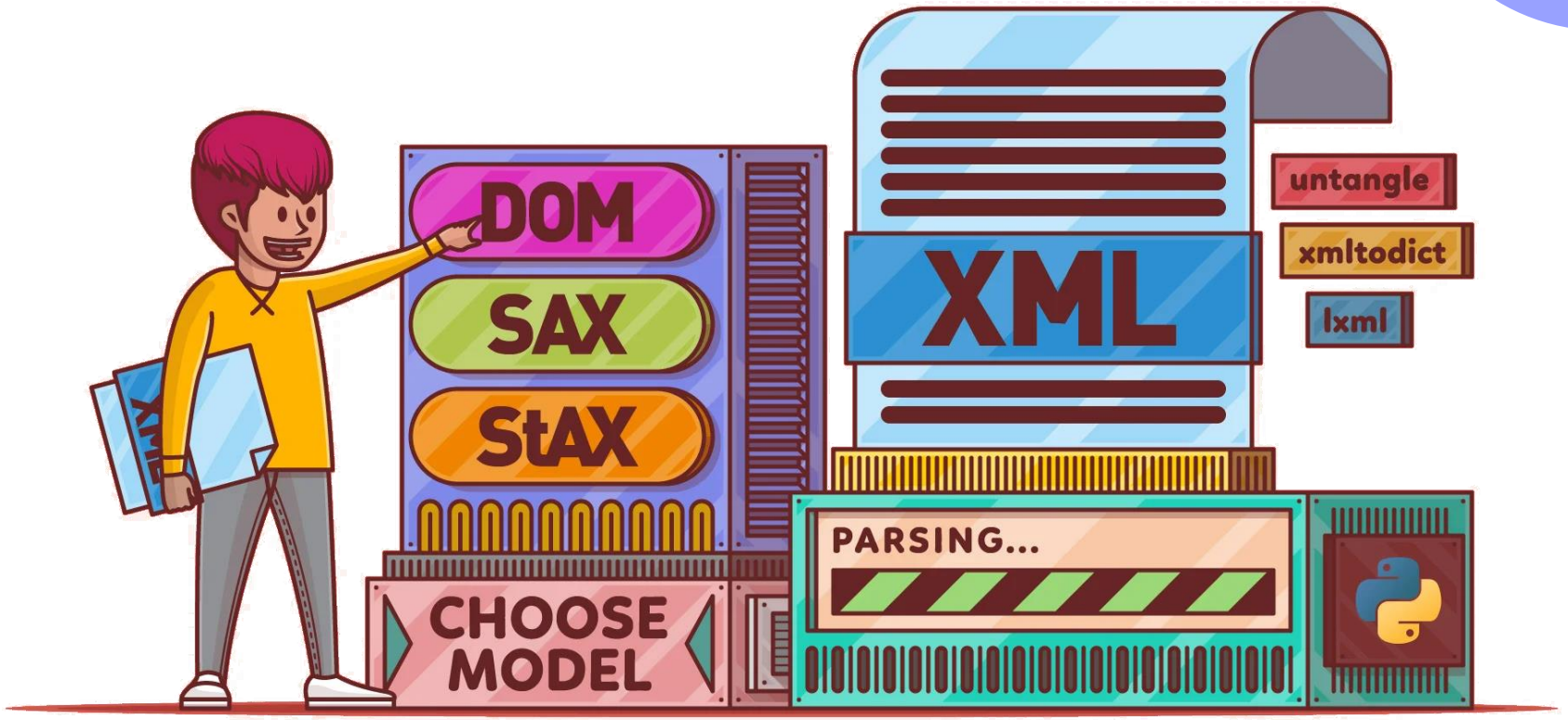
Przetwarzanie danych XML w języku Python



Real Python

Wstęp

XML

XML - (ang. E**X**tensible **M**arkup **L**anguage, rozszerzalny język znaczników) język znaczników wykorzystywany do zapisu danych w ustrukturyzowanej postaci.

```xml
<?xml version="1.0"?>
  <books>
    <book>
        <author>Carson</author>
        <price format="dollar">31.95</price>
        <pubdate>05/01/2001</pubdate>
    </book>
    <pubinfo>
        <publisher>MSPress</publisher>
        <state>WA</state>
    </pubinfo>
  </books>
```

Parsery

Parser (aka analizator składniowy) - program dokonujący analizy składniowej danych wejściowych w ramach określonej gramatyki.

