**Стандарт XML. Призначення та структура. Відмінності від HTML.**

Що таке XML?

**XML** (eXtensible Markup Language) - це розширювана мова розмітки тексту, запропонована W3C у 1996 році. Це мова, яка повною мірою визначає логічну структуру документа. Задача XML полягає в тому, щоб дані: тексти, зображення або інші частини Web-документа могли бути визначені і структуровані незалежно від платформи , що їх відтворює, постачальника і його програмного забезпечення, наприклад Web-браузерів.

При створенні документів із використанням XML, ви можете використовувати ваші власні елементи і структури для розмітки вмісту ваших документів. Можливо визначити **DTD** (a Document Type Definition), тобто визначення типу документа. DTD визначає те, що можна назвати "граматикою" документа - це список різноманітних елементів і їхніх утворень для використання у визначених документах, у чомусь це нагадує використання **CSS**, тобто ви можете зробити посилання на DTD, що знаходиться або в мережі або написати його безпосередньо у вашому документі.

Таким чином, вміст документа, його структура, типи використвуваних у ньому елементів і його видгляд визначаться окремо, тобто незалежно один від одного.

Чому XML?

Потрібно сказати, що XML корисний для автоматизованих програмних засобів, що шукають у Web. Недосконалість HTML призвела до того, що мережа перетворилася в мішанину тексту, повну різноманітних елементів і тегів, часто використовуваних, що називається Pro Forma і нічого не значущих.

XML має величезний потенціал для удосконалення гіпертекста. Наприклад у HTML для створення зв'язку використовується елемент A, XML же дозволяє створити не просто посилання, а наприклад, двонаправлений зв'язок.

Перспектива XML полягає в тому, що він буде використовуватися для опису інших мов розмітки, наприклад, JavaScript, що використовується в HTML-документах.

XML розроблений для того, щоб спростити і полегшити використання SGML, при цьому зберігши його великі можливості по створенню, поширенню і публікації Web-документів мережі.

Вступ

Незважаючи на те, що XML дуже молода (W3C затвердила специфікацію "Extensible Markup Language(XML) 1.0" на початку лютого 1998 г) і окремі компоненти цієї мови знаходяться ще в стадії доробки, уже сьогодні з'являються нові мови, створені на основі XML, виникають численні Web-сервери, що використовують цю технологію для організації інформації , що зберігається на них.

Для чого потрібна нова мова розмітки?

Мова розмітки документів - це набір спеціальних інструкцій, називаних тегами, призначених для формування в документах якоїсь структури і визначення відношень між різноманітними елементами цієї структури. Теги мови, або, як їх іноді називають, управляючі дескриптори, у таких документах якимось чином кодуються, виділяються щодо основного вмісту документа і служать у якості інструкцій для броузера.

Всю красу XML можна зрозуміти тільки при порівнянні його з HTML. Формалізована у RFC 1866 у 1995 році, HTML є найбільш популярною мовою розмітки у всьому світі. Термін “розмітка” стосовно до документа означає звичайно усе, що не відноситься до його інформаційного наповнення.

У ранню пору свого розвитку мова HTML підносилася як засіб масштабованого форматування документів, яку можна було б використовувати для обміну інформацією практично на будь-якій платформі. У основі HTML лежить украй проста ідея: ви визначаєте нескладну мову, що описує структуру документа, і чекаєте, коли компанії розроблять програмні засоби, спроможні подавати такі документи в різноманітних операційних середовищах з урахуванням обраних користувачем параметрів. За допомогою HTML можна було б створювати матеріали, що допускають представлення в будь-якому візуальному або звуковому форматі.

Проте поступово ставало ясно, що ця ідея, незважаючи на свою простоту, йде врозріз з узвичаєною практикою видавничих систем. Традиційний механізм підготування публікацій передбачає, що графічні дизайнери і компоновщики повинні брати до уваги специфічні особливості презентаційного середовища, включаючи розмір листа, якість друку, палітру кольорів і т.п. Виявилося, що переключитися з такого методу на більш простий, при якому автор відповідає лише за вміст і логічну структуру документа, перекладаючи презентаційні обов'язки на користувацькі програми, досить важко.

У файлі HTML у його вихідному виді теги форматування перемішані зі звичайним текстом. Головною особливістю розмітки HTML є, звичайно, можливість вставки посилань на зовнішні документи або на внутрішні розділи того ж самого документа.

HTML процвітав не тільки як адаптована мова розмітки, але й у якості проміжного програмного забезпечення. Завдяки своїй дешевизні і поширеності браузери Web являють собою відмінних клієнтів; за посередництвом HTML вони можуть спілкуватися з найрізноманітнішими серверами.

Проте HTML стикнувся з певними труднощами. Його обмежені можливості форматування намагалися перебороти за допомогою CSS, ініціативи TrueDoc від Bitstream і звісно ж множини специфічних розширень для браузера; а його обмежені можливості в якості проміжного ПО - за допомогою Java, Active і т.п. Проте все це не усуває його фундаментальні недоліки.

По суті, HTML - це технологія представлення інформації, вона описує те, як браузер повинний скомпонувати текст і графік на сторінці. У результаті “те, що ви бачите, - це усе, що ви одержуєте”. Немає ніякого способу описати дані незалежно від відображення цих даних (за винятком надзвичайно слабкої системи ключових слів у заголовку сторінки Web). "Байдужність" до структури документа призводить до того, що пошук або аналіз інформації усередині нього нічим не буде відрізнятися від роботи із суцільним, не розбитим на елементи текстовим файлом. Це головна причина, чому так важко знайти потрібну інформацію за допомогою механізму пошуку.

Клієнт не має ніяких менше прийнятних засобів витягу даних із сторінки Web для подальшої роботи з ними. Далі, на будь-який конкретній сторінці Web клієнт одержує тільки одне представлення конкретної множини даних.

