Ajax. Wzorce projektowe

Autor: Michael Mahemoff
Tłumaczenie: Tomasz Walczak
ISBN: 83-246-0761-7
Tytuł oryginału: Ajax Design Patterns
Format: B5, stron: 672

Praktyczne rozwiązania dla projektantów aplikacji sieciowych

- Projektowanie aplikacji z wykorzystaniem technologii Ajax
- Dynamiczne interfejsy użytkownika
- Usługi sieciowe

Technologia Ajax, będąca połączeniem języków XML i JavaScript, ostatnio zyskuje coraz większe uznanie wśród twórców witryn WWW i aplikacji sieciowych. Pozwala na ograniczenie do minimum komunikacji pomiędzy przeglądarką i serwerem, dzięki czemu aplikacje internetowe zachowują się niemal identycznie jak powszechnie wykorzystywane narzędzia biurkowe. Użytkownicy programów zaprojektowanych z wykorzystaniem Ajaksa nie muszą czekać na przeładowanie strony — odbywa się to w tle i nie wpływa na szybkość działania. W dobie rosnącej popularności narzędzi internetowych Ajax wydaje się być idealnym rozwiązaniem.

Czytając książkę „Ajax. Wzorce projektowe" opanujesz najlepsze praktyki tworzenia aplikacji sieciowych z wykorzystaniem tej technologii. Poznasz podstawowe wzorce projektowe oraz zasady ich stosowania w typowych aplikacjach. Nauczysz się korzystać z usług sieciowych, przesyłu strumieniowego i modelu DOM. Przeczytasz o optymalizacji kodu oraz powtórnym wykorzystywaniu jego fragmentów. Znajdziesz tu także wzorce dotyczące funkcjonalności i użyteczności witryn WWW oraz efektów graficznych. Ostatnią grupę wzorców opisanych w książce stanowią techniki diagnozowania tworzonych aplikacji.

- Geneza Ajaksa
- Podstawy projektowania w Ajaksie
- Zdalne wywoływanie poleceń
- Usługi sieciowe
- Transformacje XSLT
- Refaktoryzacja kodu
- Optymalizowanie wydajności aplikacji
- Efekty graficzne
- Użyteczne formularze
- Testowanie aplikacji

Wykorzystaj w swoich projektach sprawdzone rozwiązania

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

Spis treści

ROZDZIAŁ 3.

Projektowanie w Ajaksie — zasady i wzorce

Ajax może w znacznym stopniu zwiększyć użyteczność internetu, przy jego użyciu utworzono już wiele niezwykle ciekawych aplikacji i oczywiście jest to jedna z technologii będących obecnie „na topie". Jednak nie jest to rozwiązanie uniwersalne. Projekt zawsze trzeba tworzyć z rozwagą i dopasować go do stosowanej technologii. Dzięki monitorowaniu stanu aplikacji Ajax wciąż się uczymy, co działa, a co nie, a także jak programiści radzą sobie z wyborami dotyczącymi projektów. Niniejszy rozdział przedstawia te zagadnienia na wysokim poziomie i wprowadza wzorce, które opisują je w szczegółowy sposób.

Pożądane cechy aplikacji Ajax

Ajax związany jest z poprawą odczuć użytkownika i ma wartość dla organizacji posiadających aplikacje sieciowe i używających ich. W niniejszym podrozdziale przyjrzymy się kluczowym cechom idealnej aplikacji Ajax. Rzeczywistość stanowi, że nigdy nie będzie można skorzystać z najlepszych właściwości obu światów programowania, dlatego trzeba dokonywać wyborów na podstawie oceny znaczenia każdego atrybutu. Wzorce Ajax mają pomóc radzić sobie z tymi wyborami.

Użyteczność
: Aplikacje Ajax powinny być jak najbardziej intuicyjne, produktywne i wygodne.

Produktywność programistów
: Programowanie powinno być jak najbardziej wydajne dzięki przejrzystemu i łatwemu w pielęgnacji kodowi bazowemu.

Wydajność
: Aplikacje Ajax powinny w jak najmniejszym stopniu wykorzystywać przepustowość łączy i używać jak najmniej zasobów serwera.

Niezawodność
: Aplikacje Ajax powinny udostępniać precyzyjne informacje i zapewniać integralność danych.

Ochrona prywatności
: Choć można, i powinno się, używać w celu zwiększania komfortu pracy danych wygenerowanych przez użytkownika, należy także szanować prywatność klientów i powinni oni wiedzieć o tym, kiedy i jak aplikacja używa ich danych.

Dostępność
: Aplikacje Ajax powinny umożliwiać używanie ich przez osoby z różnymi rodzajami niepełnosprawności, w różnym wieku i z odmiennymi doświadczeniami kulturowymi.

Zgodność
: Rozszerzając pojęcie dostępności — aplikacje Ajax powinny działać w różnych przeglądarkach, urządzeniach i systemach operacyjnych.

Projektowanie w Ajaksie

Analizując istniejące aplikacje Ajax, jak również ich dowolnych ważnych przodków, można wyróżnić zbiór ważnych zasad projektowych, które przedstawia niniejszy punkt. Analiza zasad miała duży wpływ na proces wykrywania wzorców, a znajomość reguł pomaga stosować owe wzorce.

Zaczniemy od przyjrzenia się zasadom tworzenia projektów skoncentrowanych na użytkowniku, a następnie — tworzenia projektów skoncentrowanych na oprogramowaniu. Oczywiście nie można w pełni rozdzielić tych dwóch zagadnień i często wchodzą one ze sobą w konflikt. Radzenie sobie z takimi zagadnieniami to kluczowy problem stosowania wzorców. Warto przyjrzeć się ciekawej stronie internetowej, która przedstawia to zagadnienie z innej perspektywy. Witryna Ajax Mistakes (*http://swik.net/Ajax/Ajax+Mistakes*) zawiera długą listę błędów i kruczków z różnych aplikacji Ajax, jak również zbiór antywzorców utworzony początkowo przez Aleksa Boswortha, a obecnie przechowywany w wikipedii.

