параметров можно создавать таблицы с более сложным форматированием, а применяя разметку, можно легко изменять внешний вид таблицы.

## Диаграммы

Диаграммы позволяют создавать наглядное представление информации с использованием таких хорошо знакомых макетов, как круговая или линейчатая диаграмма. Данные, представленные в диаграмме, можно использовать во всех форматах Open XML. Одну и ту же диаграмму можно добавлять в документы WordprocessingML, SpreadsheetML и PresentationML, лишь указав индивидуальный для каждого языка элемент контейнера для хранения кода диаграммы. Ссылку на код, определяющий диаграмму, можно разместить только в таком контейнере. Сама диаграмма хранится в отдельном компоненте пакета. Поскольку диаграмма — довольно большая структура, ей будет посвящено несколько следующих разделов. Сначала мы рассмотрим содержимое базовой диаграммы, а затем различные вариации макета диаграммы.

### Элементы простой диаграммы

С помощью кода DrawingML можно создавать несколько типов диаграмм: различные типы двумерных и трехмерных диаграмм, а также сводные диаграммы, в основном применяемые в SpreadsheetML. Ко всем этим типам диаграмм можно добавлять дополнительные элементы диаграмм, например, подписи столбцов и строк или названия и легенды. Также в диаграммы можно добавлять различные фигуры для изменения их внешнего вида в презентации. На рис. 40 показаны элементы простой двумерной диаграммы. Поскольку многие элементы, применяемые во всех типах двумерных и трехмерных диаграмм, похожи, сначала рассмотрим простую диаграмму с жестко запрограммированными данными, а затем применим к ней дополнительные возможности, чтобы изменить ее внешний вид.

Чтобы создать простую диаграмму, в пакет нужно добавить отдельный компонент для хранения кода диаграммы. Корневой элемент этого кода — *chartSpace* — определяет диаграмму и другие параметры, в том числе сведения о стилях, применяемых в диаграмме.

Здесь же можно задать привязку данных к другому источнику данных, например к пакету SpreadsheetML, вложенному в презентацию. Привязку диаграммы к источнику данных мы рассмотрим ниже в этом разделе. В корневом элементе *chartSpace* единственный необходимый элемент — диаграмма. Диаграмма определяет все составляющие ее элементы, в том числе ее название, область построения и данные, на основе которых строится диаграмма. Создавать определение нужно только для элемента *plotArea*. На рис. 40 показана простая диаграмма. Область построения — это область с градиентным фоном. Область построения определяет тип отображаемой диаграммы и то, является она двумерной или трехмерной. Принцип работы области построения довольно прост. Для каждого типа диаграммы есть специальный определяющий его XML-элемент. Так, есть элементы *barChart*, *pieChart* и *barChart3D*. В качестве дочернего элемента для *plotArea* можно указать только один из них.



Рис. 40. Простая диаграмма

```

<c:chartSpace
  xmlns:c="http://.../drawingml/2006/chart"
  xmlns:a="http://.../drawingml/2006/main"
  <c:chart>
    <c:plotArea>
      <c:layout />
      <c:barChart />
      <c:catAx />

```

```

    <c:valAx />
    <c:spPr />
  </c:plotArea>
</c:chart>
</c:charSpace>

```

#### Пример 143. Простая диаграмма

В этом примере использована линейчатая диаграмма. Для ее создания в область диаграммы нужно добавить дочерний элемент *barChart*. Именно так ограничивается набор допустимых типов диаграммы. Помимо обязательного указания типа диаграммы, можно задать и другие параметры. Элемент макета в разметке диаграммы служит для указания таких параметров, как положение подписей или легенды. Используя элемент макета, вы также можете перемещать всю диаграмму в области диаграммы. В диаграмме, показанной на рисунке выше, две оси: горизонтальная — для категорий и вертикальная — для значений. Для создания осей в области диаграммы служат элементы *catAx* и *valAx*. В области диаграммы для каждого типа можно создавать по одному из этих элементов. В диаграмме всегда может быть только одна ось категорий, одна — значений, одна — времени и одна — ряда данных.

```

<c:catAx>
  <c:axId val="70435584" />
  <c:scaling>
    <c:orientation val="minMax" />
  </c:scaling>
  <c:axPos val="b" />
  <c:tickLblPos val="nextTo" />
  <c:crossAx val="70437120" />
  <c:crosses val="autoZero" />
  <c:auto val="1" />
  <c:lblAlgn val="ctr" />
  <c:lblOffset val="100" />
</c:catAx>

