Содержание

Предисловие

XSLT (Extensible Stylesheet Language Transformations) – это технология преобразования XML-документов в другие форматы. Некоторые считают, что освоить ее трудно. Однако тот факт, что в основе технологии лежат шаблоны, делают ее прекрасным кандидатом для изучения на примерах, особенно, если учесть что имеющийся образец зачастую легко приспособить для других целей. Благодаря спецификации XSLT 2.0 выразительность и элегантность XSLT заметно возросли, но язык при этом стал сложнее.

Когда я только начинал работать с XSLT (и позднее, когда приступил к изучению XSLT 2.0), мне очень не хватало сборника рецептов, из которого я мог бы черпать готовые решения стоящих передо мной задач. Первой книгой такого рода, которая попалась мне на глаза, была *Perl Cookbook*, выпущенная издательством O'Reilly. Она заставила меня сначала преодолеть нежелание изучать язык Perl, а затем и полюбить его, в чем оригинальная книга Ларри Уолла (*Programming Perl* – знаменитая «книга с верблюдом») не преуспела. Мне кажется, что такие сборники рецептов нужны, поскольку многим разработчикам недостаточно просто заставить программу работать; они хотели бы в совершенстве овладеть технологией и применять хорошо зарекомендовавшие себя приемы. При этом получить необходимую информацию желательно быстро. А нет лучшего способа глубоко изучить предмет, чем воспользоваться опытом тех, кто уже нашел удачный подход к проблеме.

Острое желание иметь подобную книгу вскоре переросло в желание самому написать ее, тем более что у меня уже подобралось несколько полезных рецептов; что-то придумали другие, а что-то – я сам. Однако я не хотел просто переписывать учебник по XSLT, по-другому сгруппировав материал. Я поставил целью создать полезный ресурс, в котором освещались бы некоторые неочевидные способы применения этой технологии. Заодно я надеялся привлечь внимание тех разработчиков, которые еще не заинтересовались XSLT и тем самым, на мой взгляд, упустили из виду один из наиболее эффективных инструментов работы с XML-документами. Если вы принадлежите к их числу, прошу вас прочитать хотя бы еще несколько абзацев, где я расскажу, чем ценен XSLT и как эта книга поможет вам осознать его потенциал.

XSLT – это язык, который находится одновременно в центре и на обочине магистрального пути развития технологий разработки программного обеспечения. Когда я работал над первым изданием этой книги, мне часто приходилось объяснять знакомым, что такое XSLT и почему так важно тратить время на написание целой книги о нем. Мои знакомые были наслышаны о Java, Perl и даже XML, но ничего не знали об XSLT. Я также видел, как растет потребность в консультациях

по XSLT, в списках рассылки и большем внимании индустрии к этой технологии, которое выражалось бы в книгах, статьях и развитых инструментах поддержки. Хотя число пользователей XSLT возрастает с каждым днем, многие профессионалы все еще не понимают, что это такое и с чем его едят. Я надеюсь, что по мере появления реализаций XSLT 2.0 эта технология распространится шире, однако уверенности в этом нет, в частности из-за конкуренции со стороны XQuery 1.0 и других способов манипулирования XML-документами. Но одно можно сказать точно: изучение XSLT 2.0 не станет потерянным впустую временем, поскольку область его применения будет расширяться, хотя, быть может, и не лавинообразно. К тому же знакомство с XSLT углубит ваши представления о том, как можно обрабатывать XML, даже если в конечном итоге вы остановитесь на другой технологии.

