ASP.NET 2.0 AJAX. Zaawansowane programowanie

Autor: Matt Gibbs, Dan Wahlin
Tłumaczenie: Krzysztof Bąbol
ISBN: 978-83-246-1300-7
Tytuł oryginału: Professional ASP.NET 2.0 AJAX
Format: B5, stron: 328

Śmiało wkraczaj w nowoczesne technologie, aby tworzyć interaktywne aplikacje internetowe!

- Jak zarządzać odwołaniami do strony?
- Jak osadzać skrypty w kontrolkach serwerowych?
- Jak używać narzędzia Fiddler do kontroli komunikatów?

ASP.NET AJAX to nazwa stworzonego przez Microsoft rozwiązania AJAX. Odnosi się do zestawu technologii klienckich i serwerowych, które koncentrują się na ulepszaniu programowania WWW pod Visual Studio. Jest to jedna z najpopularniejszych, a równocześnie najbardziej docenianych – zarówno przez użytkowników, jak i programistów – technologii do tworzenia interaktywnych witryn internetowych. ASP.NET AJAX dysponuje między innymi serwerowymi usługami aplikacyjnymi, takimi jak uwierzytelnianie i przechowywanie profilu użytkownika, oraz zawiera zestaw kontrolek ułatwiających asynchroniczną aktualizację stron i korzystanie z zasobów serwera.

Książka „ASP.NET 2.0 AJAX. Zaawansowane programowanie" szczegółowo ukazuje, na czym polega tworzenie aplikacji WWW nowej generacji. Dzięki temu podręcznikowi będziesz wiedział, jak korzystać z różnych bibliotek, jak używać zaawansowanych kontrolek z pakietu AJAX Toolkit, a także jak budować własne kontrolki. Nauczysz się asynchronicznie uaktualniać fragmenty stron i zarządzać skryptami używanymi w przeglądarce, dowiesz się, jak testować takie aplikacje i usuwać z nich błędy. Z tą książką zdobędziesz wiedzę potrzebną do kreowania nowoczesnych, interaktywnych aplikacji.

- Częściowe aktualizacje stron
- Pobieranie kodu JavaScript
- Biblioteka kliencka ASP.NET AJAX
- Używanie prototypów
- Elementy DOM
- Użycie kontrolki ScriptManager
- Dodawanie odwołań do skryptów
- Zaawansowane kontrolki z pakietu AJAX Toolkit
- Efekty interfejsu użytkownika
- Testowanie, debugowanie i wdrażanie aplikacji ASP.NET AJAX

Korzystaj z nowoczesnych technologii podczas tworzenia interaktywnych aplikacji internetowych!

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

Spis treści

1

Tworzenie nowej generacji aplikacji WWW

Aplikacje internetowe nie były dotąd tak zaawansowane i interaktywne jak zwykłe programy. Użytkownik końcowy niekoniecznie musi mieć szczegółową wiedzę na temat działania aplikacji, ale wie, że kontakt ze stroną w przeglądarce zasadniczo różni się od korzystania z lokalnie zainstalowanego programu. Kiedy zespół programistów przymierza się do nowego projektu, jednym z pierwszych pytań, na które musi odpowiedzieć, jest to, czy użytkownicy będą w stanie zaakceptować ograniczenia sieci WWW, czy potrzebują instalacji tradycyjnego programu. Aplikacje WWW są dostępne z prawie każdej przeglądarki, właściwie wszędzie, ale ograniczają się do tego, co można zrobić z kodem znaczników i skryptami działającymi w przeglądarce.

Tradycyjne aplikacje, określane też mianem „ciężkich klientów" (ang. *fat client*), wymagają zainstalowania na komputerze użytkownika, ale pozwalają programiście skorzystać z wbudowanych w system operacyjny zaawansowanych możliwości grafiki i sterowania, które niezwykle trudno zaimplementować w przeglądarce WWW, a także wykorzystać zasoby komputera do zadań takich jak zapisywanie danych. Z drugiej strony, aplikacje internetowe wystarczy zaktualizować na serwerze, a użytkownicy automatycznie będą mieli dostęp do najnowszej wersji. O wiele trudniej zaktualizować zwykły program, ponieważ trzeba przekonać użytkowników, żeby zainstalowali nową wersję, lub wprowadzić do aplikacji inteligentny system automatycznych aktualizacji.

Mówi się, że aplikacje internetowe nie wymagają wdrożenia, tradycyjne programy są zaś pracochłonne we wdrożeniu i konfiguracji. Wybór między nimi często charakteryzuje się jako kompromis między możliwościami a dostępnością (w języku ang. gra słów *rich* — bogaty i *reach* — sięgać). Aplikacje biurkowe na ogół oferują bogatsze możliwości w porównaniu z tym, co daje przeglądarka, ale aplikacje WWW bez wysiłku możesz udostępnić każdemu użytkownikowi w dowolnym systemie operacyjnym. Co więcej, w wielu firmach obowiązują restrykcyjne zasady instalacji oprogramowania na komputerach pracowników, często nie mają oni uprawnień administratora wymaganych do instalacji nowych programów, więc aplikacje internetowe są w wielu sytuacjach jedynym sensownym rozwiązaniem.

Wzbogacanie aplikacji internetowych

W minionych latach posiadanie strony WWW było czynnikiem wyróżniającym firmę. Dzisiaj już tak nie jest. Teraz sama obecność w sieci nie wystarcza. Firmy starają się odróżnić od konkurencji poprzez aplikacje WWW, które odpowiadają intuicyjnie na działania klientów i przewidują ich zachowania. Niniejsza książka ukazuje, w jaki sposób ASP.NET AJAX odnosi się do specyficznych wymagań programowania WWW i zwiększa komfort użytkownika Twoich stron. W tym rozdziale wytłumaczę, dlaczego potrzebujemy bardziej zaawansowanych platform w programowaniu WWW. Opowiem o kluczowych elementach platformy ASP.NET AJAX i przedstawię kilka innych opcji.

Zestaw technologii leżący u podstaw nowej generacji aplikacji internetowych nie jest wcale nowy. Artykuły na stronach informacyjnych oraz blogi ukazują Google, Flickr i kilka innych serwisów jako najlepsze przykłady wykorzystywania tych technologii w niespotykany wcześniej sposób. Te aplikacje mają kilka unikalnych cech, ale w rzeczywistości leżące u ich podstaw technologie są znane i używane przez prawie dekadę. Przyglądając się, w jaki sposób Microsoft Exchange Server udostępniał w aplikacji Outlook Web Access zaawansowane funkcje poczty internetowej z poziomu przeglądarki, można zobaczyć, że koncepcja wszechobecnego dostępu wykorzystującego typowy zestaw funkcji przeglądarki i dostarczającego użytkownikowi bogatych wrażeń istniała w praktyce od lat. Użytkownicy dostają naprawdę pełnowartościową aplikację bez potrzeby lokalnej instalacji i mogą korzystać z poczty praktycznie z każdego komputera.