Zasady użyteczności

Stosuj się do standardów sieciowych
: Jeśli się wystarczająco postarasz, przy użyciu technologii Ajax możesz utworzyć bardzo niezrozumiałe strony, szczególnie jeśli użyjesz złożonych efektów graficznych. Zamiast wymyślać internet od nowa, używaj technologii Ajax do tworzenia „lepszego internetu" — rozbudowanej warstwy ponad tym, co jest obecnie dostępne. Szanuj zwyczaje, do których użytkownicy zdążyli się już przyzwyczaić.

Przeglądarka to nie komputer stacjonarny
: Kontynuując rozważania z poprzedniego punktu — Ajax to raczej bogatsza wersja tradycyjnych stron internetowych niż przystosowany do sieci rodzaj tradycyjnych aplikacji stacjonarnych. To prawda, kontrolki z tych aplikacji, na przykład suwaki, migrują w kierunku technologii Ajax, ale jedynie wtedy, kiedy ich zastosowanie ma sens w kontekście sieciowym, a często kontrolki te są dostępne w zmodyfikowanej formie. Można także zauważyć kategorie aplikacji, na przykład edytory tekstu, które są udostępniane w sieci, ale także w tym przypadku trzeba pamiętać, że najlepszymi produktami będą te, które nadają się do stosowania w sieci, a nie te będące ślepą repliką ich stacjonarnych odpowiedników.

Jeśli coś jest różne, niech będzie to oczywiste

Drobne różnice mogą mylić. Jeśli zdecydujesz, że odejście od standardu lub powszechnie stosowanego idiomu jest uzasadnione, upewnij się, iż projekt jest wystarczająco odmienny, aby nie powodował nieporozumień.

Udostępniaj oferty

Oferty (ang. *affordances — http://en.wikipedia.org/wiki/Affordance*) są równie istotne w technologii Ajax co wcześniej. Możesz udostępnić nową wyszukaną technikę przeciągania służącą do aktualizowania formularza, ale czy użytkownik będzie wiedział, że jest to możliwe? Pomocne mogą być: projekt graficzny, dynamiczne ikony i obszary opisujące stan.

Płynna, ciągła interakcja

Unikaj sekwencji start-stop obecnej w standardowych aplikacjach sieciowych. Odświeżanie całej strony rozprasza i powoduje marnowanie czasu. Jeśli w ogóle chcesz stosować tę technikę, zarezerwuj ją dla istotnych, rzadkich operacji, na przykład przechodzenia do pojęciowo nowych obszarów lub przesyłania dużych formularzy.

Dostosowywanie

Preferencje w ustawieniach aplikacji nie były dotąd ważne w sieci. Po co zajmować się personalizacją kolorów tła na witrynie sklepu internetowego, z którego użytkownik korzysta raz w miesiącu, szczególnie jeśli wymaga to przechodzenia przez żmudną sekwencję przesyłania formularzy? Jednak z niektórych aplikacji sieciowych użytkownicy może korzystać po osiem godzin dziennie, pracując przy nich, a dostosowywanie nagle staje się dużo bardziej przydatne. Dzięki sterowaniu procesem dostosowywania przez technologię Ajax personalizacja jest dużo łatwiejsza.

Niech będzie ciekawie

Dzięki technologii Ajax internet jest dużo ciekawszy niż kiedyś. Techniki takie jak efekty graficzne, przeciąganie czy okresowa aktualizacja często określa się mianem „bajerów" lub „ dodatków miłych dla oka", jakby użytkownicy nie czerpali przyjemności z ich używania. Stosowanie tych dodatków z rozwagą i, w idealnych warunkach, po ich przetestowaniu, na pewno może zwiększyć wartość nawet „poważnych" aplikacji.

Zasady projektowania oprogramowania

Poznaj język JavaScript

Głównie dzięki zainteresowaniu technologią Ajax języka JavaScript (*http://www.crockford.com/javascript/javascript.html*) nie postrzega się już jako ubogiego języka programowania, którego należy unikać za wszelką cenę. W rzeczywistości może on być bardzo wartościowy (proszę się nie śmiać!) pod warunkiem, że programiści zechcą poznać związane z nim idiomy, wzorce i wyjątki.

Tam, gdzie to konieczne, akceptuj rozwiązania prowizoryczne

Ponieważ Ajax bazuje na standardowych możliwościach przeglądarek, nie ma sposobu na obejście wielu ograniczeń nakładanych przez współczesne aplikacje tego typu. Jeśli chcesz korzystać z zalet złożonych aplikacji sieciowych, które bez żadnych modyfikacji będą działać we wszystkich współczesnych przeglądarkach, a jednocześnie uważasz użyteczność za kluczowe zagadnienie, musisz w razie konieczności używać wszelkich możliwych sztuczek. Możesz narzekać, że tworzenie programów przy użyciu technologii Ajax jest

z natury kłopotliwe i marzyć o bardziej przejrzystej technice (sam tak robię). Jednak trzeba pamiętać, że programista nie może zbyt wiele zrobić w kwestii tego, z czego większość użytkowników korzysta i będzie korzystać przez najbliższych kilka lat. Może tworzenie aplikacji sieciowych *będzie* bardziej uporządkowane, ale obecnie, w świecie rzeczywistym, najlepiej jest używać dostępnych technologii i stosować — lub wykorzystywać — wszelkie dostępne tanie zagrania, jeśli w efekcie pozwoli to udostępnić użyteczną funkcjonalność niedostępną innymi sposobami.

