AJAX, eine saubere Sache

Einleitung

Webseiten werden immer interaktiver. Doch ist der Weg zur 100% Interaktivität ist allerdings schwierig. Im Vergleich zu einer GUI-Anwendung, hängt eine Webseite stark hinterher. Wenn eine Webseite dynamische Bereich hat, wird im Normalfall nach einem Userklick, die gesamte Webseite neu geladen. Das dauert lange und man erhält einen unangenehmen Neuaufbau der Webseite. In GUI-Anwendungen wird einfach der entsprechende Bereich ausgetauscht. Das macht sich die Technik AJAX zu nutze. AJAX ist die Abkürzung für Asynchronous JavaScript and XML. Über das Dokument Objekt Model (DOM) wird mit Hilfe von Javascript mehr Interaktivität in die Webseiten gebracht. Einzelne Bereiche werden mit im Hintergrund abgefragten Daten gefüllt. Dadurch braucht die Webseite nicht mehr „refreshed“ werden und das unangenehme Neuaufbauen entfällt.

Das Arbeitstier

Was passiert nun mit AJAX genau? Die Webseite wird von dem Browser normal per HTTP abgerufen und angezeigt. Der Unterschied liegt hier im Detail. Da das A im AJAX für asynchron steht, muss noch etwas im Hintergrund passieren. Während der Benutzer die Formulare in der Webseite ausfüllt, ruf ein weitere Prozess zusätzliche Daten ab. Dieser Prozess wird von dem Javascriptobjekt XMLHttpRequest erfüllt. Dieses Objekt ist zwar noch nicht Standart, aber es wird von den modernen Browsern wie Mozilla, Firefox und Opera unterstützt. Wie immer braucht der Microsoft Internet Explorer eine extra Wurst. Dieser kennt das Objekt nicht. Allerdings gibt es ein ActiveX-Objekt welches die Funktionalität abbildet.

```
If (window.XMLHttpRequest) {
  requestobj = new XMLHttpRequest();
} else if (window.ActiveXObject) {
  requestobj = new ActiveXObject(“Microsoft.XMLHTTP”);
}
```

Der oben abgedruckte Code schaut ob der Browser das XMLHttpRequest-Objekt unterstützt. Wenn er dieses tut, dann wird eine Instanz von dem Objekte erzeugt. Unterstützt der Browser das Objekt nicht, dann wird versucht ob der Browser ActiveX-Objekte unterstützt. Wenn ja, dann wird von Microsoft das XMLHTTP-Objekt erzeugt und in der Variable requestobj abgelegt. Glücklichere Weise sind die Objekte XMLHttpRequest und das XMLHTTP ähnlich implementiert. Somit müssen wir später keine Rücksicht mehr nehmen.

Eigenschaften des Objekts

onreadystatechange	Zeiger auf die Funktion welche aufgerufen wird wenn sich der readyState ändert. (Read/write)
ReadyState	Status des Objektes. 0 : Uninitialisiert 1 : Läd 2 : Geladen aber Header/Status noch nicht bereit

	3 : Interaktiv – Daten sind teilweise geladen 4 : Fertig – Alle Daten sind verfügbar. (Read-only).
responseBody	Represents only one of several forms in which the HTTP response can be returned. Read-only.
responseStream	Represents only one of several forms in which the HTTP response can be returned. Read-only.
responseText	Represents the response entity body as a string. Read-only.
responseXML	Geparstes XML aus der Antwort. Read-only.
status	HTTP Statuskode welcher der Webserver zurück geliefert hat. Read-only.
statusText	Statustext des Webservers. (Read-only).

Methoden des Objekts

abort	Abbruch der aktuellen Abfrage
getAllResponseHeaders	Liefert alle Kopfdaten
getResponseHeader	Liefert den Wert des Serverresponse
open	Initialisiert die Abfrage. Gibt auch die Abfrageart an (Get/Post).
send	Führt die Abfrage aus.
SetRequestHeader	Setzt zusätzliche Headerwerte

Lass es Blitzen

Ein Beispiel sagt meistens mehr als 1000 Worte. Um einfach mal loszulegen folgendes Szenario: Auf einer Webseite ist ein Knopf und diese soll über die Ajax-Technik Daten für eine Tabelle nachladen. Zuerst die Daten welche Abgerufen werden sollen. Selbstverständlich liegen diese im XML-Format vor.

```
<?xml version="1.0" encoding="UTF-8"?>
<Daten>
  <Bank>
    <Name>Landeszentral Bank</Name>
    <BLZ>77667766</BLZ>
  </Bank>
  <Bank>
    <Name>KSK Bonn</Name>
    <BLZ>12345678</BLZ>
  </Bank>
</Daten>
```