Technologie używane do budowania zaawansowanych aplikacji WWW przyjęło się określać mianem AJAX (ang. *Asynchronous JavaScript and XML* — asynchroniczny JavaScript i XML). Chociaż ten akronim miło brzmi, niewiele wyjaśnia. Zamiast budować aplikacje WWW w postaci serii odsłon i przeładowań stron, programiści wykorzystują JavaScript do asynchronicznej komunikacji z serwerem i dynamicznego aktualizowania części strony. Oznacza to, że strona potrafi dynamicznie zmieniać wygląd podczas interakcji z użytkownikiem, a nawet może wysyłać lub pobierać dane z serwera WWW w tle. Minęły dni brzydkiego przeładowania czyszczącego ekran użytkownika i odwracającego jego uwagę! Teraz odsyłanie do serwera jest potrzebne tylko wtedy, gdy chcemy przejść na inną stronę.

> *Można nagiąć nawet tę zasadę. Niektóre aplikacje przekraczają tę granicę i kompletnie zmieniają wygląd ekranu użytkownika, tak jakby przeszły do nowej strony, ale robią to, przesyłając dane asynchronicznie i zmieniając zawartość strony bez faktycznego przejścia do nowego adresu URL.*

Akronim AJAX odnosi się do XML jako formatu danych wymienianych między klientem a serwerem, ale w rzeczywistości tworzy się aplikacje pobierające zwykłe fragmenty tekstu, XML oraz JSON (ang. *JavaScript Object Notation* — zapis obiektów JavaScript), który omówię dokładniej w rozdziale 4. Pewna atrakcyjna cecha technologii AJAX nie jest nawet zawarta w akronimie: oprócz nieblokującej komunikacji z serwerem, programiści wykorzystują Dynamic HTML (DHTML) i Cascading Style Sheets (CSS), aby stworzyć naprawdę zdumiewające interfejsy użytkownika. Kod JavaScript działający na kliencie komunikuje się asynchronicznie z serwerem, a potem używa DHTML do dynamicznego modyfikowania strony, tworząc zaawansowane animacje, przejścia i aktualizacje treści, w czasie gdy użytkownik kontynuuje interakcję ze stroną. W wielu przypadkach użytkownicy nie będą nawet zdawać sobie sprawy, że używają aplikacji internetowej!

Kto skorzysta z technologii AJAX

AJAX oferuje korzyści zarówno użytkownikom, jak i programistom — użytkownikom, ponieważ redukuje konflikt „dostępności i możliwości"; programistom, ponieważ pomaga pokonać ograniczenia protokołu HTTP.

Dlaczego użytkownicy chcą aplikacji AJAX

Użytkownicy zazwyczaj patrzą na tradycyjne programy jako coś w rodzaju zobowiązania. Instalują program, zazwyczaj z płyty wyciągniętej z kosztowego, zafoliowanego pudełka. Program zabiera miejsce na dysku oraz pozycję w menu startowym. Być może trzeba będzie go co jakiś czas uaktualniać lub przejść później na wyższą wersję, aby uzyskać nowe funkcje. Jeśli program sam chce się aktualizować, użytkownik jest regularnie konfrontowany z oknami dialogowymi informującymi o łatkach lub uaktualnieniach. W zamian za zainwestowany czas, pieniądze i energię aplikacja odpłaca się tym, że może wykorzystać zasoby systemu operacyjnego i komputera. Jest zaawansowana. Może zapisywać dane lokalnie, szybko reaguje, ma interesujący i intuicyjny interfejs graficzny.

Coraz więcej aplikacji jest dostępnych z poziomu przeglądarki. Nie mają one dostępu do pełnych zasobów sprzętu i systemu operacyjnego, ale nie wymagają takiego zaangażowania jak tradycyjne programy. Przez lata korzystanie z aplikacji WWW odbywało się według przewidywalnego wzoru. Użytkownik klika łącze na stronie, przeglądarka migocze, a użytkownik czeka na odrysowanie ekranu (okropne przeładowanie). Ten cykl powtarza się po raz kolejny. Użytkownik patrzy na prezentowaną stronę, nawiązuje z nią interakcję i gdzieś klika. Przeglądarka potwierdza to odgłosem kliknięcia i zaczyna kontaktować się z serwerem. Ekran przeglądarki staje się czysty, jakaś ikonka kręci się lub miga, podczas gdy użytkownik czeka, aż serwer zwróci nową wersję strony. W wielu przypadkach ta nowa wersja jest prawie identyczna jak poprzednia, zmieniony jest tylko jej fragment. A potem cykl rozpoczyna się od nowa. Powstaje wrażenie powolności nawet przy szybkim połączeniu z Internetem.

Zestaw technologii AJAX zmienił oczekiwania użytkowników wobec aplikacji internetowych. Kod JavaScript uruchomiony w przeglądarce wymienia dane z serwerem WWW asynchronicznie. Nie słychać odgłosu klikania, a przeglądarka nie miga. Komunikacja z serwerem nie jest blokująca, dzięki czemu użytkownik może dalej przeglądać stronę i nawiązywać z nią interakcję. Skrypt pobiera zaktualizowane dane z serwera i dynamicznie modyfikuje stronę, używając metodologii DHTML. Użytkownik może dalej patrzeć na stronę, podczas gdy w tle aktualizują się jej fragmenty. AJAX daje internautom bardziej dynamiczne wrażenia, zbliżając zachowanie aplikacji WWW do tradycyjnych programów. JavaScript wzbogaca doświadczenia użytkownika dzięki wsparciu mechanizmu „przeciągnij i upuść", obsłudze modalnych okien dialogowych i pozornie natychmiastowej aktualizacji różnych elementów strony w zależności od działań internauty.

Duża część sukcesów w wykorzystaniu technologii AJAX to zasługa postrzeganego wzrostu wydajności. Internauci doceniają aplikacje, które przewidują ich zachowania. Jeśli użyjesz także kodu JavaScript, pobierającego w tle obrazki i dane, które mogą być potrzebne,

użytkownicy dostaną szybką odpowiedź bez typowej przerwy towarzyszącej ich akcjom. Nikt nie chce czekać na wymianę danych między klientem a serwerem, badania pokazały, że długi czas pomiędzy działaniem użytkownika a następującymi po nim zmianami w interfejsie może znacznie obniżyć produktywność i wywołać frustrujące wrażenie walki z aplikacją. Użytkownicy chcą, by aplikacje internetowe zachowywały się jak typowe programy, ale bez obciążenia związanego z instalacją. W sytuacji gdy coraz więcej aplikacji korzysta z inteligentnego buforowania, przewiduje działania użytkownika i dostarcza bogatszego interfejsu, zaciera się różnica pomiędzy aplikacjami internetowymi a biurkowymi. Wymagania względem tych pierwszych zwiększają się. Użytkownicy dostrzegli już, że można uniknąć zobowiązań związanych z instalacją tradycyjnego programu i wciąż mieć bogate, interaktywne doświadczenia.