Parsery umożliwiają przetworzenie tekstu czytelnego dla człowieka w strukturę danych przydatną dla oprogramowania komputera.

Parsery języka XML

Do najpopularniejszych parserów języka XML należą:

- DOM
- SAX

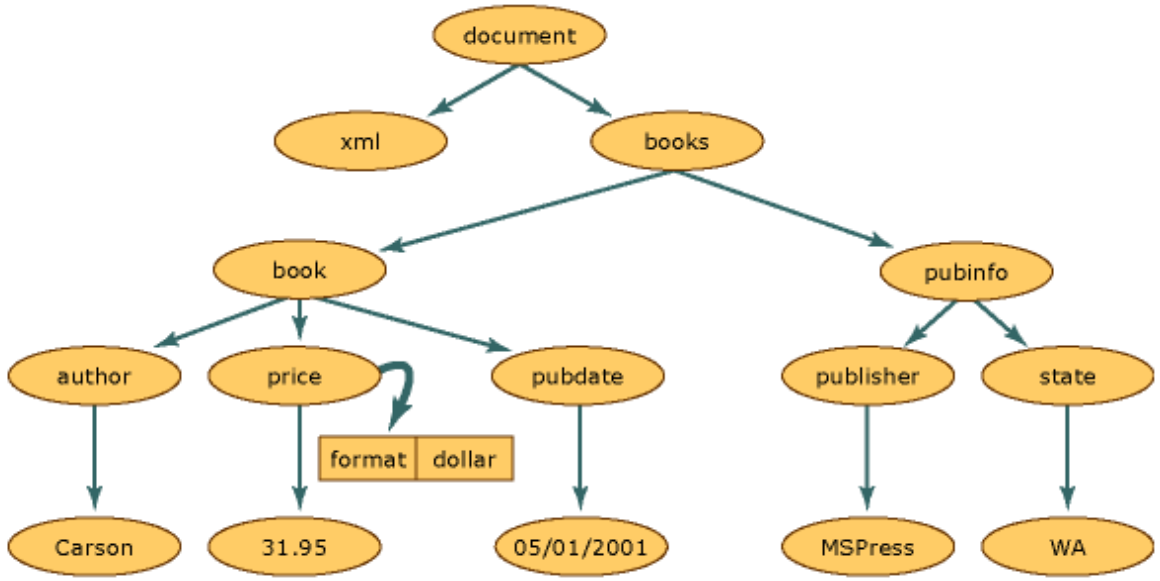
DOM

DOM - (*ang **D**ocument **O**bject **M**odel, obiektowy model dokumentu*) parser, który na podstawie zawartości xml-a tworzy w pamięci RAM drzewiastą reprezentację danych XML (tzw. DOM).

DOM

DOM - (*ang **D**ocument **O**bject **M**odel, obiektowy model dokumentu*) parser, który na podstawie zawartości xml-a tworzy w pamięci RAM drzewiastą reprezentację danych XML o tej samej nazwie (tzn. DOM).

```xml
<?xml version="1.0"?>
  <books>
    <book>
        <author>Carson</author>
        <price format="dollar">31.95</price>
        <pubdate>05/01/2001</pubdate>
    </book>
    <pubinfo>
        <publisher>MSPress</publisher>
        <state>WA</state>
    </pubinfo>
  </books>
```



DOM Python

xml.dom — The Document Obje

https://docs.python.org/3/library/xml.dom.html

Zaloguj

Python » English 3.11.1 3.11.1 Documentation » The Python Standard Library » Structured Markup Processing Tools » `xml.dom` — The Document Object Model API

