helion.pl
KSIĘGARNIA INTERNETOWA · WYDAWNICTWO

IDŹ DO
PRZYKŁADOWY ROZDZIAŁ
SPIS TREŚCI

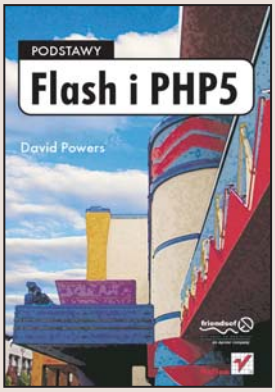
KATALOG KSIĄŻEK
KATALOG ONLINE
ZAMÓW DRUKOWANY KATALOG

TWÓJ KOSZYK
DODAJ DO KOSZYKA

CENNIK I INFORMACJE
ZAMÓW INFORMACJE O NOWOŚCIACH
ZAMÓW CENNIK

CZYTELNIA
FRAGMENTY KSIĄŻEK ONLINE

Flash i PHP5. Podstawy



Autor: David Powers
Tłumaczenie: Jarosław Dobrzański, Renata Wójcicka
ISBN: 83-246-0185-6
Tytuł oryginału: Foundation PHP5 for Flash
Format: B5, stron: 744
Przykłady na ftp: 2520 kB

Od dawna Flash wykorzystywany jest do tworzenia efektownych, interaktywnych witryn WWW i prezentacji internetowych. Język programowania Action Script, znacznie rozbudowany w najnowszych wersjach tej aplikacji, umożliwia napisanie wielu oryginalnych, ciekawych programów. Czasem jednak nawet tak olbrzymi potencjał, jakim dysponuje Action Script, okazuje się niewystarczający. Co zrobić, gdy trzeba sięgnąć do bazy danych znajdującej się na serwerze lub przechować informacje o sesji? Jak dynamicznie uaktualniać treść artykułów opublikowanych w witrynie? Odpowiedź jest oczywista, trzeba sięgnąć po inne, rewelacyjne narzędzie dla twórców stron WWW, czyli duet PHP i MySQL – dostępny nieodpłatnie w pełni obiektowy język programowania oraz stabilna i wydajna baza danych.

Książka „Flash i PHP5. Podstawy" to niezastąpiony poradnik dla tych użytkowników Flasha, którzy chcą „wycisnąć" z niego więcej, niż oferuje Action Script. Czytając ją, nauczysz się łączyć witryny WWW tworzone we Flashu ze skryptami PHP i bazą danych MySQL. Poznasz podstawy języków PHP i SQL i dowiesz się, jak skonfigurować środowisko pracy. Znajdziesz w niej informacje o możliwościach wykorzystania PHP do realizacji zadań, których wykonanie za pomocą języka Action Script byłoby niemożliwe. Te zadania to przetwarzanie danych wprowadzanych przez użytkowników, zapisywanie i odczytywanie informacji z bazy danych, zarządzanie sesjami i tworzenie mechanizmu zarządzania treścią serwisu.

- Instalacja i konfiguracja środowiska
- Wysyłanie danych z formularzy za pomocą poczty elektronicznej
- Obliczenia matematyczne w PHP
- Przetwarzanie danych tekstowych
- Tworzenie czytnika RSS
- Praca z bazami MySQL i SQLite
- Wprowadzanie informacji do bazy danych
- Obsługa sesji
- System CMS oparty o bazę danych i XML

Wykorzystaj PHP i stwórz dynamiczne witryny WWW we Flashu

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

Spis treści

O autorze **9**

O recenzencie technicznym **10**

Podziękowania **11**

Wstęp **13**

Rozdział 1. Przygotowania do pracy z PHP **21**
Dostęp filmu Flasha do danych zewnętrznych 22
Wybór właściwej technologii 22
Co oferuje PHP, Apache oraz MySQL? 24
Współpraca wszystkich elementów 25
PHP i ActionScipt: dalecy kuzyni 28
Instalowanie potrzebnego oprogramowania 29
Instalacja w systemie Windows 30
Instalacja w systemie Mac OS X 48
Konfiguracja środowiska 60
Poznawanie możliwości PHP 66

Rozdział 2. Pełne wykorzystanie możliwości Flasha **69**
Komunikacja z zewnętrznymi źródłami danych 70
Pierwsze kroki w PHP 76
Czy PHP nadaje się do tworzenia szaty graficznej strony? 76
Podstawy składni PHP 77
Wysyłanie formularza pocztą e-mail 91
Czego się dowiedziałeś do tej pory 110

Rozdział 3. Obliczenia i decyzje **111**
Dokonywanie obliczeń w PHP 112
Operatory arytmetyczne 113
Wykonywanie obliczeń we właściwej kolejności 119
Składanie operatorów 120
Mechanizmy decyzyjne w PHP 121
Użycie instrukcji if…else 121
Stosowanie operatorów porównania 122
Testowanie więcej niż jednego warunku 124
Korzystanie z operatora xor 125
Korzystanie z negacji do testowania zaprzeczeń 126

Użycie instrukcji switch w wielokrotnych warunkach ... 127
Korzystanie z operatora warunkowego ... 128
Aplikacja we Flashu — multikonwerter ... 129
Planowanie skryptu przeliczającego ... 129
Budowanie interfejsu we Flashu ... 135
Podsumowanie ... 151

Rozdział 4. Wokół ciągów znakowych ... 153
Przetwarzanie ciągów znakowych w PHP ... 154
Jak w PHP wyświetla się ciągi znakowe? ... 154
Zmiana wielkości liter ... 159
Praca z podciągami ... 161
Uzyskiwanie podciągu ... 165
Modularyzacja kodu przy użyciu funkcji ... 168
Gdzie są uruchamiane funkcje PHP? ... 168
Do czego mogą się przydać własne funkcje? ... 169
Zasięg zmiennych w PHP i w ActionScripcie ... 170
Wartość zwracana przez funkcję ... 172
Decydowanie o tym, gdzie umieścić funkcje ... 174
Wykańczanie skryptu multikonwertera ... 174
Formatowanie głównych jednostek miar ... 175
Galony, pinty i litry ... 175
Kilogramy, funty i kamienie ... 179
Metry, stopy i jardy ... 180
Udoskonalanie projektu multikonwertera ... 182
Dalsze rozwijanie projektu ... 183
Przetwarzanie wejściowych danych użytkownika ... 184
Obcinanie początkowych i końcowych białych znaków ... 184
Obcinanie znaczników HTML ... 185
Usuwanie lewych ukośników ... 185
Korzystanie z wyrażeń regularnych do rozpoznawania wzorców ... 186
Udoskonalanie aplikacji z formularzem oceniającym ... 191
Chwila odpoczynku ... 195

Rozdział 5. Inteligentne wykorzystywanie tablic i pętli ... 197
Podstawy korzystania z tablic oraz pętli ... 198
Umieszczanie elementów w tablicy ... 198
Grupowanie podobnych jednostek w tablicy wielowymiarowej ... 199
Korzystanie z pętli do wykonywania powtarzalnych zadań ... 200
Tworzenie tablic w PHP ... 200
Tablice indeksowe: porządkowanie na podstawie numeru ... 201
Tablice asocjacyjne: porządkowanie według nazw ... 205
Długość tablicy: istota różnicy pomiędzy tablicami w PHP a w ActionScripcie ... 209
Tablice wielowymiarowe, czyli zagnieżdżone ... 209
Łączenie w pętli powtarzających się zadań ... 212
Przeglądanie tablicy za pomocą pętli foreach ... 212
Pełne wykorzystywanie pętli ... 213
Najprostsze pętle: while oraz do ... 215
Kończenie działania pętli ... 216
Zagnieżdżanie pętli ... 217
Bezpieczne przekazywanie danych w zmiennej $_POST ... 219
Dlaczego ustawienie register_globals jest tak istotne? ... 221
Manipulowanie tablicami ... 222
Dzielenie i łączenie tablic ... 222
Sortowanie tablic ... 227

Budowanie czytnika wiadomości RSS ... 229
Do czego służy kanał RSS? ... 230
Analiza kanału RSS ... 231
Przetwarzanie kodu RSS w MagpieRSS ... 233
Wyświetlenie w filmie Flasha połączonych kanałów RSS ... 244
Usuwanie elementów HTML, których Flash nie potrafi zinterpretować ... 250
Opublikowanie czytnika wiadomości w internecie ... 251
Raport postępów ... 252

Rozdział 6. PHP i bazy danych: poszerzanie aplikacji ... 253
Dlaczego MySQL? ... 254
Wady MySQL ... 254
Zalety MySQL ... 255
Wybór właściwej wersji MySQL ... 257
Wybór właściwej licencji oraz jej koszt ... 259
SQLite jako alternatywa ... 260
Zalety SQLite ... 260
Wady SQLite ... 260
Wybór właściwego systemu baz danych ... 261
Instalacja MySQL w systemie Windows ... 262
Zmiana domyślnego typu tabel w Windows Essentials ... 276
Ręczne uruchamianie oraz zatrzymywanie MySQL w systemie Windows ... 278
Uruchamianie MySQL Monitor w systemie Windows ... 280
Konfiguracja MySQL w systemie Mac OS X ... 282
Praca z programem MySQL Monitor (Windows i Mac) ... 284
Utworzenie pierwszej bazy danych w MySQL ... 285
Pobieranie danych z zewnętrznego pliku ... 295
Korzystanie z graficznego interfejsu MySQL ... 298
phpMyAdmin: złoty staruszek ... 298
MySQL Administrator a MySQL Query Browser: inteligentne programy ... 303
Co nas dalej czeka? ... 305

Rozdział 7. Zabawa słowami ... 307
Tworzenie elementów graficznych gry ... 308
Używanie PHP do komunikacji z bazą danych ... 313
Klasa kopiująca działanie mysqli w dowolnej konfiguracji ... 313
Korzystanie z obiektowego interfejsu mysqli ... 314
Tworzenie klas PHP 5 i korzystanie z nich ... 329
Nazywanie i deklarowanie klas ... 330
Tworzenie właściwości klasy ... 330
Użycie konstruktora ... 331
Tworzenie metod klasy ... 333
Dostęp do metod publicznych ... 333
Udoskonalanie wybierania słów poprzez zastosowanie SQL i PHP ... 334
Tworzenie mechanizmu wyświetlającego liczbę punktów ... 343
Wprowadzenie do klasy SharedObject Flasha ... 344
Inne sposoby na wzbogacenie gry ... 347
Obsługa błędów bazy danych ... 347
Ustawienie różnych poziomów gry ... 351
SQLite: alternatywny system baz danych ... 352
Podstawy SQLite ... 352
Sprawdzenie, czy baza SQLite posiada odpowiednie uprawnienia ... 353
Dostosowywanie skryptu gry do bazy SQLite ... 363
Porównanie MySQL z SQLite ... 364
Ciągle naprzód ... 364

Rozdział 8. Tworzenie bazy danych rejestrujących się użytkowników 365
Różne rodzaje baz danych 366
Jednopoziomowe bazy danych — najprostszy typ baz danych 366
Elastyczność relacyjnych baz danych 368
Różne formaty składowania danych MySQL 373
Wybór właściwego typu kolumny 374
Typy kolumn w MySQL 375
Wartości domyślne oraz NULL 379
Wybór odpowiednich ustawień języka 380
Tworzenie systemu rejestrowania użytkownika 381
Rejestrowanie użytkowników w MySQL 381
Co zrobić, gdy nie działa poprawnie? 405
Chwila oddechu 406

Rozdział 9. Ochrona danych przy użyciu mechanizmu sesji 409
Śledzenie preferencji użytkownika za pomocą mechanizmu sesji PHP 410
Sieć jest środowiskiem bezstanowym 410
W jaki sposób działa sesja? 412
Podstawy sesji PHP 414
Korzystanie z mechanizmu sesji do ograniczania dostępu do strony 415
Inne zastosowania sesji 428
Podsumowanie 429

Rozdział 10. Sprawowanie kontroli przy użyciu systemu nadzorującego zawartość bazy 431
Cztery podstawowe polecenia SQL 432
SELECT 432
INSERT 434
UPDATE 434
DELETE 435
Tworzenie prostego systemu kontrolowania zawartości bazy danych 436
Tworzenie interfejsu aplikacji 436
Tworzenie skryptu aplikacji 444
Zabezpieczanie systemu zarządzania bazą danych 491
Dodawanie dodatkowej kolumny do tabeli 493
Mamy już solidne podstawy 495

Rozdział 11. Praca z datami 497
Jak ActionScript, PHP oraz MySQL przetwarzają daty? 498
Niepoprawne znaczniki czasowe 499
Tworzenie znacznika czasowego 501
Formatowanie dat w PHP 507
Praca z datami w MySQL 510
Korzystanie z dat w obliczeniach 510
Odnalezienie i utworzenie rekordów na podstawie tymczasowych kryteriów 517
Przetwarzanie dat w danych wejściowych użytkownika 518
Formatowanie dat pobranych z danych wejściowych użytkownika 519
Sprawdzenie w PHP poprawności daty 520
Tworzenie selektora daty we Flashu dla MySQL 522
Prawie u mety 536

Rozdział 12. Korzystanie z wielu tabel i XML 537
Projektowanie struktury tabel 538
Ustalanie podstawowych wymagań 538
Normalizacja tabel 539

Projektowanie tabeli przechowującej dane o książkach ... 539
Projektowanie tabel reprezentujących autorów i wydawnictwa ... 542
Połączenie wszystkiego w całość — tabela wyszukiwania ... 542
Przygotowania do budowy bazy danych księgarni ... 543
Tworzenie bazy i nadawanie uprawnień użytkownikom ... 543
Ogólny przegląd projektu ... 547
Wypełnianie bazy przykładowymi danymi ... 547
Szybki test systemu CMS ... 547
Tworzenie struktury bazy danych ... 549
Definiowanie tabeli books ... 550
Definiowanie tabel authors i publishers ... 553
Tworzenie tabeli wyszukiwania ... 554
Tworzenie systemu zarządzania zawartością bazy ... 555
Ustalanie podstawowej struktury ... 555
Budowanie prostych formularzy w XHTML ... 557
Uaktywnienie formularzy za pomocą kodu PHP ... 571
Wstawianie danych o nowych autorach i wydawnictwach ... 572
Tworzenie listy autorów ... 576
Tworzenie listy wydawnictw ... 579
Edycja danych autorów i wydawnictw ... 580
Wstawianie danych o nowych książkach ... 587
Pobieranie danych przechowywanych w kilku tabelach ... 596
Unikanie niejednoznacznych odwołań do kolumn ... 597
Stosowanie złączenia pełnego ... 597
Zastosowanie złączenia lewego do szukania niepełnych trafień ... 601
Prace wykończeniowe nad systemem CMS ... 602
Zarządzanie informacjami o książkach w bazie ... 602
Usuwanie rekordów z więcej niż jednej tabeli ... 613
Zachowywanie integralności odwołań przy operacjach usuwania rekordów ... 615
Aktualizacja kilku rekordów ... 622
Zastosowanie SimpleXML do odczytu danych XML ... 624
Wyświetlanie zawartości bazy danych we Flashu ... 630
Przygotowywanie bazy danych ... 630
Komunikacja z bazą danych poprzez PHP ... 630
Tworzenie interfejsu we Flashu ... 637
Tworzenie kodu ActionScript ładującego rezultaty z bazy ... 638
Długa droga za nami ... 644

Dodatek A Gdy PHP i MySQL nie chcą ze sobą współpracować ... 645

Dodatek B Dostosowanie skryptów do wersji 1.0 ActionScriptu ... 669

Dodatek C Instalacja starszych wersji MySQL w systemie Windows ... 675

Dodatek D Stosowanie innych języków w MySQL ... 685

Dodatek E Podstawowe zasady konserwacji bazy danych MySQL ... 695

Skorowidz ... 715

Rozdział 2.

