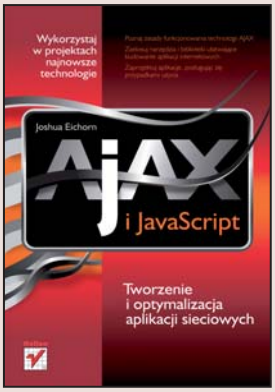
AJAX i JavaScript. Tworzenie i optymalizacja aplikacji sieciowych



Autor: Joshua Eichorn
Tłumaczenie: Tomasz Walczak
ISBN: 978-83-246-1098-3
Tytuł oryginału: Understanding AJAX: Using JavaScript to Create Rich Internet Applications
Format: B5, stron: 336
Przykłady na ftp: 691 kB

Wykorzystaj w projektach najnowsze technologie

- Poznaj zasady funkcjonowania technologii AJAX
- Zastosuj narzędzia i biblioteki ułatwiające budowanie aplikacji internetowych
- Zaprojektuj aplikacje, posługując się przypadkami użycia

Rosnąca popularność internetu sprawiła, że jego użytkownicy stawiają witrynom WWW i aplikacjom sieciowym coraz wyższe wymagania. Nie wystarczy już przygotować efektowny projekt graficzny i atrakcyjną treść. Dziś liczy się szybkość działania, bezpieczeństwo i wygoda obsługi. Ratunkiem dla programistów poszukujących kolejnych metod usprawnienia działania swoich aplikacji jest technologia AJAX. To niesamowite połączenie języka JavaScript i XML pozwoliło wyeliminować największą wadę aplikacji i witryn WWW, czyli konieczność przeładowywania strony po każdej zmianie zawartości. AJAX realizuje proces przeładowania danych w tle, w sposób niezauważalny dla użytkownika. Oczywiście, to nie jedyna zaleta technologii AJAX – jest ich znacznie więcej.

Czytając książkę „AJAX i JavaScript. Tworzenie i optymalizacja aplikacji sieciowych", odkryjesz wszystkie zalety tej technologii i dowiesz się, jak projektować i pisać wydajne, bezpieczne oraz ergonomiczne aplikacje WWW. Nauczysz się korzystać z żądań asynchronicznych, przetwarzać pobrane dane i rozbudowywać istniejące aplikacje tak, aby zastosować AJAX. Poznasz proces projektowania i tworzenia aplikacji sieciowych z wykorzystaniem przypadków użycia, zadbasz o wygodę obsługi i zdiagnozujesz oraz rozwiążesz problemy z kodem AJAKSA. W książce znajdziesz także omówienie bibliotek i narzędzi, dzięki którym proces budowania aplikacji przyspieszysz i usprawnisz.

- Obiekty XMLHttpRequest
- Żądania asynchroniczne
- Modyfikowanie aplikacji pod kątem AJAKSA
- Zwiększanie użyteczności witryn internetowych
- Przetwarzanie danych zwracanych do aplikacji ajaksowych
- Korzystanie z bibliotek Sarissa i Scriptaculous
- Przyspieszanie wyświetlania danych
- Formularze logowania oparte na AJAKSIE
- Biblioteki AJAKSA dla PHP, Javy, C# i DHTML

Powiększ swój arsenał narzędzi o technologię AJAX i popraw komfort swojej pracy oraz jakość aplikacji!

Wydawnictwo Helion
ul. Kościuszki 1c
44-100 Gliwice
tel. 032 230 98 63
e-mail: helion@helion.pl

Spis treści

Rozdział 5.

Optymalne wykorzystywanie możliwości AJAX-a

AJAX daje mnóstwo nowych możliwości, jednak aby osiągnąć korzyści, trzeba skoncentrować się na głównych celach. W rozdziale opisuję pewne ogólne cele, które programiści sobie stawiają, stosując AJAX-a, a także pokazuję, jak sprawdzić, czy udało się zrealizować zamierzenia. Z tym zagadnieniem wiąże się także określanie, w jaki sposób postawione cele pozwalają usprawnić bieżące aplikacje. Jest to możliwe poprzez porównanie programu sieciowego utworzonego za pomocą zwykłych technik z aplikacją bazującą na AJAX-ie. Jeśli sam AJAX nie spełnia wszystkich wymagań, trzeba czasem zgodzić się na pewne ustępstwa i zastosować go wraz z innymi technologiami. Bierzmy się do pracy!

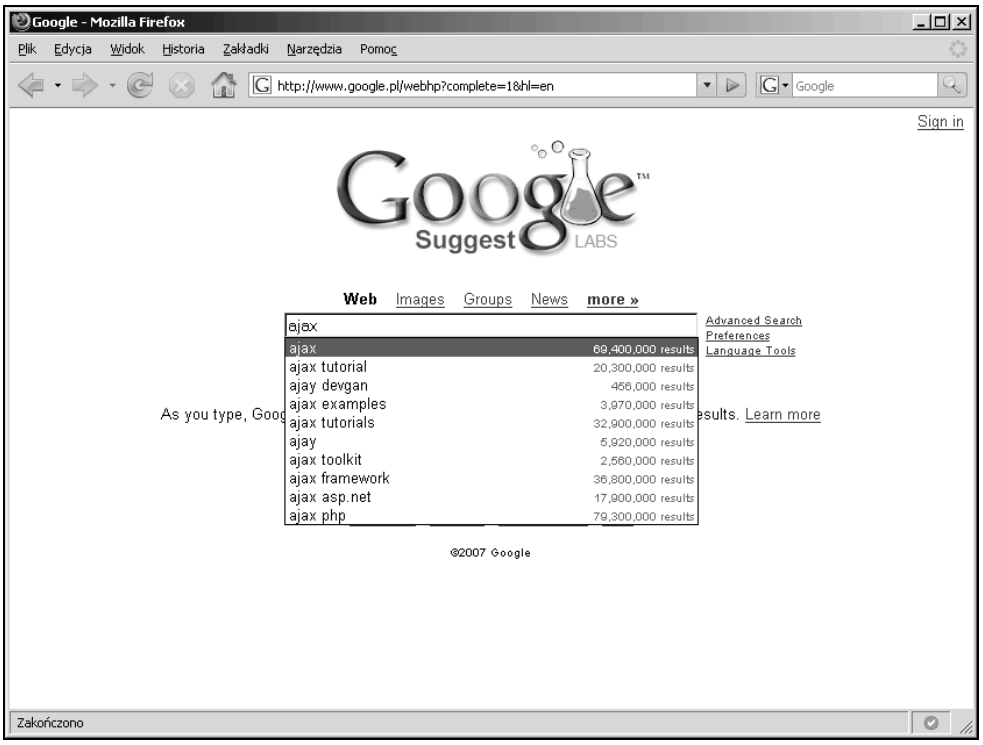
5.1. Cele stosowania AJAX-a

Pierwszym i najważniejszym celem stosowania AJAX-a jest poprawa komfortu pracy użytkownika. Usprawnienia w tej dziedzinie można podzielić na dwie kategorie: ułatwienie wykonywania aktualnych zadań oraz umożliwienie przeprowadzania wcześniej nieobsługiwanych operacji. Oczywiście łatwiej jest skoncentrować się na ułatwianiu bieżących zadań. W obszarze programowania aplikacji sieciowych można podzielić tę kategorię na dwie dalsze grupy: zwiększanie interaktywności i zmniejszanie czasu potrzebnego na wykonanie zadania. Jeśli aplikacja ma działać poza intranetem, ważny może być też techniczny cel w postaci zmniejszenia obciążenia łączy. Przesyłając do przeglądarki mniejsze ilości danych, można skrócić czas wczytywania stron i poprawić ogólny komfort pracy użytkowników.

