helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

PHP. Programowanie obiektowe



Autor: Peter Lavin
Tłumaczenie: Julia Szajkowska
ISBN: 83-246-0697-1
Tytuł oryginału: Object Oriented PHP
Format: B5, stron: 232

Język PHP od dawna cieszy się zasłużonym uznaniem wśród twórców witryn WWW i aplikacji internetowych. Prosta składnia, ogromne możliwości i dostęp na licencji open-source przysporzyły mu liczne grono zwolenników. W najnowszej wersji PHP, oznaczonej numerem 5, twórcy języka wprowadzili kolejną ciekawą i niezwykle użyteczną cechę — programowanie obiektowe. Otwiera to przed programistami nowe możliwości tworzenia wydajnych i bezpiecznych aplikacji. Dla tych, którzy nie znają jeszcze zasad programowania obiektowego, PHP 5 jest idealnym narzędziem do nauki, bowiem stosowany w nim model obiektowy jest niezwykle prosty i czytelny.

„PHP. Programowanie obiektowe" to świetny podręcznik, dzięki któremu opanujesz możliwości i sekrety tej techniki. Czytając go, poznasz nie tylko język PHP 5, ale także ogólne zasady programowania obiektowego. Dowiesz się, czym są obiekt, klasa, konstruktor, destruktor, dziedziczenie, polimorfizm i interfejs. Nauczysz się tworzyć własne klasy i korzystać z nich w projektach. Przeczytasz także o komunikacji z bazą danych, stosowaniu języka XML, protokołu SOAP oraz zaznajomisz się z automatycznym tworzeniem dokumentacji kodu.

- Podstawy programowania obiektowego
- Obiektowe cechy PHP 5
- Tworzenie klas
- Połączenia z bazami danych
- Dziedziczenie i interfejsy
- Przetwarzanie kanałów RSS
- Korzystanie z protokołu SOAP
- Generowanie dokumentacji za pomocą klas Reflection
- Praca z SQLite

Jeśli chcesz poznać PHP 5 i zasady programowania obiektowego, ta książka jest właśnie dla Ciebie

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

Spis treści

PODZIĘKOWANIA **11**

WSTĘP **13**

1
JAKĄŻ SPLĄTANĄ SIEĆ PLECIEMY **21**
Czy naprawdę potrzebujemy obiektów? 22
Tylko język skryptowy 22
Obiekty tylko dla wielkich 23
Zostawcie go w spokoju 24
Zwiększona złożoność 24
Kultura PHP 24

2
PODSTAWY PROGRAMOWANIA OBIEKTOWEGO **27**
Klasa 28
Klasy kontra rekordy 28
Spójna całość 28
Obiekty są instancjami 29
Obiekty potrzebują modyfikatorów dostępu 29
Ponowne wykorzystanie obiektu i dziedziczenie 30
Dziedziczenie wielokrotne 30
Mieć ciastko i zjeść ciastko 30
Co dalej? 31

3
NOWE CECHY OBIEKTOWE W PHP *5* 33
Modyfikatory dostępu 34
Klasy wbudowane 34
Wyjątki 35
Klasy baz danych 35
Usługi internetowe 35
Klasy Reflection 36
Iterator 36
Wsteczna kompatybilność 36
Przekazany przez referencję 37
Prognozy 37
Co dalej? 38
Adaptacja PHP 5 38
Kompromis 39

4
POKAŻ ODROBINĘ KLASY 41
Projekt 42
Definiowanie problemu 42
Niekoniecznie kod Leonarda 43
Konstruktor 44
Odwoływanie się do zmiennych wystąpienia 44
Metody osłonowe 45
Tworzenie egzemplarza 46
Co osiągnęliśmy? 47
Ale czy poleci? 48

5
ZMIEŃMY NASZĄ KLASĘ 49
Przejście do PHP 5 49
Modyfikatory dostępu 50
Konstruktor 52
Modyfikowanie klasy 53
Przebudowa konstruktora 53
Filtrowanie zawartości 55
Przywrócenie pierwotnej postaci tablicy 57
Podsumowanie zmian 58