Pełne wykorzystanie możliwości Flasha

Czym będziemy się zajmować:

- zastosowanie `LoadVars` do komunikowania się z zewnętrznymi źródłami danych;
- bliższa analiza składni PHP;
- potwierdzenie odebrania zmiennych z Flasha przez skrypty PHP;
- sprawdzenie danych z wyjścia PHP przed wysłaniem wyników do Flasha;
- sprawdzenie poprawności danych wejściowych;
- użycie PHP do wysłania formularza we Flashu pocztą e-mail.

Filmy Flasha stanowią zamknięte w sobie światy. Mimo że można do nich załadowywać pliki JPG, pliki dźwiękowe oraz sekwencje wideo, to jedyna komunikacja klipów z zewnętrznym środowiskiem polega na załadowywaniu strony w oknie przeglądarki. Dzięki połączeniu pliku Flasha z kodem PHP zyskasz bardzo wiele nowych możliwości, między innymi łączenie się z bazą danych, jednak najpierw musisz poznać podstawy składni PHP.

Przede wszystkim musisz znać zasady przesyłania danych do filmów Flasha, a także sposób ich uzyskiwania z plików flashowych. Równie istotna jest kontrola danych wejściowych i wyjściowych w programie. Jak się przekonasz podczas pracy z zewnętrznymi danymi, nie zawsze można wykorzystać polecenie `trace` do sprawdzenia wartości zmiennych. Pokażę kilka metod ominięcia tego problemu.

Pod koniec tego rozdziału przedstawię, w jaki sposób można wykorzystać zdobytą wiedzę do utworzenia we Flashu i PHP formularza, który po wypełnieniu zostanie wysłany pocztą elektroniczną do wskazanej skrzynki pocztowej.

Komunikacja z zewnętrznymi źródłami danych

Istnieje kilka sposobów na komunikowanie się filmów Flasha z zewnętrznymi źródłami danych. Do najważniejszych należą:

1. `LoadVars`: umożliwia łatwe eksportowanie i importowanie zmiennych. Wymaga użycia Flash Playera w wersji 6. lub wyższej.
2. `XML`: bardzo przydatna klasa umożliwiająca obróbkę zawartości dokumentów XML, a także wysyłanie danych sformatowanych w XML-u do plików Flasha. Wymaga użycia Flasha w wersji 5. lub wyższej.
3. `FlashVars`: umożliwia komunikację tylko w jednym kierunku poprzez przeniesienie zmiennych ze strony WWW do nowo utworzonych plików SWF. Wymaga użycia Flasha w wersji 5. lub wyższej.
4. `loadVariables()`: ta globalna funkcja została początkowo zaprojektowana do obustronnej wymiany danych pomiędzy filmami Flasha a zewnętrznym środowiskiem. Obecnie jest zastępowana bardziej funkcjonalną klasą `LoadVars`. Została po raz pierwszy użyta we Flash Playerze 4 i jest obsługiwana aż do obecnej wersji 8.

W tej książce będę się zajmował przede wszystkim pierwszym z tych sposobów — zastosowaniem klasy `LoadVars`. Jest to jedna z najbardziej efektywnych i najłatwiejszych w użyciu metod. Pozwala na ścisłe określenie, która zmienna zostanie wysłana do serwera, dzięki czemu oszczędza się czas i przepustowość łączy. Ponadto funkcja ta jest powiązana z wieloma różnymi zdarzeniami, które ułatwiają przesyłanie danych. Choć starsze funkcje są nadal obsługiwane przez Flash Playera 7, to nie wiadomo, czy w przyszłości nadal będą częścią ActionScriptu. Jeśli jesteś doświadczonym użytkownikiem Flasha, będziesz bardzo zadowolony ze zmian oferowanych przez tę funkcję. Jeśli natomiast jesteś nowicjuszem, przyjmij do wiadomości, że funkcje `loadVariables()` oraz `loadVariablesNum()` są już mocno przestarzałe.

Klasa XML jest istotna dla programistów zajmujących się dokumentami XML, jednak wymaga dobrej znajomości obiektowego modelu dokumentu XML (w skrócie *DOM*). Jej dość dużą wadą jest abstrakcyjny sposób opisywania dokumentów. Na szczęście w PHP 5 obróbka prostych dokumentów XML jest o wiele mniej skomplikowana. Toteż pokażę, jak należy zamieniać dokumenty XML na zmienne we Flashu przed przesłaniem ich dalej. Rozwiązanie to jest często szybsze niż przetwarzanie dokumentu XML w języku ActionScript.

Klasa `FlashVars` została zaprojektowana dla programistów tworzących zwykłe strony HTML (importuje zmienne podczas pierwszego załadowania klipu Flasha do przeglądarki), zatem nie będziemy się tą metodą szczególnie zajmować.

Warto teraz zapoznać się z narzędziem, z którego będziesz często korzystać — z klasą `LoadVars`.

Zastosowanie klasy LoadVars do pobierania danych we Flashu

Mimo nazwy, `LoadVars` umożliwia dwukierunkowe komunikowanie się — może zarówno pobierać, jak i wysyłać dane. Klasy tej będziemy używać jednak tylko do pobierania zmiennych, a właściwie jednej zmiennej.

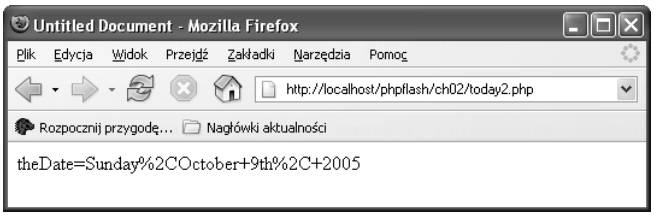
> Wszystkie utworzone skrypty należy umieszczać w folderze *phpflash*, który jest podfolderem (utworzyłeś go podczas lektury rozdziału 1.) głównego katalogu serwera. W przypadku systemu Windows ścieżka dostępu do tego podfolderu wygląda następująco: *C:\htdocs\phpflash*, natomiast użytkownicy systemu Mac OS będą używać ścieżki *Macintosh HD:username:Sites:phpflash*.

1. Utwórz nowy folder wewnątrz katalogu *phpflash* i nazwij go *ch02*. Następnie zapisz kopię pliku *today.php* z poprzedniego rozdziału jako *today2.php*.
2. Otwórz teraz plik *today2.php* i zmodyfikuj go w następujący sposób (dodatkowy kod oznaczono pogrubioną czcionką):

   ```
   <?php
    echo 'theDate='.urlencode(date('l,F jS, Y'));
   ?>
   ```

   Co prawda, jest to niewielka zmiana, jednak należy zwrócić uwagę na kilka kwestii. Nie będę teraz omawiał wszystkiego po kolei, skupię się jedynie na sposobie przekazania zmiennej z PHP do Flasha (przetwarzaniem dat zajmiemy się w rozdziale 11.). Chciałbym zwrócić Ci jednak uwagę na pewien szczegół, mianowicie na polecenie `echo`. W PHP jest to jedna z głównych metod służąca do wyświetlania danych wyjściowych w przeglądarce i z niej też będziemy korzystać w niniejszej książce. Zasady stosowania tej metody szczegółowo opisano w rozdziale 4.

   Należy się upewnić, że plik składa się wyłącznie z kodu pokazanego wyżej. **Skrypty przedstawione w tej książce (o ile nie zostało to wyraźnie powiedziane) nie powinny być zagnieżdżane w znacznikach HTML lub XHTML. Nie powinny także zawierać definicji typu dokumentu `Doctype`. Teraz PHP jest wykorzystywany wyłącznie jako język przetwarzania po stronie serwera — nie korzystasz z niego do tworzenia stron WWW.**
3. Załaduj do przeglądarki plik *today2.php* — pamiętaj o właściwej ścieżce dostępu (*http://localhost* w przypadku systemu Windows oraz *http://localhost/~username* w przypadku systemów Mac OS). Pasek adresu będzie wyglądał następująco (użytkownicy Dreamweavera mogą po prostu nacisnąć przycisk *F12*):

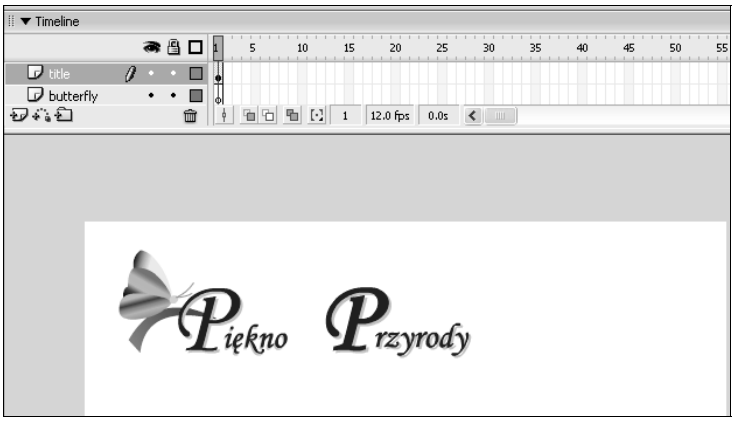


> Jeśli ujrzysz pusty ekran, może być to spowodowane przez dwa czynniki — ewentualny błąd w kopiowaniu skryptu konfiguracyjnego i ustawienie zmiennej `display_errors` na `Off`. W takim przypadku należy powrócić do poprzedniego rozdziału i postępować zgodnie z przedstawionymi tam wskazówkami.
>
> Jeśli natomiast został wyświetlony komunikat o błędzie, to bardzo dobrze! Oznacza to, że uzyskałeś odpowiedź z programu. Nie próbuj jednak wnikać, co ona oznacza (błędy w PHP oraz metody postępowania z nimi zostały omówione szczegółowo w dodatku A). Po prostu sprawdź kod jeszcze raz, zwracając szczególną uwagę na brakujące znaki: cudzysłowy, nawiasy lub kropki. Można także pobrać prawidłowy skrypt z serwera FTP.

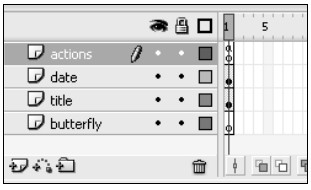
4. Zapewne sądzisz, że wynik działania skryptu jest mniej interesujący niż w przypadku pierwszej wersji, gdyż przedstawiona data jest zupełnie nieczytelna. Z pewnością takie wrażenie odniesie zwykły odbiorca, jednak symbol `%2C` nasuwa skojarzenie, że wyjście to nie zostało zaprojektowane dla ludzi, lecz dla komputera. Data została sformatowana według zasad kodowania URL (w podobny sposób używa się ciągu znakowego na końcu adresu URL). Zmienne ładowane do filmów Flasha muszą być ujęte w takim właśnie formacie. Wyświetlenie ich w oknie przeglądarki będzie dobrym testem sprawdzenia działania skryptu PHP. Jeśli zmienne nie są właściwie sformatowane, pobranie ich z zewnętrznego źródła okaże się bezcelowe. Nawet jeśli uruchomisz polecenie `trace` w kodzie ActionScript, uzyskasz jedynie niejasny komunikat o błędzie. A zatem, jeśli napotkasz problemy z ładowaniem zmiennych ze skryptu PHP, zawsze przeprowadzaj test ich prawidłowości w taki właśnie sposób.

   Gdy już uzyskasz pewność, że *today2.php* wyświetla datę poprawnie (spacje zostały zastąpione znakiem +, natomiast przecinek został zastąpiony `%2C`) i nie ma spacji w sąsiedztwie znaku równości, możesz załadować dane wyjściowe skryptu PHP do filmu Flasha.

5. Otwórz plik *biodiverse01.fla* znajdujący się w materiałach źródłowych dołączonych do tego rozdziału i zapisz go jako *biodiverse.fla*. Plik zawiera logo z motylkiem oraz nagłówek. Skorzystałem na początku z czcionki *Monotype Corsiva*, jednak zmieniłem ją ostatecznie na grafikę, gdyż czcionka ta nie jest powszechnie dostępna w systemach. Jeśli chcesz, możesz utworzyć nowy dokument Flasha, w którym dodasz swój własny nagłówek internetowy.

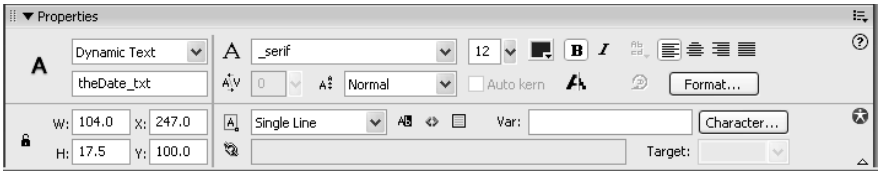


6. Nad warstwą `title` utwórz dwie nowe warstwy — jedną nazwij `date`, drugą `actions`. Zwróć uwagę, żeby przez przypadek nie umieścić grafik w warstwie `actions`.



> Jeśli poznałeś zasady kodowania w ActionScript dzięki lekturze książki Sham Bhangala, zapewne wiesz, że skrypty powinny się znajdować w klatce kluczowej warstwy przeznaczonej tylko i wyłącznie na kod ActionScript. Warstwę tę należy umieścić nad pozostałymi warstwami. Najczęściej akcje wywołuje się właśnie z tej warstwy (choć niektórzy wolą korzystać ze skryptów). Jeśli przez przypadek umieścisz tu grafikę, jej poszczególne elementy będą mogły być wywoływane w nieprawidłowej kolejności. Zablokowanie warstwy pomoże ustrzec się przed tym niebezpieczeństwem. Mimo jej zablokowania nadal można ją zaznaczać i dołączać skrypty w odpowiednim panelu.

7. W warstwie `date` wstaw dynamiczne pole tekstowe i nazwij je `theDate_txt`. Możesz je umieścić w dowolnym miejscu. Poniższy zrzut ekranu prezentuje ustawienia, których użyłem aby wyrównać pole `date` zostało wyrównane do litery `P` słowa `Przyrody`. Ustawiłem także rozmiar czcionki na 12 pt, jej typ na *_serif*, a styl na `bold`, natomiast kolor na `#730F73`. Poważny problem z dynamicznymi polami tekstowymi polega na tym, że trudno jest dokładnie ustawić ich szerokość, toteż w tym przypadku skorzystałem z kodu ActionScript.