5.1.1. Zwiększanie interaktywności

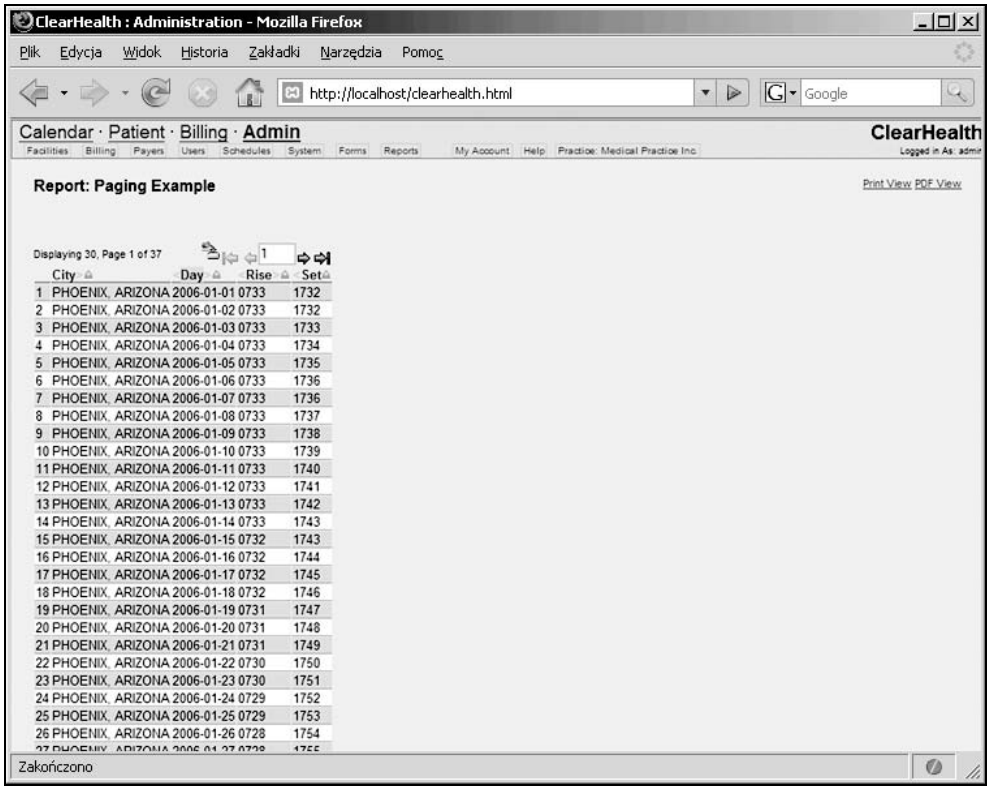
Jednym z ogólnych celów dodawania dowolnej ilości kodu JavaScript do witryny internetowej jest zwiększenie jej interaktywności. Nawet bez AJAX-a można udostępniać informacje związane z treścią, kiedy użytkownik umieści kursor myszy nad odnośnikiem, lub sprawdzić poprawność formularza bez konieczności odświeżania strony. Dodatkowa interaktywność pozwala przekazać więcej informacji bez jednoczesnego przytłaczania użytkowników danymi. Za pomocą AJAX-a można bazować na ogólnym procesie udostępniania danych, zamiast koncentrować się na dodawaniu jedynie następnych statycznych informacji. Inaczej mówiąc — można wyświetlać dodatkowe dane dynamicznie.

Dobry przykład zwiększania interaktywności za pomocą AJAX-a to dodanie funkcji wyszukiwania w czasie rzeczywistym do standardowego formularza wyszukiwania na witrynie internetowej. Jedna z takich aplikacji to Google Suggest (*www.google.com/webhp?complete=1&hl=en*), która sugeruje możliwe wyrażenia za pomocą listy rozwijanej wyświetlanej w trakcie wpisywania zapytania. Lista ta zawiera także liczbę wyników wyszukiwania powiązanych z danym wyrażeniem. Na rysunku 5.1 przedstawione jest wyszukiwanie w Google Suggest pojęcia „AJAX". Podobnej techniki można użyć w dowolnej aplikacji wyszukującej informacji. Zastosowania tego mechanizmu są bardzo szerokie — od wyboru użytkownika, dla którego należy zmienić uprawnienia, po wybór docelowego miasta dla danej przesyłki.



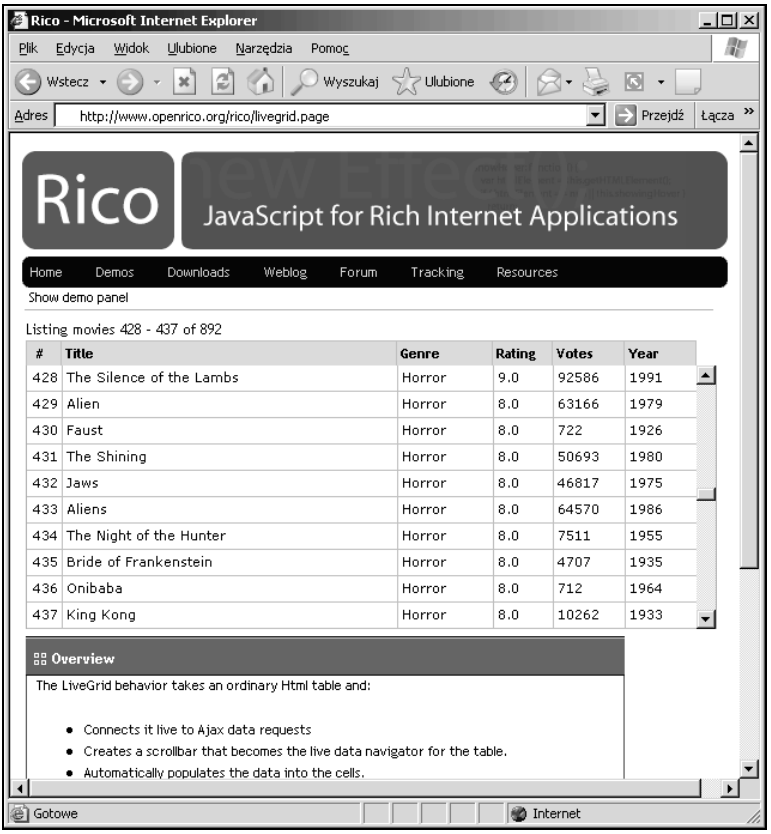
Rysunek 5.1. *Używanie strony Google Suggest do wyszukiwania pojęcia AJAX*

