XML i PHP

Denne artikel gennemgår kort nogle af de mange muligheder for brug af XML i PHP. Det sker ved brug af eksempler. Eksemplerne kræver PHP 5.

Den forudsætter et vist kendskab til både XML og PHP.

Skrevet den **14. Feb 2010** af **arne_v** I kategorien **Programmering / PHP** | ★★☆☆☆

Historie:
V1.0 - 29/12/2008 - original
V1.1 - 14/02/2010 - smårettelser

Indledning

Der er mange XML extensions i PHP. Jeg vil vise eksempler på brug af:
DOM
Simple XML
XML reader
XML writer
XSL

Eksemplerne forudsætter PHP 5 og brug af OOP.

Der findes nogle XML extensions i PHP 4, men disse er ikke kompatible med de ovennævnte. Og da PHP 4 er forældet nu vil jeg ikke bruge tid på dem.

For fuld dokumentation af PHP XML extensions se:
http://www.php.net/manual/en/refs.xml.php

For hurtig intro til XML terminologi se:
http://www.eksperten.dk/guide/627 "XML hvad, hvorfor og hvornår ?"

Gennemgangen af XML extensions vil i høj grad basere sig på simple eksempler der illusterer tankegangen bag en given extension og hvordan man kan komme igang. Til praktisk brug er flere funktioner formentligt nødvendige, men dem kan man slå op i dokumentationen (se link ovenfor).

Eksemplerne nedenfor vil bruge følgende simple XML fil som input:

test.xml

```
<?xml version='1.0' standalone='yes'?>
<medlemmer>
    <medlem no="1">
        <navn>Niels Nielsen</navn>
        <adresse>Nellikevej 19</adresse>
    </medlem>
```

```
    <medlem no="2">
        <navn>Jens Jensen</navn>
        <adresse>Jagtvej 17</adresse>
    </medlem>
    <medlem no="3">
        <navn>Ole Olsen</navn>
        <adresse>Omfartsvejen 13</adresse>
    </medlem>
</medlemmer>
```

Den genererede HTML i eksemplerne er ikke speciel advanceret, men formålet er at vise XML processing ikke HTML generering (og så er jeg iøvrigt ikke specielt HTML kyndig).

DOM

DOM extension bygger på den W3C standardiserede DOM model som også kendes fra JavaScript, ASP, Java, .NET etc..

Hele XML filen indlæses i et hierakisk træ, hvor elementer/attributter/text er children til et element.

Vær opmærksom på at et DOM træ fylder mere i memory end XML filen. DOM er derfor ikke en god løsning til meget store XML filer (3 cifrede MB eller GB størrelser).

Metoderne til at tilgå dette træ er standardiserede, så man kan genbruge sin viden fra andre sprog i PHP.

Først et eksempel som viser en XML fil som en HTML TABLE ved at løbe igennem træet på meget traditionel vis.

showdom.php

```
<html>
<head>
<title>DOM demo</title>
</head>
<body>
<table border="1">
<tr>
<th>No</th>
<th>Navn</th>
<th>Adresse</th>
</tr>
<?php
// opret DOM træ
$doc = new DOMDocument();
// indlæs XML fil i DOM træ
$doc->load('test.xml');
// find alle medlem elementer
$medlemmer = $doc->getElementsByTagName('medlem');
```

```
foreach($medlemmer as $medlem) {
    echo "<tr>\r\n";
    // find og udskriv attribut no
    echo "<td>" . $medlem->attributes->getNamedItem('no')->nodeValue .
"</td>\r\n";
    // find alle under elementer
    foreach($medlem->childNodes as $elm) {
        // find under element navn
        if($elm->nodeName == 'navn') $navn = $elm->nodeValue;
        // find under element adresse
        if($elm->nodeName == 'adresse') $adresse = $elm->nodeValue;
    }
    // udskriv de fundne under elementer
    echo "<td>$navn</td>\r\n";
    echo "<td>$adresse</td>\r\n";
    echo "</tr>\r\n";
}
?>
</table>
</body>
</html>
```

Så et eksempel der viser, hvordan man kan søge i en XML fil med XPath.

