Раздел 4 Интерфейсы и XML

Определение

XML (eXtensible Markup Language) — расширяемый язык разметки, разработанный консорциумом W3C для представления информационных ресурсов Web нового поколения. Язык определяется стандартом консорциума. Сейчас действует версия XML 1.0 (вторая редакция), принятая в октябре 2000 года.

Аббревиатуру XML часто используют для обозначения не только самого языка, но и других связанных с ним понятий — определяющего язык стандарта W3C, информационных ресурсов XML, комплекса основанных на этом языке стандартов, называемого далее платформой XML. При отсутствии контекста это приводит к двусмысленности интерпретации обозначаемых таким «усеченным» образом понятий, а иногда и к подмене предмета обсуждения.

Исторические зарисовки

В 60-х годах общество вновь обратилось к идее структурированных документов, которыми можно обмениваться и управлять, если они разработаны в стандартном и открытом формате.

Одну из подобных попыток предприняла фирма IBM, которая разработала для своих внутренних нужд **Generalized Markup Language** (**GML**). GML предназначался для разработки различных документов – от руководств и пресс-релизов до контрактов и проектной документации. GML был спроектирован таким образом, что один и тот же исходный файл мог быть преобразован в книгу, отчет или электронную версию документа. Также GML имел простой синтаксис, включая использование тегов.

В 1986 году был опубликован стандарт языка **SGML** (Standardized Generalized Markup Language) – ISO 8879. Разработанный для определения и использования переносимых форматов документов, он был спроектирован достаточно формально для доказательства целостности документов, достаточно структурированно для работы со сложными документами и был достаточно расширяем для управления большими информационными репозитариями. И хотя обычному человеку SGML может показаться жертвой «разработки комитета», он оказался успешным в качестве независимого языка для управления и обмена текстовыми документами.

Для чего потребовался новый язык разметки?

При создании Web был разработан язык гипертекстовой разметки HTML, являющийся порождением известного метаязыка SGML. В соответствии с синтаксисом SGML был определен набор тегов разметки, совокупность которых и составила язык HTML. Простота языка, его способность служить интегратором текстовых и мультимедийных данных, легкость подготовки и публикации информации, — все это способствовало быстрому росту представленных в Web информационных ресурсов.

Вместе с тем ряд ограничений языка стал сдерживать его развитие. HTML — закрытый язык, пользователь не может добавлять в него свои теги. Расширение

функциональности языка может быть осуществлено только в соответствии с длительной процедурой обсуждения и принятия новой версии стандарта в консорциуме W3C. HTML обеспечивает только форматную разметку Web-страниц, определяющую, каким образом они должны представляться на экране монитора или печататься на принтере. Однако он не предназначен для разметки содержания Web-страниц, выделения их структурных элементов, которые можно было бы использовать, например, при поиске в Web. Наконец, одна из самых существенных слабостей HTML заключается в том, что он обладает весьма скромными средствами описания содержимого Web-страниц.

Язык XML

Создание языка XML позволило преодолеть перечисленные ограничения. Пользователи получили возможность самостоятельно определять и применять нужные совокупности тегов разметки. Языки платформы XML обеспечивают разметку не только формата, но и содержания публикуемых в Web документов, предоставляя средства описания содержимого документов и некоторых их свойств.

С синтаксической точки зрения язык XML представляет собой подмножество метаязыка SGML, допускающее сравнительно простую реализацию. XML — это набор небольшого количества операторов, главное назначение которых заключается в определении типов компонентов, составляющих содержимое размечаемых единиц информации (XML-документов), и допустимой их структуры. Такие компоненты называются элементами XML-документа. Они могут иметь свое содержимое и включать в себя параметры, называемые атрибутами элемента. Содержимое элемента может быть пустым, может быть некоторым значением или содержать экземпляры элементов других типов. Атрибуты элемента имеют скалярное значение, они могут быть обязательными или факультативными. Каждый определяемый тип элементов документов идентифицируется именем, которое явно включается в экземпляры элементов этого типа и указывает их границы в содержимом документа. Поэтому имя типа элементов служит тегом разметки. Таким образом, XML служит языком описания данных для XML-документов.

Предостережем читателя. Упоминаемое в названии языка свойство расширяемости не относится к самому языку XML! Пользователи не могут добавить в язык какие-либо синтаксические конструкции. Изменить синтаксис языка может только консорциум W3C, приняв новую версию определяющего его стандарта. Однако консорциум может пополнять функциональные возможности языка, вводя его расширения как самостоятельные стандарты, не изменяя при этом синтаксис самого языка.

Свойство расширяемости относится также к набору тегов разметки, применяемых конкретным пользователем или сообществом пользователей для разметки интересующих их документов. Этот набор может быть расширен самим пользователем путем определения новых тегов.

Платформа XML

Новые средства разметки, предложенные W3C, в отличие от языка HTML, представляют собой модульный комплекс стандартов, имеющих единую синтаксическую основу. Необходимые расширения языка создаются так. С помощью средств языка XML определяется некоторый тип или набор типов элементов XML-документов и принимается соглашение об их функциональности. Такая спецификация проходит установленную процедуру и принимается в качестве стандарта W3C.

По такому принципу функциональность XML расширена стандартами, позволяющими включать в XML-документы гиперссылки (стандарт XLink), идентифицировать фрагменты документов (XPointer), определять форматную разметку документов (XSL), определять их структуру (XML Schema) и семантику (RDF), включать в документ цифровую подпись (XML-Signature), определять запросы (XQuery) и т. д.

Какие метаданные поддерживаются в среде XML?

Стандарты платформы XML позволяют явно представлять и поддерживать метаданные, то есть данные о различных свойствах XML-документов. Благодаря этому обеспечиваются автоматическая проверка правильности структуры XML-документов и снижение уровня «шума» при поиске информационных ресурсов в Web с помощью поисковых машин.

Для описания метаданных используются различные стандарты платформы. Они позволяют описывать допустимую структуру XML-документов и структурные свойства их компонентов (XML DTD — подмножество языка XML, XML Schema, NG Relax), определять содержимое XML-документа средствами простого языка представления знаний типа «объект — свойство — значение» (Resource Definition Framework, RDF), описывать онтологию предметной области (Web Ontology Language, WOL), определяющую понятия и взаимосвязи, в терминах которых описывается содержание документов в стандарте RDF.

Семантический Web

В последние годы начал реализоваться замысел создателей Web, направленный на превращение этой системы в систему семантического уровня. В то время как Web первого поколения строился с ориентацией на обработку содержащейся в нем информации человеком, технологии Web нового поколения должны обеспечивать автоматизированную интерпретацию и обработку информации. Возникает необходимость в средствах формального описания семантики XML-данных.

Где еще применяется язык XML?

Язык XML и базирующаяся на нем технологическая платформа нашли широкое применение в различных областях ИТ главным образом благодаря поддержке языка

XML в глобальной коммуникационной среде Web и возможности использования его как языка-посредника для обмена сообщениями, обеспечивающего интероперабельность различного рода систем. Применения стандартов платформы XML охватывают также ряд технологий и стандартов как горизонтальной, так и вертикальной сферы, например, XML-ориентированные базы данных. Одной из центральных областей применения стандартов XML стал электронный бизнес.

Информационная архитектура Web нового поколения

Рис.1

В последние годы мы являемся свидетелями развивающейся во Всемирной паутине «бархатной» революции, которая связана с появлением нового стандарта языка гипертекстовой разметки XML.

По нашему мнению, происходит действительно революция, поскольку интенсивным образом формируются радикально новые Web-технологии. Революцию эту вполне можно считать «бархатной», поскольку она осуществляется осмотрительно, с явно выраженными и практически осуществляемыми намерениями сохранить все те огромные информационные ресурсы и те полезные Web-ориентированные приложения, которые интенсивно создавались в среде Web за ее недолгую историю.

Одновременно с этими внутренними переменами Web подвержен также сильному влиянию внешних процессов, в значительной мере ими же стимулированных и связанных с активно развивающейся в последние годы новой тенденцией в разработках информационных систем — с интеграцией технологий баз данных, текстовых (документальных) систем, технологий Java, Web-технологий, технологии неоднородных распределенных объектных сред CORBA на основе различных подходов. Причины этой тенденции заключаются в стремлении не только к обогащению функциональных возможностей создаваемых крупных систем, но и к обеспечению интеграции, в том числе и на смысловом (семантическом) уровне, неоднородных информационных ресурсов, созданных и поддерживаемых средствами различных технологий.