6
KLASA THUMBNAILIMAGE 59
Zadania Projektanta 60
Naśladowanie Projektanta 60
Pomoc ze strony funkcji PHP 60

Klasa ThumbnailImage 61
Dane składowe 61
Przebudowa konstruktora 61
Dwa sposoby skonstruowania obiektu 62
Zachowanie wewnętrzne — metody prywatne 63
Czy musi być prywatna? 64
Metoda pomocnicza 64
Metody publiczne 65
Czyszczenie pamięci 65
Wyświetlanie obrazu 66
Metody „pobierz" i „ustaw" 66
Jakość obrazu 67
Kiedy zmieniać jakość? 68
Wyświetlanie miniaturki 68
Zbierając to razem 69
Co dalej? 70

7
BUDOWANIE KLASY PAGENAVIGATOR 71
Jak zachowuje się nawigator? 72
Różne rodzaje wyszukiwań 72
Jak to będzie wyglądać? 72
Kod 73
Konstruktor 76
Stop złemu zachowaniu 76
Inne wywołania metod konstruktora 77
Metoda getNavigator 79
„Przejdź do pierwszej" i „Przejdź do poprzedniej" 79
Ciało nawigatora 80
„Przejdź do następnej" i „Przejdź do ostatniej" 81
Bieżący i ostatni numer strony 81
Co dalej? 82

8
WYKORZYSTANIE KLASY PAGENAVIGATOR 83
Zmiany w klasie DirectoryItems 84
CSS i możliwość ponownego wykorzystania 84
Stronicowanie z klasą 86
Wyświetlanie fragmentu tablicy 88
Tworzenie obiektu PageNavigator 89
Co dalej? 90

9
KLASY BAZY DANYCH 91
Korzystając z tego, co znane 91
Jedną kostkę cukru czy dwie? 92
Klasa MySQLConnect 92
Zmienne świadome klasy 93
Tworzenie kolejnych połączeń 94
Droga do celu rozpoczyna się tutaj 95
Klasa MySQLResultSet 95
Nawigator stron 97
Sortowanie, filtrowanie i wydobywanie 98
Przemierzanie zestawienia wyników 99
Nawigator potrzebuje wskazówek 99
Co po nawigatorze? 101

10
Z DZIEDZICZENIEM LEPIEJ 103
Standardowa biblioteka PHP — Standard PHP Library 104
Rozszerzenie klasy przez dziedziczenie 104
Klasa wyjątków 105
Słowo kluczowe protected 105
Słowo kluczowe final 106
Więcej metod magicznych 106
Zamiana błędów na wyjątki 107
Klasa MySQLException 108
Zmiany w klasie MySQLConnect 110
Pchnąć klasę do przodu 111
Przechwytywanie wyjątków 112
Implementacja interfejsu 113
Kilka słów o interfejsie Iterator 114
Implementacja 115
Zostawmy ją niezdefiniowaną 117
Implementacja i dostęp 118
Iteracja przez MySQLResultSet 119
Co dalej? 119

11
ZAAWANSOWANE POJĘCIA Z ZAKRESU PROGRAMOWANIA OBIEKTOWEGO 121
Klasy abstrakcyjne 122
Metody prywatne nie mogą być abstrakcyjne 122
Interfejs czy klasa czysto abstrakcyjna? 123
Polimorfizm 123
Kontrola nad użyciem funkcji 124

Klasy statyczne 125
Statyczne klasy matematyczne 125
Instancje klas statycznych 126
Jak zapobiec tworzeniu instancji klasy statycznej? 127
Wzorce projektowe 127
Wzorzec Singleton 127
Która implementacja? 129
Co dalej? 129

12
UTRZYMAJMY ŚWIEŻOŚĆ 131
SimpleXML 132
XML 132
RSS 133
Budowa pliku RSS 133
Odczyt wątku 135
Przeszukiwanie zawartości strony 136
Funkcje Google API 136
Technologia AJAX 137
Instalacja protokołu SOAP 137
Rozszerzenie SOAP 137
Klient SOAP 138
Test działania 141
Technologia AJAX wykorzystana do przeglądania wyników 142
Uproszczenie złożonych zadań 143
A może proceduralnie? 143

