

3 PHP und MySQL auf der Serverseite

Zwar kümmert sich AJAX hauptsächlich darum, die Clients cleverer zu machen, doch die Server, mit denen diese Clients kommunizieren, müssen genauso clever sein, da die beiden sonst nicht sehr weit miteinander kommen werden.

In Kapitel 2 haben Sie lediglich statischen Text oder XML-Dateien vom Server gelesen. Doch im vorliegenden Kapitel werden wir beginnen, auch die Serverseite ans Arbeiten zu bekommen, indem wir mit PHP dynamisch Ausgabe generieren und mit MySQL die zugrunde liegenden Daten manipulieren und speichern. In diesem Kapitel werden folgende Themen behandelt:

- Mit PHP Funktionalität auf der Serverseite ausführen
- Clients per Parameterübergabe mit dem Server kommunizieren lassen
- XML auf Client und Server einsetzen
- Mit PHP-Skripten Sicherheitsprobleme mit JavaScript verhindern
- Wiederkehrende Aufgaben auf dem Client erledigen
- Mit MySQL-Datenbanken umgehen
- Die Anwendungsarchitektur optimieren

3.1 PHP und DOM

In Kapitel 2 haben wir Daten asynchron vom Server gelesen. Dieser Mechanismus ist zwar Standard und wird in diesem Buch noch viele Male verwendet, doch eine Sache war ungewöhnlich: Die Daten, die vom Server zurückgegeben wurden, waren statische Dateien (entweder einfacher Text oder XML).

Doch in den meisten realen Situationen muss der Server Daten verarbeiten und dynamische Ausgaben generieren. In diesem Buch wird diese Arbeit auf der Serverseite mit PHP erledigt. Wenn Ihre PHP-Kenntnisse dafür noch nicht ausreichen, wird eine Onlinesuche

nach dem Begriff „PHP-Tutorial“ jede Menge interessanter Quellen zu Tage fördern, einschließlich des offiziellen PHP-Tutorials unter <http://php.net/tut.php>. Wenn Sie „Learning by Doing“ bevorzugen, schauen Sie doch einfach in die zweite Auflage von *PHP 5 – Grundlagen und Profiwissen* von Jörg Krause oder einer anderen PHP-Einführung.

Sie können sogar mit der in Kapitel 6 erstellten *Suggest and Autocomplete*-Anwendung die Hilfsseite zu den PHP-Funktionen ausfindig machen. Die Anwendung liegt unter <http://ajaxphp.packtpub.com/ajax/suggest/>.

In der ersten Übung zu diesem Kapitel schreiben Sie ein PHP-Skript, das die DOM-Funktionen von PHP nutzt, um XML-Ausgabe zu erzeugen, die der Client dann liest. PHP hat eine ähnliche DOM-Funktionalität wie JavaScript; die offizielle Dokumentation liegt unter <http://www.php.net/manual/en/ref.dom.php>.

Das XML-Dokument, das Sie auf dem Server anlegen, ist fast dasselbe, wie jenes, das in Kapitel 2 als statische XML-Datei abgespeichert wurde, doch dieses Mal wird es dynamisch generiert:

```
<response>
  <books>
    <book>
      <title>Building Reponsive Web Applications with AJAX
        and PHP</title>
      <isbn>1-904811-82-5</isbn>
    </book>
  </books>
</response>
```

Showtime: AJAX mit PHP

1. Legen Sie im `foundations`-Ordner einen Unterordner namens `php` an.
2. Im `php`-Ordner erzeugen Sie eine Datei namens `phptest.html` mit folgendem Text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>Practical AJAX: Using the PHP DOM</title>
    <script type="text/javascript" src="phptest.js"></script>
  </head>
  <body onload="process()">
    The AJAX book of 2006 is:
    <br />
    <div id="myDivElement" />
  </body>
</html>
```

3. Der clientseitige Code, `phptest.js`, gleicht fast aufs Haar der `books.js`-Datei aus der XML-Übung von Kapitel 2. Die Änderungen sind hervorgehoben:

```
// speichert eine Instanz von XMLHttpRequest
var xmlhttp = createXmlHttpRequestObject();