8. Kliknij teraz warstwę `actions` i otwórz panel *Actions* (*F9*). Wpisz następujący kod:

```
//utwórz i zastosuj format tekstowy do daty:
var dateDisplay:TextFormat = new TextFormat();
dateDisplay.font = "Georgia,Times,_serif";
theDate_txt.setNewTextFormat(dateDisplay);
theDate_txt.autoSize = "left";
```

W ten sposób utworzono obiekt `TextFormat` i przypisano go do dynamicznego pola tekstowego `theDate_txt`. Większość formatowania dokonano w panelu *Properties*, jednak warto w tym przypadku użyć bardziej konkretnej czcionki niż domyślna *_serif*. Dopiero wtedy, gdy system nie będzie mógł odnaleźć żadnej z podanych czcionek, zastosuje domyślną czcionkę *_serif*.

Jeśli właściwość `autoSize` pola `theDate_txt` zostanie ustawiona na `left`, pole tekstowe automatycznie dostosuje swoją wielkość do wyświetlania pełnej daty wyrównanej do lewej strony. Przed zmianą tej właściwości należy jednak upewnić się, że będzie wystarczająco dużo miejsca na wyświetlenie najdłuższego możliwego tekstu. W przypadku dat dość łatwo można przewidzieć, jak dużo miejsca będzie potrzebne.

9. Teraz nadeszła pora na ważną część ActionScriptu, która będzie tworzyć instancję `LoadVars`, a następnie pobierać dane ze skryptu PHP. Poniżej poprzednio wpisanego kodu umieść poniższy fragment:

```
//utwórz nową instancję klasy LoadVars, która będzie pobierać dane ze skryptu PHP
var getDate:LoadVars = new LoadVars();
//załaduj datę z pliku PHP
getDate.load("http://localhost/phpflash/ch02/today2.php");
//przypisz właściwość theDate klasy LoadVars do pola tekstowego
getDate.onLoad = function() {
theDate_txt.text = this.theDate;
};
```

Zwróć uwagę, że adres *localhost* przy odnoszeniu się do pliku *today2.php* został tu użyty tylko **w celu lokalnego testowania**. Gdy umieścisz swój plik na zewnętrznym serwerze, będziesz musiał zastąpić odniesienie do *localhost* właściwą ścieżką (absolutną lub relatywną) do tego pliku.

Jeśli pracujesz na komputerze Macintosh, adres URL w drugiej linii kodu będzie następujący: *http://localhost/~username/phpflash/ch02/today2.php*. Od teraz jednak będę podawać adres właściwy tylko dla systemu Windows, gdyż ten dla komputera z systemem Mac OS jest analogiczny. Użytkownicy tych maszyn powinni pamiętać jedynie o umieszczaniu swojej nazwy użytkownika we właściwym miejscu (po znaku tyldy).

Choć jest możliwe testowanie filmów Flasha bez użycia powyższej techniki, to metoda ta spowoduje załadowywanie plików SWF do przeglądarki za każdym razem, gdy dokona się w nim zmian. Nie tylko jest to bardziej czasochłonne, ale także pozbawia możliwości skorzystania z funkcji `trace` do sprawdzenia wartości zmiennych we Flashu.

10. Przetestuj plik poprzez wciśnięcie skrótu klawiaturowego *CTRL+ENTER* (lub *CTRL+RETURN* w przypadku systemów Mac OS). Możliwe, że będziesz musiał chwilę poczekać, zanim Flash nawiąże kontakt z serwerem Apache. W końcu zobaczysz następujący komunikat:

Jeśli napotkasz po drodze jakieś problemy, porównaj swoje pliki z *biodiverse02.fla* oraz *today2.php*, które znajdują się w kodach źródłowych dołączonych do tej książki. Prawdopodobnie przyczyną błędu jest zagnieżdżenie kodu PHP wewnątrz kodu XHTML. Pamiętaj, że w tym przypadku PHP służy nie do tworzenia niezależnych stron internetowych, lecz do tworzenia zwykłych skryptów.

Choć dodanie daty do pliku filmowego nie jest czymś szczególnie efektownym, to jednak należy zwrócić uwagę, że w tych kilku liniach kodu zawarta jest ta sama funkcjonalność, co w o wiele obszerniejszym kodzie ActionScript. Funkcja `date()` (którą omówimy bardziej szczegółowo w rozdziale 11.) używa jedynie pięciu znaków do sformatowania bieżącej daty. To samo w ActionScripcie byłoby możliwe dopiero po utworzeniu tablic przechowujących nazwy dni tygodnia, miesięcy, nie wspominając o konieczności utworzenia osobnej funkcji dodającej przyrostek do daty (st, nd, rd lub th)[1]. Kolejna istotna różnica polega też na tym, że data jest generowana po stronie serwera, a nie klienta. Oczywiście, trzeba zapewnić poprawne działanie zegara serwera, jednak i tak jest to bezpieczniejsze niż opieranie się na poprawności działania zegara po stronie klienta (jeśli nie korzystasz ze środowiska anglojęzycznego, funkcja `strftime()` opisana w rozdziale 11. sformatuje datę oraz czas według zasad wybranego języka, natomiast na wyjściu funkcji `date()` data zawsze będzie podana w języku angielskim.

Jednak najważniejszą sprawą jest fakt, że utworzyłeś dynamiczne łącze do zewnętrznego źródła danych i wiesz już, w jaki sposób należy importować zewnętrzne dane za pomocą klasy `LoadVars`. Eksportowanie danych jest równie łatwe. Zajmę się tym przy okazji tłumaczenia zasad działania klas.

Teraz natomiast przedstawię podstawy składni PHP.

[1] Jeśli chce się zachować format daty w języku angielskim — *przyp. tłum.*

Pierwsze kroki w PHP

Dość często ludzie zastanawiają się, co oznacza skrót PHP. Przez pierwsze trzy lata jego istnienia rozwinięciem tego akronimu była nazwa *Personal Home Page* (co oznacza *osobista strona domowa*). Gdy w 1998 r. opublikowano wersję 3., uznano, że nazwa ta nie jest adekwatna do funkcji języka, więc zmieniono ją na *PHP:Hypertext Preprocessor*. Jest to ulubione rozwinięcie rekursywne wśród społeczności *open source* (np. *GNU — Gnu's Not Unix*).

Wraz z opublikowaniem wersji 5. PHP, język ten stał się zaawansowanym językiem programowania z bardzo rozbudowanymi możliwościami obiektowymi (jednak nie uwzględniono tutaj przestrzeni nazw ani przeciążania typów). Jednym z powodów, dla których nauka PHP jest tak łatwa, jest fakt, że obiektowość jest tylko jedną z opcji języka, toteż można zacząć naukę samego PHP, a następnie — po poznaniu podstaw — szybko przejść do tworzenia obiektów.

Czy PHP nadaje się do tworzenia szaty graficznej strony?

Do najpowszechniejszych zastosowań PHP należy dynamiczne tworzenie zawartości strony. Fragmenty PHP mogą być osadzone bezpośrednio w kodzie HTML bądź XHTML. O ile tylko mechanizm PHP będzie w stanie zinterpretować zawartość dokumentu PHP, strona będzie wyświetlana w przeglądarce jak każdy inny dokument HTML.

Spójrzmy na prosty przykład — wiele witryn posiada na dole strony informację o prawach autorskich. Zamiast ręcznego zmieniania roku, można skorzystać ze znanej nam już funkcji `date()`. Wystarczy w odpowiednim miejscu umieścić następujący kod XHTML:

```
<div id="footer">&copy; 1999-<?php echo date('Y'); ?>
David Powers </div>
```

Jeśli strona zawiera dołączony arkusz stylów, będzie mogła wyglądać następująco (pełny kod oraz arkusz stylów można znaleźć w pliku *copyright.php*, znajdującym się w kodach źródłowych dołączonych do książki):

© 1999-2005 David Powers

Jeśli zegar na serwerze jest prawidłowo ustawiony, druga liczba zawsze będzie numerem bieżącego roku, który będzie się zmieniać o północy 31 grudnia. Gdy spojrzysz na kod źródłowy strony w oknie przeglądarki, zauważysz, że cały kod źródłowy PHP zniknął:

```
<div id="footer">&copy; 1999-2005 David Powers</div>
```

Mechanizm PHP analizuje kod strony i wysyła już przetworzony z powrotem do dynamicznie tworzonej strony kod XHTML. Również Flash podczas publikowania filmu dynamicznie tworzy stronę HTML (lub także XHTML, począwszy od wersji MX 2004). Linię pokazaną wyżej mógłbyś bez problemu dodać do zwykłej strony HTML przed kończącym znacznikiem `</body>`. Wystarczyłoby tylko zmienić rozszerzenie z *.html* na *.php*.

Pracując z przykładami znajdującymi się w książce przez większość czasu będziesz wykorzystywać PHP do formatowania danych, wysyłanych następnie do filmu Flasha, jednak warto dodać, że techniki omawiane w książce stosują się także do stron XHTML. Od momentu wydania wersji 4.3.0 PHP, technologia ta może być używana także jako język poleceń, jakkolwiek w tej książce nie będziemy się zajmować tym aspektem. Jeśli chcesz dowiedzieć się czegoś więcej o zastosowaniu PHP w pisaniu skryptów powłoki, zapoznaj się z zawartością strony z dokumentacją PHP (*www.php.net/manual/pl/features.commandline.php*).

Podstawy składni PHP

Podobnie jak języki naturalne, języki komputerowe posiadają zbiór zasad gramatycznych, tworzących składnię. Zasad tych należy przestrzegać. Komputery są szczególnie wrażliwe na precyzję konstruowanych zdań. Szczęśliwie składnia PHP nie jest trudna do opanowania. Posiada dużo cech wspólnych z ActionScriptem, jednak w wielu miejscach jest znacząco różna.

Ze względu na charakter zagadnienia, możliwe, że większość informacji przedstawionych w tym rozdziale wyda Ci się zbyt sucha. Jednak trzeba je przyswoić, zanim przejdziemy dalej. Mogę zagwarantować, że po uważnej lekturze tego rozdziału nie tylko zdobędziesz dużo nowych wiadomości, ale także utworzysz pierwszy w tej książce przydatny program — formularz oceniający we Flashu i PHP.

Nadawanie nazwom stron odpowiednich rozszerzeń

Każda strona PHP musi posiadać nazwę o odpowiednim rozszerzeniu (np. *.php* — *index.php*). Bez rozszerzenia serwer nie będzie w stanie przetworzyć zawartych w dokumencie instrukcji PHP, a przeglądarka po prostu wyświetli stronę tak, jakby kod PHP był zwykłym tekstem. W ten sposób pewne istotne informacje mogą w łatwy sposób zostać przypadkowo opublikowane (np. hasła do baz danych).

Choć można tak skonfigurować serwer, aby traktował pliki o innych rozszerzeniach nazw tak samo jak skrypty PHP, to jednak nie jest to zalecane rozwiązanie. Jeśli strona nie będzie zawierała kodu, który powinien zostać przetworzony przez parser PHP, cenne zasoby komputera będą marnowane.

Korzystanie ze znaczników PHP

Nawet jeśli korzystasz z odpowiednich rozszerzeń nazw plików, cały kod PHP musi zostać ujęty w odpowiednie znaczniki. Prawdopodobnie zdążyłeś już się zorientować na podstawie podanych wcześniej przykładów, że kod PHP zaczyna się znacznikiem

<?php i kończy ?>. Mechanizm przetwarzania PHP zajmuje się wyłącznie zawartością, która znajduje się pomiędzy tymi znacznikami. Wszystkie pozostałe elementy są ignorowane przez parser PHP.

Niestandardowe znaczniki PHP. Istnieje kilka innych metod osadzania kodu PHP na stronie. Mimo wszystko nie polecam stosowania którejkolwiek z nich, jednak dla ścisłości omówię je pokrótce.

- **Krótkie znaczniki**: pierwotnie kod PHP oznaczano poprzez <? oraz ?>. Mimo zalety zwięzłości, otwierający znacznik mógł być w łatwy sposób pomylony z instrukcjami XML, toteż nie zaleca się dłużej jego stosowania. Krótkie znaczniki są domyślnie włączone w PHP 5, jednak administrator systemu może je wyłączyć. Dlatego aby zachować pewność, że skrypt będzie przenośny, lepiej używać znacznika <?php.
- **Znaczniki w stylu ASP**: aby zachować zgodność z niektórymi edytorami HTML, które nie potrafią poprawnie interpretować znaczników PHP, parser, po zmianie konfiguracji, może również interpretować znaczniki <% oraz %> jako rozpoczynające i kończące znaczniki PHP, niemniej jednak nie warto zmieniać domyślnych ustawień.
- **Bezpośrednie wyświetlanie**: niezależnie od używanego formatu znaczników, PHP oferuje skrótowy sposób na umieszczenie kodu PHP w dokumencie — jest nim symbol <?= dla znacznika otwierającego. Przykładowo <?= date ('Y'); ?> może zastąpić <?php echo date ('Y'); ?>. Choć jest to pożyteczny sposób, to jednak posiada ograniczone zastosowanie i zawiedzie, jeśli w ustawieniach wyłączono krótkie znaczniki.
- **Znaczniki skryptowe**: jest to najbardziej dosłowna metoda zaznaczenia obecności kodu PHP na stronie: <script language="PHP"></script>. Taki sposób dołączania kodu PHP zaleca się jedynie wtedy, gdy dany edytor HTML nie obsługuje żadnej z poprzednich metod.

Komentowanie kodu

Jeśli posiadasz doświadczenie w programowaniu w ActionScripcie, zapewne rozumiesz konieczność dodawania komentarzy do kodu. Najkrócej mówiąc, komentarze po prostu opisują zadania poszczególnych części kodu. Podczas pracy zespołowej umożliwiają rozeznanie się w wykonanej już pracy, natomiast indywidualnym programistom komentarze ułatwiają debugowanie i modyfikowanie programów, szczególnie gdy powracają do nich po kilku miesiącach przerwy.

W PHP istnieje kilka metod dodawania komentarzy — wszystkie zostały zaczerpnięte z innych języków programowania. Pierwsze dwa są takie same jak w ActionScripcie.