13
WIĘCEJ METOD MAGICZNYCH 145
Metody __get i __set 146
Czy było warto? 147
Metody __isset i __unset 148
Metoda __call 149
Metoda __autoload 150
Metody __sleep i __wakeup 151
Metoda __clone 152
Gdzie jest Włodek? 153
Operator clone 154
Klasy agregowane 154
Metoda pobierania dla danych składowych obiektu klasy złożonej 157
Klonom wstęp wzbroniony 158
Kilka słów o przeciążaniu 158

14
TWORZENIE DOKUMENTACJI DZIĘKI KLASOM REFLECTION 161
Czym są klasy reflection? 162
Grupa klas typu reflection 162
Klasa Reflection 163
Klasa ReflectionClass 164
Obiekty ReflectionMethod i ReflectionParameter 165
Funkcje wbudowane 166
Jak sformatować tekst? 166
Klasa Documenter 166
Opis klasy Documenter 167
Opisywanie metod i danych składowych 168
Konstruktor 169
Modyfikatory metod i danych składowych 169
Używanie klasy Documenter 170
Tworzenie bocznego menu klas i interfejsów 171
Formatowanie szczegółów dokumentacji 171
Formatowanie komentarzy dla klasy Documenter 173
Reflecting 174

15
ROZBUDOWA ROZSZERZENIA SQLITE 175
Krótki przegląd 176
Struktura katalogu 176
Jak się to robi? 177
Zaczynajmy 178
Tworzenie tabeli 179
Widoki 180
Wyzwalacze 180
Implementacja SQLite w PHP 182
Rozbudowa klasy SQLiteDatabase 182
Przesłonięte metody zapytania 183
Komunikaty o błędach 185
Metody zapytań 185
Metody użytkowe 189
Pobieranie metadanych 191
Używanie metadanych 191
Funkcje zdefiniowane przez użytkownika 193
Korzystanie z SQLite i jego ograniczenia 195

16
UŻYWANIE PDO 197
Wady i zalety 198
Zmienianie aplikacji SQLite 198
Zmiany w kodzie 198

Dodatkowe możliwości PDO 201
Klasa PDO 201
Klasa PDOStatement 202
Ocena 204
Czy to święty Graal? 205

A
KONFIGURACJA PHP *5* 207
Ustawienia php.ini 208
E_STRICT 209
Nie obcinamy dwa razy 210

B
TABELA KONWERSJI: PHP 4 I PHP *5* 211

SŁOWNICZEK 215

SKOROWIDZ 221

3

Nowe cechy obiektowe w PHP 5



PHP 3 POJAWIŁO SIĘ W POŁOWIE 1998 ROKU, MAJĄC JUŻ, NIEJAKO MACHINALNIE, WBUDOWANE PEWNE PODSTAWOWE MOŻLIWOŚCI PROGRAMOWANIA OBIEKTOWEGO, KTÓRE „DAWAŁY NOWE MOŻLIWOŚCI dostępu do tablic”.[1] W kolejnej, czwartej wersji, wypuszczonej w połowie 2000 roku, nie pojawiły się żadne znaczące zmiany. Nadal udostępniano podstawowe funkcje programowania obiektowego — tworzenie klas i dziedziczenie.

Wraz z pojawieniem się wersji 5. w 2004 roku, okazało się, że wiele rzeczy można było ulepszyć. W tym samym czasie Java, najpopularniejszy język obiektowy, był już obecny na rynku od prawie 10 lat. Dlaczego więc w przypadku PHP pełna obiektowość pojawiła się tak późno? Mówiąc zwięźle, powodem był fakt, że PHP to język do tworzenia stron internetowych, a potrzeba wprowadzenia obiektów w tego typu narzędziach pojawiła się stosunkowo niedawno.

[1] Por. Zeev Suraski, „Object-oriented Evolution of PHP”, dostępny na *http://www.devx.com/webdev/Article/10007/0/page/1* (otwierany 3 sierpnia 2006 r.) — *przyp. aut.*

Obsługa obiektów została doczepiona do istniejącego już języka. Dzięki temu można wybrać jedną z dwóch opcji — programowanie obiektowe lub proceduralne. Ta hybrydowość PHP powinna być postrzegana jako jego zaleta, a nie wada. Zdarzają się sytuacje, kiedy trzeba wstawić w kod strony krótki skrypt PHP, ale także takie, kiedy nieodzowne jest użycie obiektu.