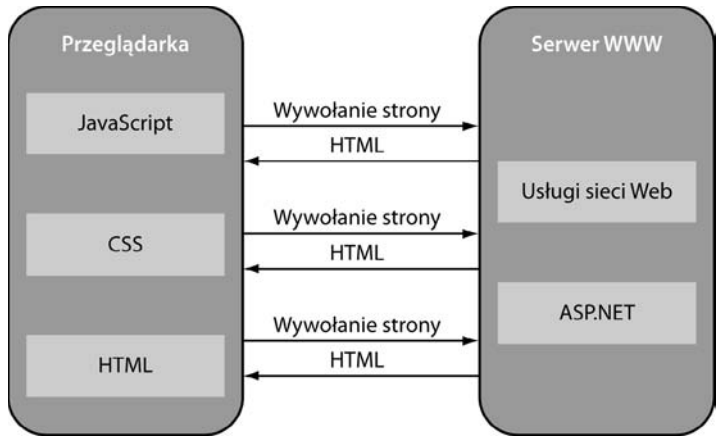
Dlaczego programiści chcą stosować AJAX

Często pierwszym pytaniem stawianym po rozpoczęciu projektu jest to, jaki typ aplikacji będziemy tworzyć. Czy to ma być aplikacja tradycyjna, czy może internetowa? To kluczowa decyzja, ponieważ w przeszłości narzucała w znacznym stopniu naturę aplikacji i przestrzeń zadań związanych z jej rozwojem. Obecnie wielu programistów wybiera domyślnie tworzenie aplikacji WWW, jeśli nie ma w tym względzie żadnych przeciwwskazań. Jeśli aplikacja musi działać bez dostępu do sieci lub wymaga interfejsu trudnego do uzyskania w języku HTML, może to wykluczać korzystanie z przeglądarki i narzucać wybór pisania samodzielnej aplikacji.

Pisząc nowoczesne aplikacje internetowe, programiści mają trudności z powodu wrodzonych ograniczeń sieci WWW nakładanych przez Hypertext Transfer Protocol (HTTP) i sposób, w jaki korzystają z niego przeglądarki. HTTP jest protokołem bezstanowym. Przeglądarka WWW wywołuje stronę, być może podając jakieś parametry łańcucha zapytania lub danych formularza, a serwer przetwarza to żądanie i wysyła odpowiedź zawierającą treść opisaną w języku HTML. Serwer może tylko reagować na informację przesłaną w bieżącym zleceniu i nie zna, na podstawie informacji w nim zawartych, żadnych szczegółów na temat drogi, jaką internauta dotarł do bieżącego widoku. Podczas wizualizacji odpowiedzi połączenie może zostać zerwane, a serwer nie ma żadnych informacji, które mógłby przechować dla przyszłego wywołania. Patrząc z perspektywy serwera, po prostu słucha wywołań nadchodzących z dowolnego miejsca i dowolnej przeglądarki, a następnie na nie reaguje. Przeglądarka zgłasza żądanie dostępu do strony i otrzymuje w odpowiedzi stronę HTML. Używa otrzymanego kodu HTML do wizualizacji interfejsu użytkownika. Użytkownik nawiązuje interakcję ze stroną, a w odpowiedzi przeglądarka czyści ekran i zgłasza do serwera nowe żądanie zawierające pewne informacje na temat danych wprowadzonych przez użytkownika lub jego działań. Znowu zwracana jest kompletna strona HTML. Zasadniczo protokół HTTP jest bezstanowy. Serwer otrzymuje wywołanie i na nie odpowiada. Wywołanie niesie ze sobą ograniczoną informację o komunikacji toczącej się między klientem a serwerem.

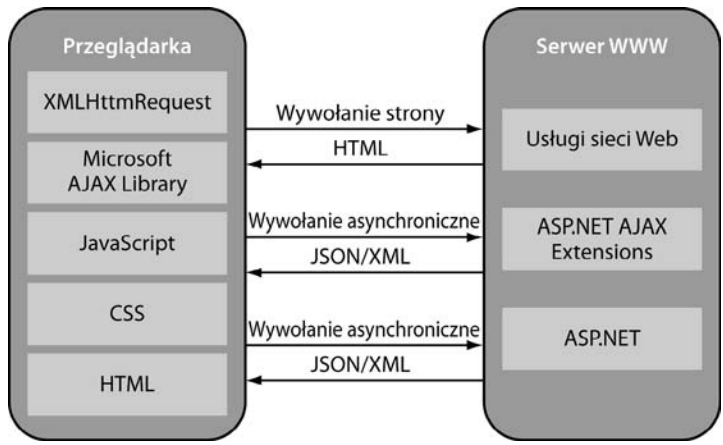
AJAX robi to dużo lepiej. Łamie ten wzór, aktualizując fragmenty stron oddzielnie, poprzez **częściowe odświeżanie strony** (ang. *partial page rendering*). Rysunek 1.1 pokazuje typową, nie-AJAX-ową serię interakcji przeglądarki i serwera. Każde wywołanie skutkuje zwróceniem całości strony, a w odpowiedzi przeglądarka aktualizuje cały widok przedstawiany użytkownikowi.

Rysunek 1.1



Na rysunku 1.2 do polepszenia wrażeń użytkownika zastosowano AJAX. Pierwsze wywołanie jest wysyłane w celu początkowej wizualizacji strony. Potem przesyłane są asynchroniczne żądania do serwera. Asynchroniczne wywołanie przesyła lub odbiera dane w tle w zupełnie niewidoczny sposób. Jest ono asynchroniczne, ponieważ interfejs użytkownika nie jest w tym czasie zablokowany, a użytkownik może kontynuować interakcję ze stroną podczas transferu danych. Tego rodzaju wywołania aktualizują stronę przyrostowo, zamiast pobierać całkiem nową. Kod JavaScript uruchomiony na kliencie reaguje na nowe dane i odpowiednio aktualizuje różne fragmenty strony. Liczba wywołań serwera może być taka sama, a w niektórych przypadkach może być ich nawet więcej, ale użytkownikowi wydaje się, że aplikacja odpowiada szybciej. Nie jest zmuszany do przerwy, choćby krótkiej, i oczekiwania na serwer, gapiąc się na biały ekran przeglądarki.

Rysunek 1.2



Prawie dziesięć lat temu zespół programistów serwera Microsoft Exchange stworzył kontrolkę ActiveX o nazwie `XmlHttpRequest`, której egzemplarze mogły być tworzone z poziomu JavaScriptu i wykorzystywane do komunikacji z serwerem. Mogła ona przebiegać bez czyszczenia ekranu i odrysowywania strony. Używając obiektu `XmlHttpRequest`, mogłeś wysyłać informacje do serwera i pobierać je stamtąd bez potrzeby tworzenia nowej strony HTML. Kod JavaScript mógł potem dynamicznie manipulować zawartością HTML na komputerze klienckim, unikając irytujących mignięć i oczekiwań zazwyczaj wiązanych z przeglądaniem

stron WWW. Ta funkcjonalność niezbyt długo ograniczała się tylko do Internet Explorera. Wkrótce również inne przeglądarki zaczęły obsługiwać obiekty `XmlHttpRequest`. Programiści mogli już zacząć pisać bardziej zaawansowane aplikacje dostępne dla różnych systemów operacyjnych.

Przeglądarki zaczęły też używać zaawansowanego modelu DOM (ang. *Document Object Model* — obiektowy model dokumentu) umożliwiającego reprezentację przeglądarki, okna, strony i zawartych w niej elementów HTML. DOM udostępniał zdarzenia i odpowiadał na akcje użytkownika, pozwalając skryptom manipulować stroną. Dynamiczny HTML (DHTML) otworzył drzwi do tworzenia zaawansowanych interfejsów w oknie przeglądarki. Programiści zaczęli pisać setki, a nawet tysiące wierszy kodu JavaScript, aby stworzyć bogate i atrakcyjne aplikacje niewymagające instalacji na komputerze i dostępne wszędzie z każdej przeglądarki. Aplikacje WWW wzniosły się na całkiem nowy poziom zaawansowania. Bez bibliotek AJAX trzeba by było pisać mnóstwo kodu JavaScript i szukać błędów w czasami subtelnych różnicach między przeglądarkami, aby osiągnąć tak wysoki poziom.