Хотя XSLT 1.0 – вполне зрелый язык и спецификация XSLT 2.0 была принята не так уж давно, у меня есть ощущение, что больше половины людей и компаний, работающих с XML, не используют XSLT. Сравнительно недавно один мой знакомый, который в курсе всех новейших технологий, говорил мне, что XSLT – это просто еще один язык стилизации. Такое недопонимание простительно, поскольку об XSLT судят по первым трем буквам аббревиатуры (XSL – расширенный язык таблиц стилей) и по директиве, с которой начинаются большинство XSLT-программ (xsl:stylesheet). Но самой важной является как раз последняя буква T, означающая *Transformations* (преобразования). Именно это делает язык таким ценным, и по этой причине я им и заинтересовался. При написании этой книги я, в частности, намеревался показать, что XSLT применим к решению самых разных задач. Кроме того, я хотел, чтобы эта книга стала для пользователей начального и среднего уровня тем местом, где можно найти большинство распространенных приемов работы с XSLT. И, наконец, я хотел продемонстрировать, что вообще можно делать с XSLT и тем самым побудить тех, кто уже знает этот язык, к более глубокому изучению, а остальных – присоединиться к компании «преобразователей XML».

Мне доводилось слышать самые разные мнения о том, что такое информатика. Высказывания типа «любое вычисление – это просто более или менее хитроумное манипулирование битами», «компьютеры – не более чем очень усложненные перемалыватели чисел» или «все, что может делать компьютер, возможно выразить в терминах манипулирования символами» в какой-то степени верны. Но я бы хотел предложить и свое собственное обобщение: «Любая задача, решаемая с помощью программы, может быть описана в терминах преобразований». Тот, кто овладел искусством преобразований, овладел и информатикой. Преобразованиями занимается процессор, преобразования лежат в основе алгоритмов, преобразования – это суть работы программистов. И именно преобразования составляют смысл языка XSLT, по крайней мере, когда на вход подается XML-документ (а иногда и нечто иное). Разумеется, XSLT – не единственный язык преобразований, и, как и для тысяч предшествовавших ему языков, неясно, будет ли он развиваться независимо или окажется поглощенной следующей «потрясающей инновацией». Очевидно лишь, что лежащие в основе XSLT идеи никуда не денутся,

поскольку они стары, как и сама информатика. Эта книга поможет читателю понять эти идеи и научиться применять их к конкретным задачам.

О структуре книги

Чтобы сделать книгу полезной максимально широкому кругу читателей, я сохранил большую часть примеров XSLT 1.0 из первого издания. Но добавил и решения на базе XSLT 2.0 в тех случаях, когда они оказывались проще или элегантнее. Иногда я также привожу «2.0-решения» задач, которые было бы очень трудно или даже невозможно решить в рамках версии 1.0. Примеры для версий 1.0 и 2.0 приводятся в разных подразделах. Надеюсь, что так читателям будет проще найти то, что их интересует. Во многих примерах я не даю отдельного решения для версии 2.0. Как правило, это объясняется тем, что, на мой взгляд, «1.0-решение» будет работать и в версии 2.0, а специальная адаптация под 2.0 мало что дает. Искренне надеюсь, что мое желание сохранить деревья и сэкономить время не вызовет раздражения у читателя.

Версии XSLT 1.0 и 2.0 опираются на фундамент, заложенный спецификациями XPath 1.0 и 2.0 соответственно. Некоторые читатели первого издания пеняли мне на то, что я не уделил должного внимания XPath. Глава 1 написана отчасти для того, чтобы удовлетворить их, а отчасти потому, что язык XPath 2.0 заметно расширился и усложнился.

Одно из самых простых преобразований – это обработка последовательности символов, то есть *строки*. В отличие от древнего языка SNOBOL и относительно современного Perl, XSLT не проектировался специально для манипулирования строками. Однако в главе 2 показано, что практически все, что необходимо при работе со строками, можно выполнить, оставаясь в рамках XSLT, причем новые функции, появившиеся в XSLT 2.0, значительно упрощают задачу.

Еще один вид низкоуровневых преобразований – это численные преобразования (собственно, это не что иное, как *математика*). Они пронизывают программирование снизу доверху просто потому, что измерения и расчеты – неотъемлемая часть самой жизни. В главе 3 показано, как обогатить «математические способности» XSLT, хотя этот язык и не задумывался как полноценная замена Фортрану.