Jak już dowodziłem w rozdziale 1., często rozwiązanie obiektowe jest jedynym możliwym. PHP 5 jest przygotowane na taką sytuację i zapewnia w pełni rozwinięty model obiektu, umacniając tym czołową pozycję PHP wśród języków skryptowych używanych po stronie serwera.

Ten rozdział, podobnie jak poprzedni, poświęcony będzie ogólnym zagadnieniom. Dokonam w nim przeglądu poprawek wprowadzonych w nowej wersji PHP, a bardziej szczegółowo, na przykładach, przedstawię te zagadnienia w kolejnych rozdziałach. Przy tej okazji poruszę problem kompatybilności ze starszymi wersjami.

Modyfikatory dostępu

Rozdział 2. określił modyfikatory dostępu jako zasadniczy element języka obiektowego. PHP 5 dostarcza tu wszystkiego, czego moglibyśmy wymagać. Poprzednie wersje nie zapewniały ochrony danych, wszystkie elementy klasy były powszechnie dostępne. Brak modyfikatorów dostępu najprawdopodobniej najbardziej odstręczał od pracy z obiektami w starszych wersjach PHP.

UWAGA *Pojęciem ściśle związanym z ochroną danych jest* ukrywanie informacji*. Modyfikatory dostępu pozwalają na taką operację, dzięki wprowadzeniu interfejsu (omówionego w rozdziale 2.). To samo odnosi się do* hermetyzacji *obiektu.*

Klasy wbudowane

Każdy język obiektowy posiada swoje wbudowane klasy i PHP nie jest tu wyjątkiem. PHP wprowadza bibliotekę Standard PHP Library (SPL), która jest źródłem wielu gotowych klas i interfejsów. Wersja 5.1, w zależności od konfiguracji PHP, posiada ponad 100 klas i interfejsów — całkiem rozsądny wzrost w stosunku do wersji 5.0.

Gotowe obiekty znacznie przyspieszają tempo pracy nad stroną, a rdzenne klasy pisane w C dostarczają dodatkowych możliwości. Nawet jeśli nie odpowiadają one dokładnie potrzebom programisty, to można je łatwo rozszerzyć, aż do uzyskania właściwego efektu.

UWAGA *Liczba klas nie pozwala omówić ich wszystkich w tej książce, szczególnie że nie wszystkie są dobrze udokumentowane. Skupimy się na tych, które są z jakichś względów wyjątkowo godne uwagi.*

Wyjątki

Wszystkie języki obiektowe mają wbudowaną obsługę *wyjątków*, co jest obiektowym odpowiednikiem obsługi błędów. Aby skorzystać z tej funkcji, programista posługuje się słowami kluczowymi `try`, `catch` i `throw`. *Blok instrukcji try* zawiera kod, który może spowodować wystąpienie błędów. Jeśli takowy się pojawi, jest wyrzucany i przechwytywany przez *blok catch*. Przewaga wyjątków nad błędami sprowadza się do możliwości ich centralnego obsłużenia, co nie zaśmieca tak kodu. Wyjątki znacznie zmniejszają ilość kodu odpowiedzialnego za wykrywanie i obsługę błędów. Dodatkowo gotowa, wbudowana klasa obsługująca wyjątki jest doskonałą podstawą do napisania własnej klasy obsługi wyjątków — oczywiście przez dziedziczenie. (Przejście z pułapkowania błędów do obsługi wyjątków zostanie szczegółowo opisane w dziale „Zamiana błędów na wyjątki").

Klasy baz danych

Skoro PHP ma ułatwiać tworzenie dynamicznych stron WWW, to musi obsługiwać zagadnienia związane z bazami danych. W PHP pojawia się rozszerzenie mysqli (MySQL Improved), które pozwala wykorzystać właściwości MySQL 4.1 i późniejszych wersji. Teraz można już używać wcześniej przygotowanych dla MySQL instrukcji (dzięki wbudowanemu interfejsowi OO). W zasadzie wszystko, co może być zrobione w procedurach, może być napisane obiektowo dzięki temu interfejsowi.