```

#### Пример 144. Определение оси категорий

Поскольку ссылки на элементы оси категорий и оси значений находятся в определении линейчатой диаграммы, прежде чем подробно ее рассматривать, нужно остановиться на этих двух элементах.

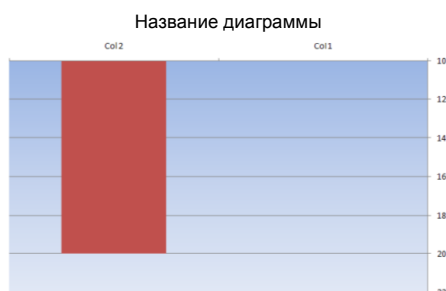


Рис. 41. Изменение определения оси диаграммы

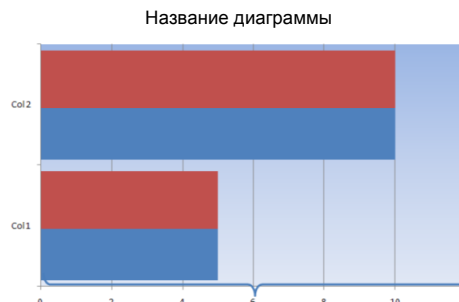


Рис. 42. Изменение параметров линейчатой диаграммы

Элементы оси определяют внешний вид осей диаграммы, точку их пересечения и отображаемые ими данные. С каждым определением оси связан идентификатор, который затем служит для создания ссылок на ось в элементе диаграммы. В примере 144 показано определение оси категорий для диаграммы с рис. 40. Этот идентификатор представляет собой случайное большое число и служит для создания ссылки на ось из других мест. Элемент масштаба определяет диапазон данных для оси; если он не был задан, как в примере, диапазон данных определяется на основе имеющихся данных. Ориентация определяет расположение значений на оси — от минимального к максимальному или наоборот. Для определения осей, с которыми пересекается данная ось, используют их идентификаторы. Можно точно задать место пересечения двух осей, в этой точке значение по этим осям будет равно 0. В оставшейся части определения оси указываются сведения о положении подписей осей, используемых делениях, а также о создании сетки делений на поверхности диаграммы. Благодаря этой сетке пользователи могут лучше «читать» диаграмму, но внешний вид диаграммы сильно меняется. На рис. 41 вы видите ту же диаграмму, что и на рис. 40. На этот раз значения по оси значений и оси категорий расположены от максимального к минимальному, а ось значений пересекается на значении «10» и все столбцы ниже этого значения оказываются скрытыми.

Теперь обратимся к определению *barChart*. Оно содержит такие элементы, как параметры форматирования для строк, ссылки на ось, отображаемую вдоль строк и, что более важно, отображаемые данные. Некоторые элементы применяются только для линейчатых диаграмм, например направление строк, а другие (скажем, определение данных или привязка оси) являются общими для всех типов диаграммы. Не внося изменений в данные, можно отформатировать исходную диаграмму (рис. 40) так, что она будет выглядеть, как на рис. 42.

```

<c:barChart>
  <c:barDir val="col" />
  <c:grouping val="stacked" />
  <c:ser>...</c:ser>
  <c:ser>...</c:ser>
  <c:ser>...</c:ser>
  <c:gapWidth val="100" />
  <c:overlap val="100" />
  <c:axId val="70435584" />
  <c:axId val="70437120" />
</c:barChart>