Манипулирование датами и временем заложено в самой природе человека, интерес к часам, календарям и точному прогнозированию часто становился движущей силой прогресса. Глава 4 содержит рецепты работы с датами и временем, которые восполняют недостатки, присущие стандартному XSLT 1.0. Там же описываются долгожданные функции, которые наконец-то были добавлены в XSLT 2.0. Рассматривая в этой главе интригующие и трудные задачи, связанные с преобразованием дат, мы приведем как готовые решения, так и ссылки на ресурсы, относящиеся к летоисчислению.

Для любого преобразования нужно прежде всего указать объект, к которому оно применяется. Если этот объект составной, то в ходе преобразования нужно будет обойти его части. Этой теме посвящена глава 5, в которой исследуются

проблемы, для решения которых и предназначен XSLT. Здесь мы представляем XML-документ в виде дерева и показываем, как XSLT позволяет манипулировать такими деревьями. Попутно приводятся советы по поводу достижения максимальной производительности при обработке XML-документов.

Глава 6 была заново написана для второго издания. Она целиком посвящена версии XSLT 2.0. Читателям, которых больше всего интересуют новшества, появившиеся в этой версии, рекомендуется сначала прочесть главы 1 и 6, а затем для более углубленного понимания познакомиться с примерами ее применения, рассыпанными по всему тексту.

Задолго до появления различных текстовых процессоров, HTML и PDF существовал старый добрый текст. Весьма важна задача преобразования данных из формата, предназначенного для обработки компьютером, к виду, удобному человеку. Если исходные данные представлены в формате XML, то для ее решения XSLT подходит идеально. В главе 7 демонстрируется, как из XML-документа можно извлечь текст, пригодный для вывода на терминал, подачи на вход текстового редактора или для импорта в программы, требующие, чтобы значения были как-то отделены друг от друга, например, запятыми.

XML быстро становится универсальным форматом для передачи информации, и есть все признаки того, что эта тенденция будет и дальше набирать силу, а не сойдет на нет. Поэтому очень часто одни XML-документы приходится преобразовывать в другие. В главе 8 рассматриваются именно такие преобразования. Здесь показано, как XML-документы можно разбивать на части, объединять, преобразовывать из иерархического вида в плоский, очищать и выполнять иные трансформации с помощью сравнительного небольших XSLT-шаблонов.

Многие преобразования сводятся просто к извлечению информации из имеющихся данных для получения ответа на тот или иной вопрос. Глава 9 – это сокровищница рецептов, иллюстрирующих применение XSLT в качестве языка запросов. В ней приведено немало примеров, моделирующих запросы, которые вполне могут возникнуть и у вас.

Еще одна важная сфера применения XSLT – это HTML-разметка. В главе 10 показано, как решать задачи генерирования Web-контента, в том числе ссылок, таблиц, фреймов, форм и выполнять другие преобразования на стороне клиента.

Задача графического программирования – преобразование данных к визуально воспринимаемому виду. Не стоит думать, что XSLT – это язык графического программирования. Однако, если нужно выполнить преобразование в формат SVG (Scalable Vector Graphics), то XSLT поможет добиться впечатляющих результатов. В главе 11 демонстрируется, как преобразовать исходные данные в столбчатые или секторные диаграммы, линейные графики и другие графические компоненты. Здесь же рассматриваются преобразования XML в древовидное представление. В этой главе подчеркивается, что преобразования – это конструкции, которые можно комбинировать для получения самых разных результатов на выходе.

Меня всегда интересовала задача автоматической генерации кода. С этим преобразованием люди пока справляются лучше компьютеров (к счастью для тех, кто зарабатывает на жизнь программированием). Однако иногда лучше написать

программу, которая будет генерировать код, чем писать этот код самому. В главе 12 описываются плюсы представления данных, управляющих генерацией кода, в виде XML и доказывается, что XSLT идеально подходит для написания генераторов кода на языках C++, Java и самом XSLT. Сюда же включен рецепт генерации кода, заимствованный из паттерна проектирования, который представлен на языке UML, записанном в XML-нотации.