Można także użyć AJAX-a do zwiększenia interaktywności strony w inny sposób, niż usprawniając wyszukiwanie. Jedną z możliwości jest zastosowanie paska przewijania do poruszania się po stronie z wynikami, zamiast udostępniania sieciowej techniki w postaci odnośników do następnej strony. AJAX działa dobrze dla elementów tego typu, ponieważ dane są pobierane wtedy, kiedy są potrzebne (podobnie jak w przypadku zwykłych tabel), ale dostęp do następnych wierszy jest dużo szybszy. Rysunek 5.2 przedstawia standardową kontrolkę do przełączania stron, podczas gdy na rysunku 5.3 widać tabelę z paskiem przewijania AJAX-a. Strona widoczna na rysunku 5.3 umożliwia także sortowanie kolumn bez konieczności wczytywania strony. Ponadto do tabeli można dodać filtry bazujące na AJAX-ie, co pozwala przeglądać dane w szybki i naturalny sposób.



Rysunek 5.2. *Standardowa kontrolka do przełączania stron na witrynie internetowej. Każdy odnośnik powoduje ponowne odświeżenie strony*

AJAX otwiera wiele nowych możliwości zwiększania interaktywności, ponieważ w razie potrzeby można wczytać i wyświetlić dodatkowe dane. Jest to szczególnie przydatne w zbiorach danych średniego rozmiaru, ponieważ można wtedy wyświetlać wszystkie informacje bez zwiększania czasu wczytywania oryginalnej strony lub potrzeby ponownego odświeżenia w celu wyświetlenia danych. Największy problem ze zwiększaniem interaktywności polega na tym, że trudno jest ją zmierzyć, dlatego jej zwiększenie jest najbardziej przydatne przy okazji zajmowania się drugim celem — zmniejszaniem czasu potrzebnego na wykonywanie operacji.



Rysunek 5.3. *Bazująca na AJAX-ie przewijana tabela, w której dane są wczytywane w wyniku przeciągnięcia paska przewijania*

5.1.2. Zmniejszanie czasu potrzebnego na wykonywanie operacji

Jedną z największych wad aplikacji sieciowych jest to, że każdy wieloetapowy proces trwa przez wiele minut. W przypadku szybkiego połączenia każde odświeżenie strony to dwie do pięciu sekund samego oczekiwania na wygenerowanie nowej strony i pobranie jej przez przeglądarkę, a jeśli łącze jest wolne, czas ten może być dwu- lub trzykrotnie dłuższy. Używanie AJAX-a do pobierania nowych danych pozwala uniknąć odświeżania stron, co umożliwia płynną pracę z krótkimi jedno- lub dwusekundowymi okresami oczekiwania na pobranie dodatkowych danych.

AJAX umożliwia skrócenie czasu trwania procesów także w wielu innych sytuacjach. Obejmują one wykonywanie zadań w wieloetapowych kreatorach czy przeglądanie oraz aktualizowanie danych dostępnych w internecie. Po wykryciu operacji, która zajmuje dużo czasu — może to być na przykład moderowanie forum pomocy technicznej — należy poszukać konkretnych najdłuższych zadań. W przypadku moderowania forum problem polega na tym, że każde odświeżenie strony zajmuje dużo czasu, ponieważ moderator może wyświetlać 20, a nawet 100 wiadomości jednocześnie. Modyfikowanie wiadomości wymaga jednego odświeżenia w celu rozpoczęcia edycji oraz

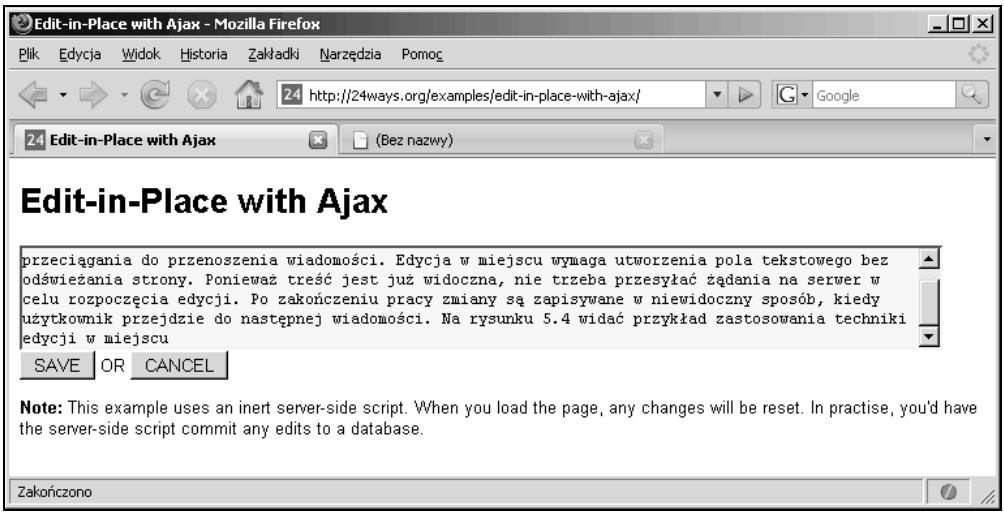
drugiego, aby zapisać zmiany. Jest to proces kosztowny. Inne zadania, takie jak przenoszenie wiadomości, także wymagają dużo czasu, ponieważ po każdym odświeżeniu strony moderator może zostać przeniesiony w inne miejsce na liście wiadomości.

Wyobraź sobie system rezerwowania sal konferencyjnych w dużej korporacji. Po wybraniu sali trzeba znaleźć każdego uczestnika zebrania i dodać go do listy, aby można wysłać mu wiadomość z powiadomieniem o spotkaniu. Ponieważ korporacja zatrudnia ponad 100 pracowników, używanie listy rozwijanej z nazwiskami nie jest dobrym rozwiązaniem. Jej użycie znacznie zwiększyłoby czas wczytywania strony, ponieważ trzeba wstępnie pobrać duże ilości danych. Ponadto niezwykle długa lista byłaby także nieporęczna w użyciu.

