AJAX – kiedy i jak?

Czyli krótka recepta na to gdzie i w jaki sposób stosować AJAX

Artykuł ma na celu pokazanie przypadków, w których totalne i bezwzględne implementowanie AJAX, gdzie tylko się da, niekoniecznie musi być korzystne dla serwisu, a kiedy jest wręcz niepożądane. Postaram się także w kilku krótkich przykładach omówić najciekawsze, moim zdaniem, podejścia do implementacji AJAX oraz pola ich zastosowań.

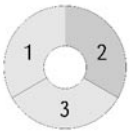
Dowiesz się...

- Jak skutecznie stosować AJAX na różnych rodzajach stron WWW;
- Zapoznasz się również z popularnymi bibliotekami AJAX dla JavaScript.

Powinieneś wiedzieć...

- Znać podstawy programowania obiektowego w JavaScript;
- Posiadać doświadczenie w tworzeniu oprogramowania komercyjnego.

Poziom trudności



Ostatnio bardzo popularna technologia AJAX (ang. *Asynchronous JavaScript* and *XML*) rewolucjonizuje świat witryn internetowych. Coraz więcej stron i serwisów internetowych z powodzeniem stosuje tę technologię, kusząc użytkowników pięknym interfacem, brakiem przeładowań strony i wizualną płynnością ładowania danych. Niewątpliwie pomysł na wykorzystanie obiektów XMLHttpRequest z JavaScript (czy innych języków skryptowych Jscript, czy VBScript), przeżywa swoją świetność i jest czymś, bez czego nowoczesny projekt wygląda, co najmniej blado, na tle pełnych gadżetów produktów konkurencji. Cechą charakterystyczną XMLHttpRequest, jest możliwość wykonywania żądań do serwera już po załadowaniu strony internetowej w trakcie interakcji z użytkownikiem. Otrzymanych odpowiedzi używamy do dynamicznej modyfikacji wyświetlanej treści. Developerzy niejednokrotnie stają w obliczu bezpośrednich żądań klienta o to, by aplikacja wyposażona była w tego typu mechanizmy. Stały się one nieomal koniecznością wynikającą z wysokich wymagań klienta. Niestety – programista najczęściej przystaje na sugestie i otrzymujemy strony bez przeładowań, których ocena oglądalności przy pomocy standardowych narzędzi staje się utrudniona.

Środowisko działania serwisu

Pierwszym pytaniem, na jakie należy sobie odpowiedzieć, jest – jaki serwis tworzymy w PHP? Z uwagi na to określamy parametry techniczne środowiska pracy naszego projektu, wymagania dotyczące dopuszczalnych transferów, oczekiwań użytkowników końcowych oraz przewidywanego czasu odpowiedzi serwera. Wyróżniłbym trzy typy, z jakimi możemy się spotkać, jeżeli Czytelnik uzna, iż jakiegoś brakuje – na pewno uda się go dopasować na podstawie opisu środowiska, do którejś z grup: intranet lub extranet, pojedyncza strona WWW lub blog (czy inne, niemasowe treści), portal lub serwis.

Sieci wewnętrzne i korporacyjne

W przypadku intranetu lub extranetu mamy do czynienia ze środowiskiem oferującym nam bardzo wiele i takim, które *wiele wybacza* od strony optymalizacji transferów klient–serwer, konieczności rejestracji statystyk, precyzji określenia parametrów technicznych klientów – rodzaj przeglądarki, system operacyjny. W tego typu aplikacjach końcowy użytkownik liczy na maksymalny *performance1* naszego systemu. Od końcowego użytkownika w najlepszym przypadku możemy oczekiwać biegłej obsługi pakietów biurowych, dlatego wyświetlanie danych, które co jakiś czas znikają, po to by strona się przeładowała, raczej ich nie zachwyci – klient bardzo doceni zastosowanie AJAX do stopnia, przy którym nasz system oparty o WWW nie będzie się wizualnie odróżniał od innych aplikacji zainstalowanych na stacji roboczej. Osobne okno bez pasków nawigacyjnych przeglądarki i dobrze oprogramowanej obsługi zdarzeń spowoduje, iż jeszcze bardziej zatrzemy różnicę. Zarówno jednolite środowisko działania, jak i pole eksploatacji sprzyja AJAX. Brak konieczności rejestrowania statystyk odwiedzin również działa na korzyść AJAX–a, gdyż technologia ta zaburza nam działanie narzędzi analitycznych, rejestrujących ruch w serwisie i musimy mieć to na uwadze przede wszystkim w trzecim z omawianych typów serwisów.