Происходящие в Web перемены затрагивают довольно широкий класс информационных систем. Не случайно поэтому, что дальнейшей судьбой Web, и прежде всего оценкой перспектив языка XML, являющегося основой новых Web-технологий, озабочено большое число специалистов, причастных к разработке и развитию информационных систем. В этой связи представляется вполне естественным состоявшееся обсуждение этих вопросов на страницах данного приложения. Однако при этом, как кажется, не были затронуты или недостаточно четко акцентировались некоторые важные моменты, на которые нужно обратить внимание читателей.

Обсуждение возможностей языка XML и перспектив его использования представляется важным, поскольку этот язык вместе с комплексом стандартов W3C, образующих его инфраструктуру, уже стал стандартом де-факто. При этом сфера его применения постоянно расширяется и включает не только непосредственно Web-технологии, но и некоторые смежные области информационных технологий, ориентирующихся на Web как на среду теледоступа к ресурсам совершенно иного рода и обмена информацией между системами, основанными на других технологиях.

Язык XML и XML-платформа

При оценке происходящих в Web перемен было бы заблуждением ограничиться рассмотрением возможностей только собственно языка XML. Наряду с созданием стандарта этого языка консорциум W3C, формирующий техническую политику развития Web и разрабатывающий стандартизованные спецификации для этой среды, на самом деле одновременно формирует новую платформу, основой которой является язык XML. Функциональность этой платформы определяется целым комплексом взаимосвязанных стандартов, часть из которых уже принята W3C, а другие находятся в стадии разработки.

За редким исключением другие стандарты платформы XML рассматриваются разработчиками как приложения XML, главным образом в связи с тем, что для их спецификаций используется синтаксис языка XML. Однако немаловажную роль играет и то обстоятельство, что определяемые этими стандартами новые возможности вводятся путем конкретизации смысла и функциональности некоторых синтаксических компонентов языка XML. Другими словами, в спецификациях языка XML и некоторых других стандартов платформы имеется ряд «открытых точек», благодаря которым обеспечивается естественный путь расширения функциональных возможностей XML в различных аспектах. Спецификации некоторых таких расширений и определяются рядом стандартов платформы XML (рис. 5).

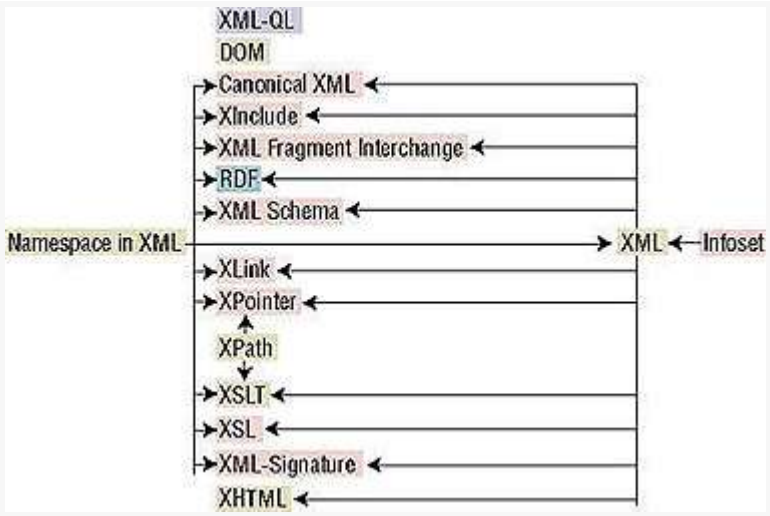

Рис. 2.

На рис.22 представлены стандарты платформы XML. Стрелки указывают на использование одних стандартов в определении других

Эти стандарты позволяют, в частности, определять множество допустимых в XML-документе тегов разметки и их атрибутов, ассоциируя с ними по умолчанию некоторую семантику (стандарт пространства имен XML — Namespaces in XML), обогащают имеющиеся в языке возможности описания с помощью DTD структуры XML-документов (стандарт спецификаций схемы — XML Schema), позволяют определять гиперсвязи между документами и/или их фрагментами (стандарты языка указателей и языка гиперссылок — XPointer и XLink), дают возможность описывать семантику XML-документов с различной степенью формализованности (стандарт среды определения ресурсов — RDF), управлять представлением XML-документов на стороне клиента (стандарты каскадных таблиц стилей CSS и расширяемого языка таблиц стилей XSL), описывать трансформации XML-документов (стандарт языка описания трансформаций XML-документов — XSLT — специальная часть стандарта XSL).

Кроме того, создан стандарт объектной модели DOM для XML- и HTML-документов, определяющий функции интерфейса прикладного программирования для их обработки.

Разрабатываются также стандарт языка запросов ресурсов XML (XML-QL), для чего сформулированы требования к базовой модели и языку и изучается ряд имеющихся претендентов, и стандарт электронной подписи для XML-документов (XML-Signature).

Особое место в рассматриваемом комплексе стандартов занимает недавно принятый W3C стандарт XHTML 1.0. Он предоставляет один из возможных путей обеспечения преемственности развития среды Web, позволяя использовать на платформе XML информационные ресурсы, накопленные в рамках технологий

HTML. Этот стандарт поддерживает средствами XML функциональность текущей версии языка HTML (HTML 4.01) с тремя различными уровнями полноты.

Рассматриваемый комплекс стандартов платформы XML включает также целый ряд вспомогательных стандартов. Приведем несколько примеров. Стандарт XML Information Set (Infoset) представляет абстрактное описание тех данных, которые составляют XML-документ. На него опирается спецификация XML [1]. Стандарт XPath определяет понятие фрагмента XML-документа, используемое в языках XPointer и XSLT. В стандарте XML Inclusions (XInclude) представлены модель и синтаксис для описания слияния XML-документов. Стандарт XML Fragment Interchange позволяет описывать контекст фрагментов XML-документа и благодаря этому просматривать и редактировать их вне полного текста документа. Упомянем также стандарт Canonical XML, в котором предлагается метод, позволяющий устанавливать эквивалентность двух XML-документов с различным синтаксическим представлением. Эта возможность существенна, в частности, для использования цифровой подписи.

Даже приведенный неполный перечень состава стандартов платформы XML и их назначение показывают, что при оценке перспектив использования языка XML неправомерно ограничиваться рассмотрением лишь собственно функциональности этого языка, и следует принимать во внимание всю совокупность стандартов, составляющих формирующуюся платформу XML.

Как XML взаимосвязан с другими стандартами платформы

Как отмечалось, в спецификации языка XML предусматривается ряд «открытых точек», которые обеспечивают взаимосвязь XML с другими стандартами базирующейся на нем платформы, а также при необходимости и со стандартами, к ней не относящимися.

В общих чертах используется следующий подход.

Главная «открытая точка» языка заключается в том, что XML — это метаязык, как и породивший его язык SGML. В отличие от языка HTML в его спецификации не фиксируется функциональная специализация элементов XML-документов, их атрибутов и семантика значений атрибутов. Именно путем конкретизации функциональности и синтаксиса элементов XML-документов можно расширять функциональность языка XML.

Второй «открытой точкой» в XML является возможность использовать так называемые пространства имен — предопределенные именованные множества имен, используемых в качестве имен типов элементов и атрибутов элементов XML-документов. Определение пространства имен позволяет также явным или неявным образом ассоциировать с именами атрибутов множества допустимых значений этих атрибутов.

Предполагается при этом, что каждому имени, принадлежащему данному пространству имен, а также значениям атрибутов соответствует некоторая семантика, определяемая по умолчанию или явным образом. Способ определения семантики при этом в стандарте Namespaces in XML никак не фиксируется. Эти определения могут базироваться на различных других стандартах или методах, требуемых для какого-либо конкретного приложения.