Язык XSLT применим и в более сложных ситуациях. В главе 13 приведены примеры такого рода, в том числе преобразование из формата Visio VDX в SVG, преобразование Microsoft Excel XML, построение тематических карт и обработка WSDL-документов.

Язык XSLT и сам по себе достаточно мощный, но его можно расширять и встраивать в другие языки, что позволяет делать поистине поразительные вещи. В главе 14 обсуждаются расширения XSLT на примере языков Java и JavaScript. Здесь же показано, как XSLT можно встраивать в программы на языках Java и Perl.

Тестирование и отладка – неотъемлемые составные части разработки на любом языке, и XSLT – не исключение. В главе 15 описываются некоторые полезные приемы поиска ошибок в XSLT-программах, которые дают результат даже в отсутствие настоящего отладчика XSLT.

В главе 16 еще раз подчеркивается мысль, что XSLT – гораздо больше, чем просто очередной язык стилизации. Основное внимание уделено использованию XSLT в качестве языка обобщенного и функционального программирования. Даже если вы не найдете в этой главе конкретных полезных рецептов, она откроет вам глаза на то, как можно использовать XSLT для создания обобщенных решений, и подтолкнет к размышлениям в этом направлении.

Типографские соглашения

В книге применяются следующие шрифты:
Курсив:

- пути, имена файлов и программ;
- адреса в Интернете, например, имена доменов и URL;
- новые термины при первом употреблении.

`Моноширинный`:

- команды и параметры, которые следует вводить буквально;
- имена и ключевые слова в программах, в том числе имена методов, переменных и классов;
- теги элементов XML.

`Моноширинный полужирный` – для выделения участка кода.
`Моноширинный курсив` – для переменных аргументов программы.

Этим значком обозначается примечание к окружающему тексту.

Этим значком обозначается предупреждение, относящееся к окружающему тексту.

О примерах кода

Эта книга призвана помогать вам в работе. Поэтому вы можете использовать приведенный в ней код в собственных программах и в документации. Спрашивать у нас разрешения необязательно, если только вы не собираетесь воспроизводить значительную часть кода. Например, никто не возбраняет включить в свою программу несколько фрагментов кода из книги. Однако для продажи или распространения примеров на компакт-диске разрешение требуется. Цитировать книгу и примеры в ответах на вопросы можно без ограничений. Но для включения значительных объемов кода в документацию по собственному продукту нужно получить разрешение.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются название книги, имя автора, издательство и ISBN, например: «*XSLT Cookbook*, Second Edition by Sal Mangano, Copyright 2006 O'Reilly Media, Inc., 0-596-00974-7.»

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу *permissions@oreilly.com*.

Для этой книги организована сопроводительная Web-страница, где публикуются сведения об ошибках, примеры и вообще дополнительная информация. Ее адрес в Интернете:

http://www.oreilly.com/catalog/xsltckbk

Замечания и технические вопросы можно направлять по адресу:

bookquestions@oreilly.com

Дополнительная информация о книгах, различные конференции, ресурсные центры и сеть O'Reilly Network можно найти на сайте:

http://www.oreilly.com/

Благодарности

Благодарности ко второму изданию

Писать книгу с нуля, – безусловно, тяжелая задача, особенно если это первая книга автора. Но, как оказалось, работа над вторым изданием по разным причинам была не менее сложной и интересной. Завершить ее не удалось бы без поддержки коллег, друзей и семьи. Кроме того, ни возможности, ни побудительных причин для второго издания не было бы, если бы не всесторонняя помощь, добрые слова и доброжелательная критика со стороны читателей первого издания.

Должен еще раз поблагодарить своего редактора Саймона Сен-Лорана (Simon St.Laurant) за его поистине безграничное терпение вопреки многократно срываемым мной срокам, а также за мудрые советы в большом и в малом.