Rozwiązanie problemu z wybieraniem osób z czasów sprzed wprowadzenia AJAX-a polegało na dodaniu systemu wyszukiwania, który pozwalał znaleźć każdego pracownika. Taki system można nawet wyświetlać w oknie wyskakującym, aby zmniejszyć ilość odświeżanych danych, jednak niezależnie od zastosowanej techniki dodanie każdej osoby wymaga od 5 do 30 sekund. Ten niewygodny interfejs nie stanowi problemu, jeśli trzeba dodać jedną lub dwie osoby, jednak staje się nie do przyjęcia przy dodawaniu 20 lub więcej pracowników. Podejście bazujące na AJAX-ie pozwala na wyszukiwanie w czasie rzeczywistym. Interfejs może wyglądać podobnie jak Google Suggest na rysunku 5.1 i wyświetlać nazwiska pracowników zamiast szukanych pojęć. W tym przypadku dzięki zastosowaniu AJAX-a dodanie 20 pracowników zajmie minutę zamiast pięciu.

Za pomocą AJAX-a można przyspieszyć proces, dodając możliwość edycji w miejscu i używając techniki przeciągania do przenoszenia wiadomości. Edycja w miejscu wymaga utworzenia pola tekstowego bez odświeżania strony. Ponieważ treść jest już widoczna, nie trzeba przesyłać żądania na serwer w celu rozpoczęcia edycji. Po zakończeniu pracy zmiany są zapisywane w niewidoczny sposób, kiedy użytkownik przejdzie do następnej wiadomości. Na rysunku 5.4 widać przykład zastosowania techniki edycji w miejscu. Przenoszenie wiadomości za pomocą przeciągania także pozwala zaoszczędzić wiele czasu, ponieważ dużo łatwiej jest zobaczyć docelową lokalizację na zwykłej stronie z wiadomościami niż na liście ich tytułów, z której trzeba korzystać, jeśli nie można użyć AJAX-a.

Jedną z przyczyn, dla których skrócenie czasu wykonywania zadań do doskonały cel przy dodawaniu AJAX-a, jest to, że bardzo łatwo je zmierzyć. Wystarczy usiąść i wykonać wybrane operacje, zliczając ilość czasu, jaki zajmują. W przypadku niektórych zadań można nawet dodać do aplikacji zegary i zarejestrować dane w trakcie normalnego użytkowania programu. Po uzyskaniu wyjściowych liczb można określić specyficzne miejsce wdrożenia AJAX-a. W połączeniu z dalszymi testami po dodaniu AJAX-a można uzyskać dane pozwalające ocenić wartość rozszerzeń. Dzięki praktycznym i powtarzalnym pomiarom ocena efektywności AJAX-a nie polega na zgadywaniu, a na prostych obliczeniach. Można nawet użyć miar czasu wykonywania zadań do usprawnienia AJAX-a i zmienić stosowane techniki lub dodać wstępne pobieranie, aby skrócić dany proces.



Rysunek 5.4. *Edycja w miejscu*

5.1.3. Zmniejszanie obciążenia łączy

W niektórych zastosowaniach AJAX-a użytecznym celem może być zmniejszenie obciążenia łączy, ponieważ mniejsza ilość danych wymaga krótszego przesyłania, co prowadzi do bardziej płynnej pracy użytkowników. Jeśli musisz płacić za hosting, zmniejszenie obciążenia łączy może być także skutecznym sposobem na zaoszczędzenie pieniędzy. Jednak jeśli korzystasz z aplikacji w sieci wewnętrznej, ten cel może być nieistotny, ponieważ w przypadku szybkich sieci czas wczytywania jest niski niezależnie od ilości przesyłanych danych.

Pomiar obciążenia łączy jest prosty. Zawsze lepiej jest używać odpowiednich pomiarów niż subiektywnych ocen. Oczywiście w przeciwieństwie do czasów wykonywania zadań obciążenie łączy nie wyraża się liczbą, którą zawsze warto zmniejszyć. Ograniczenie ilości przesyłanych danych w trakcie wczytywania wyjściowej strony może być użyteczne, zwłaszcza jeśli dane są rzadko używane lub można je łatwo pobrać w razie potrzeby. Jednak w niektórych sytuacjach największy komfort pracy użytkownik może zapewnić, pobierając wstępnie dane i zwiększając ogólne obciążenie łączy.

Dane można wstępnie pobierać bezpośrednio, w trakcie wczytywania początkowej strony, lub za pomocą wywołań AJAX-a. Można jednak zauważyć, że jeśli dane nie są potrzebne natychmiast, korzystniejsze jest stosowanie AJAX-a. Wstępne pobieranie przy użyciu AJAX-a może mieć miejsce po załadowaniu strony, co sprawia, że proces ten jest mniej zauważalny dla użytkowników. Wczytywanie danych można także powiązać z rozpoczęciem wykonywania zadania, które ich wymaga. Jest to szczególnie istotne w przypadku przeglądania dużych zbiorów danych, ponieważ użytkownicy korzystają z nich zwykle według spójnych wzorców, które można wykryć, monitorując zachowania osób korzystających z aplikacji.

AJAX nie gwarantuje spadku obciążenia łączy, a w niektórych wzorcach prowadzi nawet do wzrostu ilości przesyłanych danych. Jest to szczególnie widoczne przy obsłudze żądań AJAX-a sterowanych zdarzeniami. Każde pojedyncze żądanie może być małe, jednak sprawdzanie wszystkich wciśnięć klawiszy może szybko spowodować duże obciążenie. Te efekty można złagodzić, ograniczając liczbę zdarzeń do jednego w danym okresie, jednak z czasem obciążenie i tak wzrośnie. Celem powinno być zminimalizowanie wielkości każdego żądania, a jednocześnie należy pamiętać, że uzyskane w ten sposób zmniejszenie obciążenia może nie mieć znaczenia z powodu większej częstotliwości żądań i wstępnego pobierania danych, co pozwala na utworzenie wysoce interaktywnego interfejsu.

5.1.4. Tworzenie bogatych aplikacji

Trzy pierwsze cele dotyczyły głównie usprawniania istniejących aplikacji sieciowych, jednak AJAX umożliwia także tworzenie programów sieciowych o zupełnie nowej jakości. W trakcie tworzenia bogatych aplikacji programiści mają na celu jak największe upodobnienie ich do programów stacjonarnych przy zachowaniu zalet aplikacji sieciowych w zakresie łatwości wdrażania i implementacji. Ponadto tworzenie bogatych aplikacji ma prowadzić do zwiększenia interaktywności programów i zmniejszenia czasu potrzebnego na wykonywanie operacji, choć projekty i techniki prowadzące do spełnienia tych zamierzeń mogą być różne.