Zwykła strona

W drugim przypadku przygotowując pojedynczą stronę WWW, blog czy inny serwis dla niezbyt dużej ilości użytkowników, przy czym użytkownik końcowy to zwykły websurfer, już można poważnie się zastanowić nad elementami tego typu. Należy sprecyzować, że mówimy o tym, co ogląda użytkownik końcowy. Jeżeli przygotowywana przez nas strona zawiera CMS lub podobne narzędzia – bliższe będą one grupie pierwszej. Pojedyncze strony, nawet jeżeli zawierają sporą ilość treści, na ogół ciekawiej jest wzbogacić grafiką, animacją, obiektami, Flash, ciekawym menu, niż inwestować czas w dużą ilość gadżetów AJAX–owych. Jeżeli taka pojedyncza strona zawiera elementy jakiś dużych tabel danych, obrazów, krótkich tekstów czy opisów – warto zastosować AJAX do pojedynczych elementów położonych gdzieś na stronie, ale raczej tylko wtedy, gdy zrobi to znaczącą różnicę w stosunku do tradycyjnego podejścia, czy wpłynie na estetykę i użyteczność.

Duże serwisy

W końcu w trzeciej grupie – serwisów internetowych lub portali – zdecydowanie odradzałbym używania AJAX do całości zawartości, a nawet nie zawsze do drobnych kontrolek. Dlaczego? Powodów jest kilka. Pierwszym z nich jest kwestia kompatybilności. Duży serwis to duża ilość użytkowników i różnorodność przeglądarek, systemów operacyjnych, konfiguracji, a nawet samych urządzeń. W spo-

Listing 1. *Przykład inicjacji obiektu XML HTTP Request*

```
var xmlhttp=false;
/*@cc_on @*/
/*@if (@_jscript_version >= 5)
// JScript daje nam możliwość warunkowej kompilacji
  try {
    xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
  } catch (e) {
    try {
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (E) {
    xmlhttp = false;
  }
}
@end @*/

if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
  try {
    xmlhttp = new XMLHttpRequest();
  } catch (e) {
    xmlhttp=false;
  }
}
if (!xmlhttp && window.createRequest) {
  try {
    xmlhttp = window.createRequest();
  } catch (e) {
    xmlhttp=false;
  }
}
```

Listing 2. *Przykład implementacji żądania przy pomocy obiektu XML HTTP Request*

```
xmlhttp.open("GET", "test.php",true);
xmlhttp.onreadystatechange=function() {
  if (xmlhttp.readyState==4) {
    alert(xmlhttp.responseText)
  }
}
xmlhttp.send(null)
```

sób szczególny zwrócić uwagę należy na sferę urządzeń – gdyż ilości użytkowników oglądających strony przy pomocy telefonów komórkowych, palmtopów i podobnych urządzeń mobilnych ciągle rośnie. Przeszłość wiele nauczyła twórców o urokach JavaScript i pozostałych języków skryptowych oraz różnych związanych z tym trudnościach – po prostu zastosowanie zbyt wyrafinowanych rozwiązań może okazać się złym pomysłem w zderzeniu z rzeczywistością. Doświadczony programista użyje rozmaitych trików, aby zapewnić serwisowi działanie w sposób tradycyjny w przypadku braku dostępności wyrafinowanych narzędzi po stronie klienta, jednakże należy liczyć się tym samym z koniecznością podwójnej implementacji tych samych rozwiązań. Jeżeli termin nas goni, trzeba wziąć powyższe czynniki pod uwagę w fazie projektowania i planowania.

Warto AJAX zastosować do elementów, które możemy szybko wykonać i zabezpieczenie ich funkcjonalności nie będzie kłopotliwe, a w skrajnej sytuacji (do której dobry programista nie powinien dopuścić), nie odbierze serwisowi funkcjonalności. Elementy, jakie bym tutaj wymienił to: wszelkiego rodzaju oceny (selekcja liczbowa, *gwazdki* czy inne graficzne znaczniki), krótkie komentarze, niektóre *wybieraki*, rozwijane drzewa, krótkie ankiety, *shoutbox'y*, niektóre proste formy wyszukiwania, operowanie na elementach takich jak zamieszczone w serwisie *Goggle Maps*. Podejmując decyzję, które z elementów chcemy wykonać w AJAX trzeba osiągnąć kompromis pomiędzy korzystnym zmniejszeniem transferu, poprawieniem użyteczności, a ilością pracy, jaką będziemy musieli zainwestować oraz negatywnymi skutkami dla tradycyjnych statystyk strony.