SQLite jest silnikiem bazy danych wbudowanym bezpośrednio w PHP. Nie jest to baza ogólnego użytku, jak MySQL, ale w niektórych przypadkach jest wręcz idealnym rozwiązaniem — często pozwala tworzyć szybsze, bardziej elastyczne i wszechstronne aplikacje. Znów dostępny jest cały interfejs obiektowy.

Wersja 5.1 i wyższe również zawierają obiekty PHP Data Objects (PDO). Jeśli zachodzi potrzeba łączenia się z zapleczem więcej niż jednej bazy danych, wtedy taki pakiet jest idealnym rozwiązaniem. Interfejs potrafiący współpracować z różnymi systemami baz danych jest osiągalny jedynie dzięki nowemu modelowi obiektu.

Ze względu na swoją wagę, bazy danych i zagadnienia z nimi związane zostaną omówione szczegółowo w dalszej części książki. Praca z bazą MySQL rozpocznie się w rozdziale 9. Rozdział 15. będzie poświęcony SQLite, a rozdział 16. — PDO.

Usługi internetowe

W PHP 5 obsługa XML (Extensible Markup Language) została powierzona zestawowi narzędzi `libxml2` (*http://www.xmlsoft.org/*). Zmieniono kod leżący u podstaw Simple API dla XML (SAX) i DOM. Dodatkowo obsługa DOM jest zgodna ze standardem wprowadzonym przez World Wide Web Consortioum.

Unifikacja obsługi XML przez `libxml2` jest skierowana na bardziej wydajną i prostszą w utrzymaniu implementację. Jest to o tyle ważne, że obsługa XML w PHP 4 jest słaba, a usługi internetowe powodują wystąpienie problemów, dla których jedynym rozwiązaniem jest programowanie obiektowe.

Stworzenie klienta SOAP i odczytanie wątku RSS w PHP 4 było poważnym wyzwaniem dla programisty. Wymagało utworzenia własnej klasy lub wykorzystania klas zewnętrznych, jak `NuSOAP` (*http://sourceforge.net/projects/nusoap/*). Podczas pracy z PHP 5 nie ma potrzeby wprowadzać zewnętrznych źródeł. Rozdział 12. pokaże, jak łatwo sprostać tym zadaniom, używając wbudowanej klasy `SOAPClient` i `SimpleXMLElement`. Znów wszystko to jest możliwe dzięki ulepszonemu modelowi obiektu.

Klasy Reflection

Klasy Reflection wbudowane w PHP 5 pozwalają dokonać introspekcji (udostępnić wraz ze skompilowanym komponentem informacje o jego metodach, właściwościach itp.) obiektu i odtworzyć kod źródłowy. Przeciętny projektant może czuć pokusę, by zignorować te klasy, ale rozdział 14. udowodni ich użyteczność podczas automatyzacji wykonywania najmniej lubianego zadania — dokumentowania kodu.

Iterator

Poza wbudowanymi klasami, PHP 5 szczyci się również wbudowanymi interfejsami. `Iterator` jest najważniejszym z nich, ponieważ wywodzi się z niego spora liczba klas i interfejsów. Szczegóły przedstawi rozdział 10.

Wsteczna kompatybilność

Wsteczna kompatybilność może być problemem, jeśli kod korzysta już z obiektów. PHP 5 wprowadza kilka „magicznych" metod. Nazwa *magicznej metody* rozpoczyna się podwójnym podkreśleniem, a to pociąga za sobą zmianę nazw innych metod i funkcji zdefiniowanych przez użytkownika, jeśli korzystają one z tej konwencji. Wszystkie one zostaną opisane w rozdziale 13. Najważniejsze z nich dotyczą zagadnienia tworzenia i niszczenia obiektu. Sposób tworzenia obiektu z PHP 4 jest nadal obsługiwany, ale lepiej jest używać nowej magicznej metody.