Biblioteki JAVASCRIPT a AJAX

Programiści mieli dostęp do technologii AJAX od lat i wielu z nich wykorzystywało AJAX do przesuwania granic tego, co można było zrobić z przeglądarką. Ale najbardziej przyciągają dziś do AJAX wszechstronne biblioteki skryptów i integracja z technologiami serwerowymi. Ułatwia to pisanie zaawansowanych aplikacji i pozwala uniknąć bycia ekspertem od różnych wersji JavaScriptu. Do biblioteki JavaScript na stronie HTML odwołujemy się, używając znacznika <script>:

```
<html>
  <head>
    <script src="http://wisniewscy.pl/jakisSkrypt.js"
type="text/javascript"></script>
  </head>
...
```

Skrypt jest pobierany i buforowany przez przeglądarkę. Inne strony aplikacji mogą odwoływać się do skryptu pod tym samym adresem URL, a przeglądarka nawet nie zada sobie trudu pobrania go ponownie z serwera. Funkcjonalność tego skryptu jest dostępna na stronie pobranej do przeglądarki. Biblioteka skryptów wysyłana do przeglądarki i wykorzystywana do tworzenia bardziej interaktywnej aplikacji o bogatszym interfejsie jest sercem wszystkich bibliotek AJAX.

Czym jest ASP.NET AJAX

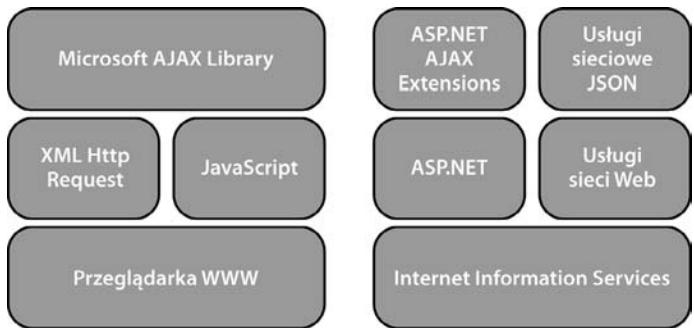
ASP.NET AJAX to nazwa stworzonego przez Microsoft rozwiązania AJAX. Odnosi się do zestawu technologii klienckich i serwerowych, które koncentrują się na ulepszeniu programowania WWW pod Visual Studio. Inne firmy mają własne rozwiązania AJAX, często zupełnie odmiennie podchodzące do zagadnienia, ale Microsoft postanowił rozszerzyć popularną technologię ASP.NET, tworząc obszerny zestaw bibliotek skryptów oraz rozszerzeń programistycznych po stronie serwera. Jeden z tych elementów, Microsoft AJAX Library, jest biblioteką JavaScript działającą na rozmaitych przeglądarkach, służącą uproszczeniu programowania w JavaScripcie. Zapewnia możliwość prostego pisania kodu w sposób obiektowy, podobnie jak w plikach kodu serwerowego ASP.NET. Można wykorzystać możliwości

Microsoft AJAX Library do operowania na DOM, dynamicznej aktualizacji fragmentów stron, asynchronicznej komunikacji z serwerem i tworzenia zaawansowanych animacji. Funkcjonalność dostępną na kliencie Czytelnik pozna w rozdziałach 6. i 7., na razie może myśleć o Microsoft AJAX Library jako o systemie typów i zestawie bibliotek klas upraszczających pisanie kodu JavaScript poprawiającego wrażenia użytkownika, manipulującego DOM i komunikującego się z serwerem WWW. Ogromną korzyścią z używania tej biblioteki jest to, że opakowuje niskopoziomowe programowanie w DHTML w model obiektowy, z którym dużo łatwiej pracować.

Drugą częścią ASP.NET AJAX są serwerowe rozszerzenia ASP.NET 2.0 AJAX Extensions. Opierają się na klasach i kontrolkach ASP.NET i korzystają z wysłanej do przeglądarki biblioteki Microsoft AJAX Library. Dzięki nim można w prosty sposób wzbogacić aplikację technologiami AJAX. Poprzez zestaw standardowych usług sieci Web ASP.NET AJAX potrafi także wykorzystać serwerowe usługi aplikacyjne, takie jak uwierzytelnianie i przechowywanie profilu użytkownika. Rozszerzenia ASP.NET 2.0 AJAX Extensions dodają nowe zachowania do istniejących kontrolek ASP.NET, jak też wprowadzają nowy zestaw kontrolek serwerowych, ułatwiających asynchroniczną aktualizację stron i korzystanie z zasobów serwera.

Serwerowe i klienckie części ASP.NET są pokazane na rysunku 1.3. ASP.NET stanowi rozszerzenie serwera WWW Microsoft Internet Information Services (IIS). ASP.NET AJAX rozbudowuje to wszystko, w tym również udostępniane przez IIS usługi sieci Web. Microsoft AJAX Library działa w przeglądarce, manipulując DOM, komunikując się asynchronicznie z serwerem WWW i wykorzystując usługi ASP.NET.

Rysunek 1.3



Instalacja ASP.NET AJAX

ASP.NET AJAX Extensions są dodatkiem do ASP.NET 2.0. Łącze do instalatora znajduje się na stronie *http://ajax.asp.net/downloads/default.aspx?tabid=47*. Po uruchomieniu zainstaluje on zestaw (ang. *assembly*) *System.Web.Extensions.dll* w buforze *Global Assembly Cache* (*gac*) Twojego komputera. Obecność w *gac* daje zestawowi uprawnienia niezbędne do zapewnienia pełnej funkcjonalności oferowanej przez ASP.NET AJAX. Jeśli na komputerze jest obecne Visual Studio 2005 lub Visual Web Developer, instalator doda także szablony projektów ASP.NET AJAX. Dzięki temu będzie można stworzyć nową witrynę korzystającą z ASP.NET AJAX. Plik *web.config* i referencje kompilacji zawierają wszystko co niezbędne, by zacząć budować aplikacje AJAX. Pakiet narzędziowy będzie zawierał kontrolkę `UpdatePanel` omówioną w rozdziale 2. oraz kontrolkę `ScriptManager` opisaną w rozdziale 5. Informacje o tym, co zawiera nowy plik *web.config*, zostaną przedstawione w rozdziale 10.

Komponent ASP.NET

Oryginalna (ASP) technologia Active Server Pages została wydana jako część składowa Internet Information Server 3.0. Została potem ulepszona przez wsparcie transakcji i dostęp do obiektów COM wraz z wydaniem NT 4.0 Option Pack prawie 10 lat temu. W tym czasie większość witryn internetowych składała się ze statycznych stron HTML. Jeśli trafiały się jakieś dynamiczne aplikacje, były to aplikacje typu CGI (ang. *Common Gateway Interface*) lub ISAPI (ang. *Internet Server API*) zazwyczaj napisane w C lub C++. Wraz z wydaniem Active Server Pages (ASP) programiści mogli zacząć używać języków JavaScript i VBScript na serwerze, wykorzystując zestaw wewnętrznych obiektów dostarczanych przez ASP. „Klasyczne ASP", jak je dzisiaj nazywamy, dostarczało obiekt sesji, zatem programiści nie musieli się przejmować bezstanową naturą HTTP. Miało obiekty `Request` i `Response` zapewniające łatwy dostęp do danych pochodzących z formularzy na kliencie oraz metodę zwracania zaktualizowanej informacji. Obiekty `Server` i `Application` stanowiły sposób dostępu do informacji z serwera WWW i wykorzystania wspólnego zestawu obiektów COM w całej aplikacji.