Стандарты платформы XML, расширяющие функциональность языка, строятся именно по этому принципу. Вводится пространство имен с зарезервированным именем, определяющее имена специальных типов элементов XML-документов и их атрибутов. Семантика этих элементов и их атрибутов и синтаксические соглашения определяются в спецификациях этих дополнительных стандартов. Принадлежащие этому пространству имена считаются общепринятыми.

Примером использования рассмотренного механизма расширения XML может служить стандарт XLink, который позволяет использовать в XML-документах специального вида ссылочные элементы, обеспечивающие различного рода гиперссылки между XML-документами. В самом языке XML концепция гиперссылки не поддерживается.

Какие данные может представлять XML

Нуждается в уточнении часто высказываемое во многих публикациях утверждение о том, что XML позволяет описывать данные самой различной природы. Это утверждение следует понимать таким образом.

XML позволяет осуществлять разметку текстовых файлов, превращая линейный текст в гипертекст. Различные файлы иной природы, называемые часто двоичными файлами, не являются объектами гипертекстовой разметки средствами XML. Однако спецификации языка XML позволяют интегрировать такие информационные ресурсы в гипертекст с помощью ссылок на содержащие их файлы, порождая тем самым гипермедийные информационные ресурсы, составляющие содержимое страниц Web. Наряду со ссылками на ресурсы, содержащиеся в двоичных файлах, XML-документ может содержать непосредственно их текстовое описание или ссылаться на другие XML-документы, которые его содержат. Эти последние, в свою очередь, могут содержать интегрированные в них двоичные файлы и т. д. Никаких иных возможностей описания и представления данных типа изображений, аудио, видео и т. д. язык XML не обеспечивает.

XML и HTML: какой язык сложнее

В обсуждении ставился вопрос о сопоставлении сложности стандартов HTML и XML. Утверждалось, что язык XML примитивнее HTML. Хотелось бы уточнить этот момент.

Если сравнивать объем описаний этих языков (документации стандартов XML и HTML), то спецификации XML занимают в несколько раз меньше места. Нужно потратить меньше труда и времени, чтобы их понять и запомнить. И в этом смысле XML проще, чем HTML.

Однако XML отнюдь не примитивнее HTML. Если сопоставлять функциональные возможности этих языков, то нужно прежде всего принять во внимание, что хотя они и имеют общие корни — известный международный стандарт обобщенного языка разметки документов SGML, но тем не менее относятся к разным уровням абстракции.

Язык XML — это метаязык, являющийся, как известно, подмножеством стандарта SGML. Как и SGML, он предназначен для порождения разнообразных конкретных языков разметки путем определения конкретных наборов тегов (в XML — типов элементов документа). Эти определяемые с помощью XML языки являются, таким образом, его конкретизациями.

Что касается языка HTML, то это — конкретный (не расширяемый) язык. Функциональность тегов разметки в нем фиксирована, в отличие от XML. HTML создавался как простейшая конкретизация SGML, представляющего собой мощный метаязык. HTML может также быть определен средствами XML (вспомним стандарт XHTML), и поэтому он представляет собой также одну из конкретизаций XML.

В силу своей абстрактности XML открыт для расширений (что отражено в его названии), и по этой причине он значительно более консервативен по сравнению с HTML, где дополнение функциональных возможностей требует прохождения процедуры принятия новой версии стандарта. Версии XML-браузеров будут появляться гораздо реже, чем для HTML, который все еще продолжает развиваться.

Многоуровневое представление XML-документов

Язык XML обеспечивает ту многоуровневость представления данных, которая является «врожденной» чертой систем баз данных. Вспомним хрестоматийные для специалистов в области баз данных понятия «физического» и «логического» представления данных или внешнюю, концептуальную, и внутреннюю схемы в трехсхемной технологии ANSI/X3/SPARC.

Более конкретно, XML поддерживает прежде всего «физический» уровень представления XML-документа — описание структуры его хранения. Строительными блоками для него служат так называемые сущности языка XML — файлы и фрагменты файлов различной природы (файлы с XML-спецификациями, например, файл DTD для какого-либо типа документов, или двоичные файлы графики, аудио- или видеоданных, повторяющиеся внутри XML-документа строки и т. д.). Структура хранения XML-документа представляет собой иерархию таких сущностей. Важно отметить, что в языке XML не предусматривается отдельного

описания физического представления XML-документа. Это представление — самоописываемое. Оно встроено в сам документ.

Далее, наряду с «сущностным» (физическим) поддерживается «логическое» представление XML-документов. Логическая структура XML-документа представляет собой иерархию составляющих его содержание структурных элементов, выделяемых тегами разметки. В то время как физическое представление XML-документов, как уже указывалось, является самоописываемым, для логического их представления предусматривается возможность отдельного явного описания. Именно для этой цели служит определение типа документов — DTD.

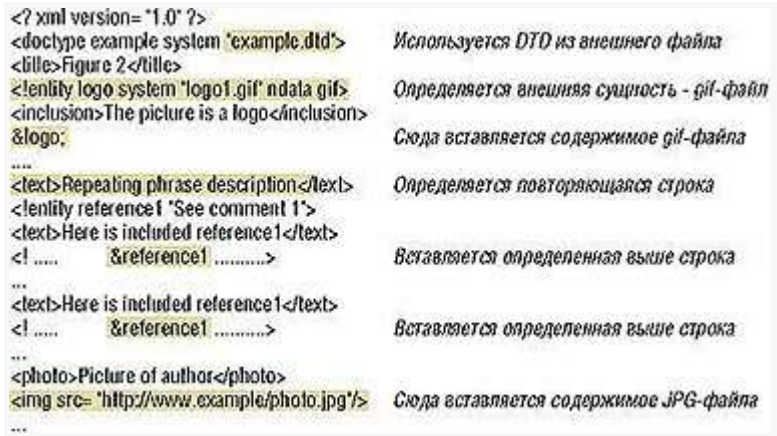

Рис. 3.

На рис.3 приведен пример XML-документа. Выделенные фрагменты синтаксиса XML относятся к описанию физического представления документа

Таким образом, хотя XML и поддерживает двухуровневое представление документов, но, в отличие от технологий баз данных в их современном воплощении, в XML описание физического (хранимого) представления не «отчуждено» от документа, а встроено в него (рис. 6). Это обстоятельство в существенной степени ограничивает в среде XML возможности поддержки независимости данных.

Верхний уровень информационной архитектуры представления XML-документов — это описание его семантики.

Поддержка слабоструктурированных данных

Главная сфера применения стандартов платформы XML — это представление слабоструктурированных данных Web-сайтов в форме XML-документов. Собственно, для этой цели и создавался язык XML. Применение XML в этой области позволяет не только представлять в среде Web гипермедийные страницы в форме XML-документов, но и поддерживать связанные с ними метаданные. Благодаря этому можно создать такие поисковые машины Web, которые будут обеспечивать в результате обработки пользовательского поискового запроса гораздо

более низкий уровень информационного шума по сравнению с нынешними HTML-технологиями.

Заметим, что недавно родившийся термин «слабоструктурированные данные» означает такие данные, которые в отличие от данных в БД не имеют регулярной структуры, определяемой с помощью предписывающей схемы. Схема для слабоструктурированных данных либо вообще не используется, и тогда они являются самоописываемыми (описание данных встроено в сами данные — в рассматриваемом случае с помощью тегов разметки), либо она задана, но не имеет предписывающего характера.

XML-сайты уже существуют в Web, как отмечалось в [2]. Доступ к их информационным ресурсам возможен, в частности, с помощью последних версий широко распространенных браузеров, например, Internet Explorer 5.0.

В последнее время активно разрабатываются инструментальные программные средства и языки для создания баз данных XML. Наряду с другими вариантами использования такие базы данных могут служить для поддержки совокупностей XML-документов, доступных через Web-сайты. В этих системах баз данных XML отводится роль языка определения данных. Операционные средства модели данных в таких системах определяются языками запросов, специально для них предназначенных. Уже существует ряд таких языков, и предпринимаются попытки создания стандарта языка запросов XML. Один из таких языков, используемый компанией Software AG в ее программном продукте Tamino, уже упоминался. Недавно совместными усилиями известных экспертов в области XML и реляционных баз данных был создан язык запросов Quilt, имеющий хорошие перспективы как претендент на роль стандарта (см. ниже). Большое достоинство Quilt заключается в том, что он удовлетворяет всем требованиям к языку запросов XML, сформулированным соответствующей Рабочей группой консорциума W3C. Эти требования служат критериями оценки претендентов на роль создаваемого стандарта языка запросов XML.