Uważaj na asynchroniczność
: Komunikacja między przeglądarką a serwerem w aplikacjach Ajax jest z natury asynchroniczna. Prowadzi to do kilku zagrożeń. Użytkownik może nie wiedzieć, że wywołanie się nie powiodło lub jego obsługa przekroczyła limit czasu oczekiwania. Wywołania mogą być przetwarzane w sposób nieatomowy. Przeglądarka może też utracić dotyczące wywołania informacje jeszcze przed nadejściem odpowiedzi. Jak wyjaśniam to w kilku wzorcach, dostępne są techniki służące do monitorowania i kontrolowania wywołań, dzięki czemu podobne sytuacje nie mają miejsca.

Dbaj o zgodność
: Jednym z problemów z językiem JavaScript jest przenośność. Na poziomie składni JavaScript jest dość spójnie obsługiwany w różnych przeglądarkach, ponieważ proces standaryzacji ECMA (próby zdefiniowania standardów języka JavaScript; zobacz *http://pl.wikipedia.org/wiki/ECMAScript*) jest w dużej mierze obsługiwany przez producentów wszystkich najważniejszych przeglądarek. Jednak język ten wciąż się rozwija, dlatego starsze przeglądarki po prostu nie obsługują niektórych właściwości. Ponadto poważnym problemem związanym z przenośnością pozostaje model DOM. Mimo stopniowej poprawy w ostatnich latach, wciąż występują drobne różnice, które w przyszłości mogą się pogłębić. Programowanie pod kątem zgodności oznacza jawne określanie wersji oprogramowania, dla których przeznaczony jest program, używanie przenośnych bibliotek tam, gdzie są one dostępne, i tworzenie architektury w taki sposób, aby zagadnienia związane z przenośnością były oddzielone od podstawowej logiki.

Zmniejszaj obciążenia łączy
: Jeśli sieć jest wykorzystywana często, warto poważnie się zastanowić nad rozmiarem komunikatów przekazywanych tam i z powrotem.

Zwracaj uwagę na opóźnienie
: Kiedy ludzie opisują, „jak szybkie" są ich łącza, zwykle używają miar przepustowości, na przykład „łącze 4-magabitowe". Doskonale pozwala to oszacować szybkość pobierania dużych plików, ale co z interaktywnością? Opóźnienie — czyli czas przekazywania bitów między przeglądarką a serwerem — jest tu zwykle ważniejszy niż przepustowość (*http://richui.blogspot.com/2005/09/ajax-reducing-latency-with-cdn.html*). W aplikacjach sieciowych, gdzie serwer może znajdować się na drugim końcu świata, nie można szybko zareagować na każde wciśnięcie klawisza lub ruch myszy właśnie z powodu dużego opóźnienia. Wyzwanie polega na utworzeniu aplikacji, która sprawia wrażenie reaktywnej, przy jednoczesnym zmniejszeniu częstotliwości interakcji. Techniki takie jak Blokowanie przesyłania czy Pobieranie na podstawie przewidywania wymagają podjęcia określonych decyzji, ponieważ zmniejszają częstotliwość przesyłania danych, ale zwiększają ich ilość.

Rozbijanie aplikacji na wiele warstw
: Podobnie jak w przypadku każdej architektury sieciowej aplikacje Ajax powinny składać się z wielu warstw, co pomaga rozdzielić różne zagadnienia. Ta wskazówka jest często mylnie interpretowana jako „upraszczaj warstwę prezentacji", co jest błędem prowadzącym, niestety, do beznadziejnie jałowych interfejsów użytkownika. Nie obawiaj się tworzyć bogatych, inteligentnych interfejsów użytkownika przy użyciu języka JavaScript. Upewnij się tylko, że zachowujesz dbałość o zgodność i oddzielasz logikę biznesową od logiki warstwy prezentacji. Ponadto staraj się stosować „nienatrętne skrypty JavaScript" i „nienatrętne style CSS". Oznacza to, że powinieneś dbać o przejrzystość początkowej strony HTML i dodawać referencje do zewnętrznych plików JavaScript i CSS — nie stosuj zagnieżdżonych skryptów i stylów. Także JavaScript i CSS powinny być rozdzielone. Wszędzie tam, gdzie to możliwe, skrypt powinien zmieniać sposób wyświetlania danych poprzez przełączanie klasy CSS, a nie poprzez bezpośrednie zarządzanie informacjami o stylu.

Nie przeciążaj przeglądarki
: Niestety, aplikacja Ajax będzie prawdopodobnie jednym z wielu programów działających na maszynie klienckiej. Sytuację pogarsza fakt, że język JavaScript jest dość wolny, co oznacza, że trzeba ograniczać operacje wykonywane po stronie przeglądarki.

Stosuj stopniową degradację
: Jeśli przeglądarka nie obsługuje zaawansowanych właściwości, aplikacje Ajax powinny stopniowo wyłączać opcje i używać tego, co *jest* dostępne. Doskonale byłoby zawsze udostępniać tę samą funkcjonalność, choć ze zmniejszoną liczbą dodatków. Jednak nawet jeśli musisz zrezygnować z funkcjonalności — a zdarza się to często — powinieneś to robić stopniowo (myśl pod kątem „delikatnych komunikatów o błędzie", a nie „skomplikowanych zrzutów stosu".)