Chociaż klasyczne ASP było wielką wygraną programistów, odkryli oni szybko, że tworzenie złożonych aplikacji stało się trudne. Brak wsparcia dla modularyzacji powodował, że aplikacje miały tak skomplikowane współzależności w skryptach, że programiści często określali je jako „kod spaghetti". W dużym stopniu brakowało wsparcia dla debugowania oraz zaawansowanych narzędzi do projektowania aplikacji. Wydajność także była słaba, bo ASP używało powolnego mechanizmu wykonywania skryptów, a błędy składni mogły być wychwycone tylko po uruchomieniu aplikacji. ASP.NET stanowi wielki krok naprzód. Zapewnia istniejącym aplikacjom zgodność, bo zawiera ten sam zestaw wewnętrznych obiektów co klasyczne ASP, ale zamiast interpretować JavaScript i VBScript „w locie", korzysta ze skompilowanego zbioru stron i modułów napisanych w C# i VB.NET. Klasyczne ASP po prostu wykonywało skrypt po kolei od początku do końca strony; ASP.NET reprezentuje model sterowany zdarzeniami, z cyklem życia strony, upodabniając programowanie WWW do pisania zwykłej aplikacji. Obecnie zamiast dołączać oddzielne pliki JavaScript reprezentujące obiekty biznesowe, możesz stworzyć obiekty w dowolnym języku wspieranym przez .NET i mieć do nich dostęp bezpośrednio z ASP.NET.

ASP.NET z zestawu stron zawierających kod i znaczniki generuje klasę `Page`, która jest następnie kompilowana i buforowana. Przy każdym wywołaniu strony wytwarzany jest egzemplarz klasy i uruchamiany kompletny cykl życia strony. Wykonuje się zestaw zdarzeń, z których część jest przesłaniana przez wygenerowaną klasę `Page`. Kontrolki na stronie także uczestniczą w tym cyklu życia, wiążąc się z danymi z końcówki bazodanowej, odpowiadając na działania użytkownika i reagując na zmiany swojego stanu, jakie zaszły od poprzedniej odsłony strony. Aby odpowiedzieć na akcję użytkownika, programista musi tylko dostarczyć dla niej procedurę obsługi zdarzenia. Przykładowo kontrolka-przycisk udostępnia zdarzenie `Click`. Jeśli ktoś chciałby je wykorzystać, nie musi pisać kodu sprawdzającego wszystkie zmienne formularza na stronie, by dowiedzieć się, że przycisk został kliknięty. Wystarczy, że doda kod przesłaniający obsługę zdarzenia. Kod ten może aktualizować HTML strony lub właściwości i dane związane z innymi kontrolkami.

Niniejsza książka zakłada pewną znajomość środowiska ASP.NET, niezbędną do zrozumienia, w jaki sposób nowe funkcje ASP.NET AJAX rozbudowują jego funkcjonalność i dostarczają środków do wzbogacenia aplikacji WWW.

Jeśli Czytelnik nie zna jeszcze ASP.NET, warto zapoznać się z zestawem podręczników QuickStart ze strony http://www.asp.net/quickstart. Witryna ASP.NET (http://forums.asp.net/) ma także liczne fora dyskusyjne, na których można porozmawiać z kolegami programującymi w ASP.NET oraz członkami zespołu tworzącego tę platformę.

Komponent JavaScript

Technologie AJAX korzystają z powszechnego wsparcia dla JavaScriptu w nowoczesnych przeglądarkach. Ponieważ istnieje standard obsługiwany przez różne przeglądarki, możesz pisać skrypty, wiedząc, że będą działały. Ale nie zawsze tak było.

W połowie lat 90. ubiegłego wielu Netscape i Microsoft (wraz z innymi) wypracowały standard języka skryptowego, który zastosowały w swoich przeglądarkach. Ten standard nazywa się EcmaScript. Implementac a firmy Microsoft nosi nazwę JScript, ale powszechnie ten język jest nazywany JavaScriptem, tak jak go nazwał Netscape. (Nie ma nic wspólnego z Javą, ale widocznie ktoś pomyślał, że takie skojarzenie będzie korzystne dla celów marketingowych). Fragmenty kodu JavaScript są wysyłane do przeglądarki wraz z HTML i uruchamiają się wewnątrz przeglądarki użytkownika, mając wpływ na przetwarzanie strony na komputerze klienckim.

JavaScript nie jest kompilowany; jest interpretowany. Nie ma statycznego sprawdzania zgodności typów jak w C++ czy C#. Można zadeklarować zmienną, nie określając jej typu, a typ, do którego ona się odnosi, zmienić w dowolnej chwili. Dzięki temu łatwo zacząć programować w JavaScripcie, ale pozwalając na podmienianie typu zmiennej w czasie wykonania skryptu, nieuchronnie narażamy się na pewne niebezpieczeństwo. Poniższy fragment kodu pokazuje, jak łatwo zmienna może odnosić się do dowolnego typu:

```
var something = 1;
something = true;
something = "łańcuch";
```

JavaScript jest językiem dynamicznym. Typy danych mogą być rozszerzane podczas wykonywania programu przez inny kod. Oznacza to, że możesz pisać kod, który tworzy typy „w locie”. Ponieważ nie można wyegzekwować bezpieczeństwa typów danych, Twój kod może otrzymać te typy w postaci parametrów lub zwracanych wartości bez żadnego problemu. Zapewnia to ogromne możliwości i elastyczność programowania.

Podstawowymi typami danych w JavaScripcie są łańcuchy tekstowe, liczby, wartości logiczne i funkcje. Istnieje także wsparcie dla obiektów i tablic, które są kolekcjami typów podstawowych. Język zawiera kilka dodatkowych obiektów uważanych za niezbędne do wielu zadań. Można tu wymienić wsparcie dla wyrażeń regularnych oraz operacji na datach i czasie.

Do łączenia łańcuchów tekstowych w języku JavaScript używa się operatora „plus”:

```
var theEnd = "KONIEC."
var result = "Początek, " + "środek i " + theEnd;
```

W powyższym przykładzie zmienna `result` jest obecnie łańcuchem tekstowym: „`Początek, środek i KONIEC.`”.

Interpretery JavaScript używają do zapisu liczb standardu zmiennoprzecinkowego IEEE. Nie wchodząc w szczegóły, można przyjąć, że w większości zadań nie będzie z tym problemów.

Typ `Boolean` w języku JavaScript jest prawie tym, czego można by się spodziewać, ale niezupełnie. Typ ten określa, czy dane wyrażenie jest prawdziwe, czy nie, jednak wzoruje się na języku C, używając wartości całkowitych `0` i `1`.