XML и другие технологии

В то время как язык XML пробивает себе дорогу к использованию в среде Web по прямому своему назначению — как выразительное средство для представления информационных ресурсов в этой среде, он вместе с тем энергично вторгается в другие технологии. Развитые выразительные возможности языка, а главное, его поддержка механизмами среды Web позволяют использовать XML в качестве языка-посредника для определения форматов обмена данными между различными системами, которые используют Internet в качестве коммуникационной среды.

Благодаря указанным возможностям XML уже применяется в целом ряде принятых или разрабатываемых стандартов. Приведем несколько примеров.

В стандарте XMI (XML Metadata Interchange) консорциума Object Management Group (OMG) на основе XML определяется формат обмена метаданными между

инструментальными средствами объектного анализа и проектирования систем, поддерживающими известный язык UML, а также между такими системами и репозитариями метаданных в распределенных неоднородных средах, которые соответствуют стандарту CORBA.

В стандарте OIM (Open Information Model) версии 1.0 консорциума Meta Data Coalition и в созданном на его основе стандарте OMG CWMI (Common Warehouse Metadata Interchange) определяется базирующийся на XML формат представления метаданных и обмена метаданными для хранилищ данных.

Разработчики стандарта ISO/IEC RDA/SQL (Remote Database Access for SQL), обеспечивающего удаленный доступ приложений к системам SQL баз данных, планируют в последующих версиях стандарта использовать XML для кодирования сообщений, которыми обмениваются клиент и сервер в коммуникационной инфраструктуре, основанной на протоколах TCP/IP.

В разрабатываемом консорциумом Workflow Management Coalition (WfMC) стандарте потоков работ определяются спецификации XML DTD, позволяющие осуществлять обмен сообщениями в среде, основанной на языке XML, между программными средствами потоков работ для поддержки их интероперабельности. Представляемые средствами XML сообщения могут содержать определения бизнес-процессов, а также данные управления процессами. Стандарт позволяет интегрировать программные средства потоков работ, соответствующие спецификациям WfMC, в среды, основанные на XML-технологиях.

Сфера экспансии XML не исчерпывается перечисленными технологиями. Однако даже приведенные примеры показывают, насколько разнообразны приложения этого языка, укрепляя оптимистическое мнение о его перспективах.

XML и интеграция информационных ресурсов

Создание языка XML вызвало к жизни ряд исследовательских проектов и разработок, посвященных решению проблем интеграции неоднородных информационных ресурсов. Начались поставки программных продуктов, обеспечивающих некоторые возможности такого рода. Одно из важных направлений всей этой деятельности связано с созданием единых однородных средств для совместного использования слабоструктурированных данных Web-сайтов и данных, представленных в реляционных БД.

Для достижения этой цели предпринимаются встречные усилия со стороны разработчиков стандарта языка SQL и разработчиков стандартов платформы XML.

Как известно, недавно, в конце 1999 года, был принят новый международный стандарт языка SQL, названный SQL:1999. Он не включает всех тех функциональных возможностей, которые предусматривались в проекте SQL3 (рабочее название этой версии стандарта на стадии ее затянувшейся разработки). Поэтому ведутся работы над новой версией стандарта, условно называемой

SQL:200n. Она должна включить недостающие функциональные компоненты проекта SQL3. Предполагается принять SQL:200n в 2003 году.

По всей вероятности, в связи с успешным продвижением платформы XML в практику летом 2000 года в планируемом составе стандарта SQL:200n появился новый ранее не предусматривающийся компонент — SQL/XML. По замыслу разработчиков, он будет определять возможности совместного использования SQL и XML. В частности, будут определяться представление схем и данных SQL в форме XML-документов, представление XML-документов и метаданных XML в терминах SQL, представление операций манипулирования данными SQL (вставка, удаление, обновление) в XML. В SQL/XML будут определены также протоколы передачи данных при совместном использовании SQL и XML. Ориентировочный срок принятия этого компонента стандарта — октябрь 2002 года. Его спецификации составят часть 14 нового стандарта SQL:200n.

Стремление к интеграции с SQL-ресурсами со стороны потребностей XML выразилось, в частности, в разработке упоминавшегося выше языка запросов для XML — Quilt. Цели его авторов заключались в создании компактного реализуемого языка, который бы удовлетворял требованиям Рабочей группы по языку запросов XML. В спецификации Quilt использованы идеи ранее предложенных для рассматриваемой цели языков — XML-QL, XQL, Xpath, YATL, XSQL, а также некоторые элементы SQL и языка объектных запросов OQL стандарта ODMG. Несмотря на такую «лоскутность» выбранного подхода, которая и определила название языка (в русском переводе — буквально «лоскутное одеяло»), авторам удалось обеспечить его концептуальное единство.

Запросы на языке Quilt основываются на структуре XML-документов. В результате их обработки порождаются XML-документы той же или иной структуры. Вместе с тем обеспечиваются возможности для запросов данных из реляционных и объектных баз данных. Язык обладает также рядом других полезных возможностей для работы с источниками данных, отличными от среды XML. Тем самым на уровне языка запросов предоставляются условия для совместного использования данных XML и SQL, объектных данных и данных из других источников.

Интересен состав авторов языка Quilt. Д. Чемберлин — сотрудник исследовательского центра IBM Almaden, один из авторов ранних версий языка SQL, на основе которых был создан международный стандарт реляционного языка запросов. Дж. Роби из компании Software AG — соавтор XQL, известного языка запросов для XML. Наконец,

Д. Флореску — сотрудница французского национального исследовательского центра INRIA, известна своими работами последних лет в области слабоструктурированных данных, а также интеграции данных, является одним из авторов языка XML-QL. Представляется, что Quilt имеет высокие шансы стать основой создаваемого W3C стандарта языка запросов XML.

Замечание о Дублинском ядре

Это замечание имеет лишь косвенное отношение к теме данной статьи. Однако нам хотелось бы воспользоваться возможностью и сделать важное дополнение к тому, что было сказано о возможностях использования набора элементов метаданных Дублинского ядра (Dublin Core, DC) для описания семантики XML-документов.

После принятия в 1999 году стандарта Дублинского ядра значительное внимание уделялось стандартизации средств, которые позволяли бы уточнять смысл его элементов и их значений в конкретных предметных областях. Для этой цели было решено использовать факультативные атрибуты (квалификаторы) для некоторых элементов DC и их значений. Был согласован также подход к стандартизации семантики и методов таких уточнений. При этом имеется в виду, что если приложение, которое использует описание содержания документов средствами DC, окажется «незнакомым» с тем или иным квалификатором, оно должно его игнорировать без каких-либо фатальных последствий для этого приложения. Вместе с тем приложение может использовать собственные «нестандартные» квалификаторы.

В июле 2000 года Dublin Core Metadata Initiative (DCMI) придал статус рекомендации предложениям рабочих групп по некоторому конкретному составу квалификаторов. Предложенные в этом документе квалификаторы делятся на две категории. К первой относятся квалификаторы, ограничивающие и уточняющие смысл элементов метаданных DC. Например, для элемента Date с помощью квалификатора можно указать одну из следующих альтернатив: время создания ресурса; время его доступности; время, в течение которого он имеет силу; время его издания; время, когда он был модифицирован. Для элемента Coverage можно уточнить, имеется ли в виду область пространства или времени. Квалификаторы второй категории уточняют интерпретацию значения элементов метаданных, указывая схему (способ) его кодирования. Например, для значения элемента Source или Relation можно указать, что оно представлено в формате URI.

Предложенный состав квалификаторов рассматривается как предварительный. Работа над ним продолжается.