```

#### Пример 145. Определение линейчатой диаграммы

Данные, составляющие диаграмму, определяют с помощью рядов данных посредством элемента *ser*. Внутри каждого ряда данных следует определить данные, используемые для категорий и значений. В диаграмме из примера есть два ряда данных, отображаемые на диаграмме разными цветами. В каждом из этих рядов данных определено два элемента данных, поэтому в диаграммах, приведенных выше, по два столбика.

```

<c:ser>
  <c:idx val="1" />
  <c:order val="1" />
  <c:cat>
    <c:strLit>
      <c:ptCount val="2" />
      <c:pt idx="0">
        <c:v>Col 1</c:v>
      </c:pt>
      <c:pt idx="1">
        <c:v>Col 2</c:v>
      </c:pt>
    </c:strLit>
  </c:cat>
  <c:val>
    <c:numLit>
      <c:ptCount val="2" />
      <c:pt idx="0">
        <c:v>5</c:v>
      </c:pt>
      <c:pt idx="1">
        <c:v>10</c:v>
      </c:pt>
    </c:numLit>
  </c:val>
</c:ser>

```

#### Пример 146. Ряд данных диаграммы

Ряд данных диаграммы из примера показан в примере 146. В заголовке элемента указан уникальный идентификатор и информация о порядке. Затем в примере задан элемент категории и элемент значений, в каждом из которых определен список значений. Элемент *strLit* определяет жестко запрограммированную строку литерала, точно так же в элементе *numLit* определяются жестко запрограммированные значения. Вместо этих жестко запрограммированных значений можно использовать элементы *strRef* и *numRef*, чтобы создать ссылки на данные в отдельном источнике данных, если такой в диаграмме применяется. Сохраняемые данные представлены в более привычном виде — как таблица в заголовке этого абзаца.

При использовании категорий и значений может быть допущено несколько ошибок. Являются ли они критическими, зависит от конкретного программного продукта. Например, в PowerPoint в ситуации, когда значений больше, чем категорий, вместо недостающих элементов будет отображаться пустая подпись. Возможны ошибки и в значениях индекса для строки и числовых литералов, но тогда диаграмма получится странной.

### Привязка данных диаграммы

При создании диаграмм данные можно жестко запрограммировать, но лучше хранить данные отдельно от определения диаграммы. Есть два типа источников данных, которые могут быть использованы для диаграммы: данные SpreadsheetML и сводные таблицы. В этом разделе рассматривается источник данных SpreadsheetML.

Источник данных SpreadsheetML определяется в диаграмме посредством компонента связей. Этот источник может быть встроен в пакет или храниться отдельно на диске, что обеспечивает полную свободу выбора места хранения данных. Чтобы клиентская программа получила доступ к данным даже в случае

недоступности сетевых подключений, данные из таблицы кэшируются в коде диаграммы. Можно задать специальный параметр, чтобы клиентская программа извлекала эти данные при открытии документа.

Сначала при создании привязки к внешнему источнику данных нужно создать новый компонент связей пакета, используя в качестве источника компонент диаграммы. Нужно выбрать тип связи *officeDocument*, поскольку для определения источника данных нет специального типа связи. Чтобы присоединить таблицу к диаграмме, нужно добавить в корневой элемент *chartSpace* простой элемент (пример 147).

```
<c:chartSpace
  xmlns:c="http://.../drawingml/2006/chart"
  xmlns:a="http://.../drawingml/2006/main"
  xmlns:r="http://.../officeDocument/2006/relationships">
  <!--Другое содержимое диаграммы -->
  <c:externalData r:id="rId1" />
</c:chartSpace>
```

#### Пример 147. Внешний источник данных диаграммы

Мы рассмотрим извлечение данных для диаграммы из внешней таблицы. Пока данные в таблице не используются. Чтобы настроить использование в диаграмме указанного источника данных, нужно изменить ряд данных так, чтобы в нем применялись строки и числа, на которые указывает ссылка, а не жестко закодированные данные. В элементе *ser*, который служит для определения ряда данных, для определения категорий и значений ряда данных точно так же применяются элементы *cat* and *val*. Только на этот раз эти элементы содержат узлы *strRef* и *numRef*, ссылающиеся на данные из внешнего источника. Ссылки на строку и число содержат функцию *SpreadsheetML*, позволяющую извлекать данные из источника данных как диапазон ячеек с данными. Затем значения, вычисленные с помощью этой функции, кэшируются в специальном узле кэша для этих ссылок, например в элементе *strCache*.

```
<c:ser>
  <c:cat>
    <c:strRef>
      <c:f>Sheet1!$A$1:$B$1</c:f>
      <c:strCache>
        <c:ptCount val="2" />
        <c:pt idx="0">
          <c:v>
            </c:v>
          </c:pt>
        <c:pt idx="1">
          <c:v>Sales</c:v>
        </c:pt>
      </c:strCache>
    </c:strRef>
  </c:cat>
</c:ser>
```

