40. PHP XML DOM

The built-in DOM parser makes it possible to process XML documents in PHP.

40.1. What is DOM?

The W3C DOM provides a standard set of objects for HTML and XML documents, and a standard interface for accessing and manipulating them.

The W3C DOM is separated into different parts (Core, XML, and HTML) and different levels (DOM Level 1/2/3):

* Core DOM - defines a standard set of objects for any structured document

* XML DOM - defines a standard set of objects for XML documents

* HTML DOM - defines a standard set of objects for HTML documents

If you want to learn more about the XML DOM, please visit our XML DOM tutorial.

40.2. XML Parsing

To read and update - create and manipulate - an XML document, you will need an XML parser.

There are two basic types of XML parsers:

- Tree-based parser: This parser transforms an XML document into a tree structure. It analyzes the whole document, and provides access to the tree elements
- Event-based parser: Views an XML document as a series of events. When a specific event occurs, it calls a function to handle it

The DOM parser is an tree-based parser.

Look at the following XML document fraction:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<from>Jani</from>
```

The XML DOM sees the XML above as a tree structure:

- Level 1: XML Document
- Level 2: Root element: <from>
- Level 3: Text element: "Jani"

40.3. Installation

The DOM XML parser functions are part of the PHP core. There is no installation needed to use these functions.

An XML File

The XML file below will be used in our example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

40.4. Load and Output XML

We want to initialize the XML parser, load the xml, and output it:

Example

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

print $xmlDoc->saveXML();
?>
```

The output of the code above will be:

```
Tove Jani Reminder Don't forget me this weekend!
```

If you select "View source" in the browser window, you will see the following HTML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
```

```
<body>Don't forget me this weekend!</body>
</note>
```

The example above creates a DOMDocument-Object and loads the XML from "note.xml" into it.

Then the saveXML() function puts the internal XML document into a string, so we can output it.

40.5. Looping through XML

We want to initialize the XML parser, load the XML, and loop through all elements of the <note> element:

Example

```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("note.xml");

$x = $xmlDoc->documentElement;
foreach ($x->childNodes AS $item)
  {
  print  $item->nodeName  .  " = "  .  $item->nodeValue  .
"<br>";
  }
?>
```

The output of the code above will be:

```
#text =
to = Tove
#text =
from = Jani
#text =
heading = Reminder
#text =
body = Don't forget me this weekend!
#text =
```

In the example above you see that there are empty text nodes between each element.

When XML generates, it often contains white-spaces between the nodes. The XML DOM parser treats these as ordinary elements, and if you are not aware of them, they sometimes cause problems.