Und nun die Webseite mit dem Javascript.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=iso-8859-1">
```

```
    <script language="JavaScript" type="text/javascript">
      var requestobj = null;

      if (window.XMLHttpRequest) {
        requestobj = new XMLHttpRequest();
      } else if (window.ActiveXObject) {
        requestobj = new ActiveXObject('Microsoft.XMLHTTP');
      }

      function AsyncResult() {
        if (requestobj.readyState==4) {
          if (requestobj.status==200) {
            var daten = requestobj.responseXML.getElementsByTagName('Bank')[0];
            var test = document.getElementById('ergebnis');
            test.innerHTML=daten.textContent;
          }
        }
      }

      function DoIt(url) {
        if (requestobj) {
          requestobj.open('GET', url, true);
          requestobj.onreadystatechange=AsyncResult;
          requestobj.send(null);
        }
      }
    </script>
  </head>
  <body>
    <p>
      <span id="ergebnis">text</span>
      <button onClick="DoIt('http://localhost/bank.xml')">doit</button><br>
    </p>
  </body>
</html>
```

Kommen wir nun zu der Erklärung des Beispiels. In dem Javascriptkopf wird versucht, wie schon oben gezeigt, das RequestObjekt zu erstellen. Die Funktion DoIt wird beim Knopfdruck aufgerufen. Der Aufruf instruiert dann das Requestobjekt das es die Datei http://localhost/bank.xml asynchron läd. Bevor das jedoch passiert wird dem Objekt mitgeteilt welche Funktion aufgerufen werden soll wenn sich der Status des Requestobjekts ändert. Wenn sich der Status ändert wird dann die Funktion AsyncResult aufgerufen. Wenn der Request abgeschlossen ist (Status=4) wird der Webserverresponsecode geprüft. Ist dieser 200 (OK) dann wird das XML-Dokument nach den Objekten Bank durchsucht. Das erste Bankelement wird dann in „Daten“ zwischen gespeichert. Nun wird über das Dom dem Dokumentenelement mit der Id „Ergebnis“ der XML Kontent zugewiesen.

Alles Sauber

Im Abschluss möchte ich noch paar Ideen an die Frau bzw. Mann bringen. Es braucht kein statisches XML-File sein welches abgerufen wird. Bezugnehmend auf meinen Vortrag „Dreamteam – Pear und Smarty“ möchte ich den Tipp geben XML-Dateien dynamisch aus einer Datenbank zu erstellen. Selbstverständlich können auch Parameter an ein PHP-Script übergeben werden, welches dann die XML generiert. Diese können dann aus HTML-Formularen generiert werden. Es sind alle Möglichkeiten offen. Alles was mit Javascript im Dom möglich ist, ist erlaubt.

Wofür kann man Ajax nun brauchen?
Ajax kann helfen dem User zur Seite stehen. Das können Eingabeprüfungen sein oder eine Suche welche im Hintergrund durchführt wird. Das Ergebnis wird dann den User noch während der Eingabe präsentiert. Ajax könnte aber auch für ein selbstlernendes Postleitzahlenverzeichnis genutzt werden. Wer sehr viel Aufwand treiben möchte, kann auch eine gesamte Adressenprüfung mithilfe von der Deutschen Post AG Adressen CD durchführen.

Über den Autor

Bernd Ott ist Jahrgang 1975 und seit Mitte der achtziger Jahre ist er im Bereich der Softwareentwicklung tätig. Er arbeitet seit 1995 als festangestellter Softwareentwickler mit den Programmiersprachen Delphi, PHP, Java und C. Seit 1999 betreibt er nebenberuflich ein Einzelunternehmen. Dieses beschäftigt sich mit Unternehmensberatung, Prozesstechnik, Lichttechnik und DTP. In seiner verbleibenden Freizeit ist er Diskjockey (house, techno musik).

Links

http://www.quantendrehung.de/
http://ajaxpatterns.org/
http://developer.mozilla.org/en/docs/AJAX:Getting_Started
http://www.mozilla.org/docs/dom/domref/dom_shortTOC.html
http://www.sitepoint.com/article/remote-scripting-ajax/
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/7924f6be-c035-411f-acd2-79de7a711b38.asp