Komentarze w stylu C++. Początek komentarza jest oznaczony przez dwa prawe ukośniki.

```
//To jest komentarz: następna linia wyświetli bieżący rok
echo date('Y'); //To także jest komentarz
```

Komentarze w stylu C. Parser PHP będzie ignorował kod, który pojawi się pomiędzy znakami /* oraz */, bez względu na liczbę linii zawartych między tymi symbolami. Dzięki temu można nie tylko dodawać komentarz do kodu, ale także — w razie potrzeby — uniemożliwiać wykonywanie wybranych fragmentów skryptu.

```
/* To jest komentarz,
który się rozciągnie
na wiele linii.
Można także w ten sposób
chwilowo wyłączać
fragmenty kodu. */
```

Komentarze w stylu powłoki systemu. PHP traktuje znak (#) tak samo jak dwa prawe ukośniki (//). Ten typ komentarza został zapożyczony z języków skryptowych. Ponieważ znak ten jest łatwo dostrzegalny w kodzie, dobrze się nadaje do oznaczania pewnych fragmentów programu.

```
#To jest komentarz
echo date('Y'); #To także jest komentarz
###################
# Sekcja baz danych
###################
```

Doświadczeni programiści ActionScriptu, podczas dołączania zewnętrznych plików do kodu PHP, zapewne zwrócą szczególną uwagę na to odmienne użycie znaku #. W ActionScripcie znak # jest wykorzystywany w dyrektywach kompilatora, natomiast w PHP sprawia, że dana linia jest ignorowana.

```
#include "myFile.as" //poprawny kod ActionScript
#include "myFile.php" //taka linia zostanie zignorowana w PHP
```

Zapewne pamiętasz z poprzedniego rozdziału, że plik konfiguracyjny PHP (*php.ini*) wykorzystuje znak średnika do zakomentowania linii. Jest to jedyne miejsce, w którym komentarze w PHP mają taką postać. Jak się dowiesz w następnym podrozdziale, w zwykłych skryptach PHP znak średnika posiada zupełnie inne znaczenie.

Korzystanie z nawiasów oraz średników

Podobnie jak ActionScript, PHP składa się z serii poleceń, które są wykonywane sekwencyjnie (o ile nie znajdują się wewnątrz funkcji — wówczas w obu tych językach jest wykonywana najpierw funkcja, a następnie kod znajdujący się poniżej jej wywołania). Każde polecenie w PHP musi być zakończone znakiem średnika.

> Opuszczanie średników na końcu poleceń jest jednym z najpowszechniejszych błędów popełnianych przez początkujących programistów PHP. Choć w ActionScripcie instrukcja zostanie wykonana nawet i bez tego średnika, to PHP zatrzyma działanie skryptu i zwróci komunikat o błędzie. Jedynym miejscem, gdzie można opuścić znak średnika bez żadnych konsekwencji jest ostatnia instrukcja kodu, tuż przed zamykającym znacznikiem PHP.

Polecenia PHP mogą być grupowane wewnątrz nawiasów klamrowych ({}), przykładowo, w instrukcjach warunkowych, pętlach lub funkcjach. Mimo że w ActionScripcie powinno się umieszczać średnik po zamykającym nawiasie funkcji anonimowych, to w podobnej sytuacji w PHP nie można już tego robić.

Zapełnianie kodu białymi znakami

W PHP każda instrukcja kończy się znakiem średnika, zatem wszystkie białe znaki niebędące częścią ciągu znakowego są tak naprawdę ignorowane (ciągami znakowymi zajmiemy się bliżej w rozdziale 4.). Oznacza to, że z punktu widzenia parsera PHP nie ma znaczenia, czy kod znajdzie się w jednym wersie, czy też — dla większej czytelności — będzie rozpisany na wiele linii. Oczywiście druga opcja, z wiadomych względów, jest lepsza.

Choć nie istnieją sztywne reguły, to ze względu na konieczność zachowania czytelnego kodu warto podzielić swój skrypt na logiczne bloki. PHP nie zezwala jedynie na umieszczanie białych znaków wewnątrz nazw zmiennych lub funkcji, jednak o ile mi wiadomo, nie jest to możliwe w żadnym języku programowania.

> Ze względu na historię ewoluowania PHP, nie istnieje oficjalny sposób formatowania kodu lub umieszczania go wewnątrz nawiasów klamrowych. Dla odmiany w ActionScripcie istnieje zalecany styl, który można włączyć poprzez opcję *Auto Format* w panelu *Actions*. W miarę czytania książki dostrzeżesz, że dla PHP i ActionScriptu używam nieco odmiennych stylów formatowania, dzięki czemu łatwo można odróżnić ich kod. Możesz jednak swobodnie korzystać ze stylu kodowania ActionScriptu, jeśli wydaje Ci się bardziej naturalny. O ile tylko składnia poleceń jest prawidłowa, format kodu nie ma w rzeczywistości znaczenia.

Nadawanie nazw zmiennym

Zmienne stanowią podstawę każdego języka programowania — jak sama nazwa wskazuje, przechowują wartości, które w programie mogą ulec zmianie. Dobrym przykładem może być stan konta w banku. Choć ciągle ulega zmianie, to odwołujesz się do niego na podstawie jego nazwy. Zasada nazywania zmiennych jest w PHP niemal taka sama jak w ActionScripcie, poza jednym wyjątkiem: **zmienne w PHP muszą się zaczynać od znaku dolara ($)**, natomiast po nim:

- musi występować litera lub znak podkreślenia, po którym może wystąpić dowolna kombinacja liter, liczb lub podkreśleń;
- **nie** może występować liczba;
- nazwa zmiennej **nie** może zawierać spacji.

Tabela 2.1 przedstawia przykłady różnych wariacji w nazwach zmiennych PHP.

Tabela 2.1. *Przykłady prawidłowych i nieprawidłowych nazw zmiennych PHP*

Zmienna	Poprawność	Przyczyna
`myVar`	Niepoprawna	Nie rozpoczyna się znakiem `$`
`$myVar`	Poprawna	Rozpoczyna się znakiem `$`; wszystkie pozostałe znaki są dozwolone
`$my Var`	Niepoprawna	Zawiera spację
`$myVar1`	Poprawna	Wszystkie znaki są dozwolone
`$1myVar`	Niepoprawna	Pierwszy znak po `$` jest cyfrą
`$_1myVar`	Poprawna	Pierwszy znak po `$` jest podkreśleniem
`$my_Var`	Poprawna	Wszystkie znaki są dozwolone

Nadawanie nazw funkcjom

Funkcje są kolejnym podstawowym składnikiem języków programowania. Czerpiąc analogię z języka naturalnego można powiedzieć, że skoro zmienne można potraktować jako rzeczowniki, to funkcje niech będą czasownikami — sprawią, że w programie coś będzie się działo. PHP posiada całe mnóstwo wbudowanych funkcji (obliczyłem, że więcej niż 2700), jednak zarówno w PHP, jak i ActionScripcie warto tworzyć swoje własne funkcje. Zasady ich nazywania będą prawie jednakowe jak w przypadku zmiennych, poza koniecznością uwzględniania znaku dolara na początku zmiennych. Oznacza to, że nazwy funkcji **muszą się zaczynać literą lub podkreśleniem**.

> Mimo że rozpoczynanie nazw funkcji i zmiennych od znaku podkreślenia nie jest błędem, to należy unikać tej praktyki. Nazwy wielu wbudowanych funkcji PHP zaczynają się właśnie od tego znaku, toteż aby uniknąć nieporządku w kodzie, lepiej nie nadawać funkcjom nazw rozpoczynających się od znaku podkreślenia.

Unikanie słów kluczowych

PHP nie pozwala na nadpisywanie nazw wbudowanych funkcji języka. Ze względu na ich liczbę niekiedy trudno będzie uniknąć wykorzystywania słów zastrzeżonych. Warto wówczas poprzedzać nazwy funkcji przedrostkami `my`, `the` lub na przykład własnymi inicjałami.

W tabeli 2.2 zostały przedstawione wszystkie słowa kluczowe PHP — nie można ich stosować do nazywania własnych zmiennych, funkcji, klas ani stałych.

Rozróżnianie wielkości liter

Wszystko byłoby prostsze, gdyby PHP zawsze rozróżniał wielkość liter. Niestety, tak nie jest:

- w przypadku nazw zmiennych wielkość liter jest istotna;
- w przypadku nazw funkcji wielkość liter **nie** ma znaczenia;
- w przypadku nazw klas wielkość liter **nie** jest istotna.

Tabela 2.2. *Słowa kluczowe PHP nie powinny być stosowane do nazywania funkcji i zmiennych tworzonych przez użytkownika*

Słowo kluczowe PHP				
and	array()	as	break	case
cfunction	class	const	continue	declare
default	die()	do	echo()	else
elseif	empty()	enddeclare	endfor	endforeach
endif	endswitch	endwhile	eval()	exception
exit()	extends	for	foreach	function
global	if	include()	include_once()	isset()
list()	new	old_function	or	php_user_filter
print()	require()	require_once()	return()	static
switch	unset()	use	var	while
xor	__CLASS__	__FILE__	__FUNCTION__	__LINE__
__METHOD__				

Oznacza to, że jeśli przez przypadek zamiast `$myVar` programista napisze `$myvar`, zmienna nie zostanie poprawnie rozpoznana. Z drugiej strony, wbudowane funkcje tego języka mogą już być zapisywane na wiele sposobów — np. nazwy `echo`, `Echo` lub `eCHo` będą w PHP traktowane tak samo.

Sprawę jeszcze bardziej komplikuje fakt, że choć w przypadku ActionScript 1.0 wielkość liter nie była istotna, to już w wersji 2.0 wprowadzono nowe, sztywne reguły dotyczące tej kwestii.

> W tej sytuacji najlepiej jest przyjąć, że w ActionScript oraz w PHP wielkość liter jest istotna. Dlatego też w przypadku nazw złożonych konsekwentnie korzystaj z jednej konwencji nazewniczej — albo używając tzw. notacji wielbłądziej (ang. *camel case*) — np. `myVar`, albo znaku podkreślenia — np. `$my_var`. Sprawisz sobie dużo kłopotów, jeśli będziesz jednocześnie korzystał z dwóch osobnych zmiennych nazwanych `$myVar` oraz `$myvar`. Prędzej czy później użyjesz niewłaściwej zmiennej. Ze względu na fakt, że w przypadku nazw funkcji i klas wielkość liter nie ma znaczenia, tworzenie dwóch zmiennych różniących się tylko wielkością liter jest od początku skazane na niepowodzenie.

Wybór sensownych nazw

Jeśli dostosujesz się do wyżej wymienionych wskazówek, będziesz mógł nazywać zmienne i funkcje wedle własnego uznania. W PHP nie ma znaczenia, czy nazwiesz zmienną `$a` czy też `$superfajnazmiennacharakterystyczna`, jednak jeśli nazwy będą jednoznacznie kojarzyły się z daną zmienną lub funkcją, bardzo ułatwisz sobie życie. Kod PHP jest przechowywany po stronie serwera, więc nikt nie odczyta tych nazw (o ile programista nie sprawi, by celowo były widoczne lub też skrypty bądź serwer zostaną nieprawidłowo skonfigurowane), toteż nie staraj się naśladować niektórych

technik celowego gmatwania kodu, propagowanych przez programistów JavaScriptu. Jedynymi osobami, które w ten sposób zostaną zmylone, będziesz Ty sam i inne osoby, które będą chciały rozwijać taki kod w przyszłości.

Jednym słowem, opłaca się nadawać zmiennym dużo mówiące nazwy. Dzięki nim o wiele łatwiej można zrozumieć przeznaczenie danego fragmentu kodu i łatwiej go także rozwijać. Mniejsze będzie również prawdopodobieństwo popełnienia błędu w nazwie zmiennej. Niemniej jednak jednoliterowe nazwy zmiennych, takie jak na przykład `$i`, powinny być stosowane jedynie jako liczniki w pętlach lub tam, gdzie ich przeznaczenie jest jasne, a wartość łatwo przewidywalna.

Na luzie z PHP

PHP, podobnie jak JavaScript oraz pierwotnie ActionScript, został zaprojektowany w celu ułatwienia programowania. Programista kodujący w tym języku został zwolniony z obowiązku pamiętania o typach zmiennych. Innymi słowy, może on najpierw zadeklarować zmienną jako ciąg znakowy, a następnie przypisać do niej wartość całkowitą — byłoby to niemożliwe w językach kontrolujących typy zmiennych (np. w Javie).

Ta swoboda została poważnie ograniczona w ActionScripcie 2.0, w którym znalazło się już zalecenie deklarowania typu dla każdej zmiennej oraz funkcji. Mimo że statyczne (bądź ścisłe) deklarowanie typów może się wydać zbyt dużym ograniczeniem elastyczności ActionScriptu 1.0, to i tak ostatecznie oszczędza się sporo czasu, ponieważ kompilator automatycznie będzie ostrzegał o jakichkolwiek potencjalnych błędach w dopasowaniu do typów zmiennych. Poza tym zyskuje się dokładną wiedzę, w którym miejscu popełniono błąd, co oznacza, że nie trzeba przeszukiwać całego kodu w poszukiwaniu usterek. Z drugiej strony, programiści PHP5 są generalnie przeciwni statycznemu definiowaniu typów. Szczęśliwie typ zmiennych nie stanowi głównego źródła problemów, głównie dzięki zastosowaniu w PHP odrębnych operatorów dodawania oraz łączenia ciągów znakowych (tak jak to opisano w następnym podrozdziale).

> Mimo że dzięki statycznemu określaniu typów można uniknąć wielu błędów w programie, to należy zauważyć, że taka kontrola jest przeprowadzana jedynie w czasie kompilacji kodu ActionScript, czyli w czasie publikowania pliku SWF. Kod Flasha jest tworzony dopiero w momencie uruchomienia programu i nie można zagwarantować, że program zasygnalizuje błąd wywołany przez zmiany wprowadzone przez fragment PHP. Gdy pojawią się nieoczekiwane wyniki, być może trzeba będzie dokonać konwersji ciągów znakowych otrzymanych z klasy `LoadVars` na właściwy typ zmiennych. Skoro wszystkie zmienne, które zostały pobrane poprzez `LoadVars` są zmiennymi znakowymi, niekiedy może się okazać, że w przypadku nieoczekiwanych wyników będziesz musiał dokonać konwersji do właściwego typu danych.

PHP posiada osiem typów zmiennych, które są mocno zbliżone do odpowiednich typów danych znanych z ActionScriptu. Poniższa lista zawiera krótką charakterystykę każdego z nich:

1. `Integer`: ten typ przechowuje liczby całkowite (np. 1, 25 lub 346). Zmiennych tego typu *nie zapisuje* się w cudzysłowie. Liczby całkowite mogą mieć postać dziesiętną, ósemkową lub szesnastkową. Liczby ósemkowe rozpoczynają się od 0, natomiast szesnastkowe — od 0x. Powoduje to istotne konsekwencje w przypadku liczb, które mogą się rozpoczynać od 0 (numery telefonów lub identyfikatory). Aby uniknąć problemów, warto przechowywać takie wartości jako ciągi znakowe.
2. **Liczby zmiennopozycyjne (`double`):** Ten typ może być zapisany albo przy użyciu kropki dziesiętnej (np. 2.1 lub 98.6), albo wykorzystując notację matematyczną (np. 0.314E1). Podobnie jak liczby całkowite, liczby zmiennopozycyjne nie są umieszczane w cudzysłowach. W rozdziale 3. omówimy dokładniej ten typ liczb.
3. `String`: ten typ jest sekwencją znaków, którą umieszcza się albo między znakami apostrofu, albo w cudzysłowie. Ciągi znakowe zostały wyczerpująco omówione w rozdziale 4.
4. `Boolean`: typ przyjmujący wartości `true` lub `false`. Poniższe zmienne są traktowane jako `false` w PHP:
   - słowo kluczowe `false`;
   - liczba całkowita 0;
   - liczba zmiennopozycyjna 0.0;
   - pusty ciąg znakowy (`''` lub `""`, bez żadnych spacji w środku);
   - zero jako ciąg znakowy (`"0"`);
   - tablica posiadająca zerową liczbę elementów;
   - `NULL`.