JavaScript dopuszcza zmienne niemające wartości, zmienna może być po prostu nieokreślona, co może prowadzić do niespodziewanych rezultatów. W poniższym fragmencie JavaScriptu zadeklarowane są trzy zmienne. Wszystkie z porównań zwracają wartość `true`.

```
<script type="text/javascript">
var one = 1;
var zero = 0;
var undefinedVar;

if(one) {
    alert("1 ma wartość prawdziwą");
}

if(!zero) {
    alert("0 ma wartość fałszywą");
}

if (!undefinedVar) {
    // ten test mówi nam, że "undefinedVar" albo zawiera 0,
    // albo jest naprawdę niezdefiniowana, w obu przypadkach ma wartość fałszywą
    alert("undefinedVar ma wartość fałszywą");
}

if (one != zero) {
    alert("jeden i zero nie są tym samym");
}
</script>
```

Możesz sprawdzić, czy zmienna została zdefiniowana, w następujący sposób:

```
if ( typeof(undefinedVar ) == "undefined" ) {
    alert("undefinedVar jest niezdefiniowane");
}
```

Zmienne mogą mieć też wartość `null` i różnią się wówczas od niezdefiniowanych, bo `null` jest jednak pewną wartością.

Funkcje w JavaScripcie są również prawdziwymi typami. Mogą przyjmować argumenty i zwracać wartość. Funkcje mogą być przekazywane do innych funkcji oraz tworzone dynamicznie przez kod skryptowy.

Oto dwie równoważne definicje funkcji o nazwie `Add` pobierającej dwie zmiennej i zwracającej rezultat zastosowania operatora „plus”. Proszę zauważyć, że nie powiedziałem, że przyjmuje dwie liczby. Warto pamiętać, że zmienne JavaScript nie mają określonego typu, więc mógłbym równie dobrze przesłać dwa łańcuchy znakowe, a funkcja `Add` by je połączyła.

```
<script type="text/javascript">
function Add(x, y) {
```

```
    return x + y;
}
var AddAgain = function(x, y) { return x + y; };
</script>
```

Po stworzeniu funkcji jednym z powyższych sposobów można ją wywołać w danym (lub dowolnym zagnieżdżonym w stosunku do niego) zakresie w celu wykonania dodawania. Żaden z tych sposobów tworzenia funkcji nie ma przewagi nad drugim. Można po prostu wybrać bardziej odpowiadający.

```
<script type="text/javascript">
var result = Add(36, 24);
alert(result);  // wyświetla 60

var  stringResult = Add("Witaj", "cie.");
alert(stringResult); // wyświetla "Witajcie."
</script>
```

Obiekty i tablice są tylko kolekcjami innych typów. Typy tablicowe nie wymagają, by wartości w nich przechowywane miały nazwę; zamiast tego są dostępne przez indeks. Wartości przechowywane w obiekcie są określane przez nazwy pola lub właściwości. Obiekty mogą też zawierać funkcje (które mogą być akcesorami, funkcjami dającymi publiczny dostęp do lokalnych zmiennych), co pozwala tworzyć w kodzie JavaScript struktury danych reprezentujące rzeczywiste obiekty. W tym swego rodzaju programowaniu obiektowym brakuje koncepcji dziedziczenia typów. Microsoft AJAX Library dostarcza zestaw klas i rekomendowanych wzorców programowania, pozwalających uzyskać w języku JavaScript dziedziczenie, czyniąc bardziej naturalnym przechodzenie pomiędzy JavaScriptem a innymi językami wysokiego poziomu. Poniższy fragment kodu zawiera definicję obiektu `Album` przechowującego i zwracającego tytuł i autora albumu muzycznego. Do przechowywania informacji o kilku albumach użyta jest tablica:

```
<script type ="text/javascript">
// zdefiniuj obiekt o nazwie Album - proszę zauważyć, że nie ma on zdefiniowanego typu
Album = function (title, artist) {
    var _title = title;
    var  artist = artist;

    this.get title = function() { return  title; }
    this.get_artist = function() {return _artist; }
}

// utwórz egzemplarz obiektu, wywołując konstruktora
var albumA = new Album("Na wylot", "Fisz");
var albumB = new Album("Polepione dźwięki", "Fisz");

// utwórz tablicę przechowującą te egzemplarze (również bez określonego typu)
var albumArray = new Array();

albumArray[0] = albumA;
albumArray[1] = albumB;

// przejdź kolejno po tablicy, by pokazać tytuły albumów
for(var i = 0; i < albumArray.length; i++) {
  alert((albumArray[i]).get title()); // wywołaj akcesor get_title
}
</script>
```

Komponent usług sieci WWW

Zasadnicza koncepcja usług sieci WWW ma ogromne możliwości, wciąż się rozwija i ewoluuje. Oryginalny standard SOAP (ang. *Simple Object Access Protocol* — protokół prostego dostępu do obiektów) polega na wykorzystaniu protokołu HTTP do przesyłania danych w formacie XML z klienta do serwera i odbierania tak samo uformowanych danych. Może się to odbywać z poziomu przeglądarki WWW przy użyciu obiektu `XmlHttpRequest` lub bezpośrednio z aplikacji biurkowej lub innego serwera. Zanim usługi sieci WWW zostały powszechnie przyjęte, nierzadko zdarzało się tworzenie programów, które pobierały stronę jako dokument HTML i wyciągały z niej potrzebne dane. Ta technika nosi nazwę **przechwytywania danych ekranowych** (ang. *screen-scraping*). Powoduje ona różnorakie frustracje, ponieważ strony są wciąż uaktualniane, a klienty przechwytujące dane ekranowe, by dotrzymać kroku zmianom, muszą wciąż modyfikować kod analizujący składnię, tak by odpowiadał nowemu kodowi HTML strony źródłowej.

Prowadziło to do frustracji, gdyż witryny prezentujące dane za pomocą wizualnych stron HTML łatwo było modyfikować, a to załamywało program przechwytujący dane ekranowe, który spodziewał się danych w oryginalnym formacie. Usługi sieci WWW zostały pomyślane jako niewizualna metoda przesyłania danych przez sieć WWW i w naturalny sposób izolują zdalne wywoływanie metod od warstwy prezentacji. Obecnie, zamiast przechwytywać dane ekranowe, możesz wywołać usługę sieci WWW i otrzymać dane w formacie XML łatwym do użycia przez program.

Dzięki przesyłaniu czystych danych tekstowych w formacie XML i usunięciu elementów wizualnych informację przesyłaną przez usługi sieci WWW dużo łatwiej zanalizować składniowo niż HTML. A ponieważ XML może zawierać osadzony schemat, kod może go sprawdzić i wykorzystać do ustalenia użytych struktur i typów danych. Możesz rozszerzać schemat wysyłany ze zwracanymi danymi, nie martwiąc się, że aplikacje konsumenckie przestaną działać. Dzięki temu czytniki XML mogą w pewnym stopniu tolerować modyfikacje, które z pewnością sprawiłyby programiście przechwytującemu dane ekranowe wiele zmartwień.

Schemat danych może być rozszerzany bez potrzeby aktualizacji wszystkich aplikacji konsumenckich. Mogą one w prosty sposób pozyskać te części dokumentu XML, które chcą przetwarzać, i zignorować resztę. Przekroczono też granicę prostych formatów XML. Inaczej niż w poprzednich implementacjach usług sieci WWW, obecnie można zdefiniować kontrakty usług sieci WWW, aby zastosować wybrany format danych i użyć dowolnego z licznych protokołów sieciowych. Czynnikiem napędzającym koncepcję usług sieci WWW jest możliwość łatwego dostępu do danych z różnych aplikacji w luźno powiązany sposób, a nowa warstwa komunikacyjna Microsoft Windows Communication Foundation wprowadziła tę koncepcję na nowy poziom, umożliwiając określenie w kontrakcie protokołów sieciowych, reguł wdrażania i infrastruktury logowania, a także zapewniając wsparcie dla transakcji.