Приложения XML

XML как высокоуровневый язык описаний не накладывает особых ограничений на его содержимое. Использование XML позволяет описывать обычный текст, математические формулы, структуру молекул и многое другое.
XML-процессором (XML Processor) называется программный модуль, который позволяет читать XML-документы и предоставляет доступ к их содержимому и

структуре. Подразумевается, что XML-процессор работает под управлением другого модуля, который называется **приложением (XML Application)**.
То есть, сам XML определяет строгие правила оформления (структурирования) документов, работы с ними – как данные и структура извлекаются из документа, как изменяется сам документ (через **DOM**), но нет никаких правил на то, как эти данные будут использоваться в системе. С точки зрения XML содержимое документа – это набор символов, организованных определенным образом.
Только приложение знает, как следует интерпретировать информацию, описанную в XML-документе.

Допустимые и правильные документы

Документы XML бывают хорошо «правильно составленными» (well-formed) и «действительными» (valid).
Правильно составленным документом считается такой документ, который состоит из элементов, имеет один (и только один!) корневой элемент, все его теги закрываются (пустые теги оформлены как «<…/>», непустые – «<>…</>») и удовлетворяет требованиям для правильно составленых документов (они описаны в стандарте XML).
Требования, предъявляемые к действительным документам существенно более строгие. Для того, чтобы документ считался действительным, он должен быть правильно составленным и удовлетворять всем остальным требованиям к XML-документу – обладать DTD, содержать определенные элементы документа в обязательном порядке (как это указано в DTD) и т.д.

Разметка

Документ XML состоит из текста и разметки (**markup**). Разметка включает в себя открывающие и закрывающие теги, пустые теги (empty-element tags), сущности и ссылки на них, ссылки на символы, комментарии, блоки CDATA, определения структуры документа (document type declarations) и инструкции обработки (processing instructions). Все, что не является разметкой, считается текстовым содержимым.

Комментарии

Комментарии могут появляться в любом месте документа, за исключением разметки. Это означает, что комментарии нельзя вставлять в середину разметки (например, в открывающий тег).
Пример комментария:

```
<!-- вот здесь идет объявление <body> -->
```

Инструкции обработки

Инструкции обработки не являются частью документа, а служат для управления разбором документа. Они содержат инструкции для того, кто будет обрабатывать документ – для некоего приложения (application).

ПРИМЕЧАНИЕ: Инструкции, имя которых начинается на “XML”, “xml”, “Xml” и им подобные зарезервированны для дальнейшего развития XML.

Пример инструкции обработки:

```
<?xml version=’1.0’?>
```

Теги

Начало всякого не пустого элемента XML отмечено открывающим тегом. В конце каждого элемента должен стоять закрывающий тег, который содержит «эхо» имени открывающего тега. В открывающем теге также могут быть указаны атрибуты и их значения.

Пример открывающего и закрывающего тегов:

```
<the-screen name="Messages">
  .. содержимое элемента ..
</the-screen>
```

Если элемент является пустым (не имеет содержимого), то можно использовать пустой тег:

```
<the-tag control-id='bububu'/>
```

Обратите внимание, что значение атрибута должно быть заключено в одинарные или двойные кавычки. Какие кавычки вы будете использовать – не имеет значения, а в пределах одного тега можно применять и одинарные, и двойные. Главное правило - значение атрибута ограничено одинаковыми кавычками (либо две двойные, либо две одинарные).

ВНИМАНИЕ: в языке XML большую роль играет регистр символов! Большие и маленькие буквы считаются разными символами!

Элементы

Текст между открывающим и закрывающим тегами называется содержимым елемента (element’s content). Он может включать в себя другие элементы, просто текст, ссылки, блоки CDATA, инструкции обработки и комментарии в любом порядке и количестве.

В документе есть один и только один элемент верхнего уровня. Он называется корневым (root) и содержит в себе все остальные элементы.

ВНИМАНИЕ: в XML-документах символы форматирования (пробелы, переводы строк и табуляции) не игнорируются! Они имеют свой смысл и являются частью элементов типа «текст».

Сущности (Entities)

Так же как и в HTML, в XML можно использовать сущности (entities). Они могут описывать специальные символы («<», «>» и им подобные), специальные строки (название документа, название организации).

Пример использования сущностей:

```
<!DOCTYPE page [
<!ENTITY other-file SYSTEM "other.xml">
<!ENTITY company-name "Rhonda Ltd.">
]>

<main>
   My organization called &company-name;.
   &other-file;
</main>
```

Следует отметить, что, например, использование символов “<” в теле элементов или в значениях атрибутов запрещаяется. Поэтому вы должны использовать вместо них сущности-символы (например, вместо «<» надо использовать «<»). К нашей радости, инструменты для работы с XML-документами могут сами производить преобразования. Например, если вы создадите атрибут со значением «x < y», то в файл будет записана строка «x < y». При чтении файла автоматически будет произведено обратное преобразование.

> *ПРИМЕЧАНИЕ: есть еще один вид сущностей, который называется сущности-параметры (parameter-entity).*

Обратите внимание, что объявление сущностей является частью объявления DTD для данного документа. DTD рассматривается в разделе «DTD».

Блоки CDATA

Блоки CDATA могут использоваться в любом месте документа, где можно использовать текст. Они используются для выделения блоков текста с символами, которые можно интерпретировать как разметку. Содержимое блока CDATA интерпретируется как «чистый» текст - в нем не может содержаться вложенных элементов.

Пример использования блоков CDATA:

```
<![CDATA[ вот здесь можно использовать символы разметки - < > /> // << >>
“ “ ‘’’’’’’ ]]>
```

Кодировка документа

В XML изначально была заложена поддержка различных языков и кодировок. Поэтому все хорошие библиотеки для работы с XML-документов при загрузке документа проверяют его на наличие «неправильных» символов (как, например, это делает MSXML). Если вдруг вы создадите XML-документ и будете использовать в нем русские буквы, но забудете указать кодировку, то при открытии документа

возникнет ошибка использования недопустимых символов. Для правильной работы должна быть указана используемая кодировка символов.
Примеры указания кодировок:

```
<?xml version="1.0" encoding="windows-1251"?>
<?xml version="1.0" encoding="UNICODE"?>
```

Описание структуры документа

Действительный (valid) XML-документ обязан иметь определенную структуру и эта структура должна быть описана на языке **DTD** (Document Type Definition). DTD - это язык описания структуры документа. С его помощью можно указать, что в документе будут вот такие элементы, что у них будут вот такие атрибуты, атрибуты могут принимать определенные значения (например, перечислить список значений), что элементы должны быть связанны определенным образом (элемент «глава» состоит из одного или более элементов «параграф», после которых обязательно есть один (и только один) элемент «заключение»). DTD можно описать как в начале XML-документа, так и в отдельном файле (в этом случае в документе будет ссылка на этот файл).
Язык DTD был придуман для SGML и он имеет свой, уникальный синтаксис. Вот образец файла DTD:

```
<!ELEMENT comment (#PCDATA)>

<!ELEMENT help (#PCDATA)>
<!ATTLIST help
  help-title CDATA #IMPLIED>

<!ELEMENT events (event*)>
<!ELEMENT event (#PCDATA)>
```

Как видно из примера, синтаксис DTD сам по себе не удовлетворяет требованиям XML (как минимум, нет тегов). Поэтому появился другой, альтернативный стандарт - **XML Schema**. Он также служит для описания структуры документов, является XML-совместимым и, в некотором смысле, более свободным, чем DTD. В частности, с его помощью можно описывать типы атрибутов элементов. Например, можно сказать, что атрибут «дата» содержит значения типа «дата». После этого хороший XML-инструмент будет проверять значения этого атрибута на соответствие указанному типу данных.
Вот пример описания структуры документа на языке XML Schema:

```
<?xml version="1.0"?>
<Schema name="SampleSchema"
   xmlns="urn:schemas-microsoft-com:xml-data"
   xmlns:dt="urn:schemas-microsoft-com:datatypes">

   <ElementType name="PublisherID" model="closed"
            dt:type="ID" content="textOnly">
   </ElementType>
```

```
  <ElementType name="PublisherName" model="closed"
          dt:type="string" content="textOnly">
  </ElementType>

  <ElementType name="Publisher" model="closed" content="eltOnly"
          order="many">
    <element type="PublisherID"/>
    <element type="PublisherName"/>
  </ElementType>

  <ElementType name="Book" model="closed" content="eltOnly"
          order="seq">
    <element type="Publisher" minOccurs="1" maxOccurs="1"/>
  </ElementType>

</Schema>
```

DOM и SAX

Для единства работы с документами XML был разработан специальный программный интерфес (API). Он называется DOM (Document Object Model) и является платформенно- и языко-независимым. DOM представляет XML-документ в виде множества объектов, которые реализуют определенные методы и свойства. С помощью этих объектов и методов документы XML можно читать, разбирать (parse) и изменять.