Quick search Go | previous | next | modules | index

Table of Contents

Previous topic

`xml.etree.ElementTree` — The ElementTree XML API

Next topic

`xml.dom.minidom` — Minimal DOM implementation

This Page

Report a Bug
Show Source

`xml.dom` — The Document Object Model API

Source code: Lib/xml/dom/__init__.py

The Document Object Model, or "DOM," is a cross-language API from the World Wide Web Consortium (W3C) for accessing and modifying XML documents. A DOM implementation presents an XML document as a tree structure, or allows client code to build such a structure from scratch. It then gives access to the structure through a set of objects which provided well-known interfaces.

The DOM is extremely useful for random-access applications. SAX only allows you a view of one bit of the document at a time. If you are looking at one SAX element, you have no access to another. If you are looking at a text node, you have no access to a containing element. When you write a SAX application, you need to keep track of your program's position in the document somewhere in your own code. SAX does not do it for you. Also, if you need to look ahead in the XML document, you are just out of luck.

Some applications are simply impossible in an event driven model with no access to a tree. Of course you could build some sort of tree yourself in SAX events, but the DOM allows you to avoid writing that code. The DOM is a standard tree representation for XML data.

The Document Object Model is being defined by the W3C in stages, or "levels" in their terminology. The Python mapping of the API is substantially based on the DOM Level 2 recommendation.

DOM applications typically start by parsing some XML into a DOM. How this is accomplished is not covered at all by DOM Level 1, and Level 2 provides only limited improvements: There is a `DOMImplementation` object class which provides access to `Document` creation methods, but no way to access an XML reader/parser/Document builder in an implementation-independent way. There is also no well-defined way to access these methods without an existing `Document` object. In Python, each DOM implementation will provide a function `getDOMImplementation()`. DOM Level 3 adds a Load/Store specification, which defines an interface to the reader, but this is not yet available in the Python standard library.

Once you have a DOM document object, you can access the parts of your XML document through its properties and methods. These properties are defined in the DOM specification; this portion of the reference manual describes the interpretation of the specification in Python.

The specification provided by the W3C defines the DOM API for Java, ECMAScript, and OMG IDL. The Python mapping defined here is based in large part on the IDL version of the specification, but strict compliance is not required (though implementations are free to support the strict mapping from IDL). See section Conformance for a detailed discussion of mapping requirements.

See also:

SAX

SAX - (*ang **S**imple **A**PI for **X**ML*) parser XML sterowany wydarzeniami (ang. event-based parser).

Podczas analizy składniowej parser SAX w odróżnieniu od parsera DOM nie ładuje całego xml-a do pamięci RAM. W pamięci przechowujemy tylko ten fragment xml-a, który jest nam aktualnie potrzebny.

Za pomocą SAX możemy czytać zawartość xml-a, ale nie możemy jej modyfikować.

SAX, jak sama nazwa wskazuje, jest jednocześnie API (Application Programming Interface) do pracy z plikami XML.

SAX

Niezabezpieczona | www.saxproject.org

Zaloguj

SAX

General
About SAX
Copyright Status
Events vs. Trees
FAQ
Links

Java API
Quickstart
Features and Properties
Filters
Namespaces
JavaDoc

SAX Evolution
Genesis
SAX 1.0 Overview
SAX 2.0 Changes
SAX 2.0 Extensions
Other Languages

SourceForge Services
Bugs/RFEs
Project Page

SOURCEFORGE

About SAX

This is the official website for SAX. It replaces David Megginson's original SAX page.

SAX is the *Simple API for XML*, originally a Java-only API. SAX was the first widely adopted API for XML in Java, and is a "de facto" standard. The current version is SAX 2.0.1, and there are versions for several programming language environments other than Java.

SAX has recently switched over to the SourceForge project infrastructure. The intent is to continue the open development and maintainence process for SAX (no NDAs required) while making it easier to track open SAX issues outside of the high-volume `xml-dev` list. Project resources include archived mailing lists and a download area. See the Project Page link for full information about project facilities which are being used, as well as news announcements. Use the *sax-users@lists.sourceforge.net* mailing list to discuss problems that come up when you're trying to use SAX.