   Wszystkie pozostałe wartości są traktowane jako `true`.
5. `Array`: istotna różnica pomiędzy PHP a ActionScriptem polega na tym, że w PHP tablice asocjacyjne posiadają długość, którą można mierzyć i którą można manipulować (w ActionScripcie to nie jest możliwe). Zajmiemy się tym szczegółowo w rozdziale 5.
6. `Object`: ten typ jest spotykany w klasach wbudowanych PHP lub zdefiniowanych przez użytkownika. W rozdziale 5. omówimy zastosowanie obiektów w tym języku, natomiast w rozdziale 7. zajmiemy się nimi bardziej szczegółowo.
7. `Resource`: gdy PHP uzyskuje dostęp do zewnętrznych danych, takich jak bazy danych MySQL lub pliki tekstowe, wymaga pewnego odniesienia do tworzonego połączenia. Z punktu widzenia programisty będzie to zwykła zmienna typu resource.
8. `NULL`: ten typ posiada tylko jedną wartość i jest ona po prostu równa `NULL`. Podobnie jak w ActionScripcie, jest ona wykorzystywana jedynie do oznaczania zmiennych, które nie posiadają żadnych wartości. Zwróć uwagę, że tego słowa kluczowego nie zapisuje się w cudzysłowie.

Śledzenie zmian w typach zmiennych

Ze względu na fakt, że PHP jest językiem, w którym nie stosuje się kontroli typu zmiennych, zmienne w skrypcie mogą przyjmować różne typy. Często sam programista nie jest świadom zachodzących zmian, jednak niekiedy może mieć to nieprzewidywalne konsekwencje. Co prawda, operatorami PHP i instrukcjami warunkowymi będziemy się zajmować dopiero w następnym rozdziale, jednak te zaprezentowane w poniższym ćwiczeniu mają takie same działanie w ActionScripcie, więc nie powinno być większych problemów z ich zrozumieniem.

1. Utwórz nowy dokument PHP w folderze *phpflash/ch02* i nazwij go *variables.php*.

2. Wpisz następujący kod:

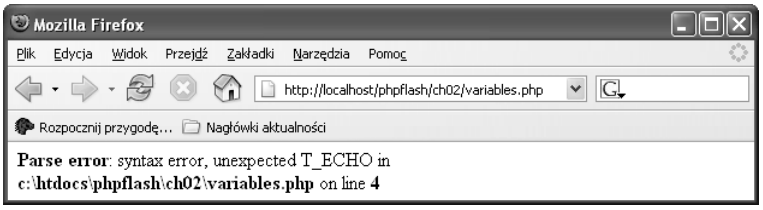
```
<?php
$myVar = '12';
$type = gettype($myVar);
echo "Wartość: $myVar Typ: $type ";
?>
```

   Funkcja `gettype()` jest odpowiednikiem operatora `typeof` w ActionScripcie, który służy do określania typu zmiennej przekazanej do niego jako argument. W powyższym kodzie wynik jej działania jest przechowywany w zmiennej `$type`, a następnie wysyłany poprzez funkcję `echo` na wyjście komputera, w tym przypadku ekran. Zwróć uwagę, że w 2. linii zastosowano znaki apostrofu, natomiast w linii 4. wartość zmiennej umieszczono już w cudzysłowie. W tym szaleństwie jest pewna metoda — przyczyny takiego postępowania wyjaśnię w następnym podrozdziale.

3. Zapisz plik, a następnie otwórz go w oknie przeglądarki. Prawdopodobnie domyślasz się już, jaki będzie wynik. Strona powinna wyglądać następująco:

4. W pliku *variables.php* umieść kursor tuż po słowie `gettype` i przed otwierającym znakiem nawiasu. Wstaw kilka spacji, zapisz plik i załaduj go ponownie (w celu ponownego załadowania strony możesz skorzystać z przycisku *Odśwież* — w trakcie wykonywania ćwiczeń opisanych w tym rozdziale będziesz wielokrotnie korzystał z tego przycisku). Zmiana zawartości pliku nie powinna spowodować żadnej różnicy w wyglądzie wyświetlanej strony. Teraz umieść kilka pustych linii pomiędzy słowem `gettype` oraz `($myVar)`. Ponownie zapisz i wywołaj plik. Wynik powinien wciąż być ten sam. Dzieje się tak dlatego, że PHP ignoruje wszystkie niepotrzebne białe znaki i poszukuje jedynie średników na końcu wyrażeń. Usuń w takim razie średnik po `($myVar)`. Zapisz plik i znów przeglądnij go

w oknie przeglądarki. Tym razem PHP stwierdzi, że nie jest zadowolony z tej zmiany:



5. Umieść średnik z powrotem na swoim miejscu i zmień drugą linię kodu w następujący sposób:

```
$myVar = '12' + 1;
```

6. Zapisz plik i uruchom go w przeglądarce. Parser PHP dokonał prostej operacji arytmetycznej, w konsekwencji zmienił się typ zmiennej.

7. Teraz dokonaj drobnej modyfikacji:

```
$myVar = '12' + 0.5;
```

8. Zapisz plik, a następnie ponownie uruchom go w oknie przeglądarki. Wyniki nie powinny Cię zdziwić. Jedyna różnica w tym przypadku polega na tym, że PHP posiada odmienny typ danych dla zmiennych przechowujących liczby całkowite (`integer`) i zmiennopozycyjne (`double`), natomiast ActionScript traktuje je wszystkie jako typ `Number`.

9. Ponownie zmodyfikuj drugą linię kodu w następujący sposób:

```
$myVar = '12' + 1.0;
```

10. Zapisz plik i wywołaj go w oknie przeglądarki. Efekt może się okazać zaskakujący. Wyświetloną wartością jest 13, jednak oznaczoną jako typ `double`.

    Dzieje się tak dlatego, że dodaliśmy liczbę z kropką dziesiętną. W takich sytuacjach, jeśli po kropce dziesiętnej występuje zero, wówczas część dziesiętna jest pomijana, niemniej jednak typ danych nadal pozostaje `double`.

Prawdopodobieństwo, że w ten sposób będzie dochodziło do powstawania błędów w kodzie jest bardzo niewielkie, niemniej jednak powyższy przykład może potwierdzić opinię, że nazbyt swobodne podejście do typu zmiennych może nieść ze sobą ukryte niespodzianki.

11. Tym razem pokażę, w jaki sposób spowodować, by zmienna `$myVar` przyjęła typ `Boolean`. Niekiedy będziesz chciał wykonać fragment kodu jedynie po sprawdzeniu pewnego warunku wyrażonego przez samą zmienną. Jeśli jej wartość będzie różna od zera, wówczas warunek zostanie spełniony. Zmień kod skryptu w następujący sposób (znaki # zostały użyte do chwilowego wyłączenia linii 3. i 4.):

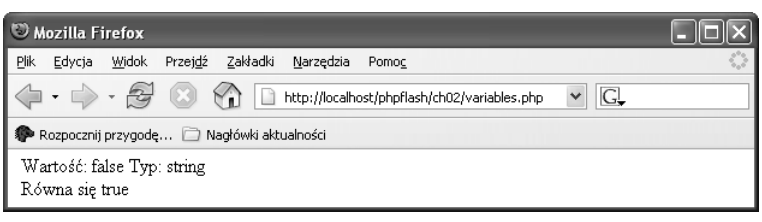
```
<?php
$myVar = '12' + 0.5;
#$type = gettype($myVar);
#echo "Wartość: $myVar Typ: $type";
if ($myVar) echo '<br> Równa się true';
?>
```

12. Zapisz plik i otwórz go w oknie przeglądarki. Ponieważ teraz zmienna `$myVar` zawiera wartość 12.5, test warunku powiedzie się i nastąpi zwrócenie wartości `true`.

13. Teraz zmień drugą linię kodu, aby wyglądała następująco (upewnij się, że dokładnie ją skopiowałeś):

```
$myVar = 'false';
```

14. Zapisz plik *variables.php* i otwórz go w oknie przeglądarki:



15. Powinieneś otrzymać ten sam wynik, co poprzednio i nie jest to pomyłka. Usuń znaki komentarza z linii 3. oraz 4. Po dokonaniu edycji strony wszystko powinno być jasne.

    Wartość `false` znalazła się w cudzysłowie, zatem stanowi ona teraz zwykły ciąg znakowy. Zajrzyj do początkowej części opisu tego ćwiczenia: jedynymi ciągami znakowymi, które mogą być równe `false` są puste ciągi znakowe (czyli zwykły cudzysłów) lub ciąg znakowy 0. Paradoksalnie, słowo „`false`" zwróci w warunku wartość `true`.

16. Usuń zatem znaki cudzysłowu okalające wartość `false`, która znajduje się w drugiej linii. W ten sposób ciąg znakowy stał się słowem kluczowym języka i wszystko powinno już zadziałać poprawnie.

> Zapisywanie wartości `true` oraz `false` w cudzysłowie jest częstą pomyłką początkujących programistów PHP oraz ActionScriptu. Wartości te stają się wówczas zwykłymi ciągami znakowymi.

Łączenie ze sobą słów oraz zmiennych

Diabeł tkwi w szczegółach. Okazuje się, że znak kropki w PHP i ActionScripcie pełni bardzo ważną rolę, jednak jego znaczenie jest różne w każdym z tych języków. Podczas gdy kropka w ActionScripcie ma przede wszystkim pozwalać na odwoływanie się do właściwości obiektów oraz zagnieżdżonych klipów filmowych, to w PHP służy do łączenia ciągów znakowych. Innymi słowy, jest to **operator konkatenacji**.

Toteż, aby połączyć dwa ciągi znakowe w każdym z tych języków, należy zastosować odrębne operatory:

- **Styl ActionScript**: "To jest" + "zdanie.";
- **Styl PHP**: "To jest" . "zdanie. ";

Zauważ, że PHP, podobnie jak ActionScript, nie dodaje żadnych spacji pomiędzy ciągami znakowymi podczas ich łączenia. Liczba białych znaków pomiędzy ciągiem znakowym a operatorem konkatenacji jest ignorowana, toteż jeśli pomiędzy nimi ma się znaleźć spacja, należy ją dodać albo do prawego, albo do lewego ciągu.

Jedną z zalet korzystania z osobnego operatora złączenia dla ciągów znakowych w PHP jest unikanie problemu związanego z dodawaniem liczby do ciągu znakowego (problem występował w ActionScripcie 1.0). Jeśli korzysta się z operatora +, ActionScript przypisze pierwszeństwo ciągom znakowym, jednak PHP po prostu zamienia ciągi znakowe na odpowiadające im wartości liczbowe.

```
"12" + "34"   //ActionScript zwróci 1234
'12' + '34'   //PHP zwróci 46
'12' . '34'   //PHP zwróci 1234
```

PHP pójdzie nawet dalej. Jeśli jest w stanie wydobyć liczbę z początku ciągu znakowego, to zwyczajnie doda ją do drugiego składnika sumy.

```
'10 zielonych butelek' - 1 //PHP zwróci 9
'zielonych butelek 10' - 1 //PHP zwróci -1
```

Zauważ, że w tym przykładzie tekst wewnątrz ciągu znakowego jest tak naprawdę ignorowany. W pierwszym przypadku PHP zwróci po prostu 9, zamiast „9 zielonych butelek", natomiast w drugim przykładzie liczba 10 zostanie pominięta, gdyż nie rozpoczyna ona ciągu znaków, który w tym wypadku jest zamieniany na liczbę 0.

Pewnym problemem w PHP może być fakt, że operator złączenia jest tak niewielki, że niekiedy można go nieumyślnie pominąć (zarówno podczas przeglądania kodu, jak i podczas jego kopiowania). Często też można się pomylić jednocześnie kodując w ActionScripcie oraz w PHP. Dlatego też zwracaj uwagę, z którego operatora konkatenacji korzystasz, w przeciwnym przypadku mogą powstawać trudne do wykrycia błędy.

Apostrofy czy cudzysłowy? Mogłeś zauważyć, że w większości przykładów w tym rozdziale wykorzystywałem apostrofy, mimo że w czwartej linii poprzedniego fragmentu kodu zastosowałem cudzysłów. Istnieje ku temu kilka powodów, choć programiści PHP nie zawsze przestrzegają tych zasad:

- To, co jest zawarte pomiędzy apostrofami jest traktowane jako stała znakowa (`string literal`).
- Cudzysłów wskazuje parserowi PHP, że przechowuje elementy, które należy przetworzyć.

Prosty przykład poniżej powinien nieco ułatwić zrozumienie tej reguły. Jeśli chcesz przetestować dowolny z poniższych przykładów w swojej przeglądarce, nie zapominaj, że cały kod PHP musi być zawarty między otwierającym i zamykającym znacznikiem (`<?php` oraz `?>`):

```
$name= 'David';
echo 'Cześć, $name'; //Wyświetla 'Cześć, $name'
echo "Cześć, $name"; //Wyświetla 'Cześć, David'
```

Gdy zastosujesz apostrofy, zmienna `$name` będzie wyświetlana bez wcześniejszego przetworzenia przez parser PHP, natomiast jeśli zapiszesz ją w cudzysłowie, zostanie wyświetlona na ekranie dopiero po przetworzeniu przez mechanizm PHP.