PHP 5 neguje pewne istniejące już funkcje, związane z programowaniem obiektowym. Przykładem może być funkcja `is_a`, która została zastąpiona nowym operatorem `isinstanceof` (zob. rozdział 14.). Ta zmiana nie ma znaczącego wpływu na wykonanie kodu w PHP 5. Jeśli użyta zostanie niezalecana funkcja, PHP wyświetli komunikat (tylko jeżeli poziom informowania o błędach ma wartość E_STRICT) — użyteczna cecha, ponieważ pokazuje, w których miejscach kod wymaga aktualizacji; szczegóły znajdują się w dodatku A. Kolejnym przykładem są funkcje `get_parent_class`, `get_class` i `get_class_methods`, które w nowej wersji zwracają wyniki rozróżniające małe i wielkie litery (ale nie wymagają takiego rozróżnienia przy podawaniu ich parametrów). Oznacza to, że kod wykorzystujący wyniki ich działania w porównaniu uwzględniającym wielkość liter, będzie wymagał zmian.

Przekazany przez referencję

Wcześniejsze przykłady pokazują niewielkie zmiany, których efekty wykrywa się względnie łatwo i równie prosto wprowadza się aktualizacje. Jednak wprowadzono też zmianę całkiem innego formatu.

Główna różnica między wersją 5. PHP a poprzednimi może być opisana stwierdzeniem, że obiekty są teraz przekazywane przez referencję. W zasadzie jest to prawdą, ale taki opis maskuje prawdziwy stan rzeczy: zmieniono sposób, w jaki operator przypisania pracuje z obiektami.

Faktycznie operator przypisania jest często wywoływany pośrednio, podczas przekazywania obiektu do funkcji lub metody, ale obecnie obiekty są przekazywane przez referencję *z powodu* niejawnego przypisania. W wersjach wcześniejszych niż PHP 5 domyślnym zachowaniem było przypisanie obiektów przez wartości i przekazanie ich do funkcji przez wartości. Takie rozwiązanie jest zupełnie odpowiednie dla funkcji pierwotnych, ale w przypadku obiektów staje się zupełnie nieopłacalne. Kopiowanie dużego obiektu, do czego w rezultacie sprowadza się przekazywanie go przez wartość, powoduje przeciążenie pamięci. W większości przypadków potrzebne jest właśnie odwołanie do oryginalnego obiektu, a nie jego kopia. Zmiana zadania operatora odwołania jest naprawdę poważnym postępem. Co więcej, silnik Zend, który jest fundamentem PHP, został przepisany specjalnie na potrzeby PHP 5.

UWAGA *W PHP 4 również istnieje możliwość przekazywania obiektów przez referencję przy użyciu operatora & i jest to uznawane za dobry zwyczaj programistyczny. Nie trzeba chyba dodawać, że taki sposób użycia operatora odwołania jest zupełnie zbędny w PHP 5. Konsekwencje wskazanej właśnie zmiany omówimy szerzej w rozdziale 13., w sekcji „Metoda __clone".*

Prognozy

Samo wymienienie kolejnych punktów wstecznej kompatybilności nie naświetla odpowiednio sprawy, która może powodować kontrowersje. Każda zmiana w ustalonym już języku programowania powoduje konflikt interesów. Wprowadzenie zmian spotyka się z ogólnym potępieniem, ale ich brak również. Weźmy na przykład niespójności w nazewnictwie funkcji — ich pozostawienie zapewnia kompatybilność ze starszymi wersjami, ale takie rozwiązanie spotka się ze słuszną krytyką za brak spójności języka.

Oczywiście zerwanie wszelkich więzi z poprzednimi wersjami spowoduje, że część istniejących już kodów przestanie działać. Decyzja, kiedy i w których miejscach przerwać ciągłość języka, nie należy do prostych, ale omawiane tu wprowadzenie przekazywania obiektów przez referencje, pomimo pewnych niedogodności, jakie powoduje, samo broni się przed krytyką. Pewnym można być tylko tego, że każda zmiana spowoduje podniesienie się głosów krytyki. Oczywiście komunikaty o pojawieniu się w kodzie niezalecanych funkcji są doskonałym sposobem na poinformowanie projektantów o nadchodzących zmianach.

Co dalej?