Второе издание не состоялось бы без титанических усилий Майкла Кэя (Michael Kay), который не только отредактировал черновики материалов, посвященных XPath 2.0 и XSLT 2.0, но – и это особенно важно – предоставил бесплатную высококачественную реализацию того и другого в Saxon 8.

Должен также выразить признательность Эвану Ленцу (Evan Lenz) и Майку Фицджеральду (Mike Fitzgerald) за совместную работу по техническому редактированию, которая принесла великолепные плоды. Если читатель сочтет, что пояснения технически недостаточно ясны, неточны или вообще неправильны, то это лишь потому, что ваш покорный слуга упрямо проигнорировал или недопонял их предложения.

Опыт работы с XPath 2.0 и XSLT 2.0 я приобрел в основном в ходе разработки сайта SD Times (www.sdtimes.com). Хотелось бы поблагодарить Тэда Бара (Ted Bahr) и Алана Цейчика (Alan Zeichick) из компании BZ Media за предоставленную возможность переработать их сайт, а также Ребекку Паппас (Rebecca Pappas), которая терпеливо мирилась с тем, что я вынужден делить время между ней и книгой. Благодарю также своих клиентов и друзей из компании SIAC, особенно Кэрол Спивак (Carol Spiewak), Фрэнка Карреру (Frank Carrera), Берта Спильмана (Bert Spielman), Ами Ху (Amy Hui) и Диану Веркавиц (Diana Verkavits), которые нанимали меня для решения многих интересных задач, в ходе работы над которыми я отточил свое мастерство разработчика программ.

Наконец, я хочу сказать спасибо своей жене Ванде и сыновьям Ленардо и Сальваторе за то, что они пережили еще одну мою книгу. Я знаю, что совсем невесело смотреть, как папа сидит за компьютером, особенно когда на улице столько всего интересного и неожиданного. Впрочем, надеюсь, вы поймете, что упорный труд и стремление к цели вознаграждаются, если, конечно, время от времени делать небольшие перерывы, чтобы побездельничать (и пошутить насчет треснувших и разбитых предметов!).

Благодарности к первому изданию

Я всегда мечтал написать книгу и очень рад, что издательство O'Reilly помогло мне реализовать свою мечту. Однако это вовсе не было подвигом одиночки. На пути к цели мне помогали многие люди, и я хотел бы отметить их заслуги.

Прежде всего, благодарю своего редактора Саймона Сен-Лорана. Он сопровождал меня на всем пути, начиная с момента получения написанного в спешке письма с предложением о сотрудничестве и до последних этапов производственного цикла. Саймон всегда готов был поддержать меня, разделить мои радости и разочарования, без которых не обходится ни одно творччекое начинание.

Во-вторых, я выражаю признательность Джени Теннисон (Jeni Tennison), моему первому техническому редактору. Ее технические знания и внимание к деталям просто бесподобны. Джени не только исправила мои откровенные и менее очевидные ошибки, но и любезно позволила включить в книгу свой код и воспользоваться идеями, которыми она щедро делится во многих посвященных XML форумах, в которых участвует. (Если остались какие-то ошибки, то только из-за моей собственной бестолковости.) Джени – просто уникум, и я уверен, что сообщество пользователей XML присоединится к моим благодарностям за ее вклад и бескорыстную помощь.

В-третьих, я хотел бы поблагодарить всех моих коллег по работе в банке Морган Стэнли за поддержку и одобрение моей работы, а особенно своего начальника

Фарида Халили (Farid Khalili) за то, что он с пониманием относился к тем случаям, когда мне нужно было срываться с работы или оставаться дома, чтобы успеть к сроку. А также его начальника Джона Рейнольдса (John Reynolds), который рекомендовал мою книгу всему возглавляемому им отделению бумаг с фиксированным доходом. Также хочу сказать спасибо моему бывшему клиенту, корпорации SIAC, и особенно Карен Халберт (Karen Halbert), позволившей мне возглавить проект, на котором я отточил навыки работы с XSLT.