Ponieważ w tym przypadku nie rozwiązujemy problemów z powolnymi fragmentami istniejącego programu, nie ma wyjściowych pomiarów szybkości działania standardowej aplikacji sieciowej. Z tego powodu trzeba porównać funkcjonowanie utworzonego programu z jego stacjonarnym odpowiednikiem. Może to być trudne, ponieważ aplikacje stacjonarne używają dużych, trwałych magazynów danych w celu zmniejszenia liczby wolnych interakcji, podczas gdy aplikacje AJAX-owe są ograniczone do korzystania z mniejszych buforów bazujących na sesji. W zależności od ilości danych potrzebnych do wykonania zadania osiągnięcie wyników porównywalnych ze stacjonarnymi programami może być niemożliwe, dlatego trzeba skoncentrować się na innych wzorcach użycia, które pozwolą ukryć problem. Najłatwiej jest naśladować te aplikacje stacjonarne, które obsługują duże zbiory danych przechowywane poza lokalnym programem klienckim. Ponieważ czas dostępu do baz danych jest podobny w obu typach programów, aplikacje sieciowe muszą współzawodniczyć jedynie w zakresie jakości interfejsu użytkownika.

Wiele bogatych aplikacji powoduje większe obciążenie łączy niż ich odpowiedniki w postaci standardowych programów sieciowych. Wynika to z szerokiego stosowania wstępnie pobranych danych, co pozwala zapewnić płynną pracę. Powoduje to, że bogate aplikacje najlepiej nadają się do zastosowań w środowiskach wewnętrznych, w których szybka sieć i brak opłat za przesył danych pozwalają pominąć cel w postaci zmniejszenia obciążenia łączy.

Nie należy pochopnie decydować się na tworzenie bogatej aplikacji zamiast rozbudowanej witryny internetowej. Bogate aplikacje działają najlepiej, kiedy mają wykonywać operacje zwykle znane ze stacjonarnych programów. Klienty pocztowe, czytniki RSS i aplikacje generujące raporty to dobre przykłady stacjonarnych aplikacji, które

łatwo jest naśladować. Jednocześnie usługi zwykle udostępniane przez witryny internetowe, na przykład zakupy w sklepie internetowym czy wyświetlanie informacji o produktach, nie przekładają się dobrze na bogate aplikacje. Te zadania lepiej jest udostępniać poprzez rozbudowane witryny internetowe, w których wolne, złożone operacja można zastąpić ich AJAX-owymi wersjami.

5.2. Pomiar usprawnień

Czas wykonywania zadania to jedna z najbardziej przydatnych miar przy ocenie powodzenia wdrożenia AJAX-a. Sam proces pomiaru można rozbić na trzy proste etapy:

1. Określenie punktu początkowego i końcowego zadania.
2. Dodanie narzędzi do pomiaru czasu rozpoczynania i kończenia operacji.
3. Połączenie wielu danych w przydatne informacje.

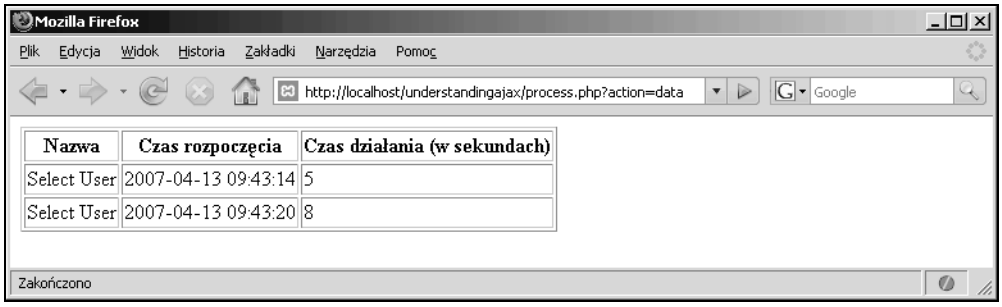
Określenie, jakie zadanie mierzyć, to zwykle prosty proces. Trzeba jedynie wykryć obszary w aplikacji, na które użytkownicy wciąż narzekają. Jeśli proces jest powolny i niewygodny, jest dobrym kandydatem na przeprowadzenie pomiarów i zastosowanie w nim AJAX-a. Po wyborze zadania należy określić jego punkty początkowy i końcowy. Ważne jest, aby zmierzyć cały proces. Nie należy koncentrować się na pobieraniu strony lub elementach technicznych, ale na operacjach wykonywanych przez użytkownika. Jeśli proces jest złożony, przydatne może być obserwowanie osób w celu zobaczenia, jak wykonują zadania.

Po wyznaczeniu punktów początkowego i końcowego trzeba dodać oprzyrządowanie. W większości przypadków można dokonać pomiaru przy użyciu prostych żądań AJAX-a kierowanych do rejestrującego skryptu. Jedno wywołanie oznacza początek procesu, a drugie — koniec. Przykładowy program rejestruje czas potrzebny na wybór użytkownika, którego dane mają być zmodyfikowane, co przedstawia rysunek 5.5. Ten przykład jest sztuczny, ale użyteczny, ponieważ pozwala pokazać, jak dodać oprzyrządowanie do procesu, a nie jak utworzyć edytor użytkowników w AJAX-ie.

Rysunek 5.5. *Wybór użytkownika*

Zadanie podzielone jest na następujące fragmenty: wczytanie strony, wyszukanie danej osoby i wybór użytkownika spośród wyników. Operacja rozpoczyna się w momencie wczytania strony lub kliknięcia przycisku *Znajdź użytkownika*. W tym przypadku używamy kliknięcia przycisku, ponieważ pomaga to zmniejszyć wariancję pomiarów. Proces kończy się wraz z uruchomieniem funkcji `selectUser` języka JavaScript. W samym kodzie ta funkcja albo otwiera edytor użytkownika, albo zapełnia formularz służący do wprowadzania zmian, który znajduje się pod panelem wyboru. Trzeba także utworzyć niepowtarzalny identyfikator, dzięki czemu można dopasować do siebie czas początkowy i końcowy, jednak jeśli klient zgłasza tylko jedno żądanie naraz, identyfikator można utworzyć i zapisać w skrypcie przechowującym dane.

Do dodawania oprzyrządowania posłuży prosta nakładka `HttpClient` na obiekty `XMLHttpRequest`. Program będzie wykonywał na początku i na końcu procesu wywołania AJAX-a do strony *process.php*. Ta strona zapisuje czas rozpoczęcia w sesji, a następnie funkcja `endProcess` z tego pliku dopasowuje go do drugiego wywołania zgłaszanego po zakończeniu procesu. Można wyświetlić prosty raport (rysunek 5.6), aby zobaczyć, jak długo trwała każda operacja wyboru użytkownika. W tym rozwiązaniu magazyn danych jest bardzo prosty i warto zastąpić go bazą danych, jeśli dane mają pochodzić z różnych komputerów.



Rysunek 5.6. *Prosty raport z długościami czasu wyboru użytkownika*

Niektóre fragmenty procesu pomiaru można wykorzystać wielokrotnie: przechowujący dane skrypt z pliku *process.php* (listing 5.1) i klasę `Monitor` języka JavaScript (listing 5.2).