// erzeugt eine XMLHttpRequest-Instanz
function createXmlHttpRequestObject()
{
  // speichert die Referenz auf das XMLHttpRequest-Objekt
```

```

var xmlHttp;
// dies müsste auf allen Browsern außer IE6 und älter funktionieren
try
{
    // versucht, ein XMLHttpRequest-Objekt zu erzeugen
    xmlHttp = new XMLHttpRequest();
}
catch(e)
{
    // für IE6 oder älter
    var XmlHttpVersions = new Array("MSXML2.XMLHTTP.6.0",
                                     "MSXML2.XMLHTTP.5.0",
                                     "MSXML2.XMLHTTP.4.0",
                                     "MSXML2.XMLHTTP.3.0",
                                     "MSXML2.XMLHTTP",
                                     "Microsoft.XMLHTTP");

    // probiert Prog-IDs durch, bis eine funktioniert
    for (var i=0; i<XmlHttpVersions.length && !xmlHttp; i++)
    {
        try
        {
            // versucht, ein XMLHttpRequest-Objekt zu erzeugen
            xmlHttp = new ActiveXObject(XmlHttpVersions[i]);
        }
        catch (e) {}
    }
}
// gibt das erzeugte Objekt oder eine Fehlermeldung zurück
if (!xmlHttp)
    alert("Error creating the XMLHttpRequest object.");
else
    return xmlHttp;
}

// liest eine Datei vom Server
function process()
{
    // nur fortfahren, wenn xmlHttp nicht leer ist
    if (xmlHttp)
    {
        // Versuch einer Serververbindung
        try
        {
            // initiate reading a file from the server
            xmlHttp.open("GET", "phptest.php", true);
            xmlHttp.onreadystatechange = handleRequestStateChange;
            xmlHttp.send(null);
        }
        // zeigt einen Fehler an, falls er auftritt
        catch (e)
        {
            alert("Can't connect to server:\n" + e.toString());
        }
    }
}

// diese Funktion wird bei HTTP-Request-Statusänderungen aufgerufen
function handleRequestStateChange()
{
    // wenn readyState 4 ist, können wir die Server-Antwort lesen
    if (xmlHttp.readyState == 4)
    {
        // nur weitermachen, wenn HTTP-Status „OK“
        if (xmlHttp.status == 200)
        {
            try
            {
                // behandelt die Antwort vom Server
                handleServerResponse();
            }
        }
    }
}

```

```

    }
    catch(e)
    {
        // zeigt Fehlermeldung an
        alert("Error reading the response: " + e.toString());
    }
    else
    {
        // zeigt Statusmeldung an
        alert("There was a problem retrieving the data:\n" +
            xmlHttp.statusText);
    }
}

// behandelt die vom Server empfangene Antwort
function handleServerResponse()
{
    // liest die Meldung vom Server
    var xmlResponse = xmlHttp.responseXML;
    // fängt potenzielle Fehler bei IE und Opera
    if (!xmlResponse || !xmlResponse.documentElement)
        throw("Invalid XML structure:\n" + xmlHttp.responseText);
    // fängt potenzielle Fehler bei Firefox
    var rootNodeName = xmlResponse.documentElement.nodeName;
    if (rootNodeName == "parsererror") throw("Invalid XML structure");
    // beschafft das document-Element von XML
    xmlRoot = xmlResponse.documentElement;
    // beschafft Arrays mit Buchtiteln und ISBNs
    titleArray = xmlRoot.getElementsByTagName("title");
    isbnArray = xmlRoot.getElementsByTagName("isbn");
    // generiert HTML-Ausgabe
    var html = "";
    // durchläuft die Arrays und erzeugt eine HTML-Struktur
    for (var i=0; i<titleArray.length; i++)
        html += titleArray.item(i).firstChild.data +
            ", " + isbnArray.item(i).firstChild.data + "<br/>";
    // beschafft eine Referenz auf das <div>-Element auf der Seite
    myDiv = document.getElementById("myDivElement");
    // beschafft eine Referenz auf das <div>-Element auf der Seite
    myDiv.innerHTML = html;
}

```

4. Zum Schluss legen Sie die Datei phtest.php an:

```

<?php
// stellt XML als Inhaltstyp der Ausgabe ein
header('Content-Type: text/xml');
// erzeugt das neue XML-Dokument
$dom = new DOMDocument();

// erzeugt das Wurzelement <response>
$response = $dom->createElement('response');
$dom->appendChild($response);

// erzeugt das <books>-Element und fügt es <response> als Child hinzu
$books = $dom->createElement('books');
$response->appendChild($books);

// erzeugt das Buchtitel-Element
$title = $dom->createElement('title');
$titleText = $dom->createTextNode
    ('Building Responsive Web Applications with AJAX and PHP');
$title->appendChild($titleText);

// erzeugt das ISBN-Element für das Buch
$isbn = $dom->createElement('isbn');
$isbnText = $dom->createTextNode('1-904811-82-5');

```

```

$isbn->appendChild($isbnText);

// erzeugt das <book>-Element
$book = $dom->createElement('book');
$book->appendChild($title);
$book->appendChild($isbn);

// fügt <book> als Child von <books> an
$books->appendChild($book);

// erzeugt die XML-Struktur in einer Stringvariablen
$xmlString = $dom->saveXML();
// gibt den XML-String aus
echo $xmlString;
?>

```

- Ein einfacher Test zeigt uns, was `phptest.php` zurückgibt. Laden Sie `http://localhost/ajax/foundations/php/phptest.php` in Ihren Browser, um sich zu vergewissern, dass die Datei eine wohlgeformte XML-Struktur generiert:

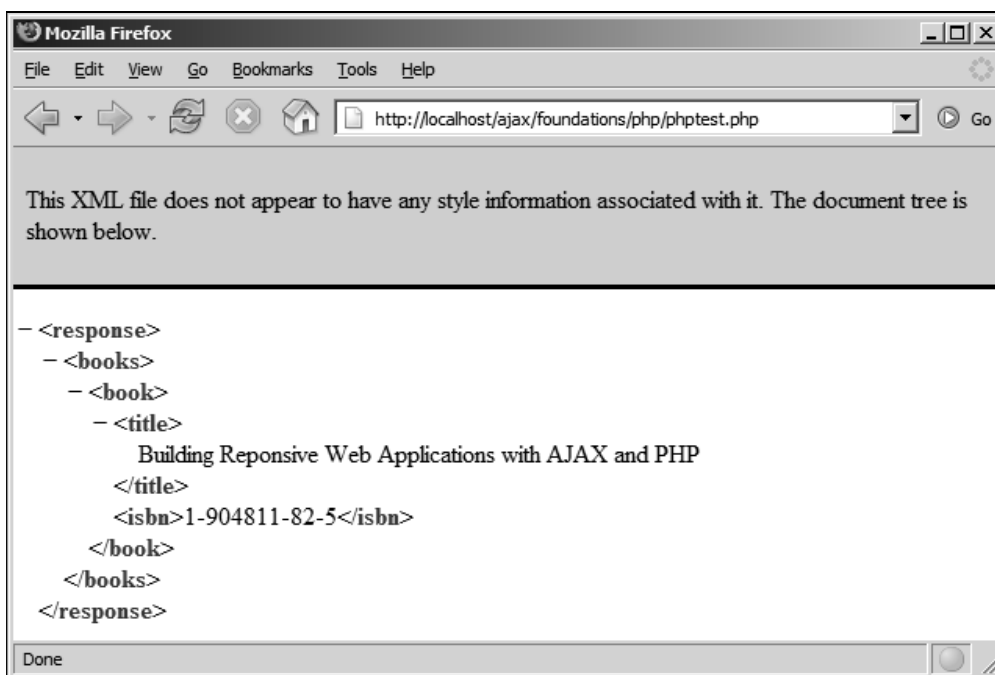


Abbildung 3.1 Eine einfache von PHP generierte XML-Struktur



Wenn das Ergebnis nicht Ihren Erwartungen entspricht, sollten Sie nicht nur den Code, sondern auch Ihre PHP-Installation überprüfen. In Anhang A erfahren Sie, wie Sie Ihren Computer richtig einrichten können.

- Wenn Sie sich überzeugt haben, dass der Server die richtige Antwort zurückliefert, testen Sie die gesamte Lösung, indem Sie `http://localhost/ajax/foundations/php/phptest.html` in den Browser laden.

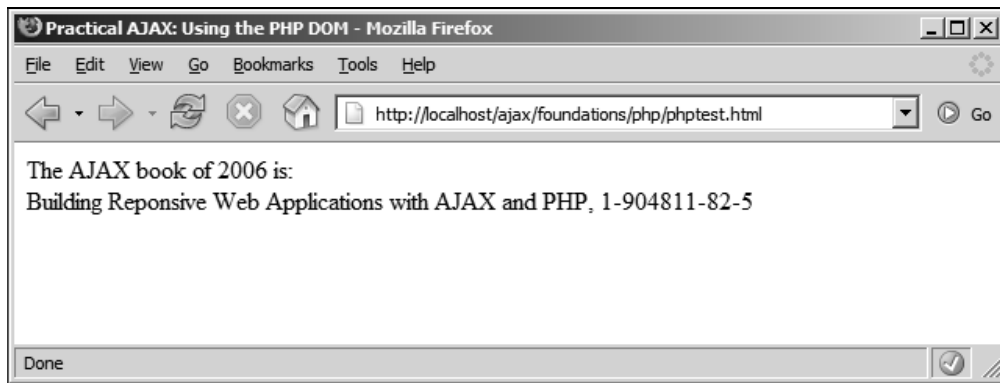


Abbildung 3.2 AJAX mit PHP

Was ist passiert?

Wenn Sie XML-Strukturen nicht nur auf dem Client, sondern auch auf dem Server generieren, müssen Sie sich entscheiden, ob Sie das XML-Dokument mithilfe des DOM oder durch Zusammenfügen von Strings erstellen möchten. Das PHP-Skript `phptest.php` setzt zuerst die Ausgabe auf `text/xml`:

```
<?php
// stellt XML als Inhaltstyp der Ausgabe ein
header('Content-Type: text/xml');
```

Die PHP-Dokumentation der `header`-Funktion finden Sie unter <http://www.php.net/manual/en/function.header.php>. (Wenn Sie 'header' als Suchbegriff in die *Suggest*-Anwendung eingeben, werden Sie auf die Hilfsseite geleitet).



Während wir in JavaScript-Dateien Strings in doppelte Anführungszeichen setzen, verwenden wir in PHP einfache Anführungszeichen. Diese werden schneller verarbeitet, sind sicherer und verursachen nicht so häufig Programmierfehler. Mehr über PHP-Strings erfahren Sie unter <http://php.net/types.string>. Zwei nützliche Artikel über PHP-Strings liegen unter <http://www.sitepoint.com/print/quick-php-tips> und http://www.jeroenmulder.com/weblog/2005/04/php_single_and_double_quotes.php.

Es nimmt kaum Wunder, dass das PHP-DOM fast genauso wie das JavaScript-DOM aussieht. Alles fängt mit der Erzeugung eines DOM-Dokument-Objekts an, das in PHP durch die Klasse `DOMDocument` dargestellt wird:

```
// erzeugt das neue XML-Dokument
$dom = new DOMDocument();
```

Dann wird die XML-Struktur mit Methoden wie `createElement`, `createTextNode`, `appendChild` usw. aufgebaut:

```
// erzeugt das Wurzelement <response>
$response = $dom->createElement('response');
$dom->appendChild($response);
```

```
// erzeugt das <books>-Element und fügt es <response> als Child hinzu
$books = $dom->createElement('books');
$response->appendChild($books);
...
```

Zum Schluss wird die gesamte XML-Struktur als String mithilfe der Funktion `saveXML` gespeichert und dieser String mit `echo` an die Ausgabe geschickt.