Przegląd wzorców Ajax

Wzorce Ajax pokazują przykłady efektywnego zastosowania zasad w prawdziwych aplikacjach. Może się wydawać śmieszne, że mamy tak wiele wzorców dotyczących technologii Ajax, mimo iż samo pojęcie „Ajax" zostało ukute zaledwie kilka miesięcy przed rozpoczęciem pracy nad wzorcami. Jednak same pomysły nie są nowe. Wiele elementów Ajaksa było dostępnych w sieci, zanim pojawiło się opisujące je pojęcie. Zdrowa ekonomia internetu także jest bardzo pomocna, zapewniając powstawanie setek nowych witryn bazujących na technologii Ajax wraz z wartościowymi narzędziami (RSS, Technorati, Google i wikipedie) służącymi do ich wyszukiwania bezpośrednio po powstaniu.

Mając ponad 60 wzorców, warto sklasyfikować je hierarchicznie. Na najwyższym poziomie książka podzielona jest na cztery części, a każda z nich odpowiada odmiennemu obszarowi — podstawowym technologiom, programowaniu, funkcjonalności i użyteczności, a także rozwojowi aplikacji. Ponadto każda część książki podzielona jest na rozdziały, które zawierają powiązane ze sobą wzorce. Na przykład wzorce dla podstawowych technologii (część II) obejmują zdalne wywoływanie poleceń (rozdział 6.) związane z kilkoma wzorcami zdalnego wywoływania poleceń. Poniżej znajduje się skrótowy opis poszczególnych części:

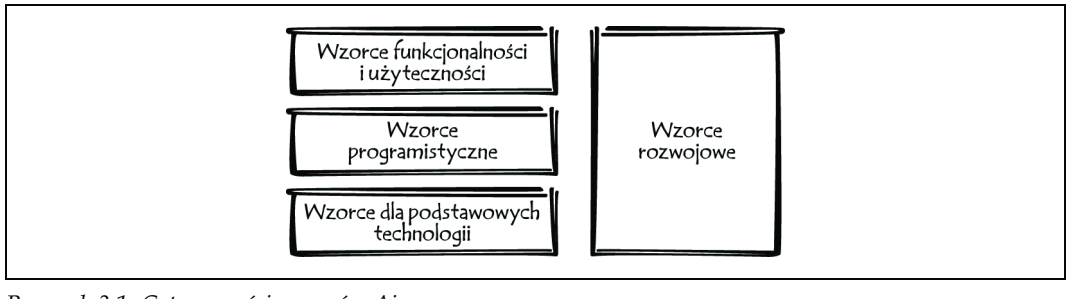
Wzorce dla podstawowych technologii (11 wzorców)
: Podstawowe technologie to cegiełki pozwalające odróżnić technologię Ajax od tradycyjnego podejścia, a ta część wyjaśnia ich typowe zastosowania.

Wzorce programistyczne (23 wzorce)
: To właściwości architektury i kodu, które służą do wdrażania wymienionych wcześniej zasad projektowych. Między innymi obejmują one projektowanie usług sieciowych, zarządzanie przepływem informacji między przeglądarką a serwerem, zapełnianie modelu DOM po otrzymaniu odpowiedzi czy optymalizację wydajności.

Wzorce funkcjonalności i użyteczności (28 wzorców)
: Te wzorce dotyczą elementów ważnych dla użytkownika, włączając w to kontrolki, techniki służące do interakcji, tworzenie struktury i zachowywanie wyglądu strony, efekty graficzne i funkcjonalność, jakiej udostępnianie umożliwia Ajax.

Wzorce rozwojowe (8 wzorców)
: Ta część zawiera wzorce związane z procesami dotyczące najlepszych praktyk rozwoju programów. Różnią się one od wszystkich wcześniejszych wzorców, które przedstawiają „obiekty" żyjące w aplikacjach Ajax. Praktyki opisane w tej części dotyczą diagnozowania problemów i testowania.

Rysunek 3.1 pokazuje, gdzie można umieścić te cztery elementy w kontekście aplikacji Ajax. Większość wzorców — te przedstawione w trzech pierwszych częściach — dotyczy *produktu*, podczas gdy pozostałe, wzorce rozwojowe, związane są z *procesem*. Spośród wzorców dotyczących produktu wzorce dla podstawowych technologii wyjaśniają, jak używać surowych technologii internetowych takich jak obiekty `XMLHttpRequest` czy model DOM. Pośredni poziom stanowią wzorce programistyczne wyjaśniające strategie stosowania podstawowych technologii. Na najwyższym poziomie znajdują się wzorce funkcjonalności i użyteczności. Podsumowując, wzorce dla podstawowych technologii to podstawa języka wzorców Ajax. Trzy pozostałe części bazują na tych wzorcach i są w miarę niezależne od siebie.



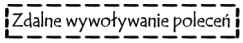
Rysunek 3.1. Cztery części wzorców Ajax

Na wewnętrznych stronach okładki książki znajdziesz alfabetycznie uporządkowaną listę wszystkich wzorców wraz z ich krótkimi opisami i numerami stron, na których są opisane. Wprowadzenie do każdej części książki oraz do każdego rozdziału także zawiera pewne informacje ogólne.

Ponadto następne strony zawierają mapy wzorców każdej z czterech ogólnych grup — dla technologii podstawowych, programistyczne, funkcjonalności i użyteczności oraz rozwojowe. Diagramy na rysunkach od 3.2 do 3.5 są utworzone zgodnie z poniższymi zasadami.