В-четвертых, спасибо всем, кто любезно предоставил мне материалы для этой книги, в том числе: Стиву Боллу (Steve Ball), Джону Брину (John Breen), Джейсону Даймонду (Jason Diamond), Никите Огивецкому и Джени Теннисон. Спасибо также пришедшим Джени на смену техническим редакторам Мике Дубинко (Micah Dubinko) и Джирке Косеку (Jirka Kosek), чьи замечания и предложения оказались весьма ценными. Ну и, конечно, всему коллективу производственного отдела издательства O'Reilly, без которого эта работа не вышла бы в свет.

И, наконец, хочу выразить признательность своим родителям, семье и друзьям. Вы, как обычно, всемерно поддерживали и подпитывали меня, помогая вести сбалансированную жизнь. Больше всего я благодарен своей жене Ванде и сыну Леонардо, без их моральной поддержки и неисчислимых жертв эта книга не состоялась бы. Спасибо тебе, Ванда, за все то, что должен был бы сделать я, а пришлось делать тебе, пока я томился как раб в темнице! Спасибо тебе, Леонардо, за то, что ты говорил: «Папа, иди работай», когда хотелось сказать: «Папа, давай поиграем». Вы вдвоем – мое самое большое достижение в жизни.

Глава 1. Язык XPath

> Нео, рано или поздно ты, как и я, поймешь, что есть разница между осознанием пути и следованием по нему.
>
> *Морфеус (Матрица)*

XPath – это язык для записи выражений. Он имеет фундаментальное значение для обработки XML-документов. Нельзя овладеть XSLT, не зная XPath, точно так же, как нельзя выучить английский язык, не зная алфавита. Некоторые читатели первого издания этой книги пеняли мне за то, что я не включил в нее основ XPath. Эта глава добавлена отчасти, чтобы удовлетворить их, но главным образом потому, что в спецификации XPath 2.0 выразительная мощь этого языка была значительно усилена. Впрочем, многие рецепты будут работать и с XPath 1.0.

В XSLT 1.0 язык XPath используется тремя способами. Во-первых, в шаблонах он служит для адресации частей преобразуемого документа. Во-вторых, он применяется для задания образцов в правилах сопоставления. В-третьих, с помощью встроенных в XPath операторов и функций выполняются простые математические операции и манипуляции со строками.

В XSLT 2.0 связь с XPath 2.0 сохранена и даже стала прочнее. В нем широко используются новые вычислительные средства, появившиеся в XPath 2.0. Можно даже сказать, что дополнительные возможности XSLT 2.0 – во многом результат нововведений в XPath 2.0, к числу которых относятся последовательности, регулярные выражения, условные и итеративные выражения, система типов, совместимая со спецификацией XML Schema, а также множество новых встроенных функций.

Каждый рецепт в этой главе – это подборка мини-рецептов для решения определенного класса задач, возникающих в XPath в контексте XSLT. Все XPath-выражения прокомментированы в соответствии с принятым в XPath 2.0 соглашением `(: комментарий :)`, но пользователям XPath 1.0 следует иметь в виду, что это недопустимая синтаксическая конструкция. Пустой результат вычисления XPath-выражения мы будем обозначать `()`, именно так в XPath 2.0 записывается пустая последовательность.

1.1. Применение осей

Задача

Требуется отобрать узлы XML-дерева с учетом сложных взаимосвязей в иерархической структуре.

Решение

Во всех приведенных ниже примерах используются оси. В каждой группе для демонстрации берется некий XML-документ, в котором контекстный узел выделен полужирным шрифтом. Поясняется, что является результатом вычисления пути, при этом показано, какие элементы отбираются относительно выделенного контекста. В некоторых случаях для иллюстрации тонкостей вычисления конкретного выражения рассматриваются и другие узлы, помимо контекстного.

