39. PHP XML Expat Parser

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

39.1. What is XML?

XML is used to describe data and to focus on what data is. An XML file describes the structure of the data. In XML, no tags are predefined. You must define your own tags.

39.2. What is Expat?

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. e.g. the Document Object Model (DOM)
- 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 Expat parser is an event-based parser.

Event-based parsers focus on the content of the XML documents, not their structure. Because of this, event-based parsers can access data faster than tree-based parsers.

Look at the following XML fraction:

```
<from>Jani</from>
```

An event-based parser reports the XML above as a series of three events:

- Start element: from
- Start CDATA section, value: Jani
- Close element: from

The XML example above contains well-formed XML. However, the example is not valid XML, because there is no Document Type Definition (DTD) associated with it.

However, this makes no difference when using the Expat parser. Expat is a non-validating parser, and ignores any DTDs.

As an event-based, non-validating XML parser, Expat is fast and small, and a perfect match for PHP web applications.

Note: XML documents must be well-formed or Expat will generate an error.

39.3. Installation

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

39.4. 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>
```

39.5. Initializing the XML Parser

We want to initialize the XML parser in PHP, define some handlers for different XML events, and then parse the XML file.

Example

```
<?php
//Initialize the XML parser
$parser=xml_parser_create();

//Function to use at the start of an element
function start($parser,$element_name,$element_attrs)
  {
  switch($element_name)
    {
    case "NOTE":
    echo "-- Note --<br>";
    break;
    case "TO":
    echo "To: ";
    break;
    case "FROM":
    echo "From: ";
    break;
    case "HEADING":
    echo "Heading: ";
    break;
```

```
        case "BODY":
        echo "Message: ";
        }
      }

    //Function to use at the end of an element
    function stop($parser,$element_name)
      {
      echo "<br>";
      }

    //Function to use when finding character data
    function char($parser,$data)
      {
      echo $data;
      }

    //Specify element handler
    xml_set_element_handler($parser,"start","stop");

    //Specify data handler
    xml_set_character_data_handler($parser,"char");

    //Open XML file
    $fp=fopen("test.xml","r");

    //Read data
    while ($data=fread($fp,4096))
      {
      xml_parse($parser,$data,feof($fp)) or
      die (sprintf("XML Error: %s at line %d",
      xml_error_string(xml_get_error_code($parser)),
      xml_get_current_line_number($parser)));
      }

    //Free the XML parser
    xml_parser_free($parser);
    ?>
```

The output of the code above will be:

```
    -- Note --
    To: Tove
    From: Jani
    Heading: Reminder
```

```
Message: Don't forget me this weekend!
```

How it works:

1. Initialize the XML parser with the xml_parser_create() function
2. Create functions to use with the different event handlers
3. Add the xml_set_element_handler() function to specify which function will be executed when the parser encounters the opening and closing tags
4. Add the xml_set_character_data_handler() function to specify which function will execute when the parser encounters character data
5. Parse the file "test.xml" with the xml_parse() function
6. In case of an error, add xml_error_string() function to convert an XML error to a textual description
7. Call the xml_parser_free() function to release the memory allocated with the xml_parser_create() function

39.6. More PHP Expat Parser

For more information about the PHP Expat functions, visit our PHP XML Parser Reference.