#### Пример 148. Ряд данных с внешним источником данных

### Создание дополнительных надписей на диаграмме

При определении диаграммы несложно добавить в нее дополнительные надписи, например название или легенду. Данные диаграммы также можно отобразить в форме таблицы внизу под диаграммой.

Для добавления в диаграмму каждой из этих дополнительных надписей служит специальный элемент, определяющий их содержимое. Кроме того, в диаграмму можно добавлять фигуры, но это сделать сложнее по сравнению с созданием стандартных надписей на диаграмме. Название таблицы, как и легенду, определяют непосредственно в элементе *chart*.

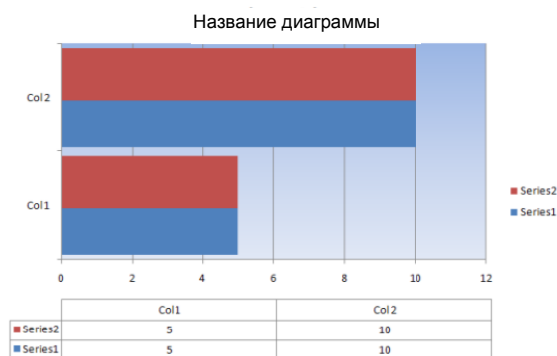


Рис. 43. Дополнительные надписи

Название диаграммы можно определить двумя способами: либо жестко запрограммировать (точно так же, как данные диаграммы) посредством блока текста, либо создать ссылку на строку в источнике данных с помощью элемента *strRef*, о котором мы говорили выше. В блоке текста помещается текст *DrawingML*, которому посвящен специальный раздел этой главы. Легенда диаграммы также находится в самом элементе *chart*. Здесь есть лишь несколько параметров, которые нужно задать значение: достаточно указать положение легенды справа от диаграммы. Используя узел свойств фигуры, можно форматировать название и легенду точно так же, как определяются свойства нестандартных фигур.



Имеются и стандартные средства для добавления под диаграммой таблицы данных. Но в этом случае этот элемент является частью области диаграммы. Как и при задании наличия таблицы данных, вам нужно лишь указать содержимое таблицы данных, а не точный способ его отображения. Затем можно форматировать область таблицы данных, используя свойства фигуры, но, к примеру, добавлять новые элементы в таблицу данных или изменять отображение надписей за пределами предустановленных значений нельзя.

```
<c:chart>
  <c:title>
    <c:tx>
      <c:rich>
        <!--Текст в формате RTF-->
      </c:rich>
    </c:tx>
  </c:title>
```

```
<c:legend>
  <c:legendPos val="r" />
  <c:layout />
</c:legend>
<c:plotVisOnly val="1" />
</c:chart>
```

Пример 149. Название и легенда диаграммы

```
<c:plotArea>
  <c:layout />
  <c:dTable>
    <c:showHorzBorder val="1" />
    <c:showVertBorder val="1" />
    <c:showOutline val="1" />
    <c:showKeys val="1" />
  </c:dTable>
</c:plotArea>
```

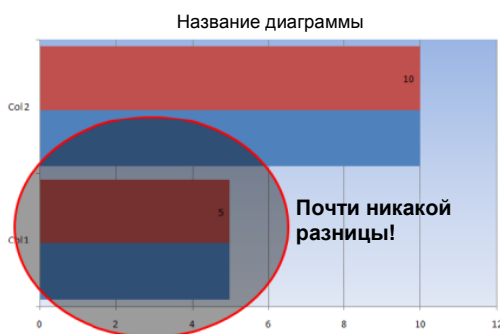
Пример 150. Подпись таблицы данных диаграммы

