**Можливості пакету XML**

Інтерфейси Python для обробки XML згруповані у так званий пакет XML. XML модулі не є захищеними від помилкових даних. Варто зауважити, що модулі в пакеті XML вимагають наявність хоча б одного SAX-сумісного XML (SAX – Simple API for XML).

Доступні такі підмодулі обробки XML:

* **xml.etree.ElementTree**: реалізовує простий і ефективний API для розбору і створення даних XML. Забезпечує читання з файлу, знаходження однакових елементів, внесення змін у XML файл, створення нового XML файлу. Містить такі функції та класи:
  + *xml.etree.ElementTree.Comment(text=None)* – створює і повертає екземпляр спеціального елемента, який представляє коментар.
  + *xml.etree.ElementTree.dump(elem)* – записує структуру елемента у sys.stdout.
  + *xml.etree.ElementTree.fromstring(text)* – аналізує XML частину із стрічки (text – стрічка, яка містить XML дані).
  + *xml.etree.ElementTree.fromstringlist(sequence, parser=None)* – аналізує документ XML із послідовності рядків (sequence – це список або інша послідовність XML даних).
  + *xml.etree.ElementTree.iselement(element)* – перевіряє чи заданий екземпляр елемента є дійсним екземпляром об’єкта.
  + *xml.etree.ElementTree.iterparse(source, events=None, parser=None)* – поступово аналізує XML частину у формі елементів дерева і доповідає про це кристувачеві.
  + *xml.etree.ElementTree.parse(source, parser=None)* – аналізує XML частину у формі елементів дерева.
  + *xml.etree.ElementTree.ProcessingInstruction(target, text=None)* – створює спеціальний елемент, який буде використовуватись як інструкція по обробці XML.
  + *xml.etree.ElementTree.register\_namespace(prefix, uri)* – реєструє префікс простору імен.
  + *xml.etree.ElementTree.SubElement(parent, tag, attrib={}, \*\*extra)* – створює екземпляр елемента та додає його до вже існуючого елемента.
  + *xml.etree.ElementTree.tostring(element, encoding="us-ascii", method="xml", \*, short\_empty\_elements=True)* – створює представлення XML елемента у вигляді стрічки, включаючи всі піделементи.
  + *xml.etree.ElementTree.XML(text, parser=None)* – аналізує XML частину з рядка.
  + *xml.etree.ElementTree.XMLID(text, parser=None)* – аналізує XML частину з рядка, повертає словник, що вказує елементи за їх id.
  + *xml.etree.ElementTree.Element(tag, attrib={}, \*\*extra)* – визначає інтерфейс елемента, забезпечує реалізацію цього інтерфейсу.
  + *xml.etree.ElementTree.ElementTree(element=None, file=None)* – представляє всю ієрархію елементів.
  + *xml.etree.ElementTree.QName(text\_or\_uri, tag=None)* – QName wrapper.
  + *xml.etree.ElementTree.TreeBuilder(element\_factory=None)* – перетворює послідовність методів виклику запуску, даних та кінця у добре сформовану структуру елемента.
  + *xml.etree.ElementTree.XMLParser(html=0, target=None, encoding=None)* – використовує xml.parsers.expat для ефективного розбору XML.
  + *xml.etree.ElementTree.XMLPullParser(events=None)* – підходить для незаблокованих додатків. Вхідна сторона API схожа до XMLParser, але замість направлення викликів до зворотньої мети, тут зберігається внутрішній список аналізу та дозволяється користувачу його читати.
  + *xml.etree.ElementTree.ParseError* – викликається при виникненні помилок виконання.
* **xml.dom**: The Document Object Model (“DOM”) – це API від World Wide Web Consortium (W3C) для доступу і модифікації XML-елементів. Реалізація DOM представлена у вигляді XML документа (структура у вигляді дерева). Містить наступні функції та модулі:
  + *xml.dom.registerDOMImplementation(name, factory)* – реєструє функцію з назвою name, яка повертатиме об’єкт, що реалізовує інтерфейс DOMImplementation.
  + *xml.dom.getDOMImplementation(name=None, features=())* – повертає відповідну реалізоцію DOM.
  + *DOMImplementation.hasFeature(feature, version)* – повертає True, якщо feature ідентифікується парою рядків і версія (version) реалізована.
  + *DOMImplementation.createDocument(namespaceUri, qualifiedName, doctype)* – повертає новий об’єкт Document із дочірнім об’єктом Element, що має задані namespaceUri та qualifiedName.
  + *DOMImplementation.createDocumentType(qualifiedName, publicId, systemId) –* повертає новий об’єкт DocumentType, який інкапсулює дані qualifiedName, publicId і systemId, що представляють інформармацію, яка міститься в оголошенні типу XML документа.