Listing 5.1. *Plik process.php*

```
<?php
session_start();

if (!isset($_SESSION['data'])) {
   $_SESSION['data'] = array();
}
if (!isset($_SESSION['id'])) {
   $_SESSION['id'] = false;
}


function startProcess() {
   if (!$_SESSION['id']) {
      $now = time();
```

```
15       $id = uniqid('m');
16       $_SESSION['id'] = $id;
17       $_SESSION['data'][$id]['start'] = $now;
18       $_SESSION['data'][$id]['name'] = $_GET['process'];
19    }
20 }
21
22 function endProcess() {
23    $now = time();
24    $_SESSION['data'][$_SESSION['id']]['end'] = $now;
25    $_SESSION['id'] = false;
26 }
27
28 function printStats() {
29    echo "<table border=1><tr><th>Nazwa</th><th>Czas rozpoczęcia</th>
30       <th>Czas działania (w sekundach)</th></tr>";
31    foreach($_SESSION['data'] as $process) {
32       echo "<tr><td>$process[name]</td><td>".
33       date('Y-m-d H:i:s',$process['start']) .
34       '</td><td>';
35       if (isset($process['end'])) {
36          echo ($process['end'] - $process['start']);
37       }
38       echo '</td></tr>';
39    }
40    echo "</table>";
41 }
42
43 switch($_GET['action']) {
44    case 'start':
45       startProcess();
46       break;
47    case 'end':
48       endProcess();
49       break;
50    case 'data':
51       printStats();
52       break;
53 }
54 ?>
```

Kod pokazany na listingu 5.1 wykorzystuje sesję PHP do przechowywania danych, dlatego skrypt rozpoczyna się od jej konfiguracji. Program otwiera sesję w wierszu 2., a następnie ustawia pewne domyślne wartości w wierszach 4 – 9. W wierszach 12 – 41 znajduje się definicja trzech funkcji — po jednej dla każdej operacji, jaką wykonuje skrypt. Funkcja `startProcess` (wiersze 12 – 20) najpierw sprawdza, czy w sesji zapisany jest bieżący identyfikator. Ten test pozwala zignorować wielokrotne żądania rozpoczęcia tego samego procesu. Jeśli tego identyfikatora nie ma, funkcja `startProcess` zapisuje bieżący czas, tworzy nowy identyfikator losowy, a następnie umieszcza te dane w sesji wraz z nazwą procesu. Funkcja `endProcess` (wiersze 22 – 26) zapisuje czas końcowy, a następnie usuwa identyfikator, aby umożliwić rozpoczęcie następnego procesu. Te dwie funkcje obsługują podstawową funkcjonalność pobierania czasu.

Trzecia funkcja, `printStats` (wiersze 28 – 41), tworzy tabelę z prostym raportem. Ta funkcja przechodzi w pętli po danych zapisanych w sesji i tworzy tabelę w kodzie HTML. W trakcie tej operacji używa czasu początkowego i końcowego do obliczenia długości każdego procesu. Na rysunku 5.6 przedstawione są dane wyjściowe tej funkcji. Wiersze 43 – 53 określają, która z tych funkcji zostanie wykonana. To, która funkcja zostanie wywołana, zależy od wartości zmiennej `action` żądania `GET`. Po stronie HTML służący do monitorowania kod JavaScript (listing 5.2) zgłasza żądania AJAX-a do strony *process.php* w celu zapisania danych na temat użytkowania aplikacji.

Listing 5.2. *Plik Monitor.js*

```
1  // Klasa do monitorowania czasu wykonywania operacji przez użytkowników
2
3  function Monitor() {
4     this.httpclient = new HttpClient();
5     this.httpclient.isAsync = true;
6     this.httpclient.callback = function() {};
7  }
8  Monitor.prototype = {
9     startProcess: function(name) {
10       this.httpclient.makeRequest(
11          'process.php?action=start&process='+name);
12    },
13    endProcess: function(name) {
14       this.httpclient.makeRequest(
15          'process.php?action=end&process='+name);
16    }
17 }
```

Klasa do monitorowania jest dość prosta. Konstruktor (wiersze 3 – 7) tworzy egzemplarz nakładki `HttpClient` do wykonywania operacji asynchronicznych, a następnie definicje dwóch funkcji. Pierwsza z nich, `startProcess`, wysyła do strony *process.php* żądanie wyzwalające funkcję `startProcess` z tego pliku. Druga funkcja, `endProcess` (wiersze 13 – 16), wysyła do strony *process.php* podobne żądanie, jednak tym razem żądanie AJAX-a wywołuje powiązaną funkcję `endProcess` języka PHP. Główne zadanie tej klasy to ułatwienie dodania oprzyrządowania do stron aplikacji, dlatego zawiera kod szablonowy, dzięki czemu programista nie musi go pisać. Jest to także dobre miejsce na dodanie nowych metod, jeśli potrzebne jest pobieranie innych danych, na przykład rejestrowanie tego, jakie operacje wykonał użytkownik.

Teraz, kiedy podstawowa platforma narzędzi jest już skonfigurowana pod kątem rejestrowania czasu procesów, trzeba dodać ją do skryptu. W przypadku stron AJAX-owych informacje o czasach wykonywania zadań mogą być przydatne do pobierania danych wygenerowanych w wyniku wprowadzenia określonych zmian. Te dane mogą być także wartościowe na stronach bez AJAX-a, gdzie pomagają w pomiarze wolnych procesów i wykryciu tych, które należy usprawnić. Pobieranie danych w taki sposób może być także pomocne w podejmowaniu decyzji o tym, które dane program ma wstępnie pobierać, jednak zwykle potrzebnych jest więcej informacji niż sam czas, ponieważ trzeba sprawdzić, jakie operacje użytkownicy wykonują najczęściej. Listing 5.3 przedstawia prosty skrypt do wybierania użytkowników, który wykorzystuje klasę `Monitor` języka JavaScript z listingu 5.2 do rejestrowania czasu każdego wyboru.

Listing 5.3. *Strona selectUser.class.php*

```
<?php
/**
 * Przykładowa klasa, która wyszukuje użytkownika w tablicy.
 * W większości przypadków takie klasy przeszukują bazę danych.
 */
class selectUser {

   var $users = array(
      1 => 'Joshua Eichorn',
      2 => 'Travis Swicegood',
      3 => 'Inna osoba 1',
      4 => 'Inna osoba 2',
      );

   function search($input) {
      $ret = array();

      foreach($this->users as $key => $name) {
         if (stristr($name,$input)) {
            $ret[$key] = $name;
         }
      }
      return $ret;
   }
}
?>
```

Samo wyszukiwanie odbywa się w klasie powiązanej, `selectUser`. Skrypt przeszukuje tablicę, aby przykład był jak najbardziej prosty, jednak w większości przypadków proces ten wymagałby korzystania z bazy danych. Przedstawiona klasa ma jedną metodę, `search` (wiersze 15 – 24), która przyjmuje dane wejściowe. Metoda ta sprawdza, nieuwzględniając wielkości znaków, czy podana nazwa znajduje się w tablicy użytkowników przechowywanej w klasie. Na zakończenie metoda tworzy tablicę na podstawie dopasowanych wyników i zwraca ją. Listing 5.4 przedstawia interfejs użytkownika w kodzie HTML. Strona ta do obsługi kierowanych do niej żądań `POST` używa klasy `selectUser`.