Припустимо, що ви переглядаєте список аукціонів eBay, упорядкований по даті відкриття торгів. Якщо ви захочете глянути на той же список, але відсортований по даті закриття торгів, то вашому браузеру прийдеться посилати новий запит серверу. У свою чергу серверу прийдеться наново відправляти повну сторінку HTML із списком аукціонів. Такого роду маніпулювання даними веде до значного збільшення числа звертань до серверів Web і утруднює, таким чином, їх подальше масштабирование.

Інша проблема з HTML у тому, що це “плоска” мова, тобто автори не можуть використовувати її для надання інформації про ієрархію даних. Далі, вона непослідовна і тому утрудняє розбір тексту програмним забезпеченням. Наприклад, хоча більшість відкриваючих тегів, (такі, як <B> або <H1>) має відповідні закриваючі теги, деякі (наприклад, <P>) їх не мають.

Істотним недоліком HTML можна назвати обмеженість набору його тегов. DTD-правила для HTML визначають фіксований набір дескрипторів і тому в розробника немає можливості вводити власні, спеціальні теги.

Простим рішенням для деяких із перерахованих проблем було би введення додаткових тегів HTML, таких, як <NAME>, <DATE> або <PRICE>. З їхньою допомогою клієнт міг би визначити, що собою являють дані, і відображати їх по-різному або експортувати по запиту користувача. Якщо ж ви вирішите не чекати зміни стандарту, то майте на увазі, що ви створюєте щось своє, нестандартне і тим самим відмовляєтеся від однієї з головних переваг HTML.

Тому в 1996 році члени робочої групи Консорціуму World Wide Web (W3C) повернулися до розгляду стандартної узагальненої мови розмітки (Standard Generalized Markup Language, SGML), сильно спрощеним нащадком якого є HTML. Запропонована у 1974 році Чарльзом Голдфарбом, SGML являє собою метамову - систему для опису інших мов. Ця мова призначена для створення інших мов розмітки, він визначає припустимий набір тегів, їхні атрибути і внутрішню структуру документа. При всіх своїх можливостях вона занадто складна для більшості браузеров Web: одна специфікацій SGML займає понад 500 сторінок.

Спростивши SGML для використання з Web, група запропонувала XML (рекомендація W3C по статусу на лютий 1998 року). XML – підмножина SGML, причому любий дійсний документ XML є дійсним документом SGML. І, як і SGML, XML - це метамова, що визначає інші мови розмітки для специфічних цілей. Наприклад, мова синхронізованої інтеграції мультимедіа (Synchronized Multimedia Integration Language, SMIL) базується на XML.

Консорціум W3C, закликаючи до використання XML у Web, фактично пропонує кожному сконструювати особисту мову для своїх гіпертекстових документів, причому для різних документів це будуть різні мови.

XML дозволяє визначити формальний синтаксис мови, наприклад правила вкладення елементів. Семантику можна, звичайно, описувати на звичайній англійській мові.

XML використовується для розмітки стандартних документів багато в чому так само, як HTML. Проте XML перевершує його при роботі зі структурованими даними, такими, як результати запиту, метаінформація про вузол Web або елементи і типи схеми.

Документ XML виглядає багато в чому схожим на HTML. Він також складається з текстових фрагментів, анотованих вкладеними в кутові дужки тегами. Проте, на відміну від HTML, зміст тега залежить від регістра, а кожний відкриваючий тег повинний в усіх випадках мати парний закриваючий тег.

XML (*Extensible Markup Language*)-э те мова розмітки, що описує цілий клас об'єктів даних, називаних XML- документами. Ця мова використовується в якості засобу для опису грамматики інших мов і контролю за правильністю впорядкування документів. XML не містить ніяких тегів, призначених для розмітки, а просто визначає порядок їх створення. Таким чином, якщо, наприклад, ми вважаємо, що для позначення елемента *rose* у документі необхідно використовувати тег *<flower>*;, то XML дозволяє вільно використовувати обумовлений нами тег і ми можемо включати в документ фрагменти, подібні такому:

<flower>rose</flower>

Таким чином, у розробників з'являється унікальна можливість визначати власні команди, що дозволяють їм найбільш ефективно визначати дані, що зберігаються в документі. Автор документа створює його структуру, будує необхідні зв'язки між елементами, використовуючи ті команди, що задовольняють його вимогам і домагається такого типу розмітки, що необхідно йому для виконання операцій перегляду, пошуку, аналізу документа.

Ще одною з очевидних переваг XML є можливість використання її в якості універсальної мови запитів до сховищ інформації. Сьогодні в глибинах W3C знаходиться на розгляді робочий варіант стандарту XML-QL (або XQL), що, можливо, у майбутньому складе серйозну конкуренцію SQL. Крім того, XML-документи можуть виступати в якості унікального засобу збереження даних, що містить у собі одночасно засоби для розбору інформації й представлення її на стороні клієнта. У цій області одним із перспективних напрямків є інтеграція Java і XML - технологій, що дозволяє використовувати міць обох технологій при побудові машинно-незалежних додатків, що використовують, крім того, універсальний формат даних при обміні інформацією.

XML дозволяє також здійснювати контроль за коректністю даних, що зберігаються в документах, робити перевірки ієрархічних співвідношень усередині документа і встановлювати єдиний стандарт на структуру документів, умістом яких можуть бути самі різноманітні дані. Це означає, що його можна використовувати при побудові складних інформаційних систем, у котрих дуже важливим є питання обміну інформацією між різноманітними додатками, що працюють в одній системі. Створюючи структуру механізму обміну інформації на самому початку роботи над проектом, менеджер може позбути себе в майбутньому від багатьох проблем, пов'язаних із несумісністю використовуваних різноманітними компонентами системи форматів даних.

На основі XML уже сьогодні створені такі відомі спеціалізовані мови розмітки, як SMIL, CDF, MathML, XSL, і список робочих проектів нових мов, що знаходяться на розгляді W3C, постійно поповнюється.