Jeg vil ikke gå i detaljer med hensyn til XPath syntax det kan man og det er der skrevet bøger om.

Den ultra korte version er:
xxxx - finder elementer med navn xxxx
//xxxx/yyyy - finder elementer med navn yyyy under elementer med navn xxxx
xxxx[yyyy='abc'] - finder elementer med navn xxxx som har et under element med navn yyyy og en tekstværdi 'abc'
xxxx[@yyyy=123] - finder elementer med navn xxxx som har en attribut med navn yyyy og en talværdi 123

searchdom.php

```
<html>
<head>
<title>DOM og XPath demo</title>
</head>
<body>
<?php
if(isset($_POST['no'])) {
    $no = $_POST['no'];
    // opret DOM træ
    $doc = new DOMDocument();
    // indlæs XML fil i DOM træ
    $doc->load('test.xml');
    // opret XPath objekt
    $xpath = new DOMXPath($doc);
```

```
    // søg efter medlemmer med det angivne no
    $medlemmer = $xpath->query("//medlemmer/medlem[@no=$no]");
    // test om der var nogen matchende medlemmer
    if($medlemmer->length > 0) {
        // tag det første matchende medlem
        $medlem = $medlemmer->item(0);
        // udskriv attribut no
        echo "no=" . $medlem->attributes->getNamedItem('no')->nodeValue .
"<br/>\r\n";
        // søg efter sub element navn og udskriv det
        echo "navn=" . $xpath->query("navn/text()",
$medlem)->item(0)->nodeValue . "<br/>\r\n";
        // søg efter sub element adresse og udskriv det
        echo "adresse=" . $xpath->query("adresse/text()",
$medlem)->item(0)->nodeValue . "<br/>\r\n";
    } else {
        echo "Ingen fundet<br/>\r\n";
    }
}
?>
<form method="POST">
No: <input type="TEXT" name="no"/>
<br/>
<input type="SUBMIT" value="Søg"/>
</form>
</body>
</html>
```

Eksempel på at rette i XML dokument (og returnere XML fremfor HTML).

changedom.php

```
<?php
header("Content-Type: text/xml");
// opret DOM træ
$doc = new DOMDocument();
// indlæs XML fil i DOM træ
$doc->load('test.xml');
// opret XPath objekt
$xpath = new DOMXPath($doc);
// søg efter medlem med no 2
$medlemmer = $xpath->query("//medlemmer/medlem[@no=2]");
$medlem = $medlemmer->item(0);
// fjern medlem
$medlem->parentNode->removeChild($medlem);
// tilføj medlem
$navn = $doc->createElement('navn');
$navn->appendChild($doc->createTextNode('Lars Larsen'));
$adresse = $doc->createElement('adresse');
$adresse->appendChild($doc->createTextNode('Ledvej 14'));
$nytmedlem = $doc->createElement('medlem');
```

```
$nytmedlem->setAttribute('no', '4');
$nytmedlem->appendChild($navn);
$nytmedlem->appendChild($adresse);
$doc->documentElement->appendChild($nytmedlem);
// udskriv DOM træ som XML
echo $doc->saveXML();
?>
```

Eksempel på at oprette et nyt XML dokument (og returnere XML fremfor HTML).

createdom.php

```
<?php
header("Content-Type: text/xml");
// opret DOM træ
$doc = new DOMDocument();
// opret document element og 3 sub elementer
$one1 = $doc->createElement('one');
$one1->appendChild($doc->createTextNode('A'));
$one2 = $doc->createElement('one');
$one2->appendChild($doc->createTextNode('BB'));
$one3 = $doc->createElement('one');
$one3->appendChild($doc->createTextNode('CCC'));
$root = $doc->createElement('all');
$root->appendChild($one1);
$root->appendChild($one2);
$root->appendChild($one3);
$doc->appendChild($root);
// udskriv DOM træ som XML
echo $doc->saveXML();
?>
```

DOM er yderst fleksibelt, men har en lidt halvtung syntax.