Listing 5.4. *Plik selectUser.php*

```
<html>
<head>
   <title>Wybieranie użytkowników (bez AJAX-a)</title>


<script type="text/javascript" src="HttpClient.js"></script>
<script type="text/javascript" src="Monitor.js"></script>
<script type="text/javascript">
var monitor = new Monitor();
function selectUser(el) {
   alert('Wybrano użytkownika o ID: '+el.value);
   monitor.endProcess('Select User');
}
</script>
```

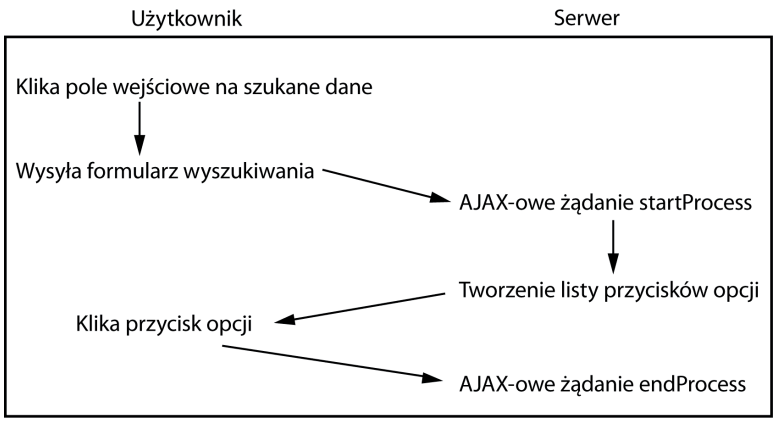
```
15 </head>
16 <body>
17
18 <div id="HttpClientStatus"></div>
19
20 <h1>Wybierz użytkownika</h1>
21
22 <form action="selectUser.php" method='post'>
23    <input name="name" onclick="monitor.startProcess('Select User')">
24    <input type="submit" value="Znajdź użytkownika">
25 </form>
26
27 <?php
28    require_once 'selectUser.class.php';
29
30    if (isset($_POST['name']) && !empty($_POST['name'])) {
31       $users = new selectUser();
32       $results = $users->search($_POST['name']);
33
34       foreach($results as $key => $val) {
35          echo "<input type='radio' name='user' value='$key'".
36          "id='user_$key' onclick='selectUser(this)'>".
37          "<label for='user_$key'>$val</label><br>\n";
38    }
39 }
40?>
```

Skrypt rozpoczyna się od prostej konfiguracji. Następnie, w wierszu 6., dołączana jest nakładka na obiekty `XMLHttpRequest`, a w wierszu 7. — klasa `Monitor` języka JavaScript. Kod w wierszu 9. tworzy egzemplarz klasy `Monitor`, dzięki czemu można łatwo wywoływać funkcje `startProcess` i `endProcess` w całym kodzie strony. W wierszach 10 – 13 znajduje się definicja funkcji języka JavaScript wywoływana na zakończenie procesu wybierania użytkownika. Ta funkcja wyświetla wybrany komunikat, a następnie uruchamia funkcję `endProcess`. Kod w wierszach 20 – 25 to prosty interfejs użytkownika w języku HTML. Jest to formularz, który wysyła do bieżącej strony żądanie `POST` z wynikami. Wejściowe pole wyszukiwania uruchamia funkcję `startProcess` po kliknięciu go w celu rozpoczęcia wpisywania szukanego wyrażenia.

Kod w wierszach 27 – 40 odpowiada za proces wyszukiwania po przesłaniu do strony żądania `POST` z formularzem. Za samo wyszukiwanie odpowiada egzemplarz klasy `selectUser`. Skrypt przechodzi następnie w pętli po wynikach tej operacji, tworząc przycisk opcji dla każdej pozycji. Do każdego z tych przycisków program dodaje akcję `onclick`, która wywołuje funkcję `selectUser` zdefiniowaną w wierszach 10 – 13.

Poniższe punkty opisują przepływ żądań, a podstawowy schemat działania strony przedstawia rysunek 5.7.

1. Użytkownik klika pole wejściowe wyszukiwania, wysyłając żądanie `startProcess`.
2. Użytkownik klika przycisk *Wyszukiwanie użytkowników*, przesyłając formularz w żądaniu `POST`.



Rysunek 5.7. *Przepływ strony umożliwiającej pomiar czasu wybierania użytkowników*

3. Skrypt używa nazwy wysłanej w żądaniu `POST` do utworzenia listy przycisków opcji umożliwiających wybór określonego użytkownika.
4. Użytkownik klika przycisk opcji, zgłaszając żądanie `endProcess`.

Pomiar czasu to istotny pierwszy etap w procesie podejmowania trafnych decyzji dotyczących tego, jak i kiedy zaimplementować AJAX-a. Jeśli można już szybko wybierać użytkowników, nie ma sensu usprawniać tej operacji poprzez dodawanie AJAX-a. W zamian warto zająć się innym procesem. Jeśli mierzona operacja jest powolna, można utworzyć wersję z AJAX-em zawierającą oprzyrządowanie, a następnie dokonać pomiarów efektywności zmian. Może się okazać, że wersja formularza z AJAX-em nie pozwala w wystarczającym stopniu zwiększyć szybkości, ponieważ wyszukiwanie według nazw nie skaluje się względem liczby użytkowników systemu. Najlepsze wyniki może dać ograniczenie wyników wyszukiwania według określonych kryteriów, na przykład według wydziału lub stanowiska.

5.3. Wady i zalety łączenia AJAX-a z innymi nowymi technologiami

Korzystając z AJAX-a, można natrafić na powiązane technologie, które mogą z nim współdziałać. Dzielą się one na dwie główne kategorie: dojrzałe technologie dostępne w wielu współczesnych przeglądarkach oraz nowe technologie obsługiwane tylko w niektórych z nich. Technologie dojrzałe obejmują Javę i Flasha. Flash jest najbardziej istotny, ponieważ wtyczka tej technologii jest szeroko instalowana, a jej budowa jest zoptymalizowana pod kątem umieszczania interaktywnych elementów i animacji na witrynach internetowych. Także Javy można używać do zwiększania interaktywności witryn, jednak jej popularność zmalała w ciągu pięciu ostatnich lat, a platforma ta nie jest już domyślnie instalowana na każdej maszynie.