Структура документа

Не обмежуючи автора яким-небудь фіксованим набором тегів, XML дозволяє йому вводити будь-які імена. Ця можливість є ключовою для активного маніпулювання даними.

Приклад для порівняння представлення списку імен і адрес на HTML і на XML.

От фрагмент HTML:

<H1>Еditor Сontacts</H1>

<H2>Ім'я: Джонатан Эйнджел</H2>

<P>Посада: старший редактор</P>

<P>Видання: Network Magazine</P>

<P>Вулиця і будинок: Гарісона, 600 </P>

<P>Місто: Сан-Франциско</P>

<P>Штат: Каліфорнія</P>

<P>Індекс: 94107</P>

<P>Електронна пошта:

jangel@mfi. com</P>

Теги розміщають дані на екрані, але нічого не повідомляють про їхню структуру.

У випадку XML той же самий фрагмент буде поданий у такий спосіб (і збережений у файлі EDITORS. XML).

<?xml version = '1.0' ?>

<?xml-stylesheet type='text/xsl' href='editors.xsl' ?>

<editor\_contacts>

 <editor>

 <first\_name>Jonatan</first\_name>

 <last\_name>Andjel</last\_name>

 <title>chif editor</title>

 <publication>Network

 Magazine</publication>

 <address>

 <street>Garissona, 600 </street>

 <city>San-Francisko</city>

 <state>California</state>

 <zip>94107</zip>

 </address>

 <e\_mail>jangel@mfi.com</e\_mail>

 </editor>

</editor\_contacts>

У XML теги не можуть накладатися, як у HTML, проте вони можуть бути вкладені один в одний. Насправді, вкладення навіть рекомендується як засіб створення ієрархії даних (підпорядковані або рівноправні відношення). Як очевидно з приведеного приклада, такі елементи, як <first\_name> і <e\_mail>, містять дані, у той час як інші (<address>) присутні тільки з метою структурування.

Теги початку і кінця елемента є основними використовуваними в XML розмітками, але ними справа не вичерпується. Наприклад, елементам можуть бути привласнені атрибути. Ця можливість аналогічна наявній в HTML, де, наприклад, елементу <table> може бути привласнений атрибут align=”center”. У XML елемент може мати один або більше пов'язаних із ним атрибутів, причому при упорядкуванні документа ви можете видумати їх стільки, скільки побажнете, наприклад <publication topic=”networking” circulation=”controlled”>.

Документи XML можуть містити посилання на інші об'єкти. Посилання являють собою рядок, що починається з амперсанта і закінчується “;”. Ці посилання дозволяють, зокрема, вставити в документ спеціальні символи. Посилання XML на об'єкти надають набагато більше можливостей, тому що вони можуть посилатися на визначені автором розділи тексту в тому ж самому або в іншому документі.

Наприклад, посилання на об'єкти дозволяють застосувати об'єктно-орієнтований підхід при створенні журнальної статті:

<article>

&introduction;

&body;

&sidebar;

&conclusion;

&resources;

</article>

Найпростіший XML- документ може виглядати так, як це показано в Прикладі 1

<?xml version="1.0"?>

<list\_of\_items>

<item id="1"><first/>Перший</item>

<item id="2">Другий <sub\_item>підпункт 1</sub\_item></item>

<item id="3">Третій</item>

<item id="4"><last/>Останній</item>

</list\_of\_items>

У XML існують відкриваючі, закриваючі і порожні теги (у HTML поняття порожнього тэга теж існує, але спеціального його позначення не потрібно).

Тіло документа XML складається з елементів розмітки (markup) і безпосередньо вмісту документа - даних (content). XML - теги призначені для визначення елементів документа, їхніх атрибутів і інших конструкцій мови.

Любий XML-документ повинний завжди починатися з інструкції <? xml? >, усередині якої також можна задавати номер версії мови, номер кодової сторінки й інші параметри, необхідні програмі-аналізатору в процесі розбору документа.

Правила створення XML- документа

У загальному випадку XML- документи повинні задовольняти таким вимогам:

У заголовку документа поміщається оголошення XML, у якому вказується мова розмітки документа, номер її версії і додаткова інформація

Кожний відкриваючий тег, що визначає деяку область даних у документі обов'язково повинний мати відповідний закриваючий тег

У XML враховується регістр символів

Всі значення атрибутів, використовуваних у визначенні тегів, повинні бути взяті в лапки

Вкладеність тегів у XML строго контролюється, тому необхідно стежити за порядком слідування відкриваючих і закриваючих тегів

Вся інформація, що розташовується між початковим і кінцевими тегами, розглядається в XML як дані і тому враховуються всі символи форматування

Якщо XML- документ не порушує приведені правила, то він називається *формально-правильним* і всі аналізатори, призначені для розбору XML- документів, зможуть працювати з ним коректно.

З XML-документом пов'язані три рівні коректності:

*Правильно побудований* XML-документ - це такий, у якому елементи правильно структуровані у вигляді дерева з коректно розставленими відкриваючих і закриваючих тегами.

*Діючий* XML-документ правильно побудований і містить теги, що відповідають оголошенню типу документа. Він містить тільки елементи і значення атрибутів, що відповідають DTD. Хоча XML-документ може підготовлятися і читатися без DTD, DTD істотно для встановлення дієвості.

*Синтаксически коректний* XML-документ знаходиться поза контролем XML. Розробник такого документа відповідає за його логічну структуризацію.

Проте крім перевірки на формальну відповідність граматиці мови, у документі можуть бути присутнім засоби контролю над вмістом документа, за дотриманням правил, що визначають необхідні співвідношення між елементами і формуючою структурою документа. Наприклад, наступний текст, будучи цілком правильним XML- документом, буде абсолютно безглуздим:

<country><title>Russia</title><city><title>Novosibirsk</country></title></city>

Для того, щоб забезпечити перевірку коректності XML-документів, необхідно використовувати аналізатори, що роблять таку перевірку і називаються верифікованими.