Дочерняя ось и ось потомков

Дочерняя ось принимается в XPath по умолчанию. Иными словами, явно указывать ось `child::` необязательно, но, если вы хотите быть педантом, то можете и указать. Спуститься по XML-дереву глубже, чем на один уровень, позволяют оси `descendant::` и `descendant-or-self::`. Первая не включает сам контекстный узел, вторая – включает.

```
<Test id="descendants">
    <parent>
            <X id="1"/>
            <X id="2"/>
            <Y id="3">
                    <X id="3-1"/>
                    <Y id="3-2"/>
                    <X id="3-3"/>
            </Y>
            <X id="4"/>
            <Y id="5"/>
            <Z id="6"/>
            <X id="7"/>
            <X id="8"/>
            <Y id="9"/>
    </parent>
<Test>

(: Отобрать все дочерние элементы с именем X :)
X   (: то же, что child::X)

Результат: <X id="1"/> <X id="2"/> <X id="4"/> <X id="7"/> <X id="8"/>
```

```
(: Отобрать первый дочерний элемент с именем X :)
X[1]
Результат: <X id="1"/>

(: Отобрать последний дочерний элемент с именем X :)
X[last()]
Результат: <X id="8"/>

(: Отобрать первый дочерний элемент при условии, что его имя X. Иначе
пусто :)
*[1][self::X]

Результат: <X id="1"/>

(: Отобрать последний дочерний элемент при условии, что его имя X. Иначе
пусто :)
*[last()][self::X]

Результат: ()

(: Отобрать последний дочерний элемент при условии, что его имя Y. Иначе
пусто :)
*[last()][self::Y]

Результат: <Y id="9"/>

(: Отобрать всех потомков с именем X :)
descendant::X

Результат: <X id="1"/> <X id="2"/> <X id="3-1"/> <X id="3-3"/>
<X id="4"/> <X id="7"/> <X id="8"/>

(: Отобрать контекстный узел, если его имя X, а также всех потомков с
именем X :)
descendant-or-self::X

Результат: <X id="1"/> <X id="2"/> <X id="3-1"/> <X id="3-3"/>
<X id="4"/> <X id="7"/> <X id="8"/>

(: Отобрать контекстный узел и всех его потомков :)
descendant-or-self::*
Результат: <parent> <X id="1"/> <X id="2"/> <Y id="3"> <X id="3-1"/>
<Y id="3-2"/> <X id="3-3"/> </Y> <X id="4"/> <Y id="5"/> <Z id="6"/>
<X id="7"/> <X id="8"/> <Y id="9"/> </parent>
```

Оси братьев

Оси братьев называются `preceding-sibling::` и `following-sibling::`. Ось `preceding-sibling` содержит братьев, предшествующих контекстному узлу, а ось `following-sibling` – следующих за ним. Братьями, естественно, называются дети одного родителя. Почти во всех примерах ниже используется ось `preceding-sibling::`, но вам не составит труда вычислить результат и для оси `following-sibling::`.