5.3.1. Łączenie AJAX-a z technologią Flash

Flash to dobra technologia towarzysząca technikom AJAX-a, ponieważ pozwala korzystać z różnych funkcji niedostępnych w czystym języku JavaScript. Obejmują one płótno, na którym można dodawać obrazy i przemieszczać je, a także interfejs API pozwalający na rysowanie grafiki. Ponadto Flash umożliwia strumieniowanie wideo i dźwięku oraz obsługę animacji wektorowych. Największą wadą Flasha jest to, że wymaga nowego, odrębnego środowiska programistycznego, a choć możliwe jest przekazywanie wywołań między kodem JavaScript stron a kodem ActionScript filmu we Flashu, nie można ściśle powiązać tej technologii z resztą strony. Ponadto elementy strony we Flashu wyglądają inaczej niż pozostałe fragmenty strony HTML, co utrudnia udostępnianie spójnego wyglądu i operacji, jeśli tej technologii używa się do obsługi drobnych funkcji w obrębie większego rozwiązania.

Wady Flasha — słabą integrację z językiem JavaScript oraz odmienny wygląd i styl — można przezwyciężyć, jednak powodują one, że wiele osób decyduje się na wybór rozwiązań bazujących całkowicie na tej technologii przy tworzeniu bardziej skomplikowanych programów. Pomaga to kontrolować złożoność, jednak powoduje całkowitą zależność od jednego producenta i oznacza, że w warstwie komunikacji trzeba używać zdalnych wywołań Flasha zamiast AJAX-a. Użycie Flasha w aplikacji AJAX-owej umożliwia dodanie obsługi wielu graficznych operacji niemożliwych w innych warunkach, jednak programista musi liczyć się z bardziej złożonym procesem projektowania i diagnozowania.

5.3.2. Skalowalna grafika wektorowa (SVG)

Nowe technologie w świecie przeglądarek nie są tak szeroko spopularyzowane jak Flash, a niektóre z nich, na przykład język XAML (ang. *Extensible Application Markup Language*) Microsoftu, doczekały się jedynie wersji beta. Mają one zalety w postaci pełnej integracji z przeglądarką, co sprawia, że można używać ich w skryptach języka JavaScript oraz jako elementów najwyższego poziomu na stronach internetowych. Skalowalna grafika wektorowa (ang. *Scalable Vector Graphics* — SVG) to nowy ustandaryzowany przez W3C język do tworzenia grafiki wektorowej. Ma wiele tych samych funkcji co Flash i umożliwia dodawanie do stron animacji oraz interaktywnych elementów graficznych. SVG pozwala uniknąć wielu problemów występujących przy korzystaniu z Flasha, ponieważ w skryptach tej technologii można używać języka JavaScript. Ponadto można zagnieżdżać takie skrypty bezpośrednio w stronach HTML i modyfikować je podobnie jak wszystkie inne elementy.

Największy problem z SVG polega na tym, że obsługa tej technologii w przeglądarkach rozwija się dość powoli. Obecnie wtyczka firmy Adobe pozwala korzystać z większości funkcji SVG, jednak występują podobne problemy z integracją jak w przypadku Flasha. Następne wersje przeglądarek Firefox i Opera będą miały wbudowaną obsługę SVG, jednak wciąż jest to niszowa technologia, której można używać jedynie w wewnętrznych projektach.

Ponadto SVG nie udostępnia płótna, dlatego nie ma sposobu na rysowanie elementów, co jest możliwe we Flashu. Aby rozwiązać ten problem, firma Apple utworzyła płótno bitmapowe dla przeglądarki Safari. To narzędzie zostało od tego czasu unormowane przez grupę WHATWG (ang. *Web Hypertext Application Technology Working Group*). Jest ona wspierana przez wielu producentów przeglądarek, między innymi Mozilli i Opery, dlatego w nowych wersjach tych programów znajdzie się obsługa SVG, jednak w przypadku przeglądarki Internet Explorer jest ona mniej pewna.

5.3.3. Języki interfejsu użytkownika bazujące na XML-u

Wiele nowych technologii nie jest obsługiwanych w przeglądarkach, a języki interfejsu użytkownika bazujące na XML-u nie są tu wyjątkiem. Języki te — XUL (ang. *XML User Interface Language*) Mozilli i XAML — umożliwiają opis standardowych elementów aplikacji (takich jak menu, paski przewijania, pola tekstowe i przyciski) za pomocą łatwych w użyciu znaczników XML. Języki bazujące na XML-u Microsoftu i Mozilli mają umożliwiać tworzenie wysoce interaktywnych bogatych aplikacji, które bez dodatkowych modyfikacji wyglądają i działają podobnie jak programy stacjonarne. Jednak ponieważ technologie te są obsługiwane tylko przez jednego producenta, powodują problem w postaci powiązania z określoną przeglądarką. Ta zależność znacznie zmniejsza atrakcyjność rozwiązania w porównaniu z bardziej zgodnymi technikami bazującymi na językach JavaScript i HTML.

Język XUL został utworzony w celu definiowania interfejsu użytkownika w przeglądarce Mozilla i był dostępny przez kilka lat. Ostatnio ma swoje pięć minut wraz ze wzrostem popularności przeglądarki Firefox, jednak nie przestanie być produktem niszowym dopóty, dopóki inne przeglądarki nie zaczną go obsługiwać. XAML został utworzony przez Microsoft jako część projektu .NET i ma być udostępniony wraz z systemem Windows Vista. Trudno stwierdzić, jaki wpływ będzie miała ta technologia. Będzie można to ocenić dopiero wtedy, kiedy zostanie szeroko rozpowszechniona i kiedy znany będzie jej poziom dostępności dla programistów aplikacji sieciowych.

Wzrost popularności nowych przeglądarek internetowych doprowadził do utworzenia ciekawych technologii sieciowych. Największy problem polega na tym, że obsługa większość z nich ogranicza się do jednej przeglądarki. Wraz ze wzrostem dostępności nowych technologii mogą one zacząć odgrywać większą rolę, umożliwiając zwiększenie interaktywności ponad poziom zapewniany przez sam AJAX i dynamiczny język HTML (DHTML).

5.4. Podsumowanie

Aby optymalnie wykorzystać możliwości AJAX-a, trzeba wykryć i usprawnić obszary, w których jego zastosowanie da największe korzyści, a następnie podjąć kroki do zmierzenia tego efektu. AJAX-a można używać w większości projektów witryn internetowych, jednak najbardziej przydatny jest do przyspieszania wykonywania wyszukiwania

i wieloetapowych procesów. Przy rozwiązywaniu prostszych problemów dobre efekty daje wybór odpowiednich elementów AJAX-a, jednak wraz ze wzrostem złożoności warto dodać do aplikacji oprzyrządowanie i uzyskać twarde dane dotyczące zmian wywołanych przez AJAX-a. Używając tych informacji, można dostosować zastosowane podejście i rozpocząć drogę do efektywnego wykorzystania tej technologii. Ostatni etap w skutecznym korzystaniu z AJAX-a to analiza użyteczności interfejsu graficznego. Ten proces opisuję w następnym rozdziale.