На сьогоднішній день існує два способи контролю правильності XML-документа: DTD - визначення (Document Type Definition) і схеми даних (Semantic Schema). Визначення DTD- правил у XML не є необхідністю.

Конструкції мови

Вміст XML- документа являє собою набір елементів, секцій CDATA, директив аналізатора, коментарів, спецсимволів, текстових даних.

Елементи даних

Елемент - це структурна одиниця XML- документа. Вкладаючи слово rose в у тэги <flower> </flower> , ми визначаємо непустий елемент, названий <flower>, вмістом якого є *rose*. У загальному випадку в якості вмісту елементів можуть виступати як простий текст, так і інші, вкладені, елементи документа, секції CDATA, інструкції з опрацювання, коментар, - тобто практично будь-які частини XML- документа.

Любий непустой елемент повинний складатися з початкового, кінцевого тегов і даних, між ними заключених. Наприклад, наступні фрагменти будуть бути елементами:

<flower>rose</flower>

<city>Novosibirsk</city>

,а ці - ні:

<rose>

<flower>

rose

Набором всіх елементів, що містяться в документі, задається його структура і визначаються всі ієрархічні співвідношення. Плоска модель даних перетворюється з використанням елементів у складну ієрархічну систему з множиною можливих зв'язків між елементами. Наприклад, у такому прикладі ми описуємо місце розташування Новосибірських університетів (вказуємо, що Новосибірський Університет розташований у місті Новосибірську, що, у свою чергу, знаходиться в Росії), використовуючи для цього вкладеність елементів XML :

<country id="Russia">

 <cities-list>

<city>

<title>Новосибірськ</title>

<state>Siberia</state>

<universities-list>

<university id="2">

<title>Новосибірський Державний Технічний Університет</title>

<noprivate/>

<address URL="www.nstu.ru"/>

<description>дуже гарний інститут</description>

</university>

<university id="2">

<title>Новосибірський Державний Університет</title>

<noprivate/>

<address URL="www.nsu.ru"/>

<description>теж не поганої</description>

</university>

</universities-list>

</city>

</cities-list>

</country>

Проводячи пошук у цьому документі, програма клієнта буде спиратися на інформацію, закладену в його структуру - використовуючи елементи документа. Тобто, якщо, наприклад, потрібно знайти потрібний університет у потрібному місті, використовуючи приведений фрагмент документа, то необхідно буде переглянути вміст конкретного елемента <university>, що знаходиться всередині конкретного елемента <city>. Пошук при цьому, природно, буде набагато більш ефективним, ніж знаходження потрібної послідовності по всьому документу.

У XML документі, як правило, визначається хоча б один елемент, названий кореневим і з нього програми-аналізатори починають перегляд документа. У приведеному прикладі цим елементом є <country>

У деяких випадках теги можуть змінювати й уточнювати семантику тих або інших фрагментів документа, по різному визначаючи ту саму інформацію, тим самим надаючи додатку-аналізатору цього документа зведення про контекст використання описуваних даних.

У випадку, якщо елемент не має вмісту, тобто немає даних, які він повинний визначати, він називається порожнім. Необхідно тільки пам'ятати, що початковий і кінцеві теги порожнього елемента ніби об'єднується в один, і треба обов'язково ставити косу риску перед кутовою закриваючою (наприклад**, <empty/**>;)

Коментар

Коментарями є будь-яка область даних, поміщена між послідовностями символів <! -- і --> Коментар пропускаються аналізатором і тому при розборі структури документа в якості значущої інформації не розглядається.

Атрибути

Якщо при визначенні елементів необхідно задати якісь параметри, що уточнюють його характеристики, то є можливість використовувати атрибути елемента. Атрибут - це пару "назва" = "значення", що треба задавати при визначенні елемента в початковому тегу. Приклад:

<color **RGB**="true">#ff08ff</color>

<color **RGB**="false">white</color>

або

<author id=0>Ivan Petrov</author>

Прикладом використання атрибутів у HTML є опис елемента <font>:

<font color=”white” name=”Arial”>Black</font>

Cпеціальні символи

Для того, щоб включити в документ символ, використовуваний для визначення яких-небудь конструкцій мови і не викликати при цьому помилок у процесі розбору такого документа, потрібно використовувати його спеціальний символьний або числовий ідентифікатор. Наприклад, &lt; , &gt; &quot; або &#036;(десяткова форма запису), &#x1a (шестнадцатеричная) і т.д.

Директиви аналізатора

Інструкції, призначені для аналізаторів мови, описуються в XML документі за допомогою спеціальних тегів - *<?* і *? >*;. Програма клієнта використовує ці інструкції для керування процесом розбору документа. Найбільше часто інструкції використовуються при визначенні типу документа (наприклад, <? Xml version=”1.0”? >) або створенні простору імен.

CDATA

Розділи символьных даних - це частини документа, аналізовані винятково як символьные дані, що не піддаються розборові, але, у відмінності від коментарів, використовуються застосуванням, виглядають так:

<![CDATA[

Цей текст, навіть якщо він містить інструкції JavaScript або елементи коду HTML, такі, як <B>жирныйшрифт</B> або <H1>заголовок</H1>, не піддається граматичному розборові. Замість цього він відображається як їсти.

]]>

Таблиці стилів

Таблиці стилів узагалі, і каскадні таблиці стилів (Cascading Style Sheets, CSS) зокрема, дозволяють відокремити структуру й вміст документа від рівня представлення. У застосуванні до Web і HTML це означає, що мова HTML не містить у собі презентаційних можливостей: характер представлення формується окремими інструментальними засобами.

Технологія CSS помітно спрощує упорядкування і супровід документів. Створивши одну таблицю стилів, ви зможете використовувати її в сотнях документів. Вже в CSS1, першої версії CSS, були передбачені елементи уявлення, узагалі немислимі в HTML (наприклад, регулювання фізичних розмірів шрифтів).

XML/CSS як метод публікації можна зіставити з використанням програмного засобу опрацювання текстів, що підтримує стилі або макрокоманди: XML/CSS здійснює структурування документів, але виникаюча структура не має незалежну загальнодоступну семантику.