Wzorzec

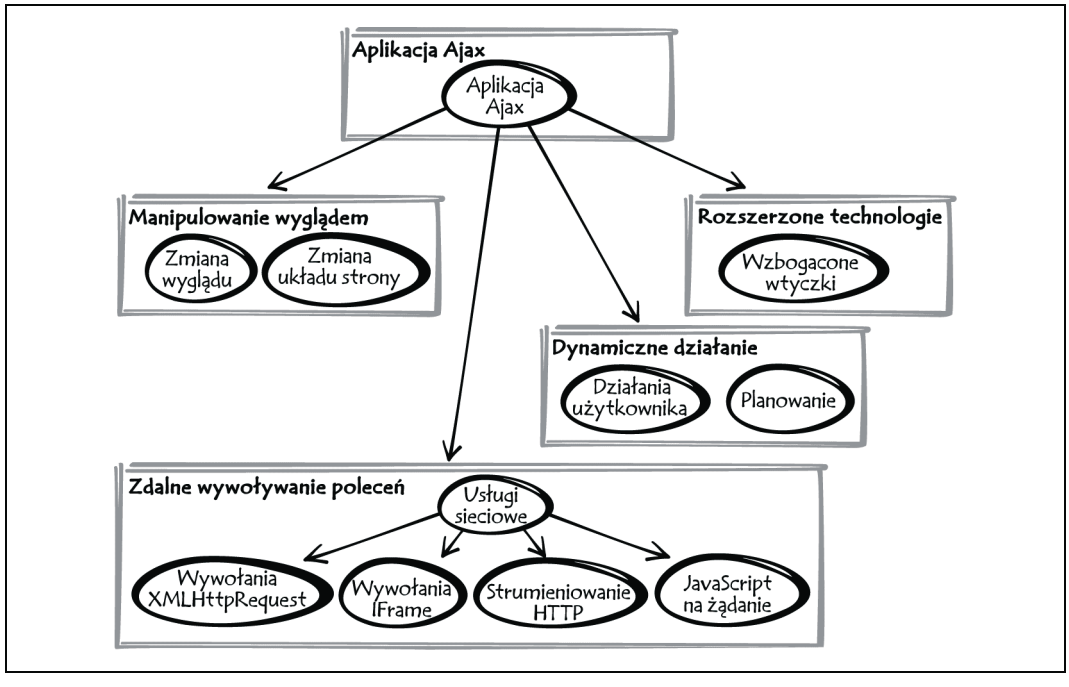
Grupa wzorców



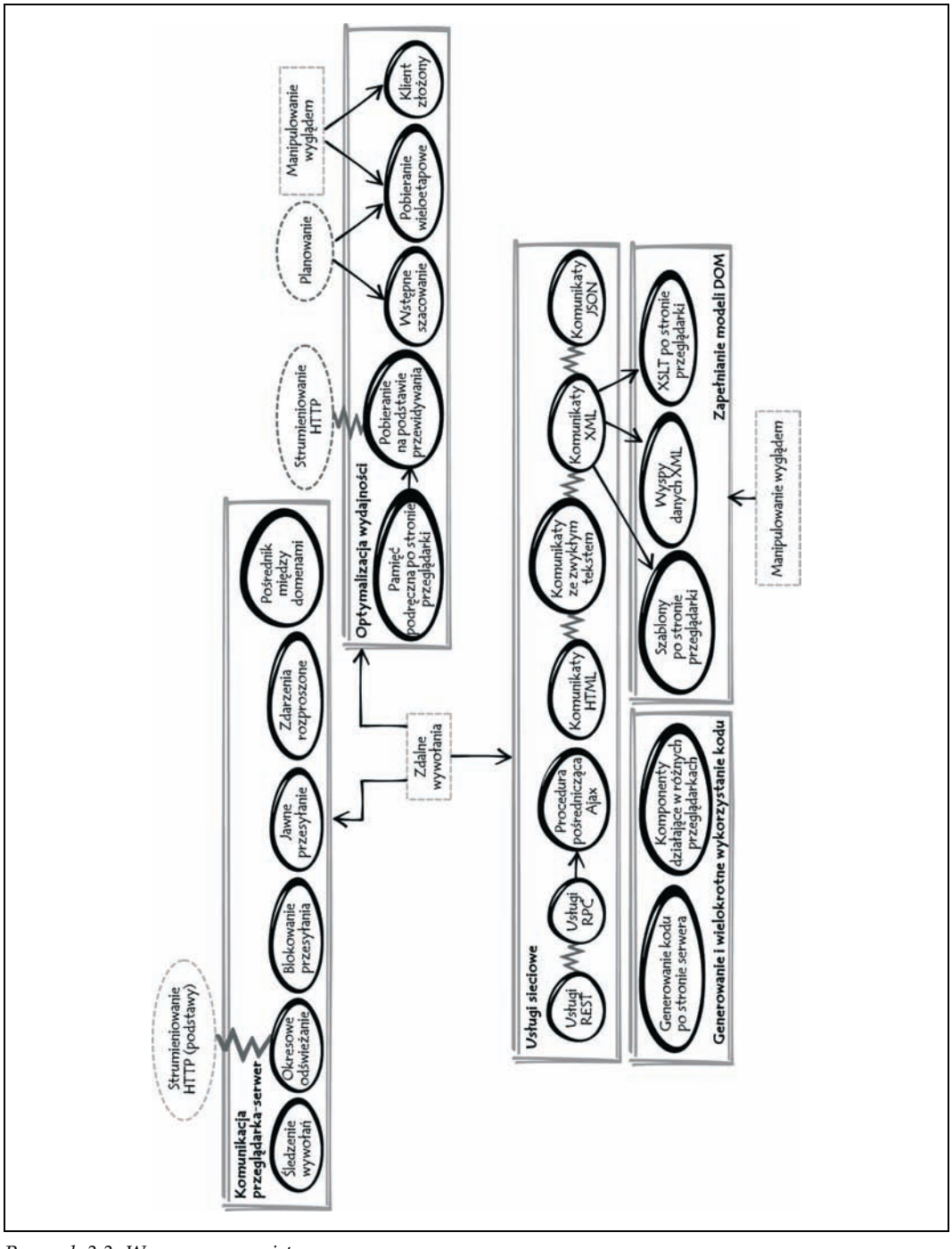
Grupa wzorców dla podstawowych technologii