Powód, który jest z mojego punktu widzenia najważniejszy, a stawia pod znakiem zapytania wszechobecne używania AJAX, dotyczy potencjalnych dochodów z serwisu – im więcej przeładowań tym więcej odsłon reklam, a im więcej odsłon reklam tym więcej w nie kliknięć. Istnieją ciekawe rozwiązania na rzecz wymiany reklam

bez przeładowań (np. *http://ajax.phpmagazine.net/2006/02/rotating_ads_with_ajax.html*), ale każdy zauważy, że w ten sposób nie osiąga się oczekiwanego przez klienta rezultatu. Klient, jeśli zamawia u nas duży serwis publiczny, chce na nim zarobić – ograniczając przeładowania (te odbywające się w tradycyjny sposób), obcinamy mu potencjalne dochody i powodujemy nie lada kłopot w interpretacji statystyk dostarczonych przez *Webalizer* czy nawet *Goggle Anaylics* (chociaż to drugie wykazuje pewną odporność przy odpowiedniej konfiguracji). Z drugiej strony użytkownicy końcowi często poirytowani są częstymi przeładowniami – znów musimy znaleźć kompromis.

Odrobina sceptycyzmu, analityczny koszmar

Istnieje szereg powodów, dla których używanie AJAX musi być przemyślane, zważywszy na statystyki. Szerokie grono analityków zauważyło komplikację w sposobach mierzenia popularności strony ze względu na nowe elementy i jasne jest, że miara liniowa oglądalności strony w stosunku do ilości wejść, odsłon, itp. zdezaktualizowała się w obliczu nowych technologii. Konkurencyjność serwisów wymaga dostarczania jeszcze większej ilości informacji na jednej stronie i ciągłej interakcji z użytkownikiem dla realizacji postulatów WEB2.0. Często chybionym przybliżeniem jest uznanie, że wybór informacji przez użytkownika wiąże się z przeładowaniem. Otwiera się tutaj szerokie pole dla nowych pomysłów na badanie popularności serwisu, jednak nie ma w tym zakresie wyznaczonych żadnych standardów. Co jakiś czas pojawiają się pomysły oparte o mierzenie użytkowania widgetów – czyli elementów składowych strony, lecz jak na razie nie znalazły one globalnego poparcia w sensie sformalizowanego standardu, który pomógłby nam za pomocą zewnętrznej aplikacji zbierać dane analityczne – możemy co najwyżej zaimplementować to we własnym zakresie. Wciąż w zastosowaniach uniwersalnych zmuszeni jesteśmy poruszać się w miarach aktualnie stosowanych, a na ich wyniki AJAX nie wpływa pozytywnie. W sieci znajdują się dwa dość ciekawe artykuły budzące uzasadniony dystans wobec wszechobecnego entuzjazmu dla AJAX: *Is AJAX The Page View Killer?* autorstwa Sean'a Michael'a Kerner: *http://www.internetnews.com/xSP/article.php/3666656* oraz kontrowersyjny i wymagający na dziś drobnego uaktualnienia (lecz nadal ciekawy) *Why Ajax Sucks (Most of the Time)* autorstwa Jakob'a Nielsen'a: *http://www.usabilityviews.com/ajaxsucks.html*. W dalszej części przedstawię odrobinę podejścia praktycznego do stosowania AJAX.

Rysunek 1. *Jedno z najczęstszych zastosowań AJAX przy weryfikacji logowania użytkownika, na przykładzie systemu Google Analytics*

Wszystko w AJAX

W aplikacjach z grupy pierwszej, które rozwijamy najczęściej na bazie gotowych kompleksowych frameworków, przy odpowiednim doborze być może istnieje możliwość, iż nie będziemy musieli znacząco się napracować, aby zastosować AJAX. Przykładem wybranym przeze mnie jest popularny framework PRADO, który swoją prostotą *uruchomienia* AJAX moim zdaniem wyróżnia się znacząco od innych.