## Добавление нестандартных фигур

При создании диаграммы может потребоваться привлечь внимание к какой-то ее области. Для этого на поверхности диаграммы можно создавать фигуры по тому же принципу, что и задание внешнего источника данных.

```
<c:chartSpace>
  <c:userShapes r:id="rId1" />
</c:chartSpace>
```

Пример 151. Ссылка на рисунки на диаграмме



В качестве ссылки на компонент, содержащий фигуры диаграммы, используется специальный тип связи:

<http://schemas.openxmlformats.org/officeDocument/2006/relationships/chartUserShapes>

В этом компоненте список фигур и групповых фигур хранится в специальных пространствах имен DrawingML. Принцип работы этого списка аналогичен фигурам в PresentationML. Основное отличие в том, что у каждого элемента фигуры *sp* есть специальная привязка, позволяющая поместить элемент на поверхности диаграммы. Эти привязки позволяют перемещать фигуру и изменять ее размеры одновременно с перемещением и изменением размеров диаграммы.

```
<c:userShapes
  xmlns:c="http://.../drawingml/2006/chart"
  xmlns:a="http://.../drawingml/2006/main"
  xmlns:cdr="http://.../drawingml/2006/chartDrawing">
  <cdr:absSizeAnchor>
    <cdr:from>
      <cdr:x>0.03516</cdr:x>
      <cdr:y>0.35156</cdr:y>
    </cdr:from>
    <cdr:ext cx="600000" cy="600000" />
    <cdr:sp macro="" textlink="">
      <!--Содержимое фигуры -->
    </cdr:sp>
```

```
</cdr:absSizeAnchor>
</c:userShapes>
```

### Пример 152. Задание абсолютного положения пользовательской фигуры

В примере 152 показано применение абсолютной привязки, при которой размер фигуры не изменяется при изменении размера диаграммы. Привязка относится ровно к одной фигуре, но это может быть и групповая фигура, содержащая дочерние фигуры. В элементах *userShapes* может быть несколько привязок для различных фигур. Сама разметка фигуры похожа на PresentationML, но в ней используется пространство имен DrawingML. Различные свойства позволяют создавать определения фигур и указывать их стили в диаграмме. Для абсолютной привязки требуется указать положение и размер фигуры. Чтобы сделать абсолютную привязку фигуры относительной, которая позволяет изменять размер фигуры одновременно с родительской диаграммой, элемент *absSizeAnchor* следует заменить на *relSizeAnchor*, остальная часть кода будет той же за исключением размера фигуры. При абсолютной привязке определяют ширину и высоту, а при относительной размер фигуры задают в процентном отношении от размера всей диаграммы. Для задания относительного размера фигуры служат элементы *from* и *to*.

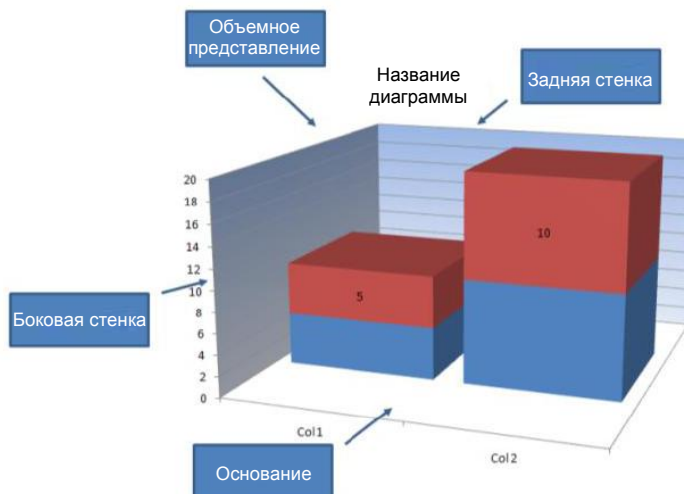
```
<cdr:relSizeAnchor>
  <cdr:to>
    <cdr:x>0.94923</cdr:x>
    <cdr:y>0.98438</cdr:y>
  </cdr:to>
</cdr:relSizeAnchor>
```

### Пример 153. Относительная привязка

## Создание трехмерных диаграмм

Помимо плоских двумерных диаграмм, о которых мы говорили до сих пор, аналогично можно создавать трехмерные диаграммы. Для них можно задавать такие дополнительные трехмерные свойства, как угол обзора и другие. Диаграмма отображается на трех плоскостях, для которых можно указать параметры стиля.