CSS можуть служити і для форматирования документів XML, але це не дуже удалий вибір. Головна перевага XML у тому, що вона подає формат документа, для можливих маніпуляцій, у виді деревоподібної структури. На жаль, CSS не спроможні взаємодіяти з деревом і можуть тільки форматувати документи XML “як вони є”. Ви можете вивести документ на екран у будь-якому форматі, але не можете здійснити якесь вибіркове представлення його даних без застосування мови сценаріїв.

Дані обмеження призвели до створення XSL. Найбільше важлива особливість XML і супутньої йому технології розширюваної мови таблиці стилів (Extensible Stylesheet Language, XSL) складається у відділенні форматирования від інформаційного наповнення.

Таблиці стилів XSL описують, як документи XML повинні перетворюватися в інші формати, такі, як HTML або RTF. Але таблиці стилів XML - це щось більше, ніж просто перетворювачі форматів; вони також надають механізм для маніпулювання даними. Наприклад, дані можна сортувати, робити по ним пошук, видаляти або додавати прямо з браузера.

XSL спроможна також здійснювати умовну трансформацію виведення в залежності від значень різноманітних елементів або атрибутів. Більш того, вона дозволяє запитувати дані з використанням множини різноманітних операторів шаблонів, символів підстановки, фільтрів, булевых операторів і виражень множини. XML і XSL ніяким чином не призначені для заміни SQL, до того ж навряд чи знайдеться багато бажаючих берегти свої бази даних безпосередньо у форматі XML. Проте XSL відчиняє можливість різноманітного пошуку по даним після їх завантаження в браузер. Вам ніколи вже не знадобиться використовувати для пошуку інформації примітивну вмонтовану команду браузера Find.

Значний потенціал XML у якості проміжного програмного забезпечення підкріплюється об'єктною моделлю документа (Document Object Model, DOM), версія 1.0 якиа була прийнята в якості рекомендації W3C у жовтні 1998 року.

Визначення Типу Документів (DTD)

Якщо теги й елементи XML використовуються винятково заради зручності на вашому власному вузлі Web, то не має ніякого значення, що ви даєте цим елементам і тегам імена, зміст яких відрізняється від стандартного і відомий тільки вам. Якщо ж, з іншого боку, ви хочете надавати дані зовнішньому світу й одержувати інформацію від партнерів по бизнесу, те ця обставина набуває величезне значення. Елементи й атрибути повинні вживатися вами точно так само, як і всіма іншими людьми, або принаймні ви повинні документувати те, що робите.

Для цього використовується визначення типів документів (Document Type Definition – DTD). Збережені на початку файла XML або назовні у виді файла \*.DTD, ці визначення описують інформаційну структуру документа. DTD перераховують можливі імена елементів, визначають наявні атрибути для кожного типу елементів і описують сполучуваність одних елементів з іншими.

Кожний рядок у визначенні типу документа може містити декларацію типу елемента, іменувати елемент і визначати тип даних, що елемент може містити. Вона має такий вигляд

<!ELEMENT ім'я\_елемента

 (тип\_даних)>

Наприклад, декларація визначає<!ELEMENT publication (#PCDATA)> елемент з ім'ям publication, що містить символьні дані (тобто текст).

Декларація <!ELEMENT special\_report (article\_1, article\_2, article\_3)> визначає елемент з ім'ям special\_report, що містить піделементи article\_1, article\_2 і article\_3 у зазначеному порядку, наприклад:

<special\_report>

<article\_1>XML:час прийшов</article\_1>

<article\_2>XML перевершує саме себе</article\_2>

<article\_3>Керування мережами і системами за допомогою XML</article\_3>

</special\_report>

Після визначення елементів DTD можуть також визначати атрибути за допомогою команди !ATTLIST. Вона вказує

елемент, іменує пов'язаний із ним атрибут і потім описує його припустимі значення.!ATTLIST дозволяє управляти атрибутами і багатьма іншими засобами: задавати значення по замовченню, знищувати пробіли і т.д. DTD можуть також містити декларації !ENTITY, де визначаються посилання на об'єкти, а також декларації !NOTATION, що вказують, що робити з двійковими файлами не у форматі XML.

Серйозне і дещо надзвичайне обмеження DTD полягає в тому, що вони не припускають типізації даних, тобто обмежують дані конкретним форматом (таким, як дата, ціле число або число з плаваючою точкою). DTD використовують інший синтаксис, ніж XML, і не дуже-то інтуїтивно зрозумілі. По названих причинах DTD будуть, напевно, замінені на більш потужні і прості у використанні схеми XML, робота над який ведеться в даний час.

Схеми даних

Схеми даних (Schemas) є альтернативним засобом створення правил побудови XML-документів. У порівнянні з DTD, схеми мають більш потужні засоби для визначення складних структур даних, забезпечують більш зрозумілий засіб опису грамматики мови, спроможні легко модернізуватися і розширюватися. Безумовною перевагою схем є також те, що вони дозволяють описувати правила для XML- документа засобами самого ж XML.

Проте це не означає, що схеми можуть цілком замінити DTD-описи - цей засіб визначення грамматики мови використовується зараз практичними всіма верифікуючими аналізаторами, XML і, більш того, самі схеми, як звичайні XML- елементи, теж описуються DTD. Але серйозні можливості нової мови і її відносної простоти, безумовно, дають підстави підтверджувати, що майбутній стандарт знайде широке застосування в якості зручного й ефективного засобу перевірки коректності упорядкування документів.

В даний час у W3 консорціумі йде робота над першою специфікацією схем даних.

Консорціум World Wide Web (W3C) не збирається давати своє благословення ніяким додаткам XML (у термінології XML “додатком” називається опис галузевих термінів за допомогою деякого набору тегов XML). Іншими словами, конкретні вертикальні ринки повинні самостійно узгодити усередині галузі імена для своїх об'єктів. Щоб сприяти відкритості і передбачуваності при упорядкуванні схем XML у вертикальних галузях, Microsoft висунула ініціативу, названу BizTalk. За станом на серпень 1999 року цю ініціативу підтримало понад 25 компанії.