W *PRADO* po utworzeniu formularza, jeżeli chcemy, aby był on *AJAX-owy* po prostu zmieniamy użyte kontrolki z np. `TButton`, na `TActiveButton` – i to wystarczy! Dokumentacja *PRADO* prezentuje nam łatwość i skuteczność tego rozwiązania za pomocą prostego przykładu konwertera walut *http://www.pradosoft.com/demos/currency-converter/* . Więcej o samym użyciu tzw. *Active Controls* Czytelnik znajdzie pod adresem *http://www.pradosoft.com/demos/quickstart/index.php?page=ActiveControls.Home*. Poza samym bardzo elegancko przygotowanym mechanizmem implementacji, PRADO oferuje nam również wsparcie dla obiektów DOM, komunikacji obiektowej pomiędzy JavaScript, a PHP oraz wiele innych interesujących mechanizmów. Skorzystanie z pełnego wachlarza rozwiązań wymaga poświęcenia sporej ilości czasu na naukę, ale jeżeli chcemy wyciągnąć maksimum wydajności na rzecz użytkownika końcowego, zwłaszcza przy aplikacjach z dużą ilością formularzy i danych, mających zachowywać się, jak *desktopowe* – jest to dobre rozwiązanie.

Małe kontrolki

W przypadku projektów z grupy drugiej i de facto trzeciej, implementujemy małe kontrolki, jak oceny (*ratings*), komentarze, soutbox'y, krótkie teksty czy inne podobne. Nie potrzebujemy więc dużych objętościowo i funkcjonalnie frameworków, kompleksowych rozwiązań do wszystkiego. Najczęściej nasze zastosowanie AJAX będzie się sprowadzało do wysłania kilku wartości do bazy danych i odebrania krótkiego rezultatu, czy to w postaci komunikatu, czy jakiegoś kodu HTML, dlatego zastosujemy pojedynczą biblioteczkę do komunikacji albo też skorzystamy bezpośrednio z obiektu `XMLHttpRequest`, dostępnego w JavaScript, także w JScript i VBScript. Przedstawione przykłady przygotowane są w JavaScript i w dalszej część artykułu skupię się właśnie na implementacjach w tym języku.

Rysunek 2. *Popularne zastosowanie AJAX do szybkiego wysyłania ocen użytkowników na przykładzie popularnego serwisu Youtube*

XMLHttpRequest

Bezpośrednie użytkowanie `XMLHttpRequest` (XHR) może być odrobinę kłopotliwe, jednakże wiele osób z powodzeniem używa bezpośrednio tego obiektu. Obiekt XHR nie ma ograniczeń do wyłącznie plików XML, może żądać albo przesyłać każdy typ dokumentów, jednakże działanie na danych binarnych jest problematyczne. Jak zazwyczaj nazwa obiektu zależy od nazwy przeglądarki. Pod Internet Explorer tworzymy obiekt używając `new ActiveXObject("Msxml2.XMLHTTP")` lub `new ActiveXObject("Microsoft.XMLHTTP")` w zależności od wersji MSXML, która jest zainstalowana (wersji tej przeglądarki). W produktach Mozilli i Safari powinniśmy używać `new XMLHttpRequest()`. Dodatkową ciekawostką jest, że przeglądarka IceBrowser, wymaga jeszcze innej metody: `window.createRequest()`.

W praktyce oznacza to, że do każdej przeglądarki musimy używać innej metody, gdyż to, co działa w jednej spowoduje błędy w drugiej. Kod na Listingu 1. prezentuje przykład inicjacji obiektu z uwzględnieniem konieczności dywersyfikacji. Na Listingu 2. mamy prosty przykład jak pozyskać zawartość pliku test.php. XHR daje nam najszerszą z możliwych kontrolę nad wysyłanymi zapytaniami i ich treścią – łącznie z kontrolą nagłówków i pozyskiwaniem wielu danych jak Content–Type, czy Last–Modified. Aby szczegółowo zapoznać się z możliwościami używania tego obiektu, zachęcam do przeczytania artykułu autorstwa Jima Leya pod adresem *http://www.jibbering.com/2002/4/httprequest.html*. Autor w sposób dość wyczerpujący, na przykładach, prezentuje przykłady użycia tego obiektu.