Також містить:

* + *Document об’єкти*: Document представляє весь документ XML, включаючи його складові елементи, атрибути, інструкції обробки та коментарі.
  + *DocumentType об’єкти*: інформація про нотації та юридичних осіб, оголошених в документі.
  + *Node об’єкти*: всі компоненти XML документу є підкласами Node.
  + *Element об’єкти*: є підкласом Node, тож успадковує всі атрибути цього класу.
  + *Attr об’єкти*: успадковуються від Node, тож відповідно успадковує всі його атрибути.
  + *Comment об’єкти*: представляють коментар у XML документі, є підкласом Node.
  + *Text and CDATASection об’єкти*: представляють текст у XML документі.
  + *ProcessingInstruction об’єкти*: представляють інструкцію обробки в XML документі, успадковуються від Node.
  + *Exceptions*: вийнятки.
* **xml.dom.minidom**: це мінімальна реалізація інтерфейсу об’єктної моделі документа. Він призначений, щоб бути простішим, ніж повний DOM:
  + *xml.dom.minidom.parse(filename\_or\_file, parser=None, bufsize=None)* – повертає документ, який формується із введеної інформації.
  + *Node.unlink()* – перериває внутрішні посилання в DOM так, що в них буде збирання сміття у тих версіях Python, в яких GC (garbage collection) немає.
  + *Node.writexml(writer, indent="", addindent="", newl="")* – записує XML в записуючий об’єкт.
  + *Node.toxml(encoding=None)* – повертає рядок або байт рядок, що містить XML, який представлений вузлом DOM.
  + *Node.toprettyxml(indent="", newl="", encoding="")* – повертає друковану версію документа.
* **xml.dom.pulldom**: забезпечує “аналізатор”, до якого може подаватися запит на створення DOM-доступних фрагментів документа. Основна концепція включає в себе витягання “подій” з потоку вхідного XML і їх обробку. Користувач є відповідальним за явне витягання подій з потоку і очікування поки закінчиться процес обробки або виникне помилка. Дозволяє виконувати такі дії:
  + *xml.dom.pulldom.parse(stream\_or\_string, parser=None, bufsize=None)* – повертає DOMEventStream від введеної інформації; stream\_or\_string – це або назва файлу, або файл-подібний об’єкт. Якщо задано критерій аналізу (parser), то він має бути об’єктом XMLReader.
  + *xml.dom.pulldom.parseString(string, parser=None)* – якщо XML представлено у вигляді стрічки, можна використовувати дану функцію замість попередньої. Повертає DOMEventStream, що представляє стрічку.
  + *xml.dom.pulldom.default\_bufsize* – значення за замовчуванням для параметра bufsize функції parse(). Значення цієї змінної може бути змінено перед викликанням функції parse().
  + *getEvent()* – повертає кортеж, який містить event і такий поточний node як xml.dom.minidom; документ, якщо подія еквівалентна START\_DOCUMENT, xml.dom.minidom; елемент, якщо подія еквівалентна START\_ELEMENT або END\_ELEMENT, або xml.dom.minidom; текст, якщо подія еквівалентна CHARACTERS.
  + *expandNode(node)* – розширює можливості знаходження інформації із однаковими даними.
  + *reset()*.
* **xml.sax**: надає ряд модулів, які реалізовують SAX інтерфейс для Python:
  + *xml.sax.make\_parser(parser\_list=[])* – створює і повертає об’єкт SAX XMLReader.
  + *xml.sax.parse(filename\_or\_stream, handler, error\_handler=handler.ErrorHandler())* – створює SAX аналізатор і викорустовує його, щоб проаналізувати документ. Параметр handler має бути екземпляром SAX ContentHandler.
  + *xml.sax.parseString(string, handler, error\_handler=handler.ErrorHandler())* – схожа до parse(), але аналізує з буфера рядка, отриманого як параметр.

Також надає перелік вийнятків SAX:

* + *xml.sax.SAXException(msg, exception=None)* – інкапсулює XML помилку або попередження.
  + *xml.sax.SAXParseException(msg, exception, locator)* – збуджується при помилках аналізу.
  + *xml.sax.SAXNotRecognizedException(msg, exception=None)* – збуджується, коли SAX XMLReader стикається з невизначеною функцією.
  + *xml.sax.SAXNotSupportedException(msg, exception=None)* – виникає, коли SAX XMLReader просять встановити властивість значень, реалізація яких не підтримується.
  + *SAXException.getMessage()* – повертає повідомлення для користувача, яке описує помилку.
  + *SAXException.getException()* – повертає інкапсульований об’єкт виключення або None.
* **xml.parsers.expat**: це Python інтерфейс для Expat XML аналізатора. Містить дві функції:
  + *xml.parsers.expat.ErrorString(errno)* – повертає пояснювальний рядок для заданого номера помилки.
  + *xml.parsers.expat.ParserCreate(encoding=None, namespace\_separator=None)* – створює і повертає новий xmlparser об’єкт.

XMLParser об’єкти містять наступні методи:

* + *xmlparser.Parse(data[, isfinal])* – аналізує вміст даних рядка, викликаючи відповідні функції для їх обробки.
  + *xmlparser.ParseFile(file)* – аналізує XML дані із об’єкту file.
  + *xmlparser.SetBase(base)* – встановлює базу, яка буде використовуватися для вирішення відповідних URI у системних ідентифікаторах.
  + *xmlparser.GetBase()* – повертає рядок, що містить базу, встановлену попереднім викликом SetBase(), або None, якщо SetBase() не була викликана.
  + *xmlparser.GetInputContext()* – повертає вхідні дані, що створили поточну подію у вигляді рядка.
  + *xmlparser.SetParamEntityParsing(flag)* – контролює аналіз параметрів.
  + *xmlparser.UseForeignDTD([flag])* – виклик методу із flag, який матиме значення True, призведе Expat до виклику ExternalEntityRefHandler із None для всіх аргументів, щоб міг бути завантажений альтернативний DTD (*Document Type Definition*).