ASP.NET AJAX dostarcza zestaw obiektów zastępczych (ang. *proxy object*) w celu zapewnienia dostępu do niektórych nowych usług sieci WWW wbudowanych w ASP.NET. Dane profilu, usługi członkostwa i zarządzanie rolami mogą być łatwo dostępne z poziomu klienta. Programiści nie muszą tworzyć własnej infrastruktury wsparcia dla tych podstawowych usług aplikacyjnych. Wystarczy dodać kilka wierszy kodu, by wykorzystać zasoby serwera z poziomu kodu JavaScript uruchomionego w przeglądarce. Dzięki temu bardzo wzrósł zasięg ASP.NET, obejmując zarówno klienta, jak i serwer. Ponieważ biblioteki JavaScript zaprojektowano

tak, by były łatwe w użyciu dla programistów znających programowanie w .NET po stronie serwera, cała ta dodatkowa funkcjonalność jest wygodnie opakowana i łatwa do wykorzystania.

Komponent Dynamicznego HTML

Dynamiczny HTML nie jest oddzielną technologią, ale użyciem zestawu technologii w określony sposób. Po wywołaniu serwera WWW przeglądarka otrzymuje dokument HTML. Następnie rysuje stronę, a użytkownik może ją zobaczyć. Przeglądarka udostępnia również Document Object Model (DOM) reprezentujący strukturę wyświetlanego HTML. Do DOM może mieć dostęp kod JavaScript, który jest osadzony w stronie albo strona do niego odsyła. Na wygląd HTML wpływa zastosowanie CSS (ang. *Cascading Style Sheets* — kaskadowych arkuszy stylów), które kontrolują kolory, czcionki, pozycję, widoczność itd. Możesz dowiązać kod JavaScript do zdarzeń, które przeglądarka wywołuje, gdy użytkownicy wykonują pewne czynności jak na przykład najeżdżanie myszą na określony element lub wprowadzanie informacji w polu tekstowym. Kod JavaScript potrafi zmieniać teksty i manipulować ustawieniami CSS elementów strony. Oczywiście może także komunikować się z serwerem, pogłębiając jeszcze bardziej dynamiczną naturę strony WWW. Użytkownik widzi dynamicznie zmieniający się interfejs, odpowiadający na jego działania w czasie rzeczywistym, co ogromnie wzbogaca jego doświadczenia, a zarazem zwiększa produktywność i zadowolenie z aplikacji.

Inne biblioteki AJAX

Poza ASP.NET AJAX dostępnych do użycia w ASP.NET jest również wiele bibliotek AJAX innych dostawców, chociaż nie wszystkie były zaprojektowane właśnie w tym celu. Niektóre z nich skupiły się głównie na dostarczeniu bibliotek JavaScript wykorzystywanych z poziomu przeglądarki dla ułatwienia operacji na DOM (ang. *Document Object Model*). Inne mają pewną funkcjonalność serwerową do użycia wewnątrz stron ASP.NET (w których kontrolki serwerowe są rysowane po stronie klienta). Ten punkt pobieżnie naświetli niektóre funkcje oferowane przez te biblioteki. ASP.NET AJAX Framework może koegzystować ze skryptami i kontrolkami pochodzącymi z innych bibliotek, jednak dynamiczna natura języka JavaScript, która pozwala na rozszerzanie typów, otwiera też drogę do konfliktów. Łączenie bibliotek w wielu zastosowaniach będzie dobrze działać, ale niekiedy mogą się ze sobą kłócić.

- **Ajax.NET Professional**: Michael Schwartz stworzył Ajax.NET Professional jako narzędzie używane przede wszystkim do uproszczenia mechanizmu transportu danych, który pozwala programowi JavaScript na kliencie komunikować się z programem na serwerze. Kod serwerowy jest prosty w użyciu: trzeba jedynie zarejestrować kontrolkę na stronie i oznaczyć kilka metod kodu serwera atrybutami wyznaczającymi, które z nich mogą być wywoływane z poziomu klienta. Potem można korzystać z biblioteki skryptów do wykonywania wywołań i przesyłania danych. Biblioteka ta jest przeznaczona dla programistów dobrze znających się na DHTML, a gotowych kontrolek wizualnych nie ma zbyt wiele. Jest to bardzo lekkie rozwiązanie z minimalnym narzutem na transfer danych

i cykle procesora zarówno na kliencie, jak i serwerze. Kod źródłowy jest dostępny, a pakiet jest darmowy (*http://www.ajaxpro.info*).

- **Anthem.NET**: Anthem.NET jest projektem umieszczonym na SourceForge, gdzie użytkownicy mogą pobrać jego źródła. Jest przeznaczony dla ASP.NET 1.1 i ASP.NET 2.0. Ma zestaw kontrolek serwerowych wykorzystujących odpowiednią bibliotekę JavaScript do komunikacji z serwerem. Istnieje możliwość uzyskania informacji o stanie kontrolki podczas asynchronicznego wywołania zwrotnego. W czasie pisania tej książki strona WWW Anathem.NET (*http://anathem-dot-net.* ↪*sourceforge.net*) wskazywała, że użytkownik, który chce w pełni wykorzystać tę bibliotekę, musi być doświadczonym programistą .NET. Jednak na ogół jest ona łatwiejsza w użyciu niż Ajax.NET Professional, zwłaszcza dla programistów niezbyt biegłych w DHTML. Ten projekt jest podobny w wielu aspektach do ASP.NET AJAX, jednak nie tak wszechstronny.

- **Dojo**: zestaw narzędziowy Dojo można znaleźć pod adresem *http://dojotoolkit.com*. Jest to kliencka biblioteka do programowania AJAX niezwiązana z żadną technologią serwerową. Dojo ma system typów dla JavaScriptu i możliwość dowiązywania skryptu do zdarzeń pochodzących z obiektów JavaScript i elementów DHTML. Jedną z mocnych stron jest zaawansowane wsparcie dla dynamicznego ładowania skryptów. Można określić zależności i porządek, w jakim skrypty będą pobierane i przetwarzane.

- **Prototype**: biblioteka skryptów Prototype jest dostępna na stronie *http://prototype.conio.net*. Nie jest nakierowana na integrację z żadną technologią serwerową[1]. Ma system typów umożliwiający pisanie skryptów w bardziej obiektowy sposób oraz kilka skrótów składniowych ułatwiających pracę z tablicami w JavaScripcie oraz dostęp do elementów HTML i manipulowanie nimi. Prototype dostarcza funkcjonalność sieciową oraz metodę automatycznej aktualizacji elementu HTML rezultatami wywołania HTTP po podaniu URL. Biblioteka Prototype ma też funkcje wiązania obiektów i metod skryptu z obiektami i zdarzeniami DOM. Biblioteka koncentruje się na upraszczaniu uciążliwych i męczących zadań. Nie pomaga zbytnio w tworzeniu bogatszego interfejsu użytkownika, ale dostarcza gotowe klocki usprawniające pisanie skryptów WWW.