David Megginson, who runs an XML consulting company, has resumed maintaining SAX after a period of excellent work by David Brownell (if you use SAX, you should think about buying David Brownell's SAX2 book).

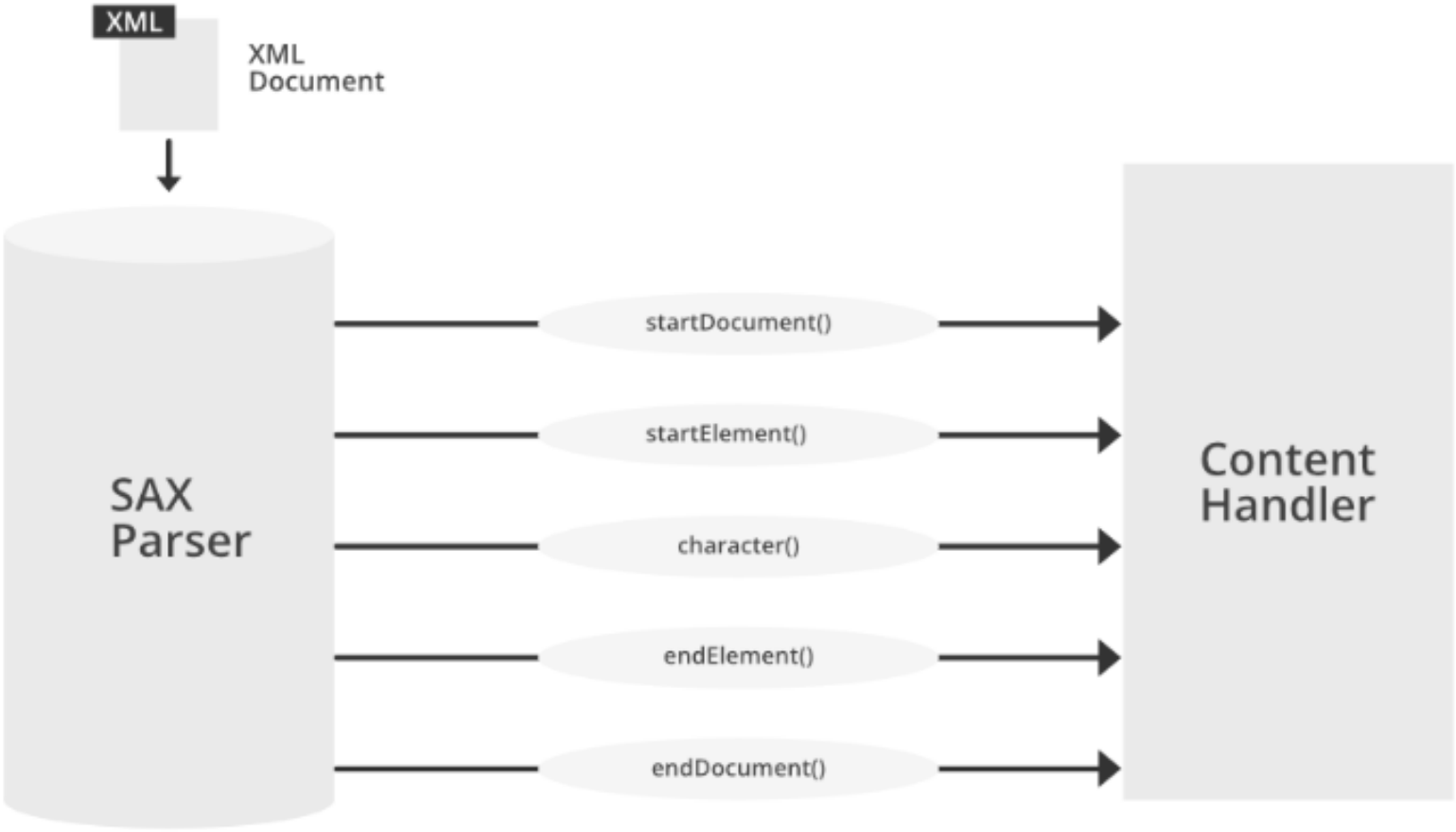
27-April 2004: SAX 2.0.2 (sax2 r3) is out. Download it by going to the Sourceforge download area. That download includes full source, pre-generated javadoc, and a JAR file you can install.