Dodatkowo, cudzysłów umożliwia zapisanie ciągów znakowych zawierających apostrofy.

```
echo "Pożyczyłem auto Cabe'owi";   //ciąg znaków będzie wyświetlony
echo 'Pożyczyłem auto Cabe'owi';   //wyświetli komunikat o błędzie
```

Powodem, dla którego w drugim przypadku zostanie wygenerowany komunikat o błędzie, jest fakt, że interpreter PHP traktuje drugi znak apostrofu (pomiędzy `t` a `s`) jako znak zamykający ciąg znakowy. Można też zaradzić temu w inny sposób — poprzez użycie znaków specjalnych, które spowodują, że parser PHP potraktuje dany ciąg liter w szczególny sposób. Wszystkie znaki specjalne rozpoczynają się od lewego ukośnika (tzw. znaku sterującego). Gdy umieścisz taki ukośnik przed apostrofem, parser PHP nie potraktuje tego apostrofu jako znaku zamykającego i ciąg znakowy zostanie wyświetlony tak jak zaplanowano.

```
echo 'It\'s fine';  //wyświetli się prawidłowo
```

Ze względu na przydatność takiego rozwiązania wielu programistów PHP korzysta tylko z cudzysłowów, niemniej jednak według oficjalnych zaleceń należy unikać zbędnego przetwarzania dokumentu przez parser PHP. Jeśli ciągi znakowe zawierają zmienne, które wymagają przetworzenia, warto użyć cudzysłowów. Jeśli natomiast zawierają tylko stałe znakowe, wówczas parser PHP niepotrzebnie będzie zużywał zasoby komputera do wyszukiwania zmiennych w celu ich przetworzenia. Choć w przypadku krótkich skryptów odbywa się to w ułamku sekundy, to w dłuższych programach może to zająć już nieco czasu. Jeśli powyższe argumenty nadal Cię nie przekonują, zauważ, że aby wpisać znak cudzysłowu, należy skorzystać z dwóch przycisków na klawiaturze (*SHIFT+'*), podczas gdy apostrof uzyskuje się jedynie za pomocą jednego przycisku.

Spójrzmy teraz na tabelę 2.3 przedstawiającą większość znaków specjalnych. Będziemy z niej korzystać podczas budowania aplikacji we Flashu.

Tabela 2.3. *Znaki specjalne stosowane w ciągach znakowych w języku PHP*

Znak sterujący	Reprezentowany znak w ciągu
`\"`	Cudzysłów
`\n`	Nowa linia
`\r`	Powrót kursora
`\t`	Tabulator
`\\`	Lewy ukośnik
`\$`	Znak dolara
`\{`	Otwierający nawias klamrowy
`\}`	Zamykający nawias klamrowy
`\[`	Otwierający nawias kwadratowy
`\]`	Zamykający nawias kwadratowy

Może się wydawać, że nawiasy nie są niczym szczególnym, jednak podczas pracy z zapytaniami do baz danych szybko się przekonasz, że umiejętność budowania ciągów znakowych prawidłowo ujętych w apostrofy bądź cudzysłowy jest szczególnie istotna.

Wysyłanie formularza pocztą e-mail

Mam nadzieję, że nie zanudziłem Ciebie do tej pory, gdyż teraz chciałbym pokazać, w jaki sposób można wykorzystać tę wiedzę w praktyce. Nie przejmuj się, jeśli w tym momencie nie wszystko jest dla Ciebie jasne. Książka ta powinna stanowić swego rodzaju kompendium, do którego w razie jakichkolwiek wątpliwości będziesz mógł w przyszłości powrócić.

Teraz utworzymy formularz dla odwiedzających Twoją stronę, za pomocą którego użytkownicy będą mogli wysyłać swoje komentarze, szczegóły dotyczące subskrypcji oraz innego typu informacje. Formularz może posiadać dowolną liczbę pól, jednak aby maksymalnie uprościć sprawę, zdecydowałem się tylko na pięć pól. Gotowy już formularz jest widoczny na rysunku 2.1.

Rysunek 2.1.
Gotowy formularz oceniający

Konstruowanie formularza

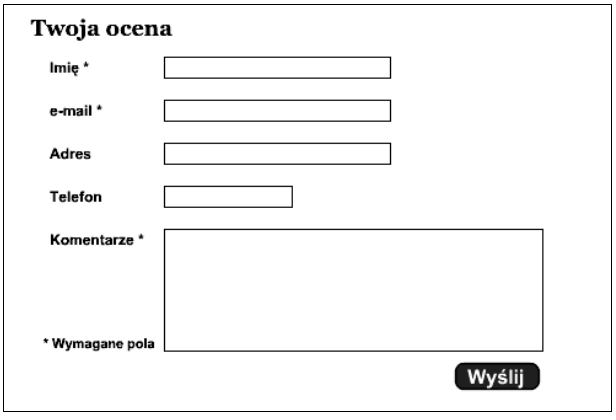
Formularz można utworzyć albo samodzielnie, na podstawie podanych niżej wskazówek, albo — jeśli chcesz się skoncentrować na PHP i ActionScripcie — wzorując się na pliku *biodiverse05.fla*, dostępnym w materiałach źródłowych tego rozdziału.

Utworzenie takiego formularza jest doskonałą okazją do wypróbowania wersji 2. komponentów *TextInput* oraz *TextArea*, po raz pierwszych uwzględnionych we Flashu MX 2004. Niemniej jednak nie będę z nich tutaj korzystać, gdyż zapewne część Czytelników nadal używa Flasha MX. Co więcej, Macromedia ogłosiła, że kolejne istotne zmiany w architekturze komponentu zostaną wprowadzone dopiero wraz z nową wersją Flasha. Dlatego też, zamiast gotowych komponentów wolałem tutaj użyć zwykłych pól statycznych oraz dynamicznych. Gdy już podstawowe funkcje PHP i Flasha będą działały poprawnie, możesz dostosować aplikację do wykorzystania odpowiednich komponentów.

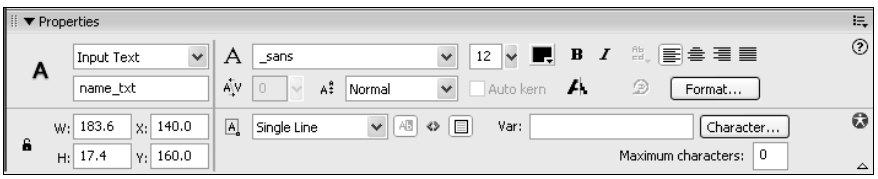
> Umiejscowienie plików FLA zależy wyłącznie od Ciebie. Radzę jednak, aby przechowywać je w osobnym folderze w katalogu głównym serwera. W tym celu kliknij *File/Publish Settings* (*CTRL+SHIFT+F12* w Windows lub *OPT+SHIFT+F12* w systemie Mac OS), a następnie ustaw ścieżkę do plików SWF oraz HTML w katalogu głównym serwera. W przypadku systemu Windows katalogiem tym będzie *C:\htdocs\phpflash\ch02*, natomiast w systemie Mac OS będzie to analogiczny folder *Sites*.
>
> Jeśli korzystasz z plików źródłowych, zapisz każdy z nich pod nazwą *biodiverse.fla*. W ten sposób zawsze w razie jakichkolwiek problemów będziesz mógł odzyskać poprzednią wersję.

1. Wykorzystaj opisany na początku tego rozdziału dokument Flasha, służący do wyświetlania daty sformatowanej przez kod PHP lub użyj pliku *biodiverse02.fla* z kodów źródłowych. Wstaw trzy nowe warstwy pomiędzy warstwę `date` i `actions`. Teraz linia czasowa powinna wyglądać następująco:

2. W warstwach `static text` oraz `form elements` rozplanuj układ formularza. W przypadku statycznych etykiet użyłem pogrubionej czcionki Arial o wielkości znaków 12 pt i zastosowałem 35-pikselowy odstęp pomiędzy sąsiadującymi ze sobą etykietami. Dzięki temu powstało miejsce na pojawiające się komunikaty o potencjalnych błędach.



3. Obok etykiet umieść pola tekstowe, w których użytkownik będzie mógł wprowadzać odpowiednie dane. Nadaj każdemu z pól odpowiednią nazwę, np.: `name_txt`, `email_txt`, `snail_txt`, `phone_txt` oraz `comments_txt`. Poniższy zrzut ekranu obrazuje ustawienia dla pierwszego pola.

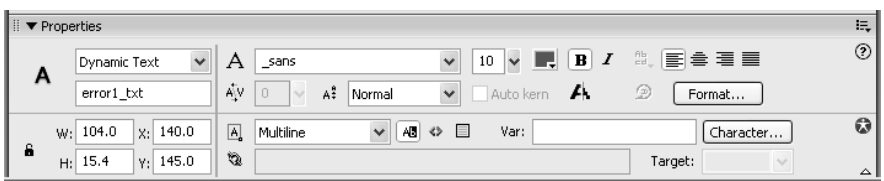


4. W ostatnim polu użytkownik będzie mógł umieszczać komentarze dotyczące strony, toteż dla tego pola należy zarezerwować szczególnie dużo miejsca. Aplikacja ta jest jedynie pewnym przykładem, więc użyłem jednoliniowego pola dla adresu użytkownika. W rzeczywistej aplikacji należałoby uwzględnić osobne pola na nazwę miasta, kod pocztowy itd.

5. Utwórz przycisk, za pomocą którego użytkownik będzie zatwierdzał dane wprowadzone do formularza, nazwij go `submit_btn` i umieść go na dole strony. Jeśli korzystasz z pliku *biodiverse02.fla*, taki przycisk będzie się znajdował w bibliotece Flasha (możesz ją otworzyć poprzez kombinację klawiszy *CTRL+L* w Windows lub *⌘+L* w systemie Mac OS).

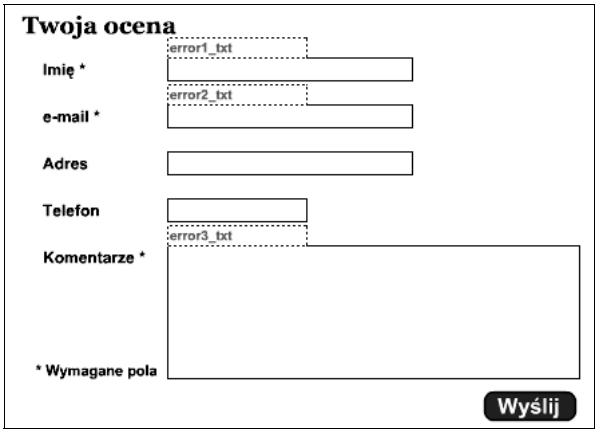
6. Przetestuj teraz swój film (*CTRL+ENTER* w Windows lub *⌘+RETURN* w systemie Mac OS). Powinien wyglądać tak jak na rysunku 2.1.

7. Nie ma nic bardziej irytującego niż otrzymywanie niekompletnych informacji, toteż upewnij się, że użytkownik wypełnił wszystkie wymagane pola. Tuż nad polem tekstowym `name_txt` wstaw dynamiczne pole i nazwij je `error1_txt`. Aby wyświetlany tekst wyróżniał się na tle strony, ustaw kolor na czerwony (`#FF0000`). Na poniższym rysunku przedstawiono ustawienia, z których korzystałem:



8. Utwórz dwa dodatkowe pola dynamiczne o nazwach `error2_txt` oraz `error3_txt` i umieść je ponad polami tekstowymi przeznaczonymi do wprowadzania adresu e-mail oraz komentarzy. Na rysunku 2.2 widać, w którym miejscu należy je umieścić. Nazwy tych pól zostały pokazane jedynie w celach demonstracyjnych. Gdy będziesz budował taki formularz, pozostaw je puste. Także i w tym przypadku skorzystam z ActionScriptu w celu zapewnienia, że pola dynamiczne będą wystarczająco szerokie do pomieszczenia wyświetlanego tekstu. Jeśli przetestujesz teraz swój film, powinien wyglądać dokładnie tak jak na rysunku 2.1. W razie potrzeby porównaj swój kod z kodem z pliku *biodiverse03.fla*.

Rysunek 2.2. *Rozmieszczenie trzech dynamicznych pól tekstowych, wyświetlających wiadomości o błędach*

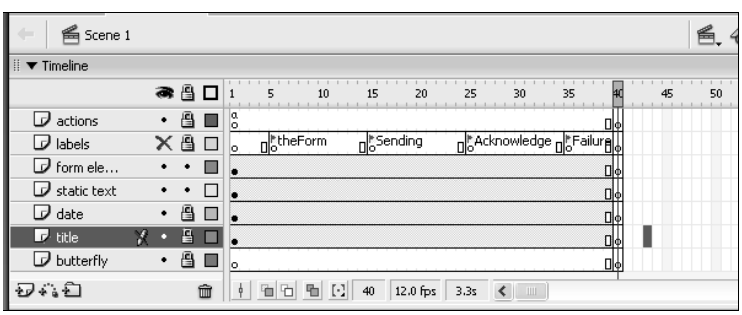


Budowanie mechanizmu przetwarzania formularza

Gdy użytkownik kliknie przycisk *Wyślij*, trzeba sprawdzić, czy wymagane pola zostały uzupełnione. W razie jakichkolwiek błędów powinna się wyświetlić odpowiednia informacja. Nawet gdy wszystko pójdzie dobrze, trzeba o tym fakcie poinformować użytkownika.

1. Nadal pracując na tym samym pliku (lub otwierając plik *biodiverse03.fla* znajdujący się w materiałach źródłowych dla tego rozdziału) na linii czasowej, we wszystkich warstwach podświetl klatkę 40. i wstaw pustą klatkę kluczową (ang. *keyframe*) — możesz skorzystać z przycisku *F5*. Jeśli nie masz ochoty na samodzielne pisanie kodu, skorzystaj z pliku *biodiverse04.fla*.

2. Na warstwie `labels` wstaw klatki kluczowe w klatkach 5., 15., 25. oraz 35. i nazwij je następująco: `theForm`, `sending`, `acknowledge` oraz `failure`. Teraz linia czasowa powinna wyglądać jak poniżej:



3. W warstwach `form elements` oraz `static text` wstaw klatki kluczowe w klatkach 5. oraz 15. Dzięki temu zarówno pola formularza, jak i sam tekst zostaną skopiowane do obu klatek. Wszystkie elementy powinny znaleźć się w klatce 5., natomiast w klatce 15. nie powinno być żadnych pól, usuń więc oprócz logo, tytułu i pola z datą całą zawartość z klatki 15.

4. Utwórz statyczny tekst informujący, że formularz jest wysyłany.

5. Jeśli przetestujesz teraz swój plik filmowy, będzie on stale powtarzał wyświetlanie formularza i pokazywanie wiadomości z klatki 15. Aby temu zapobiec, będziesz musiał wyłączyć ciągłe odtwarzanie. Zaznacz klatkę 1. z warstwy `actions` i otwórz panel *Actions*. Jeśli korzystasz z plików źródłowych książki, odpowiedni kod skryptowy już się tam znajduje. Na końcu skryptu dopisz następującą linię:

```
gotoAndStop("theForm");
```

Formularz został umieszczony w osobnej klatce kluczowej, gdyż w przeciwnym przypadku skrypt starałby się załadować datę z pliku *today2.php* za każdym przetwarzaniem klatki 1. Ponieważ data zmienia się jedynie jeden raz na 24 godziny, takie postępowanie powodowałoby marnowanie zasobów komputera.