Wzorzec A „prowadzi do" wzorca B

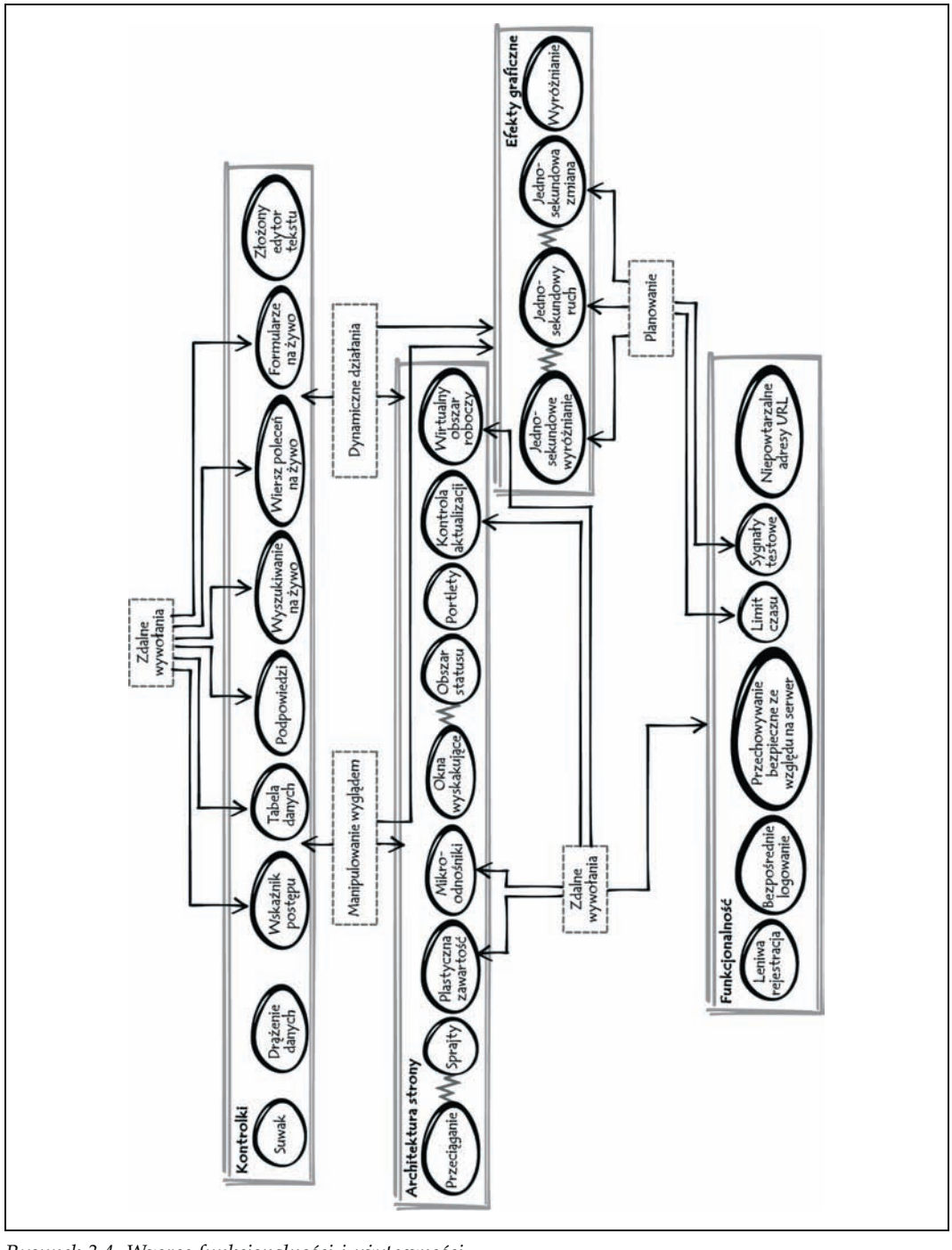
Wzorce A i B rozwiązują podobny problem, ale w inny sposób.



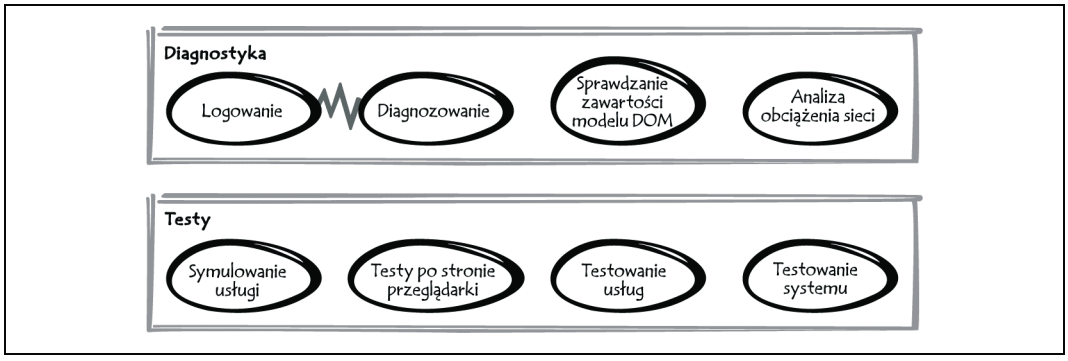
Rysunek 3.2. Wzorce dla podstawowych technologii



Rysunek 3.3. Wzorce programistyczne



Rysunek 3.4. Wzorce funkcjonalności i użyteczności



Rysunek 3.5. Wzorce projektowe

Budowa wzorca

Wszystkie wzorce mają ten sam podstawowy format, choć w niektórych przypadkach poszczególne punkty są pominięte. Niniejszy podrozdział wyjaśnia znaczenie wszystkich punktów.