`XMLHttpRequest` to rzeczywiście bezpośrednie użytkowanie natywnych metod obiektu JavaScript, który po przyjrzeniu się bardziej skomplikowanym przykładom trochę odstrasza, dlatego warto pokusić się o użycie jakiejś małej, lecz efektywnej biblioteczki. Mi bardzo bliskie są dwa dosyć popularne rozwiązania AJAX–owe, z resztą naszego rodzimego pochodzenia. Pierwsze to *advAJAX*, a drugie to *mintAjax*.

advAJAX

AdvAJAX był już wielokrotnie opisywany i jest bardzo ceniony przez osoby, które go używają. Prostota użycia, łatwość operowania na większej ilości przesyłanych zmiennych i wygodny sposób niemal zdarzeniowego programowania czynią z tej biblioteki bardzo wygodne narzędzie. Listing 3. zawiera przykład implementacji funkcji przesyłającej dane z formularza z Listingu 4. Jak widać na przykładzie, mamy do dyspozycji szereg zdarzeń związanych z transmisją. Biblioteka *advAjax* jest narzędziem godnym polecenia osobom początkującym, jak i z doświadczeniem. Więcej dowiedzieć się można na stronie projektu *http://advajax.anakin.us/index-pl.htm*.

mintAjax

Sama koncepcja *mintAjax* jest zbliżona do działania *advAJAX*, jednakże nie należy dać się zwieść. *MintAjax* to odrobinę świeższe spojrzenie na temat. Na Listingu 5. znajduje się wzorcowy przykład implementacji funkcji wysyłającej dane formularza z Listingu 4. Zachęcam Czytelnika do analizy kodu, a więcej przykładów porównawczych znaleźć można na stronie projektu w wyjątkowo rzeczowej i dobrze przygotowanej dokumentacji *http://mintajax.pl/Przewodnik/*. Myślę, że jednym z ciekawszych pomysłów zastosowanych w *mintAjax* jest funkcja `$`, która stanowi ciekawy pomysł i spore ułatwienie. Chodźby z uwagi na funkcję `$` warto przyjrzeć się *mintAjax* bliżej.

Listing 3. *Przykład implementacji funkcji wysyłającej formularz o id=„moj_forumlarz" przy użyciu advAjax*

```
function wyslijDane () {
  var moj_formularz = document.getElementById("moj_formularz");
  advAJAX.submit( moj_formularz, {
    url: ( "skrypt_obslugi_formularza.php" ),
    onSuccess : function(obj) {
      alert(obj.status);
    },
    onError : function(obj) {
      alert("Error: " + obj.status + ' ' + url);
    }
  });
}
```

Listing 4. *Przykład formularza o id=„moj_forumlarz"*

```
<form id="moj_formularz" method="POST" action="form_post.php">
  Pole tekstowe:
    <input name="text" type="text" /><br />
  Checkbox:
    <input name="checkbox" type="checkbox" /><br />
  Radio:
    <input name="radio" type="radio" value="Lorem" checked /> Lorem
    <input name="radio" type="radio" value="Ipsum" /> Ipsum
</form>
<div id="response"></div>
<button onclick="wyslijDane()">Wyślij formularz</button>
```

Listing 5. *Przykład implementacji funkcji wysyłającej formularz przy użyciu mintAjax*

```
function wyslijDane () {
  var req = mint.Request();
  req.OnSuccess = function() {
    $("response").innerHTML = this.responseText;
  }
  req.SendForm("moj_formularz");
}
```

Podsumowanie

Mam nadzieję, że ten krótki artykuł pomoże właściwie, pomysłowo i efektywnie wykorzystywać technologię AJAX. Niewątpliwie AJAX jest czymś niezbędnym w nowoczesnym webdesignie, a zastosowanie tej technologii na pewno doceni użytkownik końcowy. Dosyć rozbudowany wstęp teoretyczny daje podstawę do właściwego wyboru jednego z zaproponowanych rozwiązań. Poza zaprezentowanymi istnieją setki innych, dlatego Czytelnik, jeżeli nie będą mu odpowiadać zaprezentowane tu przykłady, bez problemu znajdzie w sieci bibliotekę AJAX w sam raz dla siebie, choćby przeglądając stronę *http://ajaxpatterns.org/*. A jeżeli nie – to już wie choćby z tego artykułu, że może stworzyć ją sam na bazie `XMLHttpRequest` – do czego gorąco zachęcam, gdyż daje to najszerszy pogląd na działanie AJAX i inspiruje do znajdowania coraz ciekawszych zastosowań.

MACIEJ WIŚNIEWSKI

Maciej Wiśniewski jest szefem iCEAM.com Webdesigner Team oraz studentem Wyższej Szkoły Informatyki Stosowanej i Zarządzania WIT w Warszawie, na specjalizacji inżynieria oprogramowania. Od 2001 roku profesjonalnie zajmuje się tworzeniem aplikacji, stron i serwisów internetowych dla dużych, małych i średnich firm.

Kontakt z autorem: *maciej.wisniewski@iceam.com*