6. Przejdź na samą górę panelu *Actions* i wpisz następujący fragment kodu:

```
function checkForm():Boolean {
    //funkcja sprawdza, czy uzupełniono wymagane pola
    //inicjalizowanie flagi missing przy założeniu, że wszystko zostało poprawnie
    //uzupełnione
    var missing:Boolean = false;
    //czyszczenie wszystkich pól tekstowych dla błędów
    error1_txt.text = error2_txt.text = error3_txt.text = "";
    //sprawdź każde pole
    // - jeśli nie zostało uzupełnione, pokaż odpowiedni komunikat
    //oraz ustaw wartość flagi missing na true
    if (name_txt.text = "") {
        error1_txt.text = "Wpisz swoje imię";
        missing = true;
```

```
        }
        if (email_txt.text.indexOf("@") == -1) {
            error2_txt.text = "Podaj właściwy adres e-mail";
            missing = true;
        }
        if (comments_txt.text = "") {
           error3_txt.text = "Nie wpisałeś żadnych komentarzy";
           missing = true;
        }
        //jeśli missing jest równe true zwróć false
        //w przeciwnym razie zwróć true
        return missing ? false : true;
    }
```

Kod ActionScript przedstawiony w książce stosuje statyczne (ścisłe) wpisywanie daty za pomocą składni uwzględnionej po raz pierwszy w ActionScripcie 2.0. Jeśli korzystasz z *Flasha MX* lub po prostu wolisz wpisywać datę w klasyczny sposób, nie podawaj typów zmiennych. W dodatku B znajdują się wskazówki, w jaki sposób dostosowywać skrypty do wersji MX Flasha.

Aby uzyskać więcej informacji na temat definiowania typów zmiennych, zajrzyj do książki *Foundation ActionScript for Flash MX 2004* autorstwa *Shama Bhangala* (wyd. *friends of ED*).

Zadaniem powyższej funkcji jest sprawdzenie, czy formularz został prawidłowo wypełniony. Niestety, ActionScript nie nadaje się specjalnie do tego typu testów. O wiele łatwiej to samo zrobić w PHP, jednak wadą takiego rozwiązania jest konieczność łączenia się serwerem, gdy podczas walidacji pól użytkownika powstaną błędy. Toteż lepiej jest wstępnie sprawdzić formularz za pomocą kodu ActionScript, a następnie zacząć go szczegółowo analizować za pomocą parsera PHP.

Działanie funkcji rozpoczyna się od zadeklarowania zmiennej `Boolean` nazwanej `missing` i ustawienia jej na wartość `false`. Zakładamy tutaj, że użytkownik dopiero wtedy kliknie przycisk *Wyślij*, gdy wszystkie pola zostaną uzupełnione. Wówczas też wszystkie komunikaty o błędach będą wyzerowane (ustawione na pusty ciąg znakowy). Jest to konieczne, aby wyczyścić poprzednie informacje o błędach. Następnie w trzech instrukcjach warunkowych skrypt przeprowadza proste sprawdzenie pól `Imię`, `e-mail` oraz `Komentarze`. Jeśli pole `Imię` lub `Komentarze` jest puste lub jeśli wysłany adres e-mail nie zawiera znaku `@`, wyświetlana jest odpowiednia informacja, a zmienna `missing` przybiera wartość `true`. Ostatecznie funkcja korzysta z operatora warunkowego do sprawdzenia, czy zmienna `missing` została ustawiona na `true`, po czym jest zwracana odpowiednia wartość.

Operator (`?:`), niekiedy nazywany trójargumentowym operatorem warunkowym jest przydatnym skrótem używanym w ActionScripcie, PHP i innych językach programowania. Może być zamiennie stosowany z instrukcją `if...else`. Jego zastosowanie zostało omówione w następnym rozdziale.

7. Tuż za funkcją `checkForm()` wstaw następujący fragment kodu (tuż koło linii 26.):

```
function sendMessage():Void {
    //sprawdź, czy formularz został poprawnie wypełniony
    var formOK:Boolean = checkForm();
    //jeśli nie ma żadnych problemów, przetwórz formularz i wyślij zmienne
    // do skryptu PHP
    if(formOK) {
        //tutaj będzie wykonywane przetwarzanie formularza
        //wyświetl informację o wysyłaniu wiadomości e-mail
        gotoAndStop("sending");
    }
};
```

Ta funkcja posłuży do wyświetlania wiadomości, gdy użytkownik kliknie przycisk *Wyślij*. Rozpoczyna się od wywołania funkcji `checkForm()`, której wynik będzie przypisany do zmiennej `Boolean` nazwanej `formOK`. Jeśli `checkForm()` zwróci wartość `true`, wówczas formularz zostanie przetworzony, a następnie wysłany w wiadomości e-mail. W tym miejscu klasa `LoadVars` wykona większość zadania. W powyższym fragmencie jedynie zaznaczyliśmy miejsce, w którym powinien być umieszczony kod analizowania formularza. Później przetwarzanie kodu jest przekazywane do klatki oznaczonej etykietą `sending`, gdzie zostanie wyświetlona wiadomość z kroku 4.

8. Teraz w 5. klatce musisz oprogramować zdarzenie `submit_btn.onRelease`. Wielu programistów Flasha przypisuje zdarzenia przycisków bezpośrednio do samych przycisków, jednak ja wolę oprogramowywać wszystkie zdarzenia w osobnej warstwie na linii czasowej. Dzięki temu rozwijanie kodu jest łatwiejsze. Zamknij panel *Actions*, wybierz klatkę 5. w warstwie `actions` i wstaw klatkę kluczową. Zaznacz ją, otwórz panel *Actions* i wpisz następującą linię kodu:

```
submit_btn.onRelease = sendMessage;
```

W ten sposób funkcja `sendMessage()` została przypisana do zdarzenia `onRelease` przycisku *Wyślij*. Zwróć uwagę, że nie powinieneś dodawać nawiasu do zdarzenia `sendMessage`, gdyż w tym miejscu jedynie przypisujemy funkcję do pewnego zdarzenia, jednak jej nie uruchamiamy.

9. Przetestuj teraz swój klip filmowy — kliknij przycisk *Wyślij* bez wpisywania czegokolwiek w polach formularza. Informacje o błędach powinny się pojawić nad trzema obowiązkowymi polami (patrz rysunek na następnej stronie).

Jeśli komunikaty o błędach nie zostaną wyświetlone, upewnij się, że pola edycyjne oraz przycisk *Wyślij* posiadają prawidłowe nazwy zarówno w klatce 1., jak i 5. Jeśli nazwałeś je przed dodaniem klatki kluczowej w klatce 5., nazwy te powinny zostać automatycznie skopiowane.

Skoro wszystkie komunikaty o błędach zostały wyzerowane, to trzeba będzie je odpowiednio ustawić, aby mogły być automatycznie wyświetlane.

10. Otwórz panel *Actions* w 5. klatce warstwy `actions` i dodaj następujący fragment kodu przed funkcją, którą dodałeś w kroku 8.:

```
error1_txt.autoSize = true;
error2_txt.autoSize = true;
error3_txt.autoSize = true;
```

11. Jeszcze raz przetestuj plik filmowy. Wpisz dowolne słowo w polu `Imię`, a następnie kliknij przycisk *Wyślij*. Nad polem `Imię` nie powinno być żadnej informacji o błędzie, niemniej jednak dwa pozostałe komunikaty nadal będą widoczne. Gdy już wszystkie pola zostaną wypełnione, zostanie wyświetlona strona utworzona w kroku 4.

W tym momencie prawie ukończyliśmy stronę wizualną pliku filmowego. Jedyne co zostało, to dodanie dwóch prostych klatek oraz kilka linii kodu *ActionScript*. Na koniec przejdziemy do sedna sprawy — połączenia aplikacji z klasą `LoadVars` oraz kodem PHP.

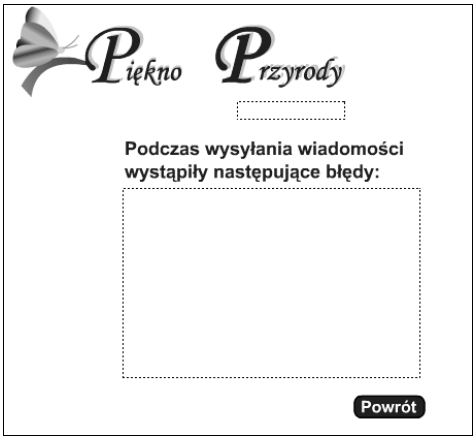
12. Zaznacz klatkę 25. w warstwie `static text`, a następnie wstaw klatkę kluczową. Jest to klatka, w której zostanie wyświetlony komunikat informujący o prawidłowym zakończeniu przetwarzania formularza. Wyświetlany tekst zmieniłem jedynie na `Dziękuję`, jednak zapewne w rzeczywistej aplikacji będziesz chciał pokazać użytkownikowi inne treści.

13. Aby podczas testowania korzystanie z aplikacji stało się bardziej przystępne, wstaw klatki kluczowe w 25. klatce warstwy `elements` oraz `actions`. Do warstwy `elements` dodaj przycisk oznaczony *Powrót* i nazwij go `back_btn` (w bibliotece pliku *biodiverse03.fla* znajduje się już gotowy taki przycisk). Wybierz klatkę kluczową 25. w warstwie `actions` i w panelu *Actions* wpisz następujący fragment kodu:

```
back_btn.onRelease = backToForm;
```

Funkcją `backToForm()` zajmiemy się wkrótce.

14. W warstwie `static text` oraz `form elements` zaznacz klatkę 35. (tą, którą nazwałeś `failure` w kroku 2.) i wstaw klatki kluczowe. Utwórz podobny formularz do tego pokazanego poniżej. Przycisk *Powrót* powinien już tam się znajdować począwszy od klatki 25., mimo że prawdopodobnie będziesz musiał zmienić rozplanowanie pól, aby dodać miejsce na zawartość tekstową. W warstwie `static text` wstaw komunikat informujący użytkownika o zaistniałym problemie, a w warstwie `form elements` umieść obszerne dynamiczne pole tekstowe o nazwie `failure_txt`.



15. Zaznacz 35. klatkę warstwy `actions` i wstaw klatkę kluczową. Otwórz panel *Actions*, a następnie przypisz funkcję `backToForm` przyciskowi *Powrót*, tak jak pokazano to w kroku 13.:

```
back_btn.onRelease = backToForm;
```

16. Teraz należy utworzyć funkcję `backToForm()`, którą umieścisz w pierwszej klatce warstwy `actions`. Kod wstaw bezpośrednio za funkcją `sendMessage()` (tuż koło linii 36.). W ten sposób sterowanie działaniem programu zostanie przekazane do klatki nazwanej `theForm`.

```
function backToForm():Void {
  //przejście do głównego formularza
 gotoAndStop ("theForm");
}
```

Możesz teraz porównać swój kod oraz stronę graficzną klipu filmowego z plikiem *biodiverse04.fla*, znajdującym się w materiałach źródłowych książki. W następnym podrozdziale zajmiemy się istotnym zagadnieniem wysyłania i otrzymywania danych z PHP za pomocą klasy `LoadVars`.

Korzystanie z funkcji LoadVars do uzyskiwania i wysyłania zmiennych

Jak wspomniałem wcześniej, to co sprawia, że funkcja `LoadVars` jest tak efektywna w porównaniu z jej poprzedniczką `loadVariables()` jest fakt, że, w przeciwieństwie do funkcji `loadVariables()`, w klasie `LoadVars` można ściśle określić wysyłane zmienne. Wprawdzie jeśli jest ich więcej, trzeba utworzyć dłuższy kod źródłowy, jednak samo

przesyłanie danych jest szybsze. Można także tworzyć różne instancje klasy `LoadVars` w odpowiedzi na różne zdarzenia — dzięki temu, jak się okaże podczas pracy z bazami danych, o wiele łatwiej postępować z dużymi ilościami danych wejściowych. Toteż zalety korzystania z `LoadVars` są znacznie większe od potencjalnej niedogodności każdorazowego wpisywania wywołania funkcji w kodzie.

Przed wykorzystaniem metod klasy `LoadVars` należy utworzyć nowy obiekt. Dobrą praktyką jest tworzenie osobnych instancji służących do wysyłania i otrzymywania danych. Dzięki temu łatwiej kontrolować wejście i wyjście programu.

1. Nadal możesz pracować z tym samym plikiem lub też po prostu wykorzystać plik *biodiverse04.fla* (gotowy kod jest dostępny w pliku *biodiverse05.fla*), znajdujący się w materiałach źródłowych książki. Wybierz klatkę 1. w warstwie `actions`. Otwórz panel *Actions* i wpisz następujący kod poniżej istniejącego skryptu. Jeśli korzystasz ze skryptu z datą, omawianego na początku tego rozdziału, poniższy kod powinien być wpisany tuż po wywołaniu funkcji `getDate` klasy `LoadVars` (tuż koło linii 47.).

   ```
   //zainicjalizuj zmienną LoadVars w celu wysyłania danych z formularza i
   //otrzymywania odpowiedzi ze skryptu PHP
   var message:LoadVars = new LoadVars();
   var messageSent:LoadVars = new LoadVars();
   ```

   Tworzone są w ten sposób dwie instancje klasy `LoadVars`: zmienna `message` posłuży do wysyłania zawartości formularza do skryptu PHP, natomiast zmienna `messageSent` zostanie wykorzystana do sprawdzenia odpowiedzi ze skryptu PHP i do przekazania informacji, czy wiadomość została pomyślnie wysłana.

   Klasa `LoadVars` traktuje wszystkie zewnętrzne dane jako swoje właściwości, toteż wszystkie wysłane zmienne są przypisywane do zmiennej `message`, natomiast wszystkie otrzymywane będą traktowane jako właściwości zmiennej `messageSent`. Nie martw się, jeśli nie wiesz jeszcze co to są właściwości — niedługo się tym zajmiemy.

2. Podczas wykonywania zadania opisywanego na początku tego rozdziału korzystałeś z klasy `LoadVars` do otrzymywania danych ze skryptu PHP. Spójrzmy bliżej na ten proces. Gdy korzysta się z `LoadVars` do otrzymywania zewnętrznych danych, następuje wywołanie zdarzenia `LoadVars.onLoad` w celu sprawdzenia, czy zewnętrzne dane zostały otrzymane. Skrypt PHP (który wkrótce utworzymy) odeśle natomiast zmienną `sent`. W przypadku wystąpienia błędów wyśle także zmienną `reason`. Wartości reprezentowane przez te zmienne automatycznie staną się właściwościami zmiennej `messageSent`, toteż wewnątrz pliku Flash dostęp do zmiennych PHP uzyskasz poprzez odwołanie się do `messageSent.sent` oraz, jeśli zmienna nie została ustawiona, do `messageSent.reason`.

   Poniższą funkcję wstaw tuż przed ostatnią linią kodu (`gotoAndStop("theForm");`) w panelu *Actions* (niedaleko 57. linii):

   ```
   messageSent.onLoad = function() {
      if (this.sent == "OK"){
         gotoAndStop("acknowledge");
   ```

```
    } else {
       gotoAndStop("failure");
       failure_txt.text = this.reason;
    }
};
```

Funkcję tę przypisano do zdarzenia `messageSent.onLoad`, więc można teraz odwoływać się do zmiennych otrzymanych ze skryptu PHP poprzez słowo kluczowe `this`. Toteż `this.sent` jest tym samym, co `messageSent.sent`. Jeśli wysyłanie wiadomości zakończy się powodzeniem, do zmiennej `messageSent` zostanie odesłana następująca para nazwa-wartość:

```
sent = OK
```

Wewnątrz pliku Flasha zmiennej `messageSent.sent` zostanie przypisana wartość `OK`. Po wystąpieniu tego zdarzenia działanie programu zostanie przekazane do klatki oznaczonej `acknowledge`.