Popularność
: Trzypunktowa skala, która graficznie obrazuje, jak często dany wzorzec jest stosowany w praktyce. Są to dość subiektywne szacunki, ale możesz używać poniższych ikon jako wskazówek:

  ○○○

  Informuje, że pomysł jest czysto spekulatywny.

  ●○○

  Informuje, że dostępna jest przynajmniej demonstracyjna wersja wzorca lub przypadek jego wczesnego zastosowania.

  ●●○

  Informuje, że dostępnych jest sporo zaawansowanych przykładów.

  ●●●

  Informuje, że wzorzec jest powszechnie stosowany.

Znaczniki
: Znaczniki (lub słowa kluczowe) pomagają umiejscowić wzorzec i zrozumieć jego znaczenie.

W okamgnieniu
: Zarys przedstawiający tło wzorca.

Historia zadania, historia programisty
: Historia to typowy scenariusz wyjaśniający, jak wzorzec jest używany lub jakie korzyści przyniesie użytkownikom końcowym. Dany wzorzec ma już implementację. Każda historia bazuje na „bohaterze" — fikcyjnej osobie (*http://www.evolt.org/article/Practical_Persona_Creation/4090/56111*) — dzięki czemu scenariusz jest bardziej realistyczny.

W opisach wzorców wykorzystuję małą grupę bohaterów. Ich nazwiska są mnemoniczne i odzwierciedlają role pełnione przez te osoby. To nieco tani chwyt, ale mam nadzieję, że pomoże zapamiętać charakter wszystkich postaci bez konieczności ciągłego wracania do tego punktu.

Historie we wzorcach dla podstawowych technologii (część II) oraz wzorcach funkcjonalności i użyteczności (część III) to „historie zadań". Ilustrują, w jaki sposób użytkownicy wchodzą w interakcję z systemem po jego wdrożeniu. Poniżej wymienione są osoby pojawiające się w tych historiach:

Jan Zwyczajny
: Zwykły obywatel, z żoną, 2,4 dziecka i psem nazwanym na cześć postaci z nadawanego właśnie serialu komediowego.

Zygmunt Rakieta
: Doktor — miłośnik nowych technologii.

Waldemar Bezpieczny
: Kierownik zmiany w fabryce. Często kładzie nacisk na bezpieczeństwo pracy i wydajność robotników.

Magda Spóźniona
: Kierownik wciąż opóźniającego się projektu.

Marianna Superkiecka
: Sprzedawca modnych ubrań wysokiej klasy.

Joasia Kontaktowa
: Towarzyska osoba, która spędza dużo czasu w witrynach, gdzie grupy towarzyskie tworzą własne katalogi stron, a także na towarzyskich blogach.

Leszek Ziomal
: Student z dużą ilością wolnego czasu, który poświęca na słuchanie muzyki i inne zainteresowania niezwiązane ze studiowaniem.

Krystyna Pieniążek
: Energiczny handlowiec instrumentami finansowymi zajmujący się dowolnego typu zasobami, które dają zysk.

We wzorcach projektowych (część III) znajdują się historie pokazujące, jak programista stosuje konkretne wzorce. Historie te występują we wszystkich wzorcach „programistycznych" i „rozwojowych", odzwierciedlając naturę tych punktów. W historiach pojawia się dwóch poniższych programistów:

Aleksander Program
: Starszy programista znający różne technologie Ajaksa.

Aleksandra Program
: Starszy programista, także znający różne technologie Ajaksa.

Problem
: Problem, z którym trzeba sobie poradzić.

Czynniki
: Czynniki, które odgrywają rolę w czasie rozwiązywania problemu.

Rozwiązanie
: Krótki opis rozwiązania (pierwsze zdanie tego podrozdziału), po którym następuje wyjaśnienie zastosowanej techniki.

Decyzje
: Każda decyzja jest przedstawiona jako pytanie. Decyzje *nie są* punktami z odpowiedziami na często zadawane pytania, który mają pomóc wyjaśnić rozwiązanie — takie informacje należą do punktu „Rozwiązanie". Są to „decyzje nadające się do wielokrotnego wykorzystania", czyli takie, które często trzeba podjąć w czasie wdrażania danego wzorca. Zwykle nie można podać precyzyjnego rozwiązania dylematu, ponieważ wszystkie decyzje trzeba podejmować pragmatycznie. Opis pomaga jedynie podjąć decyzję w pierwszej kolejności dzięki temu, że w ogóle wskazuje potrzebę jej dokonania, a ponadto zwraca uwagę na zmienne, które trzeba uwzględnić, a także konsekwencje wyboru jednej lub drugiej drogi.

Przykłady praktyczne
: Przykłady praktyczne demonstrują działania wzorca. Tam, gdzie nie ma przykładów praktycznych, używane są demonstracyjne aplikacje lub biblioteki. Niektóre przykłady nie są w pełni zgodne z Ajaksem, ale zostały dołączone w celu zobrazowania danej techniki.

Przykładowy kod — ilustracja refaktoryzacji
: Przykładowy kod i jego analiza.

  Jeśli system, to raczej wersja demonstracyjna wzorca Ajax niż zewnętrzna aplikacja, zwykle jest to „ilustracja refaktoryzacji", gdzie wcześniejsza wersja aplikacji jest poddawana refaktoryzacji zgodnie z danym wzorcem. Wyjaśniam to w dalszej części rozdziału. Zwróć uwagę na to, że Martin Fowler początkowo zdefiniował „refaktoryzację" jako modyfikację kodu bez zmiany działania widocznej dla użytkownika. Ja używam szerszej definicji, stosowanej w żargonie informatycznym, według której refaktoryzacja to mała, przyrostowa zmiana, która może być zauważalna.