- **Script.aculo.us**: biblioteka Script.aculo.us znajduje się na stronie o tej samej nazwie: *http://script.acoulo.us*. Slogan jej twórców brzmi „it's about the user interface, baby!" („to dotyczy interfejsu użytkownika, dziecinko!"), co dokładnie opisuje ich nastawienie. Script.aculo.us rozbudowuje bibliotekę Prototype, rozpoczynając tam, gdzie kończy się jej zakres. Wzbogaca aplikacje o wsparcie dla techniki „przeciągnij i upuść". Ma mnóstwo kodu odpowiadającego za efekty takie jak wygaszanie lub rozjaśnianie, skalowanie, przesuwanie i inne animacje elementów DOM. Script.aculo.us ma także kontrolkę suwaka i bibliotekę do operacji na listach elementów.

- **Rico**: biblioteka Rico również rozbudowuje system Prototype. Pozwala dodawać zachowanie „przeciągnij i upuść" do elementów DOM przeglądarki. Ma też kilka kontrolek wiążących obiekty JavaScript z elementami DOM w celu operacji na

[1] Powstała jednak jako część pakietu Rails i jest mocno inspirowana użytym w tym pakiecie językiem Ruby — *przyp. tłum.*

danych. Posiada konstrukcje pokazujące i ukrywające części stron przy użyciu stylu akordeonowego (ang. *accordion*). Ma także gotowe do użycia efekty animacji, skalowania i wygaszania. Te skrypty wspomagające tworzenie interfejsu użytkownika są dostępne na stronie *http://openrico.org*.

Zachowywanie równowagi między programowaniem klienckim a serwerowym dzięki ASP.NET AJAX

Jeśli przeglądarka nie wykonuje zaawansowanego kodu JavaScript, to zasadnicza część kodu aplikacji WWW działa na serwerze. Oznacza to, że potencjalnie niewielkie aktualizacje widoku użytkownika wymagają wielu odświeżeń strony. Dzięki AJAX duża część kodu związanego z interakcją użytkownika może być przesunięta do klienta. To z kolei niesie ze sobą inne wyzwania. AJAX bywa niekiedy używany do strumieniowego przesyłania obszernych zestawów danych do przeglądarki, zarządzanego wyłącznie przez JavaScript. Chociaż JavaScript ma potężne możliwości, udogodnienia w usuwaniu błędów i opcje ich obsługi są bardzo ograniczone. Umieszczanie złożonego kodu aplikacji na kliencie może pochłonąć wiele czasu, wysiłku i cierpliwości. ASP.NET AJAX pozwala na naturalną migrację do klienta pewnych fragmentów przetwarzania aplikacji przy wykorzystaniu częściowego generowania kodu HTML, pozwalającego serwerowi kontrolować niektóre aspekty widoku strony.

Na niektórych witrynach internetowych do uruchomienia całej aplikacji wystarczy pojedyncze wywołanie strony; JavaScript i AJAX mają tam mnóstwo pracy. Niesie to ze sobą parę trudnych wyzwań. Na ogół użytkownicy spodziewają się, że przycisk *Wstecz* przywróci aplikację do stanu, w którym była przed chwilą, ale w przypadku aplikacji AJAX niekoniecznie tak jest. Klient może wysyłać pewne informacje do serwera dla potrzeb trwałego zarządzania stanem (na przykład w pamięci serwera albo w bazie danych), ale wymaga to dodatkowego kodu i szczególnej uwagi przy obsłudze błędów i przywracaniu aplikacji.

Wydaje się, że najbardziej zaawansowane i najłatwiejsze w utrzymaniu aplikacje to te, które równoważą zasoby klienta i serwera tak, by zapewnić szybkie czasy odpowiedzi i łatwy dostęp do zasobów serwera oraz zminimalizować operacje blokowania podczas pobierania nowych widoków stron.

ASP.NET AJAX łączy ze sobą cechy programowania klienta i serwera. Microsoft AJAX Library jest ukierunkowana na programowanie klienckie. Dostarcza systemu typów do obiektowego programowania w języku JavaScript. Ułatwia rejestrowanie kodu odpowiadającego na zdarzenia. Dostarcza użytecznych funkcji ułatwiających częste czynności, jak znajdowanie elementów na stronie, dołączanie procedur obsługi zdarzeń czy też dostęp do serwera. Funkcjonalność serwerowa obejmuje zarządzanie kodem JavaScript wysyłanym do klienta, oznaczanie fragmentów strony, które mają być aktualizowane asynchronicznie, tworzenie liczników dla ciągłych aktualizacji i dostęp do usług ASP.NET, takich jak dane profilu użytkownika i uwierzytelnianie.

Podsumowanie

Sieć WWW w ostatniej dekadzie przeszła drogę od zapewniania statycznej obecności do bycia domyślnym wyborem dla programistów piszących aplikacje. Aplikacje internetowe mogą uzyskać szeroki zasięg bez potrzeby zajmowania się zagadnieniami wdrażania i serwisu typowymi dla tradycyjnych programów. Ale poprzeczka dla aplikacji WWW ciągle jest podnoszona, a użytkownicy wciąż spodziewają się więcej. Technologie AJAX prowadzą aplikacje internetowe do rywalizacji z zaawansowanymi aplikacjami biurkowymi. Można skorzystać z asynchronicznej komunikacji z serwerem WWW do aktualizacji fragmentów strony bez zmuszania użytkownika do zaprzestania pracy i oczekiwania na przeładowanie i ponowne odrysowanie strony. Dynamiczny HTML pozwala tworzyć bogaty interfejs graficzny z przejściami i animacjami przy wykorzystaniu CSS do kolorów, czcionek, pozycjonowania itd.

ASP.NET AJAX zawiera bibliotekę Microsoft AJAX Library, która ułatwia pisanie kodu JavaScript dla przeglądarki i upraszcza wiele typowych zadań programowania klienckiego. Dzięki niej łatwo dołączać kod do zdarzeń DOM, pisać JavaScript w sposób obiektowy i uzyskiwać dostęp do serwera w celu uwierzytelniania, trwałego przechowywania i aktualizacji danych.

ASP.NET AJAX zawiera także rozszerzenia do wersji 2.0 .NET Framework, które mogą znacznie ulepszyć Twoją aplikację internetową. Posiada wsparcie dla zwracania danych w formacie JSON, łatwym do zużytkowania przez JavaScript przeglądarki.

Książka ta pokazuje, że funkcje klienckie i serwerowe ASP.NET AJAX pozwalają z łatwością przesunąć granice tego, co można zrobić z aplikacją WWW! Nauczy, jak asynchronicznie uaktualniać fragmenty stron i jak zarządzać skryptami używanymi w przeglądarce. Poinformuje, jak korzystać z udogodnień sieciowych, zgłębiając przy okazji temat dostępu do usług ASP.NET, takich jak uwierzytelnianie i przechowywanie profili. Przedstawi bliżej język JavaScript i sposób, w jaki rozbudowuje go do ułatwienia zadań programistycznych Microsoft AJAX Library. Pokaże również, co ASP.NET AJAX oferuje w dziedzinie wzbogacania interfejsu aplikacji WWW, jak testować takie aplikacje i usuwać z nich błędy. Zapozna też Czytelnika z niektórymi szczegółami wdrażania i debugowania aplikacji i odkryje inne zasoby dostępne do pracy z ASP.NET AJAX.