Почасти BizTalk являє собою не що інше, як суспільний сервер Web, де публікуються всі схеми, запропоновані для використання в різноманітних галузях. Проте BizTalk не ставить своєю ціллю об'єднати всі галузі в спробі скласти одну гігантську схему для усіх використовуваних у якому б то ні було бізнесі даних.

BizTalk складається з трьох окремих елементів. По-перше, це сховище на сервері Web разом із рекомендаціями і тегами XML, використовуваними для додавання нових схем у сховище. По-друге, це розробка програмного продукту, серверу BizTalk. І по-третє, це будуть інтерактивні послуги на базі технології BizTalk.

Відмова від DTD

У тому, що стосується відображення галузевих даних, BizTalk виходить із безперспективності визначень типів документів (Document Type Definition, DTD). Замість того щоб заохочувати розробку XML DTD, прихильники BizTalk описують свої ієрархії даних за допомогою XML Schema (як передбачається, цей стандарт повинний прийти на зміну DTD).

В даний час W3C намагається узгодити різноманітні підходи до схем, але запропонована версія стандарту - XML Schema - дає достатньо ясне уявлення про те, як буде виглядати заміна DTD. XML Schema має значно більш широкі можливості, ніж DTD, причому описи даються за допомогою безпосередньо XML, без створення ще однієї системи розмітки, як того потребує DTD.

DTD цілком достатньо для базового визначення документа, але вони мають декілька недоліків. По-перше, вони даються не на XML. З огляду на високий ступінь адаптованості і розширюваність XML, наявність ще одного формату для визначення документів є зайвою.

По-друге, елементи DTD усередині документа XML потребують повного визначення усього, що знаходиться усередині цих елементів. Іншими словами, ніякі піделементи “на перспективу” не припускаються - якщо такі будуть присутні в документі, те, по визначенню, документ не буде бути правильно складеним. Тим часом визначення XML Schema використовують модель відкритого інформаційного наповнення, у котрої невизначені елементи цілком припустимі.

По-третє, DTD обмежуються тільки граматикою і синтаксисом (тобто відношенням одного елемента до іншого), тоді як XML Schema може також задавати безпосередні обмеження на тип даних, що елемент може містити. Це значно спрощує реалізацію передачі даних додатка в порівнянні з більш традиційним текстовим документом. Наприклад, точно так само, як це роблять розроблювачі в мовах програмування, ви можете явно зазначити, що дана область збереження може містити тільки целочисленные дані. Нарешті, розроблювачам, що працюють у середовищах Wintel, буде дуже зручно те обставина, що XML Schema легко відображається на Microsoft Document Object Model. Таким чином, що працює з документами XML програма може запросити у відповідної схеми наявне визначення для елемента документа по своєму виборі. Код виглядає в такий спосіб:

var bookNode = doc. documentElement

Проте як же буде виглядати сам документ, що містить схему, зсередини? По-перше, він буде містити теги XML, що повідомляють, що це схема, на зразок:

<Schema name=”schema\_sample\_1”>

... вміст схеми

</Schema>

Кожний пункт усередині схеми об'являється потім індивідуально, причому особливості кожного елемента розшифровуються за допомогою вкладених тегів, наприклад:

<ElementType name=”PERSONA” content=”textOnly” />

визначає елемент <Inventor> як здатний містити тільки текстові дані.

Подібні схеми можуть виявитися дуже важкі для читання, але вони легко піддаються розборові за допомогою інструментів XML. Іншими словами, вам не буде потрібно спеціальний редактор для роботи з документом XML Schema, як у випадку DTD.

У випадку правил на базі XML для форматів комерційних даних можна використовувати для відображення однієї схеми на другу вмонтовані функціональні можливості перетворення XML - розширювана мова таблиць стилів (Extensible Stylesheet Language, XSL).

На загальному рівні BizTalk Framework потребує, щоб видавці XML Schema притримувалися визначених рекомендацій. Так, тегам пропонується давати осмислені імена зі зрозумілим нескороченим написанням; ці імена повинні відповідати функціональному призначенню інформації, а не її місцю в приватній структурі даних (наприклад, “PartLocation” замість “PartFieldFourteen”), а інформація, що міститься в тегу, не повинна потребувати спеціального, відмінного від XML, декодування (наприклад, позначення валюти грошової суми повинно зберігатися у виді елемента XML, а не приєднуватися до суми як у “$30US”).

Необхідними складовими BizTalk Framework є спеціальні, загальні для всіх галузей теги XML. Ці теги покликані звільнити розроблювачів від турбот із приводу трьох найважливіших проблем взаємодії додатків. По-перше, від того, як дані передаються з одного додатка в інший; по-друге, від того, як “викликати” інший додаток - відправлення додатку даних у форматі XML повинно бути достатньо; по-третє, від того, у якому порядку повинні випливати елементи даних.

Один із тегів визначає код, за допомогою якого XML програма, що приймає дані у форматі, може встановити, що за схема BizTalk використовується. За допомогою інших тегів додаток може з'ясувати, хто є відправником даних, що відправник від нього хоче і кому дані повинні бути потім передані.