Simple XML

Simple XML har også hele XML filen i memory, men man tilgår elementer og attributter som PHP objekter.

Eksempel som viser en XML fil som en HTML TABLE.

showsimple.php

```
<html>
<head>
<title>Simple XML demo</title>
</head>
<body>
```

```
<table border="1">
<tr>
<th>No</th>
<th>Navn</th>
<th>Adresse</th>
</tr>
<?php
// opret simple struktur og indlæs XML fil
$xml = new SimpleXMLElement(file_get_contents('test.xml'));
// find alle medlem elementer
foreach($xml->medlem as $medlem) {
    echo "<tr>\r\n";
    // find og udskriv attribut no
    echo "<td>" . $medlem['no'] . "</td>\r\n";
    // find og udskriv sub element navn
    echo "<td>" . $medlem->navn . "</td>\r\n";
    // find og udskriv sub element adresse
    echo "<td>" . $medlem->adresse . "</td>\r\n";
    echo "</tr>\r\n";
}
?>
</table>
</body>
</html>
```

Eksempel der viser, hvordan man kan søge i en XML fil med XPath.

searchsimple.php

```
<html>
<head>
<title>Simple XML og XPath demo</title>
</head>
<body>
<?php
if(isset($_POST['no'])) {
    $no = $_POST['no'];
    // opret simple struktur og indlæs XML fil
    $xml = new SimpleXMLElement(file_get_contents('test.xml'));
    // søg efter medlemmer med det angivne no
    $medlemmer = $xml->xpath("//medlemmer/medlem[@no=$no]");
    // test om der var nogen matchende medlemmer
    if(count($medlemmer) > 0) {
        // tag det første matchende medlem
        $medlem = $medlemmer[0];
        // udskriv attribut no
        echo "no=" . $medlem['no'] . "<br/>\r\n";
        // udskriv sub element navn
        echo "navn=" . $medlem->navn . "<br/>\r\n";
        // udskriv sub element adresse
        echo "adresse=" . $medlem->adresse . "<br/>\r\n";
```

```
    } else {
        echo "Ingen fundet<br/>\r\n";
    }
}
?>
<form method="POST">
No: <input type="TEXT" name="no"/>
<br/>
<input type="SUBMIT" value="Søg"/>
</form>
</body>
</html>
```

Som man kan se er simple XML syntaxen noget nemmere end DOM syntaxen.

Bemærk dog at simple XML ikke er så godt til at ændre/oprette XML dokumenter med.

Og brug af simple XML til XML med namespaces er også kendt for at give store problemer.

XML reader

XML reader er en event driven pull parser, som kan processe en XML fil uden at læse det hele ind i memory og er derfor velgnet til at processe meget store XML filer med.

Eksempel som viser en XML fil som en HTML TABLE.

showreader.php

```
<html>
<head>
<title>XML reader demo</title>
</head>
<body>
<table border="1">
<tr>
<th>No</th>
<th>Navn</th>
<th>Adresse</th>
</tr>
<?php
// opret reader og åben XML fil
$rdr = new XMLReader();
$rdr->xmL(file_get_contents('test.xml'));
// læs hele filen
while($rdr->read()){
    // test på node type
    switch ($rdr->nodeType) {
        case XMLReader::TEXT:
            // gem text til senere
```

```
            $s = $rdr->value;
            break;
        case XMLReader::ELEMENT:
            if($rdr->localName == 'medlem') {
                // hvis start på medlem element så hent attribut no
                $no = $rdr->getAttribute('no');
            }
            break;
        case XMLReader::END_ELEMENT:
            if($rdr->localName == 'navn') {
                // hvis start på navn element så hent gemt text
                $navn = $s;
            } else if($rdr->localName == 'adresse') {
                // hvis start på adresse element så hent gemt text
                $adresse = $s;
            } else if($rdr->localName == 'medlem') {
                // hvis slut på medlem element så udskrive alt
                echo "<tr>\r\n";
                // udskriv attribut no
                echo "<td>" . $no . "</td>\r\n";
                // udskriv sub element navn
                echo "<td>" . $navn . "</td>\r\n";
                // udskriv sub element adresse
                echo "<td>" . $adresse . "</td>\r\n";
                echo "</tr>\r\n";
            }
            break;
    }
}
?>
</table>
</body>
</html>
```