Alternatywy
: Inne wzorce, które rozwiązują ten sam problem, ale w inny sposób. Są to często, choć nie zawsze, wzorce Ajax opisane w innym miejscu książki.

Powiązane wzorce
: Inne wzorce powiązane w jakiś sposób z omawianym, ale różne od wzorców wymienionych wśród alternatyw. Zwykle oznacza to, że powiązany wzorzec może stanowić rozszerzenie omawianego. Może to także wskazywać, że wzorce są pod jakimś względem podobne do siebie, na przykład bazują na tej samej technologii.

Metafora
: Metafora pomaga zapamiętać wzorzec. Niektóre osoby uznają ten punkt za irytujący (dlatego powinny go pomijać), a inne — za pomocny sposób pogłębienia zrozumienia wzorca i zapamiętania go.

Chcesz wiedzieć więcej?
: Odnośniki do oryginalnych źródeł i inny przydatnych materiałów.

Podziękowania
: Większość wzorców w tej kolekcji nie wzięło się z rozmyślań, ale zostało wykrytych na podstawie rozwiązań innych osób. Podobnie jak w przypadku wcześniejszych przykładów i punktu „Podziękowania" z początku książki ten punkt to miejsce, gdzie dziękuję innym za ich wkład.

Programy demonstracyjne wykorzystujące wzorce Ajax

Demonstracyjne wersje wzorców pojawiają się w wielu punktach jako ilustracje refaktoryzacji, a także jako rozwiązania. Wszystkie programy demonstracyjne są dostępne pod adresem *http://przyklady.helion.pl/ajaxwp/run/* i warto wypróbować ich działanie. Z tej samej strony można pobrać pełny kod rozwiązań. Sposób instalacji kodu opisuje dodatek B. Dostępny kod zawiera programy demonstracyjne, jak również pełne rozwiązania ćwiczeń z rozdziału 2. Programy demonstracyjne zostały przetestowane na przeglądarkach Firefox 1.5 i IE 6, choć większość przykładów powinna działać także w przeglądarkach podobnej klasy.

Kod działający po stronie serwera jest w całości napisany w PHP, ale większość fragmentów jest całkiem prosta, dlatego programiści nie powinni mieć problemów ze zrozumieniem go. Język PHP został wybrany, ponieważ na wszystkich platformach można stosunkowo łatwo dodać jego obsługę. Ponadto sam język jest dość „uniwersalny", ponieważ wszyscy programiści mający doświadczenie z programowanie aplikacji sieciowych nie powinni mieć problemów ze zrozumieniem kodu.

Programy demonstracyjne są zorganizowane według tematu refaktoryzacji. W przypadku większości przykładów dostępna jest początkowa, „zarodkowa" aplikacja Ajax. Następnie na tym samym programie demonstracyjnym wykonywanych jest kilka równoległych refaktoryzacji, każda z nich w odrębnym podkatalogu. Każda z tych refaktoryzacji może podlegać dalszej refaktoryzacji zapisywanej w podkatalogu niższego poziomu. Na witrynie powstaje w ten sposób struktura drzewiasta, a każda aplikacja ewoluuje w odmienny sposób.

Na przykład przyjrzyj się ewolucji programu Finite Cache Sum (*http://przyklady.helion.pl/ajaxwp/run/sum/xml/cached/expiry/*), do którego prowadzi ścieżka */sum/xml/cached/expiry/*.

/sum/
: W katalogu */sum/* znajduje się podstawowy program demonstracyjny. Możesz wpisać liczby, a serwer zwróci ich sumę. Jako podstawowa aplikacja Ajax — niewymagająca przesyłania formularza — program ilustruje niektóre podstawowe technologie, ale nic więcej.

/sum/xml/
: Następnie program jest poddawany refaktoryzacji, tak aby pobierać wyniki w formacie XML, ponieważ aplikacja demonstracyjna jest napisana według wzorca Komunikaty XML. Przyrostek „xml" w adresie URL informuje o możliwościach programu.

/sum/xml/cache
: Jedną z zalet nowej wersji, bazującej na XML, jest możliwość wygodnego zapisywania danych, dlatego dalsza ewolucja programu wiąże się z udostępnieniem podstawowej pamięci. Jest to ilustracja refaktoryzacji według wzorca Pamięć podręczna po stronie przeglądarki.

/sum/xml/cache/expiry
: Na koniec ulepszana jest obsługa pamięci. Tym razem idziemy na ustępstwo prawom fizyki i ograniczamy rozmiar pamięci. Nieużywane elementy są usuwane. Jest to rozwinięcie ilustracji faktoryzacji według wzorca Pamięć podręczna po stronie przeglądarki.

Wnioski

Rozdział opisuje, jak należy projektować aplikacje Ajax, a także wprowadza wzorce Ajax. Język wzorców (podobnie jak sama technologia Ajax) nie jest rozwiązaniem uniwersalnym, ale narzędziem mającym usprawnić tworzenie aplikacji sieciowych przy użyciu technologii Ajax. Możesz używać wzorców jako źródła odniesienia do szybkiego rozwiązywania problemów, ale prawdopodobnie więcej skorzystasz, jeśli wykorzystasz tę książkę do poznania powtarzających się problemów i ich rozwiązań przy użyciu technologii Ajax. Reszta książki zawiera same wzorce, podzielone na cztery części według obszarów ich zastosowań — podstawowe technologie, programowanie, funkcjonalność i użyteczność oraz rozwój oprogramowania.