> *ПРИМЕЧАНИЕ: MSXML позволяет использовать DOM при работе с HTML-документами.*

Интерфейс SAX (Simple API for XML) предназначен для упрощенной работы с документами XML. DOM - это достаточно сложный и «тяжелый» интерфейс, потому что он позволяет работать со всем документом сразу - изменять его, читать или добавлять в него записи. В свою очередь, SAX является более быстрым и легким. Работа с XML-документом через SAX ведется посредством обработки событий. Например, в роли таких событий могут служить начало или конец элемента. Основное назначение SAX'а - это быстро и последовательно прочитать большой XML-документ.

Преобразования XML-документов (XSLT и XPATH)

Одними из языков, тесно используемых с XML, являются **XSLT** (eXtensible Stylesheet Language for Transformations) и **XPath** (язык запросов для XML).
XSLT предназначен для преобразования документов XML. С его помощью можно описать правила преобразования, которые позволят преобразовать документ в другую форму (структуру) или формат. Примером такого преобразования является представление XML-документов в виде HTML-страниц; также документы XML можно преобразовывать в RTF или любой другой текстовый формат.

При преобразовании документов XML широко используется язык XPath, который является языком запросов к содержимому XML. С его помощью можно адресовать определенные фрагменты документов XML.

Пространства имен (NAMESPACES)

В процессе развития языка XML появилась проблема одинаковых тегов в различных приложениях. Например, два книжных магазина могут придумать свои схемы документов, в которых будет участвовать элемент «книга», но содержимое этого элемента в каждом магазине будет свое. В этом случае могут возникнуть сложности, если какой-то пользователь будет работать с документами обоих магазинов.
Для того, чтобы избежать подобного пересечения имен, было принято решение разработать и внедрить некоторое средство их предотвращения. Этим средством явились пространства имен (namespaces).
Пространства имен объявляются с помощью **URI** (Universal Resource Identificator).
Пример использования пространства имен:

```
<html xmlns="http://www.w3.org/HTML/1998/html4"
   xmlns:doc="http://www.example.com"
   xmlns:y="http:/x/y/z">
 <head><title>Book Review</title></head>
 <body><doc:bookreview>
 <doc:title>A Primer</doc:title>
  <table>
  <tr align="center">
     <th>Price</th><th>Pages</th><th>Date</th></tr>
  <tr align="left">
     <td><doc:price>31.98</doc:price></td>
     <td><doc:pages>352</doc:pages></td>
     <td><doc:date>1998/01</doc:date></td></tr>
 </table>
 </doc:bookreview>
 </body></html>
```

В этом примере объявляется несколько пространств имен. Первое, которое идентифицируется строкой «*http://www.w3.org/HTML/1998/html4*», является пространством имен по-умолчанию для элемента «html» и его содержимого. Остальные два («*http://www.example.com*» и «*http:/x/y/z*») имеют префиксы «doc» и «y» соответственно.
Обратите внимание, что имена пространст имен представляют собой не URL (Universal Resource Locator), а URI – универсальный идентификатор ресурса. То есть, на самом деле это строка символов, которая теоретически должна быть уникальна. Например, в качестве названия для пространства имен можно взять значение GUID – «{BE0DC9A3-DF1C-414A-A648-14EEDB846D72}».

Описание структуры документа

Обзор DTD и XML Schema

Язык XML хорош не только своей гибкостью в описании структуры данных. При его разработке уделялось особое внимание описанию этой структуры и ее автоматической проверке.
Для этого используется **DTD** (Document Type Definition). Он является частью SGML и позволяет описывать структуру документа XML. Чтобы XML-документы являлись действительными (valid), для них должна быть определена и указана DTD.
С течением времени стали очевидны недостатки DTD. Во первых, это отличающийся синтаксис – он не является XML-совместимым. Вторым недостатком DTD является то, что он дает строгое описание структуры документа – документ должен полностью соответствовать ей (не должно быть даже малейших отклонений от описанной структуры). Третий недостаток DTD - недостаточно точное описание структуры документа. Язык **XML Schema** был разработан для устранения этих недостатков.
Синтаксис XML Schema является XML-совместимым, XML-документ может содержать дополнительные элементы, а не только те, которые описаны в его структуре. Также с помощью XML Schema можно описать типы атрибутов (например, в атрибуте указана дата или целое число).

DTD

Рассмотрим основные элементы DTD.

Введение в DTD-описание

Описание структуры документа можно расположить как в самом документе, так и вне его.
Пример расположения в документе:

```
<?xml version="1.0"?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```

Пример расположения DTD в отдельном файле:

```
XML-документ:
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "sample_1.dtd">
<greeting>Hello, world!</greeting>

DTD-документ (файл "sample_1.dtd")
<!ELEMENT greeting (#PCDATA)>
```

Хранение DTD в виде отдельного файла представляется более разумным. Ведь определение структуры может быть достаточно большим. Также отдельное хранение DTD может облегчить его изменение.

Следует уделить особое внимание тому, если для документа указана DTD, то парсер поддерживающий проверку документа на валидность (например, MSXML) при открытии этого документа может проверить его на соответствие указанной DTD. Если документ будет иметь другую структуру, то произойдет ошибка.

Основные части DTD

Описание XML-элементы

Основным элементом DTD является ELEMENT:

```
<!ELEMENT order (item)+>
```

Это выражение говорит, что элемент «order» содержит в себе один или более элементов «item».

Символ «+» называется индикатором повторяемости (occurrence indicators). Следующая табличка содержит возможные индикаторы:

Таблица 1. Индикаторы повторяемости в DTD

Индикатор	Смысл
?	содержимое встречается либо один раз, либо ни разу
*	содержимое встречается любое количество раз (или ни разу)
+	содержимое встречается более одного раза (как минимум один раз)
[ничего]	содержимое встречается ровно один раз

Эти индикаторы могут использоваться вместе со скобками для создания очень сложных выражений. Например:

```
<!ELEMENT x (a, (b | c | d)+, e)*>
```

Запятые в выражении говорят о последовательности элементов, вертикальные палочки (|) означают выбор одной из альтернатив.
Также мы можем указать следующее содержимое элемента – (#PCDATA) или (EMPTY). PCDATA говорит о том, что содержимое элемента – смешанное, что в нем в беспорядке могут быть другие элементы и/или обычный текст, EMPTY – у элемента нет содержимого.

Описание атрибутов

Для каждого элемента можно указать список атрибутов:

```
<!ELEMENT returning-airline (EMPTY)>
<!ATTLIST returning-airline  flightNum CDATA #REQUIRED
    carrierName (Alitalia | American | Delta | Northwest
       | Pacific | TWA | United) "American">
```

На примере показано, что элемент «returning-airline» является пустым элементом и что у него есть два атрибута – «flightNum» и «carrierName».

Первый атрибут (flightNum) содержит текстовые данные и должен обязательно присутствовать в документе. Второй атрибут (carrierName) может принимать только указанные значения. Причем если для какого-то узла «returning-airline» он не будет указан, то будет считаться, что его значение равно "American" (значение по-умолчанию).

Описание сущностей

В описании структуры документа можно определять сущности (entities). В процессе обработки сущности будут автоматически заменяться на свое определение. Например:

```
<!ENTITY % name-id "name CDATA #REQUIRED id ID #IMPLIED">
<!ATTLIST screen %name-id;>
```

Использование сущностей может уменьшить объем DTD, определить общие части документа (например, название или адрес компании).

Описание структуры документа

Указанные части DTD - сущности, элементы и списки атрибутов - идут друг за другом, образуя линейную последовательность. Например:

```
<!ELEMENT software-project (uic-information?)>
<!ATTLIST software-project
  name CDATA #REQUIRED
  type (pps) "pps">