29-January 2002: SAX 2.0.1 (sax2 r2) is out! Download it by going to the Sourceforge download area. That download includes full source, pre-generated javadoc, and a JAR file you can install. Or, consult the javadoc link at left. That's current, and includes the preliminary "SAX2 Extensions 1.1" APIs.

12-November 2001: There are some SAX2 conformance tests available, using the JUNIT testing framework. Download the "sax2unit" tests at the xmlconf project. These are in addition to the SAX2-hosted XML conformance tests mentioned on the "links" page, and address different issues.

http://www.saxproject.org/

SAX (**S**imple **A**PI for **X**ML)



SAX Python

xml.sax — Support for SAX2 pars

https://docs.python.org/3/library/xml.sax.html

Python » English 3.11.1 3.11.1 Documentation » The Python Standard Library » Structured Markup Processing Tools » `xml.sax` — Support for SAX2 parsers

Quick search Go | previous | next | modules | index

Table of Contents

Previous topic

`xml.dom.pulldom` — Support for building partial DOM trees

Next topic

`xml.sax.handler` — Base classes for SAX handlers

This Page

Report a Bug
Show Source

`xml.sax` — Support for SAX2 parsers

Source code: Lib/xml/sax/__init__.py

The `xml.sax` package provides a number of modules which implement the Simple API for XML (SAX) interface for Python. The package itself provides the SAX exceptions and the convenience functions which will be most used by users of the SAX API.

Warning: The `xml.sax` module is not secure against maliciously constructed data. If you need to parse untrusted or unauthenticated data see XML vulnerabilities.

Changed in version 3.7.1: The SAX parser no longer processes general external entities by default to increase security. Before, the parser created network connections to fetch remote files or loaded local files from the file system for DTD and entities. The feature can be enabled again with method `setFeature()` on the parser object and argument `feature_external_ges`.

The convenience functions are:

`xml.sax.`**`make_parser`**`(parser_list=[])`

Create and return a SAX `XMLReader` object. The first parser found will be used. If *parser_list* is provided, it must be an iterable of strings which name modules that have a function named `create_parser()`. Modules listed in *parser_list* will be used before modules in the default list of parsers.

Changed in version 3.8: The *parser_list* argument can be any iterable, not just a list.

`xml.sax.`**`parse`**`(filename_or_stream, handler, error_handler=handler.ErrorHandler())`

Create a SAX parser and use it to parse a document. The document, passed in as *filename_or_stream*, can be a filename or a file object. The *handler* parameter needs to be a SAX `ContentHandler` instance. If *error_handler* is given, it must be a SAX `ErrorHandler` instance; if omitted, `SAXParseException` will be raised on all errors. There is no return value; all work must be done by the *handler* passed in.

`xml.sax.`**`parseString`**`(string, handler, error_handler=handler.ErrorHandler())`

Similar to `parse()`, but parses from a buffer *string* received as a parameter. *string* must be a `str` instance or a bytes-like object.

Changed in version 3.5: Added support of `str` instances.

A typical SAX application uses three kinds of objects: readers, handlers and input sources. “Reader” in this context is another term for parser, i.e. some piece of code that reads the bytes or characters from the input

DOM vs SAX

DOM	SAX
Wczytuje całą zawartość XML do pamięci RAM	Nigdy nie wczytuje całej zawartości XML do pamięci RAM. Wczytuje jedynie fragment XML, który aktualnie jest nam potrzebny
Umożliwia zarówno odczytywanie jak i modyfikację zawartości XML	Umożliwia jedynie odczytywanie zawartości XML

Pakiet xml

W bibliotece standardowej Pythona znajduje się pakiet xml. Jest to pakiet grupujący najpopularniejsze narzędzia do pracy z danymi w formacie xml.