Для забезпечення сумісності документ BizTalk повинний починатися і, відповідно, закінчуватися тегом BizTalk, щоб одержувач знав, що він вступив у сектор BizTalk. Тег MsgType задає простір імен XML (вашу конкретну схему), що визначає припустимі елементи документа. Тому що ваша схема використовує формат даних XML, то тип даних, котрими ви наповняєте свій документ, буде легко встановити. Нарешті, ви можете також вставити блок маршрутних документів, наприклад:

<Route>

<From locationID=”11111111”

locationType=”DUNS”

 process=”” path=”” handle=”3”/>

<To locationID=”222222222”

locationType=”DUNS”

process=”” path=””

handle=”23CF15”/>

</Route>

BizTalk Framework нічого не говорить про те, які дані повинні входити в чотирьох атрибута тегів і<FROM><TO>, вона просто встановлює призначення кожного з них. Теги location ідентифікують мережний вузол (можливо, за допомогою URL), куди направляється документ, у той час як теги process і handle визначають додаток і конкретний примірник (наприклад, номер транзакции), до якого відносяться дані. Тег path служить свого роду вмістилищем, де проміжні сервери можуть берегти відомості про дату й іншу інформацію, щоб маршрут (і за допомогою розширення зворотний маршрут) був видимий усім серверам уздовж шляху.

Бізнес-модель BIZTALK

Microsoft випустить серверний продукт для регулювання обміну BizTalk-сумісними повідомленнями XML між партнерами по бізнесу (бета-версія наприкінці осені 1999 року; готовий продукт повинний вийти після Windows 2000).

Як це виглядає

Інструкції в схемах складають набір правил, використовуючи який, програма-клієнт буде робити висновок про те, коректний документ або ні. Схема даних, наприклад, може виглядати таким чином:

<schema id="OurSchema">

<elementType id="#title">

<string/>

</elementType>

<elementType id="photo">

<element type="#title">

<attribute name="src"/>

</elementType>

<elementType id="gallery">

<element type="#photo">

</elementType>

</schema>

Якщо ми включимо приведені правила всередину XML- документа, програма-клієнт зможе використовувати їх для перевірки. Тобто, вона тепер зможе визначити, що правильним буде бути такий фрагмент:

<gallery>

<photo id="1"><title>My computer</title></photo>

<photo id="2"><title>My family</title></photo>

<photo id="3"><title>My dog</title></photo>

</gallery>

, а некоректним цей:

<gallery>

<photo id="1"/>

<photo index="2"><title>My family</title></photo>

<photo index="3"><title> My dog </title><dogname>Sharik</dogname></photo>

</gallery>

Всі конструкції мови схем описуються правилами "XML DTD for XML-Data-Schema".

Область схеми даних

Створюючи схеми даних, ми визначаємо в документі спеціальний елемент, **<schema>**;, усередині якого містяться описи правил:

<schema id="OurSchema">

<!-- послідовність інструкцій -->

</schema>

Якщо використовувати окремий простір імен, то повний XML-документ, що містить у собі схему даних, буде виглядати в такий спосіб:

<?XML version='1.0' ?>

<?xml:namespace href="http://www.mrcpk.nstu.ru/schemas/" as="s"/?>

<s:schema id="OurSchema">

<!-- послідовність інструкцій -->

</s:schema>

Опис елементів

Для визначення класу елемента, до якого надалі будуть застосовуватися інструкції, що описують його вміст і структуру, призначений спеціальний елемент схеми **elementType**,

<elementType id="issue">

<descript>Елемент містить інформацію про черговий випуск часопису</descript>

</elementType>

Назва елемента задається атрибутом id. Всі подальші інструкції, що ставляться до описуваного класу, визначають його внутрішню структуру і набір припустимих даних, містяться всередині блока, заданого тегами <elementType> і </elementType>.

Як очевидно з приклада, при визначенні класу елемента, можна також використовувати коментар до нього, що заключають у тэги <**descript></descript>**

Атрибути елемента

Для того, щоб в описі елемента визначити його атрибути й описати властивості цих атрибутів ми повинні використовувати елемент **attribute**:

<elementType id="photo">

<attribute name="src"/>

<empty/>

</elementType>

У даному прикладі елементу *<photo>* визначається атрибут *src*, значенням якого може бути будь-яка послідовність дозволених символів:

<photo src="0"/>

<photo src="some text">

Подібно DTD, схеми даних дозволяють встановлювати обмеження на значення і засіб використання атрибутів. Для цього в дескрипторі <attribute> необхідно використовувати параметр **atttype**.

Наприклад, якщо ми хочемо зазначити, що значення атрибута повинно використовуватися програмою-аналізатором як унікальний ідентифікатор, то нам необхідно створити таке правило:

<elementType id="bouquet">

<attribute name="id" atttype="ID">

</elementType>

Якщо ж потрібно задати список можливих значень атрибута, то приклад будет виглядати в такий спосіб:

<attribute name="flower" atttype="ENUMERATION" values="red green blue" default="red">

Модель вмісту елемента

Під моделлю вмісту в схемі даних розуміють опис усіх припустимих об'єктів XML- документа, використання котрих усередині даного елемента є коректним. Модель вмісту визначається інструкціями, розташованими всередині блока <elementType>.

<elementType id="article">

<attribute name="id" atttype="ID">

<element type="#title">

<string/>

</elementType>

Для цього правила коректним буде бути такий фрагмент документа:

<article id="0">

<title>Психи і маніяки в Інтернет</title>

</article>

Вкладені елементи описуються за допомогою інструкції **element**, у якій параметром type указується клас об'єкта - посилання на його визначення:

<elementType id="article">

<element type="#title"/>

<element type="#author"/>

</elementType>

Якщо потрібно зазначити режим використання вкладеного елемента, то треба визначити параметр **occurs**:

<elementType id="article">

<element type="#title" occurs="REQUIRED"/>

<element type="#author" occurs="OPTIONAL"/>

<element type="#subject" occurs="ONEORMORE"/>

</elementType>

Можливі значення цього параметра такі:

REQUIRED - елемент повинний бути обов'язково визначений

OPTIONAL - використання елемента не є обов’язковим

ZEROORMORE - вкладений елемент може зустрічатися декілька разів або жодного разу

ONEORMORE - елемент повинний зустрічатися хоча б один раз

Приклади правильних XML-документів, що використовують приведену вище схему:

<article>

<title>Навіщо він потрібний, XML?</title>

<author>Іван Петров</author>

<subject>Що таке XML</subject>

<subject>потрібний чи він нам</subject>

</article>

або

<article>

<title>Навіщо він потрібний, XML?</title>

<subject>Що таке XML</subject>

</article>

Крім елементів, вмістом XML-документа можуть також бути звичайним текстом і областями CDATA. Для позначення типів вмісту поточного елемента в схемах використовуються такі інструкції:

**<string/>** - вказує на те, що вмістом елемента є тільки вільна текстова інформація (секція PCDATA) :

<elementType id="flower">

<string/>

</elementType>

**<any/>** - вказує на те, що вмістом елемента повинні бути тільки елементи, без тексту, незаключенного ні в один елемент:

<elementType id="issue">

<any/>

</elementType>

**<mixed>** - будь-яке сполучення елементів і тексту

<elementType id="contacts">

<mixed/>

</elementType>

**<empty>** - порожній елемент

Приклад:

<elementType id="title">

<string/>

</elementType>

<elementType id="chapter">

<string/>

</elementType>

<elementType id="chapters-list">

<any/>

</elementType>

<elementType id="content">

<element type="#chapters-list" occurs="OPTIONAL">

</elementType>

<elementType id="article">

<mixed><element type="#title"></mixed>

<element type="#content" occurs="OPTIONAL">

</elementType>

Що в імені твоєму?

Розширювана мова розмітки (Extensible Markup Language, XML) дозволяє вам створювати свої власні теги, документувати їх за допомогою визначень типів документів (Document Type Definition, DTD) або схеми XML і потім без проблем обмінюватися даними з іншими джерелами. Все це добре, але може виявитися, що інші використовують ті ж самі, що і ви, імена для елементів і атрибутів, але при цьому спираються на інші DTD. Це прямий шлях до проблем.

Щоб уникнути подібних конфліктів W3C розробив концепцію просторів імен і ключового слова xmlns. Завдяки їм в одному документі можуть використовуватися імена елементів і атрибутів, що інакше вступили б у конфлікт один з одним. Тепер же вони різняться різними префіксами простору імен і визначаються по різноманітним DTD або схемах.

От, наприклад, фрагмент коду XML із використанням просторів імен:

<inventory xmlns:storea=

“http://www.knowknew.com/

books.dtd” xmlns:storeb=

“http://www.amazon.com/schema”>

<storea:magazine>

<storea:title>Network

Magazine</storea:title>

</storea:magazine>

<storeb:magazine>

<storeb:magazine storeb:title=

“Data Communications”>

</storeb:magazine>

</inventory>

У визначенні DTD магазина А назва книги є піделементом часопису. У схемі магазина Б назва є атрибутом часопису.

Завдяки розрізненню імен за допомогою різних префіксів просторів імен вони можуть застосовуватися разом. Місцезнаходження DTD і схеми вказується в даному прикладі за допомогою URL, але воно може також визначатися за допомогою Uniform Resource Name (URN, див. RFC 2141) або Uniform Resource Identifier (URI, див. RFC 2396).

Використання для опису даних **(Intelligent Enterprise, August 03, 1999, Volume 2, Number 11)**

Однією з особливостей XML, що привертає увагу промисловості, є можливість опису структур даних і даних, що зберігаються. З використанням XML можна визначити нові теги спеціально для опису еквівалента таблиць і стовпчиків (або сутностей і атрибутів) у структурі реляційної бази даних. Ще більш істотно те, що теги для набору стовпчиків або атрибутів можуть зв'язуватися з тегами для їхньої батьківської таблиці або сутності. Хоча теговая структура здається гарним механізмом для опису і розуміння структури бази даних, спосіб організації даних потребує як ніколи раніше суворої дисципліни. XML не забороняє мати повторювані групи, жахливі структури даних і т.д.

OMG сформувала набір тегов, названий *XML Metadata Interchange* (XMI), із метою надання можливості опису в стандартних термінах структури даних про дані ("метаданих"). Цей стандарт буде корисний для обміну метаданими між CASE-засобами і для опису "репозиторія метаданих" у проектах сховищ даних. Рухаючись у тому ж напрямку, група компаній ( щовключає, зокрема, IBM і Oracle) знаходиться в процесі визначення *Common Warehouse Metadata Interchange (CWMI)*, підмножини XMI для підтримки сховищ даних.

Це означає, що є два підходи до опису структури бази даних на XML:

По-перше, прикладну базу даних може описувати DTD XML-документа. У цьому випадку операційні дані бази даних можуть бути розміщені між наборами описаних тегів. Таке DTD може, наприклад, генеруватися одним CASE-засобом, а читатися іншим, забезпечуючи засіб передачі структури даних.

По-друге, можна розмістити самі визначення таблиці і стовпчиків між тегами XMI, визначеними на більш високому рівні абстракції. Цей підхід трохи більш хитрий, оскільки метамодель XMI *дуже* абстрактна, але використання метамоделі XMI дозволяє описувати набагато більше, чим таблиці і стовпчики.

Проте зауважимо, що проблема визначення репозиторія метаданих або обміну метаданими між CASE-засобами не пов'язаний із використанням XML або якогось іншої мови. Проблемою є структура і семантика бази даних. Важливе питання полягає не в тому, *як* буде представлятися універсальний репозиторій метаданих. (Можна легко уявити репозиторий у виді набору реляционных таблиць або діаграм сутність/зв'язок.) Питання полягає в тому, *що знаходиться в репозиториї* і *що це означає*? Які об'єкти є істотними і повинні бути описані? Це набагато складіша тема, і вона усе ще знаходиться в стадії обговорення. Наявність нової мови не вносить істотний внесок у це обговорення.

Насправді при наявності розуміння, що XML є гарним засобом для опису структури бази даних, найбільше очевидним висновком є те, що використання цієї мови накладає велику відповідальність на адміністраторів даних із приводу коректності визначення даних. XML не забезпечує таку коректність; XML усього лише реєструє будь-який проект даних, що надходить від розробника.

Поява XML підвищує важливість моделювання і проектування даних.