Czytelnik, który kupił tę książkę i dotarł w lekturze aż tutaj, z pewnością jest zainteresowany technikami programowania obiektowego. Dla osób znających PHP nauka obiektowej odmiany tego języka nie będzie skomplikowana. Biorąc pod uwagę względną prostotę modelu obiektowego w PHP, takie zadanie z pewnością będzie wymagało mniejszego wysiłku niż przejście z C do C++. Jednakże zmiana języka, czy choćby jego wersji, zawsze wymaga poświęcenia pewnego czasu i nakładu pracy, szczególnie jeśli wiąże się to ze zmianami w istniejących bibliotekach.

Omówiliśmy część zagadnień związanych z kompatybilnością w odniesieniu do programowania obiektowego. Prawie każda procedura uruchomi się bez problemu w PHP 5, nie wymagając przy tym przepisania jej na styl obiektowy.

Inna sprawą jest sprawienie, by istniejąca aplikacja wykorzystała możliwości, jakie daje obecnie PHP 5. W przypadku dużych programów taka aktualizacja może wiązać się ze znacznym nakładem pracy. Wiele z nich skorzysta na zmianach. Czytelnik, który próbował już dostosowywać do własnych potrzeb oprogramowanie typu phpBB (popularne forum o otwartym dostępie do kodu źródłowego), wie, że takie zadanie byłoby znacznie prostsze, jeśli aplikacja należałaby do gatunku obiektowych. Niestety, aktualizacja niektórych programów, oznacza pracę u samych podstaw.

Są też inne zagadnienia, poza kompatybilnością, które należałoby poruszyć. Czy samo poznanie niuansów programowania obiektowego w PHP 5 wystarczy, aby go używać? Czy w sieci działają serwery obsługujące PHP 5?

Adaptacja PHP 5

Z tego, co dotąd napisałem, wynika, że PHP 5 jest dopiero na początku swojej drogi. Ale co o nim wiemy: istnieje od ponad roku, a już wykryto w nim i naprawiono wiele błędów. Jest wersją stabilną. Wszędzie tam, gdzie projektanci mają kontrolę nad konfiguracją serwera WWW, przejście do PHP 5 będzie posunięciem korzystnym. Ale projektanci nie zawsze mają w tej kwestii wiele do powiedzenia. W sytuacji, kiedy nie mają kontroli nad hostem WWW, decyzja o aktualizacji pozostaje poza ich zasięgiem.

PHP padło ofiarą własnego sukcesu. Popularność i stabilność PHP 4 spowolniły adaptację PHP 5. PHP 4 jest w pełni rozwiniętym językiem, o który oparto mnóstwo aplikacji — open source i komercyjnych. Oczywiście pewne środowiska (szczególnie te, które stosują hosting dzielony) pozostają niechętne zmienianiu ustalonego porządku rzeczy, co owocuje spowolnionym przechodzeniem do PHP 5.

UWAGA *Inne środowiska hostingowe znacznie szybciej przyswoiły sobie PHP 5. Różne prywatne serwery wirtualne oraz hosty dedykowane (Virtual Private Server) najczęściej dopuszczają już PHP 5. VSP, będąc bezpieczniejszą i coraz tańszą usługą hostingową, zaczynają dominować na rynku.*

Kompromis

Powszechne korzystanie z PHP 5 jest tylko kwestią czasu. Jednak pisząc tę książkę, skłoniłem się ku poglądowi, że, przynajmniej na razie, wielu programistów zechce pisać swoje aplikacje zgodnie ze standardem PHP 4. Dlatego tam, gdzie to możliwe, obok kodu PHP 5 pojawia się wersja pisana w PHP 4.

W pewnym sensie PHP 5 jedynie wprowadza formalny zapis funkcji dostępnych już w PHP 4. Na przykład, rozsądniej jest wpisać metody dostępowe do zmiennych klasy w PHP 4, niż nadawać im wartość lub odczytywać bezpośrednio (choć PHP 4 dopuszcza taką możliwość). Wymaga to samodyscypliny, ale zaowocuje, umożliwiając łatwiejsze dostosowanie do standardu PHP 5. Dodanie restrykcyjnych modyfikatorów dostępu do kodu, który już posiada metody dostępowe, jest zadaniem stosunkowo prostym. Skutki założenia, że kod będzie kiedyś aktualizowany, jedynie poprawiają jego czytelność.

To wszystko, jeśli chodzi o teorię programowania obiektowego. W kolejnych rozdziałach zajmiemy się praktyką.