Pakiet xml

Pydoc: package xml

localhost:8888/xml.html

index
c:\users\user\appdata\local\programs\python\python311\lib\xml__init__.py
Module Reference

xml

```
Core XML support for Python.

This package contains four sub-packages:

dom -- The W3C Document Object Model.  This supports DOM Level 1 +
       Namespaces.

parsers -- Python wrappers for XML parsers (currently only supports Expat).

sax -- The Simple API for XML, developed by XML-Dev, led by David
       Megginson and ported to Python by Lars Marius Garshol.  This
       supports the SAX 2 API.

etree -- The ElementTree XML library.  This is a subset of the full
       ElementTree XML release.
```

Package Contents

dom (package) | etree (package) | parsers (package) | sax (package)

Data

__all__ = ['dom', 'parsers', 'sax', 'etree']

xml.dom — The Document Obje

https://docs.python.org/3/library/xml.dom.html

Table of Contents

xml.dom — The Document Object Model API

Source code: Lib/xml/dom/__init__.py

The Document Object Model, or “DOM,” is a cross-language API from the World Wide Web Consortium (W3C) for accessing and modifying XML documents. A DOM implementation presents an XML document as a tree structure, or allows client code to build such a structure from scratch. It then gives access to the structure through a set of objects which provided well-known interfaces.

The DOM is extremely useful for random-access applications. SAX only allows you a view of one bit of the document at a time. If you are looking at one SAX element, you have no access to another. If you are looking at a text node, you have no access to a containing element. When you write a SAX application, you need to keep track of your program’s position in the document somewhere in your own code. SAX does not do it for you. Also, if you need to look ahead in the XML document, you are just out of luck.

Some applications are simply impossible in an event driven model with no access to a tree. Of course you could build some sort of tree yourself in SAX events, but the DOM allows you to avoid writing that code. The DOM is a standard tree representation for XML data.

The Document Object Model is being defined by the W3C in stages, or “levels” in their terminology. The Python mapping of the API is substantially based on the DOM Level 2 recommendation.

xml.sax — Support for SAX2 parsers

Source code: Lib/xml/sax/__init__.py

The `xml.sax` package provides a number of modules which implement the Simple API for XML (SAX) interface for Python. The package itself provides the SAX exceptions and the convenience functions which will be most used by users of the SAX API.

> **Warning:** The `xml.sax` module is not secure against maliciously constructed data. If you need to parse untrusted or unauthenticated data see XML vulnerabilities.

Changed in version 3.7.1: The SAX parser no longer processes general external entities by default to increase security. Before, the parser created network connections to fetch remote files or loaded local files from the file system for DTD and entities. The feature can be enabled again with method `setFeature()` on the parser object and argument `feature_external_ges`.

The convenience functions are:

`xml.sax.`**`make_parser`**`(`*`parser_list=[]`*`)`

Create and return a SAX `XMLReader` object. The first parser found will be used. If *parser_list* is provided, it must be an iterable of strings which name modules that have a function named `create_parser()`. Modules

Table of Contents

xml.etree.ElementTree — The ElementTree XML API¶

Source code: Lib/xml/etree/ElementTree.py

The `xml.etree.ElementTree` module implements a simple and efficient API for parsing and creating XML data.

Changed in version 3.3: This module will use a fast implementation whenever available.

Deprecated since version 3.3: The `xml.etree.cElementTree` module is deprecated.

Warning: The `xml.etree.ElementTree` module is not secure against maliciously constructed data. If you need to parse untrusted or unauthenticated data see XML vulnerabilities.

Tutorial

This is a short tutorial for using `xml.etree.ElementTree` (`ET` in short). The goal is to demonstrate some of the building blocks and basic concepts of the module.

XML tree and elements

Dziękujemy!

software
development
academy