XML writer

XML writer muliggør at man skriver et XML dokument uden at have hele dokumentet i memory og er derfor velgnet til at generere meget store XML filer med.

Eksempel på at oprette et nyt XML dokument (og returnere XML fremfor HTML).

createwriter.php

```
<?php
header("Content-Type: text/xml");
// opret writer
$wrt = new XMLWriter();
// skriv direkte til output
$wrt->openURI('php://output');
// skriv XML
$wrt->startDocument();
```

```
$wrt->startElement('all');
$wrt->startElement('one');
$wrt->text('A');
$wrt->endElement();
$wrt->startElement('one');
$wrt->text('BB');
$wrt->endElement();
$wrt->startElement('one');
$wrt->text('CCC');
$wrt->endElement();
$wrt->endElement();
$wrt->endDocument();
?>
```

XSL

XSLT er en måde at transformere XML på.

Det kan laves client side. Men det kan også laves server side.

Laver man det server side, så kan man outputte HTML udfra XML uden at kode presentationen i PHP.

Eksempel.

showxsl.php

```
<html>
<head>
<title>XSL demo</title>
</head>
<body>
<?php
// opret DOM træ
$xml = new DOMDocument();
// indlæs XML fil i DOM træ
$xml->load('test.xml');
// opret DOM træ
$xsl = new DOMDocument();
// indlæs XSL fil i DOM træ
$xsl->load("test.xsl");
// opret XSLT processor og bed den bruge XSL
$t = new XSLTProcessor();
$t->importStylesheet($xsl);
// udskriv XML transformeret til HTML
echo $t->transformToXML($xml);
?>
</table>
</body>
</html>
```

test.xsl

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method='html'/>

<xsl:template match="/">
<table border="1">
<tr>
<th>No</th>
<th>Navn</th>
<th>Adresse</th>
</tr>
<xsl:apply-templates/>
</table>
</xsl:template>

<xsl:template match="medlemmer">
<xsl:apply-templates/>
</xsl:template>

<xsl:template match="medlem">
<tr>
<td><xsl:value-of select="@no"/></td>
<td><xsl:value-of select="navn"/></td>
<td><xsl:value-of select="adresse"/></td>
</tr>
</xsl:template>

</xsl:stylesheet>
```

XSLT er en særdels kraftfuld teknologi. Men vær også opmærksom på, at det er et helt nyt sprog som man skal sætte sig ind i. Og det er et svært sprog - sværere end PHP !

Anbefaling

Efter at have set 5 forskellige XML extensions i brug, så er det jo naturligt at spørge om, hvilken man skal vælge.

Min anbefaling er:

modificere eksisternde XML => DOM
oprette nyt XML => XML writer
konvertere XML til andet XML format => XSLT
læse XML med namespaces => DOM
læse store XML filer => XML reader
læse XML generelt => simple XML

Altså simple XML i de fleste tilfælde !

Kommentar af jensgram d. 29. Dec 2008 | 1

Jaja, ok :) Fint som referenceværktøj og til det indledende overblik (og mere gør artiklen vel egentlig heller ikke krav på at være).

Kommentar af jakobdo d. 07. Jan 2009 | 2

God artikel Arne_v, vil bestemt blive brugt som opslag i fremtiden, da jeg "godt" kan anvende XML, men støder dog tit ind i problemet: Hvordan er det nu man gør??? :o)

Kommentar af kodak d. 19. Jan 2009 | 3

Jeg kan ikke få den til at skrive ÆØÅ æøå