Чтобы превратить плоское представление диаграммы в объемное, нужно сделать три вещи. Сначала — изменить тип диаграммы в узле *plotArea*. В данном примере использован элемент *bar3DChart*, содержимое которого отличается от его аналога для двумерной диаграммы лишь тем, что в нем появляется ссылка на третью ось.



Если в качестве идентификатора этой оси задано значение «0», отображается ось «по умолчанию», без значений. Это позволяет легко переключаться между двумерным и трехмерным представлениями диаграммы.

Также можно задать стили для видимых поверхностей трехмерного пространства, в котором конструируется диаграмма. Эти настройки задают прямо в элементе *chart*. Вы можете задать стили для задней стенки, боковой стенки и основания. Суть в том, что каждый из трех элементов, определяющих стили плоскостей, содержит узел свойств фигуры, в котором задают преобразования DrawingML, применяемые к соответствующей стенке.

```
<c:plotArea>
  <c:layout />
  <c:bar3DChart>
    <!-- Те же данные, что и в элементе barChart -->
  </c:bar3DChart>
</c:plotArea>
```

### Пример 154. Определение трехмерной линейчатой диаграммы

Последний элемент, который нужно указать, — положение камеры, отображающей диаграмму. Камера определяет точку, с которой отображается диаграмма. С ее помощью можно создавать профессиональное представление данных в диаграмме или непрофессиональное, когда угол камеры выбран неудачно. Положение камеры задают с помощью элемента *view3D*, в котором указывают поворот и направление камеры: в перспективе или справа. Направление «в перспективе» дает более привычный вид, поскольку похоже на взгляд человека.

```

<c:chart>
  <c:view3D>
    <c:perspective val="30" />
  </c:view3D>
  <c:sideWall>
    <c:spPr />
  </c:sideWall>
  <c:floor>
    <c:spPr />
  </c:floor>
</c:chart>

```

Пример 155. Задание трехмерных свойств

## Темы

Темы позволяют задавать стиль документа, таблицы или презентации, используя одинаковые цвета во всех языках Open XML. Тема — это отдельный компонент в пакете, содержащий определения шрифтов, цветов и эффектов DrawingML для фигур, о которых говорилось ранее. Есть несколько типов параметров цвета и заливки, позволяющие привязать стиль фигуры к теме. При изменении темы все фигуры, основанные на ней, также изменятся.

```

<a:theme
  xmlns:a="http://.../drawingml/2006/main"
  name="My Scheme">
  <a:themeElements>
    <a:clrScheme name="My Scheme" />
    <a;fontScheme name="My Scheme" />
    <a:fmtScheme name="My Scheme" />
  </a:themeElements>
  <a:objectDefaults />
  <a:extraClrSchemeLst />
</a:theme>

```

Пример 156. Элементы темы

Определение темы задают в элементе *theme*, в котором нужно создать узел *themeElements*, содержащий все сведения о теме. Здесь же можно указать дополнительную информацию или переопределить имеющуюся в элементах темы. Элементы темы конструируются посредством набора дочерних узлов, каждый из которых определяет отдельный элемент данных, например шрифты или цвета. Элемент *clrScheme* служит для задания списка цветов. Список цветов темы содержит такие элементы, как *accent1* или *visited link* (посещенная ссылка). Для каждого из этих цветов темы используется отдельный элемент XML, в котором имя узла определяет цвет темы. Внутри этого элемента цвета задаются так же, как было описано в разделе, посвященном установке цвета фигуры с помощью DrawingML. В примере 157 определено два цвета темы — *dk1* и *accent1*. Этот пример не полон: нужно определить значение для всех двенадцати цветов темы.

```

<a:clrScheme name="My Scheme">
  <a:dk1>
    <a:sysClr val="windowText" lastClr="000000" />
  </a:dk1>
  <a:accent1>
    <a:srgbClr val="4F81BD" />
  </a:accent1>
</a:clrScheme>