<!ELEMENT uic-information (screens?)>

<!ELEMENT screens (screen*)>

<!ELEMENT screen (margins, %basic-contents;, events, controls)>
<!ATTLIST screen %name-id;>

<!ELEMENT margins EMPTY>
<!ATTLIST margins
  enabled (yes|no) "yes"
  left CDATA #REQUIRED
  top CDATA #REQUIRED
  right CDATA #REQUIRED
  bottom CDATA #REQUIRED>
```

Таким образом, мы видим, что DTD может описать структуру документа, но имеет ряд недостатков. Одним из этих недостатков является линейная последовательность описаний элементов и полная несовместимость синтаксиса с требованиями XML.

XSLT и XPATH

Как мы помним, XSLT - это язык для описания преобразования XML-документов. С его помощью мы можем изменять структуру XML-документов, преобразовывать их в HTML, RTF, обычный текст или в любую другую текстовую форму.

Введение в XSLT и XPATH

XSLT полностью удовлетворяет стандарту XML - документы XSLT являются правильно составленными (well-formed) XML-документами и обладают определенной структурой (набором тегов и атрибутов).
Преобразование документа описывается набором правил (декларативный подход).
Если для обработки XML-документа будет использоваться MSXML, то можно использовать и функции на скриптовых языках (про другие инструменты автору документа ничего неизвестно).
В преобразованиях документов очень широко используется **xPath**. Он применяется для выделения определенных частей XML-документов, для управления строками, числами и логическими значениями.

Основные элементы XPATH

Язык xPath имеет дело с элементами и атрибутами. Элементы XML-документа организованы в иерархическую структуру - почти как в файловой системе. В отличие от файловой системы, на одном уровне могут повторяться названия элементов. В xPath работа ведется с множествами элементов, которые удовлетворяют указанным условиям.
Рассмотрим следующий пример XML-документа:

```
<authors>
  <author>
    <name>Victor Hugo</name>
    <nationality>French</nationality>
  </author>
  <author period="classical">
    <name>Sophocles</name>
    <nationality>Greek</nationality>
  </author>
  <author>
    <name>Leo Tolstoy</name>
    <nationality>Russian</nationality>
  </author>
  <author>
    <name>Alexander Pushkin</name>
    <nationality>Russian</nationality>
  </author>
  <author period="classical">
    <name>Plato</name>
    <nationality>Greek</na
tionality>
  </author>
```

```
</authors>
```

Запрос узлов по именам и значениям

Следующий запрос позволяет получить множество узлов с именами авторов:

```
authors/author/name
```

Также в xPath можно использовать и шаблоны (wildcards):

```
authors/*/name
authors/author/*
```

Рассмотрим более сложный пример запроса xPath:

```
authors/author[nationality]/name
```

Этот запрос возвращает имена авторов, для которых указана национальность (возвращается множество узлов “name”).
Также мы можем запросить всех русских авторов:

```
authors/author[nationality='Russian']/name
```

Запрос узлов по атрибутам

Работа с атрибутами ведется следующим образом:

```
authors/author[@period="classical"]
authors/author/@period
```

Запрос узлов по шаблонам

Шаблоны – это выражения xPath, которые возвращают опеределенное множество узлов:

```
*|@*
node()
text()
```

Первый шаблон выделяет любые узлы и любые атрибуты.
Второй - все узлы за исключением корневого и атрибутов.
В третьем шаблоне выделяются все текстовые узлы.

Основные элементы XSLT

Чтобы парсер воспринял XML-документ как XSLT-определение, требуется указать специальный корневой тег и определить специальное пространство имен. Например:

```
<?xml version="1.0" standalone="no"?>
<xsl:stylesheet
  xmlns="http://www.w3.org/TR/REC-html40"
  xmlns:xsl="http://www.w3.org/TR/WD-xsl"
  version="1.0">

  ...
```

```
</xsl:stylesheet>
```

Корневой элемент «xsl:stylesheet» содержит в себе множество элементов «xsl:template», которые определяют правила преобразования. Например:

```
<xsl:template match="*|@*">
   <xsl:copy>
      <P><xsl:apply-templates select="@*|node()"/></P>
   </xsl:copy>
</xsl:template>

<xsl:template match='text()'>
   <P><xsl:value-of/></P>
</xsl:template>
```

В ходе обработки существует «текущий» узел обработки. Изначально, им является корневой элемент обрабатываемого XML-документа.
Парсер (например, MSXML) начинает просматривать шаблоны с самого последнего к самому первому – поиск шаблонов идет с конца XSLT. Каждый шаблон (элемент «xsl:template») содержит в себе атрибут “match” – его значение представляет собой выражение xPath. Если текущий узел (его параметры - имя, относительно местоположение, содержание) удовлетворяет этому выражению, то происходит обработка его содержимого, иначе происходит переход к следующему шаблону (который находится ближе к началу документа).
Специальный тег «xsl:apply-templates» позволяет произвести новый поиск шаблонов для указанных узлов (которые по-очереди будут текущими), причем множество обрабатываемых узлов указывается с помощью выражений xPath (см. пример выше).

Генерация узлов

В ходе обработки мы можем добавлять текст в результирующий документ. Но если, например, нам надо добавить что-нибудь вроде «<font color="red">», причем значение цвета будет известно только в момент обработки? На первый взгляд, сделать это не просто. Ведь XSLT должен удовлетворять стандарту XML, что делает невозможным следующую запись:

```
<font color="
<!-- вычисляем значение цвета -->
">
```

Поэтому в XSLT есть возможность генерации тегов:

```
<xsl:element name="font">
   <xsl:attribute name="color">
     <!-- вычисляем значение цвета -->
   </xsl:attribute>
</xsl:element>
```

Таким способом можно генерировать сложные документы с новой структурой или создавать узлы с различными значениями атрибутов.

Сортировка

Правила XSLT позволяют указывать порядок обработки узлов (сортировка узлов). Следующий пример показывает применение сортировки:

```
<xsl:template match="employees">
 <ul>
  <xsl:apply-templates select="employee">
   <xsl:sort select="name/family"/>
   <xsl:sort select="name/given"/>
  </xsl:apply-templates>
 </ul>
</xsl:template>

<xsl:template match="employee">
 <li>
  <xsl:value-of select="name/given"/>
  <xsl:text> </xsl:text>
  <xsl:value-of select="name/family"/>
 </li>
</xsl:template>
```

Элементы «ul» и «li» являются частью генерируемого HTML-документа.

Условная обработка

Другим, не менее интересным свойством XSLT является условная генерация. Например, некоторые части результата должны генерироваться только при определенных условиях.

```
<xsl:choose>
  <xsl:when test='@is-new[.="yes"]'>
  (new)
  </xsl:when>
  <xsl:otherwise>
  (old)
  </xsl:otherwise>
</xsl:choose>
```

В данном примере используются тэги «xsl:choose», «xsl:when» и «xsl:otherwise».

Также есть тег «xsl:if», но в нем нельзя определить не содержит «ложной» части – только «истинную»:

```
<xsl:if test='@was'>
  (was <xsl:value-of select="@was"/>)
</xsl:if>
```

Скрипты

Использование парсера MSXML позволяет применять скриптовые функции в XSLT-преобразованиях:

```
<xsl:script xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<![CDATA[
function GetDummyGeneralTime(node)
{
   total = parseInt (node.getAttribute("meetings"));
   total += parseInt (node.getAttribute("projects_rework"));
   total += parseInt (node.getAttribute("process_activity"));
   total += parseInt (node.getAttribute("out_of_office"));
   total += parseInt (node.getAttribute("downtime"));
   total += parseInt (node.getAttribute("general_study"));
   total += parseInt (node.getAttribute("other"));
   return total;
}
]]></xsl:script>

...
<xsl:eval>GetDummyGeneralTime(this)</xsl:eval>
...
```

Циклы

При преобразовании документов можно использовать циклы. Например, следующий код применяется ко всем подэлементам типа «timesheet»:

```
<xsl:for-each select="timesheet">
    ...
</xsl:for-each>
```

Шаблоны

В языке XSLT заложена возможность использования различных шаблонов. Для этого есть специальная конструкция «xsl:apply-templates».

В дейсвительности, использование шаблонов гораздо удобнее, чем использование циклов:

```
<xsl:template match="the-base">
...
<xsl:apply-templates select="*"/>
...
</xsl:template>

<xsl:template match="timesheet">
  ...
</xsl:template>
```

Применение шаблонов дает более лучший код - он разбит на логические части (а не представляет собой цикл длинной в несколько сотен строк с вложенными циклами), позволяет более просто указывать ограничения на обработку (через параметры «xsl:apply-templates» и «xsl:template»).

DOM

Мы рассмотрели документы XML, их строение, правила определения и языки описания структуры этих документов. Обладая полученными знаниями, мы можем очень просто создавать XML-документы в обычных текстовых редакторах, просматривать их в Microsoft IE 5.1 (например), преобразовывать в документы с другой структурой.
Создатели XML предусмотрели необходимость работы с документами XML из различных программ и разработали специальный **API** (Application Programming Interface) для этих целей. Он называется DOM – Document Object Model.
Спустя некоторое время корпорация Microsoft разработала «облегченный» API для чтения XML-документов. Он называется SAX – Simple API for XML.

Основы DOM

DOM представляет объектный взгляд на документ XML. Он представляет документы XML как совокупность объектов определенных видов с определенными связями между ними. Каждый из объектов имеет некоторый набор методов и свойств.
Использование DOM дает возможность программно создать новый документ, изменить или прочитать и проанализировать уже существующий. Также DOM предоставляет единый API для всех платформ. Например, метод «removeChild» должен называться именно таким образом во всех реализациях DOM.
Следует отметить, что существует достаточно много реализаций DOM. В ОС Windows существует библиотека **MSXML** (Microsoft XML). При разработке этой библиотеки программисты Microsoft достаточно сильно расширили стандартные наборы методов и свойств. В документации на MSXML все эти методы прокомментированы специальным образом. Хотя MSXML и расширяет стандартный DOM, но это делается согласно рекомендациям и соглашениям самого DOM.
Все дальнейшее изложение будет основываться на MSXML (в примерах используется язык VBScript).

MSXML

Корпорация Microsoft регулярно выпускает новые версии библиотеки MSXML. Обычно некоторая версия MSXML устанавливается вместе с Microsoft IE или с каким-либо другим продуктом (см. статью Q269238 в MS Knowledge Base [7]).
Но также ее можно установить и отдельно. В этом случае мы получаем возможность установки самой последней версии библиотеки (с новыми возможностями и исправленными ошибками).

Создание документов XML

Рассмотрим следующий пример создания документа XML:

```
Set xml_doc = WScript.CreateObject("Msxml2.DOMDocument")
```

```
Set xml_version = xml_doc.createProcessingInstruction _
  ("xml", "version=""1.0"" encoding=""windows-1251""")

xml_doc.appendChild (xml_version)

Set root_element = xml_doc.createElement ("the_base")
root_element.setAttribute "x", 10
xml_doc.appendChild (root_element)

xml_doc.save ("the_file.xml")
```

Документ (xml_doc)

Основным инструментом работы с документом в DOM является объект ДОКУМЕНТ – объект типа DOMDocument. Именно он содержит все остальные объекты. Также он содержит специальные методы для создания новых объектов документа.

> *ПРИМЕЧАНИЕ: новые XML-элементы создаются в контексте некоторого документа.Но! Их следует «подключить» к какому-нибудь другому элементу. В противном случае они не попадут в результирующий XML-документ.*
>
> *«Подключение» можно производить через вызов метода appendChild.*

Инструкция обработки (xml_version)

Далее мы создаем инструкцию обработки (вместе с ее некоторыми параметрами) и «подключаем» ее к объекту «xml_doc».

Корневой элемент (root_element)

Аналогично инструкции обработки мы создаем корневой элемент нашего документа и также «подключаем» его к нашему документу.

Атрибут «x»

Для корневого элемента мы указываем значение атрибута «x» равное 10.

Сохранение документа в файл

Метод «save» объекта «xml_doc» позволяет сохранить текущее содержимое документа в файл.

> *ПРИМЕЧАНИЕ: Метод «save» является Microsoft-расширением стандарта DOM.*

Результат

В результате выполнения нашего скрипта мы получим файл «the_file.xml» со следующим содержимым:

```
<?xml version="1.0" encoding="windows-1251"?>
```

```
<the_base x="10"/>
```

Чтение документа XML

В следующем примере демонстрируется чтение XML-документа с помощью MSXML:

```
'**********************************************************
Set xml_doc = WScript.CreateObject("Msxml2.DOMDocument")

'**********************************************************
xml_doc.load ("the_file.xml")

'**********************************************************
Set xml_node = xml_doc.documentElement

WScript.Echo xml_node.nodeName

'**********************************************************
for each xml_attribute in xml_node.attributes
  WScript.Echo xml_attribute.nodeName & " = " & xml_attribute.value
next
```

В этом примере мы загружаем некоторый документ XML, после чего выводим имя его корневого элемента и список атрибутов в виде «имя-атрибута = значение-атрибута».

Дополнительные данные

Использование MSXML в Visual C++

MSXML содержит библиотеку типов (type library), которая может быть использована для работы с XML-документами из программ на C++. Рассмотрим следующий пример создания простого XML-документа:

```
#import "msxml.dll" named_guids

void someFunction (void)
{
   //XX.*****************************************************
   MSXML::IXMLDOMDocument * xml_doc;
   MSXML::IXMLDOMProcessingInstructionPtr xml_version;
   MSXML::IXMLDOMElementPtr xml_node;

   VARIANT theInt;

   //XX.*****************************************************
   CoCreateInstance (MSXML::CLSID_DOMDocument, NULL
      , CLSCTX_INPROC_SERVER, MSXML::IID_IXMLDOMDocument
      , (void**) & xml_doc);
```

```
    //XX.*****************************************************
    xml_version = xml_doc->createProcessingInstruction
       (L"xml", L"version=\"1.0\""
        L"encoding=\"windows-1251\"");
    xml_doc->appendChild (xml_version);

    //XX.*****************************************************
    xml_node = xml_doc->createElement (L"the_base");
    theInt.vt = VT_I4;
    theInt.intVal = 10;
    xml_node->setAttribute (L"x", theInt);
    xml_doc->appendChild (xml_node);

    //XX.*****************************************************
    xml_doc->save ("the_file.xml");
}
```

Директива «import» позволяет достаточно просто добавить в проект какую-либо библиотеку типов (tlb), в данном случае - «msxml.dll».

Сохранение в UNICODE

Библиотека MSXML не имеет методов для сохранения XML-документов в различных кодировках. Она сама распознает нужную кодировку по содержимому инструкции обработки «xml».
В следующем примере документ будет сохранен в кодировке «windows-1251»:

```
xml_version = xml_doc->createProcessingInstruction
   (L"xml", L"version=\"1.0\""
    L"encoding=\"windows-1251\"");
...
xml_doc->save ("the_file.xml");
```

Этот пример демонстрирует сохранение в формате UNICODE:

```
xml_version = xml_doc->createProcessingInstruction
   (L"xml", L"version=\"1.0\""
    L"encoding=\"UNICODE\"");
...
xml_doc->save ("the_file.xml");
```

Промежуточный код не изменяется – все изменения заключаются только в замене «windows-1251» на «UNICODE».

Заключение

Положительные стороны XML

Использование языке XML имеет положительные стороны.

Во первых, создаваемые документы обладают четко видимой структурой.
Во вторых, полная переносимость документов - мы можем разрабатывать документ одними инструментами, а читать другими.
В третьих, можно возложить проверку правильности документа (его структуры, значений атрибутов) на инструменты. Для этого нужно воспользоваться таким средством как XML Schema.
В четвертых, документы XML можно достаточно легко отображать в произвольном виде с помощью Microsoft IE 5.X и выше - достаточно создать необходимый XSL-файл.

Завершая наше обсуждение, можно с уверенностью сказать, что платформа XML имеет благоприятные перспективы для широкого практического применения. В пользу этого свидетельствуют не только богатые функциональные возможности рассмотренного семейства стандартов, но и высокая активность в области разработки и развития стандартов, а также производства программного обеспечения, на них основанного.