XML (eXtensible Markup Language, розширювана мова розмітки) є програмно і апаратно-незалежний інструмент для зберігання і транспортування даних. XML є мовою розмітки так само, як HTML, а також був розроблений, щоб бути самостійно описовим. А також XML нічого не виконує, це є лише інформація записана в теги.

XML спрощує бугато речей, а саме: обмін даних, передачу даних, зміну платформи, доступність даних. Адже багато комп’ютеринх систем можуть містити несумісні системи. Багато інформації повинні бути перетвореі, і несумісні часто втрачаються. Саме тому XML зберігає дані в текстовому форматі, що забезпечує програмно та апаратно незалежний спосіб зберігання та обміну. А також він полегшує розширення або оновлення до нових операційних систем без втрати інформації.

XML не містить взначених тегів, як це є у HTML. За допомогою XML автор повинен визначити тег і структуру документу. Також він не переносить інформації про те, як він повинен бути відображеним, що дозволяє використовувати одні дані у різних сценах. Саме це надає різницю між інформацією і її відображенням. Це дозволяє полегшити відображення інформації, бо не потрібно постійно редагувати HTML файл для відображення нової інформації.

Синтаксис:

XML документи мають форму елементного дерева, кожен елемент може мати елементів-дітей, текстовий зміст та атрибути. Оскільки XML використовує самостійно описовий синтаксис, то на початку документу потрібно написати пролог, в якому вказати версію XML та, для уникнення помилок, кодування. Також обов’язковим є вказання одного елемента кореня, який буде батьком усіх інших елементів.

<?xml version="1.0" encoding="UTF-8**"**?> - версія та тип кодування  
<note> - корінь  
  <to>Tove</to>  
  <from>Jani</from>  
  <heading>Reminder</heading>  
  <body>Don't forget me this weekend!</body>  
</note>

Імена атрибутів повинні дотримуватися певних правил: повинні починатися з літери або нижнього підкреслення, не можуть починатися з слів xml(Xml, XML, …), можуть містити літери, числа і символи, але не можуть містити пробілів. Ще однією особливістю XML є те, що кожен тег повинен містити тег закриття, окрім прологу, бо він не є частиною XML документу. Також XML теги є чутливими до регістру літер(data і Data є відмінними). Також теги повинні бути правильно вкладені одни в одного, що не є обов’язковим в HTML (помилка: <b><i>a</b></i>). Ще однією відмінністю є те, що значення атрибуту повинне обов’язково бути в ламках. Якщо ви хочете у тексті тегу вказати сиволи *<, >, &, “, ‘* то потрібно використовувати відповідні замііни: &lt, &gt, &amp, &quot, &apos, хоча суворо заборонено використання тільки символів *<* та *&*. Також у XML документах можна використовувати коментарі(<!-- comment -->).

Як було вже раніше сказано, кожне значення атрибуту обов’язково повинно містити своє значення у лапках. Якщо у значенні атрибуту містяться лапки, то достатньо використати інший вид лапок (наприклад замість “ використати ‘), а також можна використати відповідний замінник (&quot). Існують деякі речі, які слід враховувати при використанні атрибутів є: атрибути не можуть містити кілька значень (елементи можуть), атрибути не можуть містити деревовидні структури (елементи можуть), атрибути нелегко розгортаються (для майбутніх змін)

XMLHttpRequest:

Давайте тепер розглянемо вокористання XML документів. Об'єкт XMLHttpRequest може бути використаний для запиту даних з веб-сервера та може виконувати: оновлення веб-сторінки без перезавантаження, запит даних з сервера - після завантаження сторінки, отримання даних з сервера - після завантаження сторінки, відправка даних на сервер - в фоновому режимі.

Наведемо найпростіший приклад використання об’єкту XMLHttpRequest:

var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
       // Typical action to be performed when the document is ready:  
       document.getElementById("demo").innerHTML = xhttp.responseText;  
    }  
};  
xhttp.open("GET", "filename", true);  
xhttp.send();

У першому рядку ми створюємо об’єкт XMLHttpRequest, потім у xhttp.onreadystatechange = function() ми вказуємо дію, яка буде виконуватись, коли статус об’єкту змінюється. У if (this.readyState == 4 && this.status == 200) ми перевіряємо чи запит є готовим.

Python XML пакет:

Давайте розглянемо дії, які ми можемо виконати над XML документом за допомогою бібліотеки Python XML. Для цього необхідні наступні класи з бібліотеки: ElementTree (відповідає за цілий документ) та Element (відповідає за окремо кожен елемент документу). Ці класи можна імпортувати з xml.etree.ElementTree.

Отримати дерево з XML документу можна за допомогою ET.parse('file.xml'), назвемо цей об’єкт tree. Після цього ми можемо получити корінь даного дерева (root – позначимо як об’єкт Element) tree.getroot().

Клас Element має наступні основні методи:

* Element.tag – повертає назву тегу даного елементу у форматі стрічки.
* Element.attrib – повертає перелік атрибутів даного елементу у форматі словника стрічок.
* Element.text – повертає текстове значення даного атрибуту.
* Element.iter('tag') – ітерація усіх дітей і дітей даного елемента за даним тегом.
* Element.findall('tag') – ітерація усіх дітей даного елемента за даним тегом.
* Element.find('tag') – повертає перший знайдений елемент за даним тегом.
* Element.findtext('text') – певертає перший елемент, з відповідним текстом.
* Element.get('attr') – повертає значення атрибуту даного елементу.
* Element.items() - повертає атрибути елемента у вигляді послідовності пар (ім'я, значення). Атрибути повертаються в довільному порядку.
* Element.keys() - для отримання елементів імен атрибутів у вигляді списку. Імена повертаються в довільному порядку.

Клас ElementTree має наступні основні методи:

* ElementTree.find() - те ж Element.find(), починаючи з кореня дерева.
* ElementTree.findall() - те ж Element.findall(), починаючи з кореня дерева.
* ElementTree.findtext() - те ж Element.findtext(), починаючи з кореня дерева.
* ElementTree.gerroot() - повертає кореневий елемент для цього дерева.
* ElementTree.iter('tag') - створює і повертає дерево ітератор для кореневого елемента. Ітератор перебирає всі елементи цього дерева, в порядку розділу тегу для пошуку (за замовчуванням, щоб повернути всі елементи).