```

Пример 157. Цвета темы

Схема шрифта работает по тому же принципу, что и параметры цвета схемы. Нужно задать два шрифта, которые будут применяться в документе Open XML — основной и дополнительный. Затем эти два шрифта подразделяются на конкретные шрифты для каждого языка. Для любого естественного языка можно задать основной и дополнительный шрифт. Шрифт, используемый в документе, зависит от региональных параметров ОС.

```

<a;fontScheme name="Office">
  <a:majorFont>
    <a:latin typeface="Calibri" />
    <a;font script="Jpan" typeface="MS Pゴシック" />
  </a:majorFont>
  <a:minorFont>
    <a:latin typeface="Calibri" />
    <a;font script="Jpan" typeface="MS Pゴシック" />
  </a:minorFont>
</a;fontScheme>

```

```
</a:minorFont>
</a:fontScheme>
```

#### Пример 158. Шрифты темы

И последний элемент, который можно определить для темы, — это параметры форматирования фигуры, состоящие из четырех обязательных групп: заливки, линии, эффекты и цвета фона. В каждой из них можно задать параметры заливки и стили линий, а также другие параметры форматирования, упоминавшиеся в этом разделе.

```
<a:fmtScheme name="Office">
  <a:fillStyleLst />
  <a:lnStyleLst />
  <a:effectStyleLst />
  <a:bgFillStyleLst />
</a:fmtScheme>
```

#### Пример 159. Форматирование фигур темы

Сохранив параметры темы в пакете, их также нужно применить к элементам документа. При этом надо учесть два момента. В DrawingML есть лишь несколько элементов, позволяющих ссылаться на параметры темы. Эти элементы должны храниться в том или ином Open XML-контейнере. В примере 160 показан элемент контейнера, используемый для фигур PresentationML. Элементы внутри него имеют формат DrawingML. Чтобы создать ссылку на конкретный стиль линии в теме, используют элемент *lnRef*, а для ссылок на заливку и эффекты — элементы *fillRef* и *effectRef* соответственно. Ссылки во всех этих элементах реализованы на основе индекса элемента в определении темы. Для элемента *fontRef* ссылка на основе индекса не нужна, поскольку можно определить лишь два шрифта: основной и дополнительный. Для задания информации о цветах и добавления ее к теме служит элемент *schemeClr*. Его можно использовать в любом месте документа, где нужно применить выделение цветом. Список значений атрибута *val* ограничен заданным набором имен цветов темы. Значения этих цветов хранятся в именах узлов в определении темы. Механизм их действия рассмотрен ранее в этом разделе.

```
<p:style>
  <a:lnRef idx="2">
    <a:schemeClr val="accent1">
      <a:shade val="50000" />
    </a:schemeClr>
  </a:lnRef>
  <a:fillRef idx="1">
    <a:schemeClr val="accent1" />
  </a:fillRef>
  <a:effectRef idx="0">
    <a:schemeClr val="accent1" />
  </a:effectRef>
  <a:fontRef idx="minor">
    <a:schemeClr val="lt1" />
  </a:fontRef>
</p:style>
```

#### Пример 160. Применение тем к фигурам

## Единицы измерения

### EMU

---

Единица измерения EMU (English Metric Unit) применяется в DrawingML для указания таких свойств, как положение и размер. EMU может быть преобразована в другие популярные единицы измерения (например, сантиметры и дюймы) без использования чисел с плавающей запятой.

Единица измерения	EMU
Дюйм	914400
Сантиметр	360000
Пункт	12700

### Twip

---

Twip — это единица измерения, не зависящая от аппаратного обеспечения устройства вывода. Благодаря ей сохраняются пропорции объектов при выводе изображений на любые устройства. 1 twip равен 1/1440 дюйма.

Единица измерения	twip
Дюйм	1440
Сантиметр	567