Имейте в виду, что в позиционном выражении вида `preceding-sibling::*[1]` вы ссылаетесь на непосредственно предшествующего брата в порядке обратного отсчета от контекстного узла, а не первого брата в порядке документа. Некоторых смущает тот факт, что результирующая последовательность возвращается в порядке документа вне зависимости от того, используется ось `preceding-sibling::` или `following-sibling::`. В выражении `../X`, строго говоря, никакая ось не указана; это просто способ отобрать предшествующего и последующего брата с именем X, а также сам контекстный узел, если он называется X. Формально это сокращенная запись выражения `parent::node()/X`. Отметим, что выражения `(preceding-sibling::*)[1]` и `(following-sibling::*)[1]` отбирают первого предшествующего или последующего брата в порядке документа.

```
<!-- Тестовый документ с выделенным контекстным узлом ->
<Test id="preceding-siblings">
    <A id="1"/>
    <A id="2"/>
    <B id="3"/>
    <A id="4"/>
    <B id="5"/>
    <C id="6"/>
    <A id="7"/>
    <A id="8"/>
    <B id="9"/>
</Test>

(: Отобрать всех братьев с именем A, предшествующих контекстному узлу :)
preceding-sibling::A

Результат: <A id="1"/> <A id="2"/> <A id="4"/>

(: Отобрать всех братьев с именем A, следующих за контекстным узлом :)
following-sibling::A

Результат: <A id="8"/>
(: Отобрать всех братьев, предшествующих контекстному узлу :)
preceding-sibling::*

Результат: <A id="1"/> <A id="2"/> <B id="3"/> <A id="4"/> <B id="5"/>
```

```
<C id="6"/>

(: Отобрать первого предшествующего брата с именем A в обратном порядке
документа :)
preceding-sibling::A[1]

Результат: <A id="4"/>

(: Отобрать первого предшествующего брата в обратном порядке документа
при условии, что его имя A :)
preceding-sibling::*[1][self::A]

Результат: ()
(: Если бы контекстным был узел <A id="8"/>, то в результате мы получили
бы <A id="7"/> :)

(: Отобрать всех предшествующих братьев, кроме элементов с именем A :)
preceding-sibling::*[not(self::A)]

Результат: <B id="3"/> <B id="5"/> <C id="6"/>

(: В следующих примерах используется такой тестовый документ :)

<Test id="preceding-siblings">
    <A id="1">
           <A/>
    </A>
    <A id="2"/>
    <B id="3">
           <A/>
    </B>
    <A id="4"/>
    <B id="5"/>
    <C id="6"/>
    <A id="7"/>
    <A id="8"/>
    <B id="9"/>
</Test>
(: Элемент, непосредственно предшествующий контекстному узлу при усло-
вии, что у того есть дочерний элемент
с именем A :)
preceding-sibling::*[1][A]
Результат: ()
(: Первый из предшествующих контекстному узлу элементов,
у которого есть дочерний элемент с именем A :)
```

```
preceding-sibling::*[A][1]

Результат: <B id="3"/> ...

(: XPath 2.0 позволяет более гибко выбирать элементы с учетом про-
странств имен. В следующих примерах используется такой XML-документ :)

<Test xmlns:NS="http://www.ora.com/xsltckbk/1" xmlns:NS2= "http://
www.ora.com/xsltckbk/2">
    <NS:A id="1"/>
    <NS2:A id="2"/>
    <NS:B id="3"/>
    <NS2:B id="3"/>
</Test>
(: Отобрать всех предшествующих братьев контекстного узла, для которых
пространство имен ассоциировано с префиксом NS :)
preceding-sibling::NS:*

Результат: <NS:A id="1"/>

(: Отобрать всех предшествующих братьев контекстного узла, для которых
локальное имя равно A :)
preceding-sibling::*:A

Результат: <NS:A id="1"/> <NS2:A id="2"/>
```

Родительская ось и ось предков

Родительская ось (`parent::`) относится к родителю контекстного узла. Не путайте выражение `parent::X` с `../X`. Первое порождает последовательность, которая содержит в точности один элемент, если имя родителя контекстного узла – X, и пуста в противном случае. Второе – это сокращенная запись выражения `parent::node()/X`, которое отбирает всех братьев контекстного узла с именем X, в том числе и сам этот узел, если он называется X.

С помощью осей `ancestor::` и `ancestor-or-self::` можно перемещаться вверх по XML-дереву (переходя к родителю, деду, прадеду и т.д.). В первом случае сам контекстный узел исключается, во втором – включается.

```
(: Возвращает родителя контекстного узла, при условии, что он называется
X, в противном случае — пустую последовательность. :)
parent::X
(: Возвращает родителя контекстного узла. Результат может быть пустым,
только если контекстный узел является элементом верхнего уровня. :)
parent::*


(: Возвращает родителя, если его пространство имен ассоциировано с
префиксом X. Префикс должен быть определен, иначе произойдет ошибка. :)
```