```
$xmlString = $dom->saveXML();
// gibt den XML-String aus
echo $xmlString;
?>
```

Dann wird das XML-Dokument gelesen und auf der Clientseite mit den Techniken aus Kapitel 2 angezeigt.



In den meisten Fällen generieren Sie XML-Dokumente auf dem Server und lesen sie auf dem Client, doch natürlich ist auch der umgekehrte Weg möglich. In Kapitel 2 haben Sie gesehen, wie man XML-Dokumente und -Elemente mit dem DOM von JavaScript generiert. Diese Strukturen können Sie dann an PHP übergeben (mit `GET` oder `POST`, wie es in der folgenden Übung gezeigt wird). Um XML-Strukturen von PHP einzulesen, können Sie neben dem DOM auch eine einfacher benutzbare API namens **SimpleXML** einsetzen. Mit SimpleXML werden Sie in Kapitel 9 üben, wenn Sie eine RSS Reader-Anwendung erstellen.

3.2 Parameter übergeben und PHP-Fehler behandeln

In der vorigen PHP-Übung wurden zwei Aspekte übergangen, die jedoch beim Schreiben von PHP-Skripten sehr häufig sind:

- Normalerweise müssen Parameter an das PHP-Skript auf der Serverseite gesendet werden.
- Da nun die Clientseite recht gut geschützt ist, sollten Sie auch auf der Serverseite irgendeine Fehlerbehandlung implementieren.

Parameter können Sie an das PHP-Skript entweder mit `GET` oder mit `POST` übergeben. Die Fehlerbehandlung in PHP wird mit einer PHP-spezifischen Technik erledigt. In der nächsten Übung übergeben Sie einem PHP-Skript Parameter und implementieren einen Fehlerbehandlungsmechanismus, den Sie dann mit Bogus-Werten testen werden. Die Anwendung wird aussehen wie in Abbildung 3.3.

Diese Seite sendet einen asynchronen Aufruf an einen Server, in dem er den Server bittet, zwei Zahlen für Sie zu dividieren. Wenn alles klappt, gibt der Server das Ergebnis als XML-Struktur zurück:

```
<?xml version="1.0"?>
<response>1.5</response>
```

Wenn ein PHP-Fehler auftritt, meldet das Serverskript in einfachem Text den Fehler, der vom Client abgefangen wurde, anstatt einen XML-String zu generieren. (Warum es dies tut, werden Sie nach der Übung verstehen).

Showtime: PHP-Parameterübergabe und Fehlerbehandlung

1. Im foundations-Ordner legen Sie jetzt einen neuen Ordner namens morephp an.
2. Erstellen Sie nun im Ordner morephp eine Datei namens morephp.html:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html>
  <head>
    <title>Practical AJAX: PHP Parameters and Fehlerbehandlung</title>
    <script type="text/javascript" src="morephp.js"></script>
  </head>
  <body>
    Ask server to divide
    <input type="text" id="firstNumber" />
    by
    <input type="text" id="secondNumber" />
    <input type="button" value="Send" onclick="process()" />
    <div id="myDivElement" />
  </body>
</html>
```

3. Erzeugen Sie eine weitere Datei namens morephp.js:

```
// speichert eine Instanz von XMLHttpRequest
var xmlhttp = createXmlHttpRequestObject();

// erzeugt eine XMLHttpRequest-Instanz
function createXmlHttpRequestObject()
{
  // speichert die Referenz auf das XMLHttpRequest-Objekt
  var xmlhttp;
  // dies müsste auf allen Browsern außer IE6 und älter funktionieren
  try
  {
    // versucht, ein XMLHttpRequest-Objekt zu erzeugen
    xmlhttp = new XMLHttpRequest();
  }
  catch(e)
  {
    // für IE6 oder älter
    var XmlHttpVersions = new Array("MSXML2.XMLHTTP.6.0",
                                     "MSXML2.XMLHTTP.5.0",
                                     "MSXML2.XMLHTTP.4.0",
                                     "MSXML2.XMLHTTP.3.0",
                                     "MSXML2.XMLHTTP",
                                     "Microsoft.XMLHTTP");

    // probiert Prog-IDs durch, bis eine funktioniert
    for (var i=0; i<XmlHttpVersions.length && !xmlhttp; i++)
    {
      try
      {
        // versucht, ein XMLHttpRequest-Objekt zu erzeugen
        xmlhttp = new ActiveXObject(XmlHttpVersions[i]);
      }
      catch (e) {}
    }
  }
  // gibt das erzeugte Objekt oder eine Fehlermeldung zurück
  if (!xmlhttp)
    alert("Error creating the XMLHttpRequest object.");
  else
    return xmlhttp;
}

// liest eine Datei vom Server
function process()
```