Jeśli ta operacja się nie uda, to do filmu Flasha zostanie odesłana wiadomość rozpoczynająca się następująco:

```
sent=failed&reason=there...
```

Wewnątrz pliku Flasha sprowadzi się to do:

```
messageSent.sent = "failed";
messageSent.reason = "there...";
```

Jeśli wartość zmiennej `sent` nie jest równa `OK`, działanie pliku filmowego przejdzie do ujęcia oznaczonego `failure` i ustawi zawartość `failure_txt` na dowolną wartość otrzymaną ze zmiennej `reason`.

3. Do tej pory zajmowaliśmy się odpowiedzią ze skryptu, jednak najpierw trzeba przesłać do niego odpowiednie zmienne. Proces ten jest analogiczny do otrzymywania danych, niemniej jednak działa w drugą stronę. Wysyłane dane muszą najpierw zostać przypisane do właściwości obiektu `LoadVars` służącego do wysyłania danych — czyli do zmiennej `message`.

Przejdź do funkcji `sendMessage()` i znajdź komentarz `Tutaj będzie wykonywane przetwarzanie formularza` (tuż koło 31. linii). W formularzu znajduje się pięć pól edycyjnych, toteż ich zawartość trzeba będzie przypisać do pięciu właściwości `message`. Tuż po komentarzu wstaw następujący kod:

```
message.from = name_txt.text;
message.email = email_txt.text;
message.snail = snail_txt.text;
message.phone = phone_txt.text;
message.comments = comments_txt.text;
```

Klasa `LoadVars` wysyła każdą właściwość do skryptu PHP za pomocą metody HTTP POST, toteż zmienna `message.from` w kodzie PHP stanie się zmienną `$_POST['from']`, zmienna `message.email` stanie się `$_POST['email']` itd. Celem przedstawionych wyżej linii kodu jest przypisanie zawartości każdego pola tekstowego do zmiennych, które potem mogą być z łatwością wykorzystane w kodzie PHP.

Zwróć uwagę, że podczas tworzenia nowych właściwości zmiennej `message` nie korzysta się ze słowa kluczowego `var`, gdyż po prostu dodaje się nowe właściwości do istniejącego już obiektu. Nazwy, które nadasz tym właściwościom, mogą być dowolne, ważne jedynie, aby były to poprawne nazwy zmiennych zarówno w PHP, jak i ActionScripcie.

4. Teraz trzeba jedynie wysłać nowe zmienne do skryptu PHP. Wystarczy w tym celu wywołać metodę `sendAndLoad()` klasy `LoadVars`. Metoda wymaga dwóch argumentów: ciągu znakowego, który będzie zawierał adres do zewnętrznego skryptu oraz referencji do obiektu, który otrzyma odpowiedź z serwera. Trzeci argument jest opcjonalny. Jest nim ciąg znakowy określający metodę HTTP (`GET` lub `POST`), która będzie użyta do wysyłania danych. Domyślnie dane są wysłane metodą `POST`.

> Podstawowa różnica między metodą `GET` i `POST` polega na tym, że metoda `GET` wysyła zmienne jako część adresu URL, natomiast metoda `POST` przesyła je oddzielnie. W metodzie `GET` nałożono też pewne ograniczenia dotyczące wielkości przesyłanych informacji, toteż w większości aplikacji lepiej będzie korzystać z metody `POST`. Domyślną wartością trzeciej zmiennej jest `POST`, więc trzeci argument uwzględniaj tylko wówczas, gdy faktycznie chcesz użyć metody `GET`.

Pod liniami wstawionymi przed chwilą dodaj następujący fragment kodu:

```
message.sendAndLoad("http://localhost/phpflash/ch02/feedback.php?ck="
➥+ new Date().getTime(), messageSent);
```

Możesz się zastanawiać, co oznacza fragment: `?ck=" + new Date().getTime()` na końcu adresu URL. Jest to sposób na uniknięcie problemu z przeglądarkami, które przechowują w pamięci podręcznej dane z zewnętrznych źródeł. Większość przeglądarek będzie mogła potraktować odwołanie do zewnętrznego pliku jako już przetworzone zadanie i dostarczy klipowi Flasha nieaktualne już dane. Aby wymusić ich odświeżanie, trzeba dodać ciąg znakowy wskazujący na źródło najnowszych danych. Adres taki można zakończyć nawet losowo wygenerowaną liczbą. Ja wybrałem akurat skrót `ck` (od *cache killer*[2]) oraz bieżący czas. Możesz porównać swój kod z plikiem *biodiverse05.fla*, znajdującym się w materiałach źródłowych tej książki.

Plik Flasha jest już gotowy. Teraz nadszedł czas na oprogramowanie wszystkiego w PHP!

Sprawdzenie, czy PHP otrzymuje właściwe zmienne

Na początku tego rozdziału testowałeś dane wyjściowe skryptu PHP w celu upewnienia się, że do filmu Flasha zostały wysłane właściwe informacje. Teraz przyszła pora na sprawdzenie tego samego, jednak w drugą stronę. Gdy już zaznajomisz się ze sposobem, w jaki Flash współpracuje z PHP, tworzenie poniższego kodu nie będzie konieczne, niemniej jednak sama technika jest bardzo prosta. Będziesz mógł ją wykorzystywać zarówno w dużych, jak i w małych projektach.

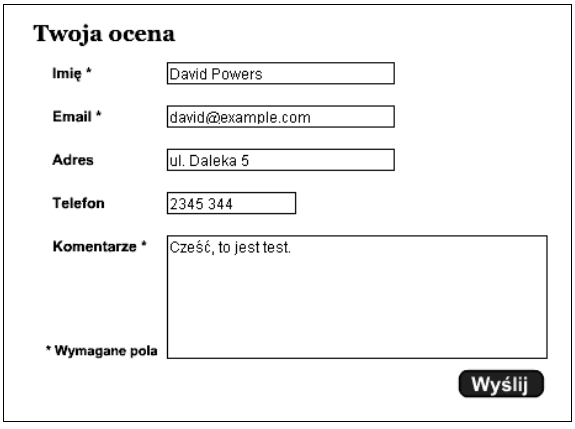
[2] Zabójca pamięci cache — *przyp.tłum.*

1. Otwórz dowolny edytor skryptów PHP i wpisz następujący kod:

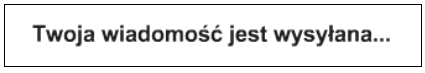
```
<?php
foreach ($_POST as $key=>$value ) {
   $received .= "$key = $value\r\n";
}
$printout = fopen('variables.txt','w');
fwrite($printout, $received);
fclose($printout);
?>
```

2. Zapisz plik w katalogu *phpflash/ch02* pod nazwą *feedback.php*.

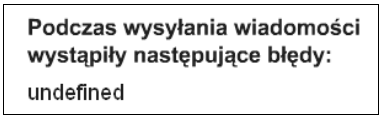
3. Przetestuj plik Flasha poprzez jednoczesne naciśnięcie *CTRL+ENTER* w Windows lub *⌘+RETURN* w systemie Mac OS. W każdym z pól wpisz dowolne dane testowe i kliknij przycisk *Wyślij*.



4. Jeśli wszystko poszło dobrze, powinieneś zobaczyć następującą wiadomość:



Po której nastąpi wyświetlenie informacji:



Nie należy jednak się tym specjalnie przejmować. Zauważ, że wiadomość nie precyzuje miejsca powstania błędu. Mimo wszystko wskazuje, że wszystko poszło dobrze.

5. Otwórz folder *phpfash/ch02* (albo za pomocą Windows Explorera w Windows, albo poprzez Findera w Mac OS), znajdujący się w folderze *C:\htdocs* w Windows lub w katalogu domowym w systemie Mac OS. Powinieneś znaleźć plik tekstowy *variables.txt* o następującej zawartości:

```
comments = Cześć, to jest test.
phone = 2345 344
snail = ul. Daleka 5
email = david@example.com
from = David Powers
```

Zmienne prawdopodobnie nie będą się znajdowały w takiej kolejności, jakiej się spodziewałeś, niemniej jednak zostały wymienione wszystkie. W przeciwnym przypadku trzeba się cofnąć i sprawdzić zarówno plik PHP, jak i kod ActionScript w pliku FLA. Jeśli nadal nie potrafisz określić źródła błędu, porównaj swój kod z plikami źródłowymi strony.

6. Skrypt wykorzystuje prostą pętlę do pobierania wartości wszystkich zmiennych, a następnie poprzez kilka funkcji PHP, służących do obróbki ciągów znakowych, wypisuje wyjście do pliku tekstowego. Kod będzie działał w przypadku dowolnej aplikacji we Flashu, gdzie dane będą wysyłane poprzez klasę `LoadVars`. Może okazać się to szczególnie przydatne, jeśli używasz ActionScriptu do przetwarzania danych przed wysłaniem ich do PHP. Zapamiętaj motto każdego programisty: „Błędne dane, błędne wyniki" (ang. *Garbage In, Garbage Out*, w skrócie *GIGO*). Nieprawidłowe dane na wejściu mogą zaowocować nieprzewidywalnymi danymi wyjściowymi, toteż zawsze należy przeprowadzać kontrolę danych. Zmień teraz nazwę pliku *feedback.php* na *variable_checker.php*.

> W rozdziale 5. zajmiemy się pętlami bardziej szczegółowo. Aby dowiedzieć się czegoś więcej o funkcjach PHP działających na plikach, zajrzyj do książki *Beginning PHP 5 and MySQL: From Novice to Professional* autorstwa *W. Jasona Gilmore'a* (wyd. Press) lub też sprawdź dokumentację online na stronie *www.php.net/manual/pl/ref.filesystem.php*.

Przetwarzanie danych i wysyłanie ich mailem przez kod PHP

Na koniec zostawiliśmy najciekawszą część — wysyłanie danych w wiadomości e-mail. PHP posiada bardzo przydatną funkcję nazwaną `mail()`. Może ona przyjmować do czterech argumentów — trzy z nich są wymagane, natomiast jeden jest opcjonalny:

- adres e-mail, na który formularz będzie wysyłany;
- temat wiadomości e-mail (który pojawi się w polu `Temat` wiadomości);
- zawartość wiadomości;
- inne dodatkowe nagłówki, rozpoznawane przez protokół poczty e-mail (opcjonalne).

Teraz jedynie trzeba zebrać odpowiednie informacje i przekazać je do funkcji `mail()`, toteż zabierzmy się do pracy!

1. Teoretycznie, można by bezpośrednio w nawiasie za nazwą funkcji umieścić wszystkie szczegóły dotyczące wiadomości, jednak lepiej jest przypisać do zmiennych odpowiednie argumenty. Otwórz plik *feedback.php* i usuń kod, który wpisałeś w poprzednim podrozdziale (jego kopia powinna być zapisana

w pliku *variable_checker.php*, dzięki czemu będziesz mógł do niej zajrzeć w przyszłości). W pliku należy pozostawić jedynie otwierające i zamykające znaczniki PHP.

2. Określ adres odbiorcy oraz temat wiadomości. Obie wartości przypisz do zmiennych, tak jak poniżej:

```
$to = 'david@example.com';
$subject = 'Formularz ze strony flash';
```

3. W następnym kroku utworzysz wiadomość e-mail. Flash przesyła na wejście skryptu pięć zmiennych, jednak kod PHP wymaga, aby wiadomość została przekazana jako pojedynczy ciąg znakowy. Wpisz zatem następującą linię kodu pomiędzy tym, co już wpisałeś, a zamykającym znacznikiem PHP:

```
$message = 'Od: '.$_POST['from']."\n\n";
```

Znaczenie pierwszej części — `$message = 'From: '` — jest dość jasne. Powoduje przypisanie do zmiennej `$message` ciągu znakowego zawierającego słowo `From`, po którym następuje dwukropek oraz spacja. Następujący potem znak kropki jest operatorem złączenia. Oznacza to, że bez względu na zawartość zmiennej zostanie ona dołączona do ciągu znakowego Wszystkie dane wysłane przez klasę `LoadVars` były przesyłane za pomocą metody HTTP `POST`. PHP gromadzi te dane w specjalnej tablicy `POST`. Dostęp do nich uzyskuje się poprzez podanie nazwy zmiennej (**bez** początkowego symbolu dolara) zawartej w zmiennej `$_POST[]`, toteż zmienna `$_POST['from']` przechowuje zmienną `message.from` (tablicami bardziej szczegółowo będziemy się zajmować w rozdziale 5.).

Później następuje operator konkatenacji, za którym umieszczono ciąg `"\n\n"`. Zwróć uwagę, że tym razem skorzystałem z cudzysłowu, więc ciąg znaków zostanie przetworzony, a cztery zawarte w nim znaki zostaną zinterpretowane jako dwa symbole nowej linii.

Tak więc, jak widać, w powyższym kodzie sporo się dzieje! Szczęśliwie, następne dwie linie będą bardzo podobne.

> Mógłbyś się zastanawiać, czemu nie zapisałem w cudzysłowie całej linii. Jest tak, gdyż zawiera ona zmienne, które mają zostać przetworzone. Podobnie jak inne języki programowania, PHP zawiera pewne specyficzne cechy — przykładowo, jeśli zmienną `$_POST['from']` zapiszemy w cudzysłowie, spowoduje to powstanie błędów. W rozdziale 5. zajmiemy się tym bardziej szczegółowo.

4. Zanim przejdziemy do wpisywania kodu w następnej linii, spójrz na następujący fragment:

```
$message .= 'Email: '.$_POST['email']."\n\n";
```

Czy dostrzegasz coś zastanawiającego w sposobie przypisania wartości do zmiennej? Z lewej strony znaku równości znajduje się znak kropki. To zestawienie posiada to samo znaczenie co znak += w ActionScripcie

(przynajmniej jeśli chodzi o ciągi znakowe). Spowoduje ono dodanie wartości znajdującej się po prawej stronie operatora = do wartości już przechowywanej w zmiennej `$message`.

> Operator konkatenacji jest bardzo niewielki, więc łatwo można go przeoczyć. Choć programiści Flasha uwielbiają niewielkie litery, to jednak podczas pracy z PHP warto zwiększyć nieco domyślny rozmiar czcionki, dzięki czemu wyświetlany kod będzie bardziej czytelny.

5. Znaczenie kolejnych trzech linii kodu jest już jasne. Dodają do zmiennej przechowującej wiadomość zawartość trzech pozostałych pól edycyjnych.

```
$message .= 'adres: ' . $_POST['snail']."\n\n";
$message .= 'telefon: ' .$_POST['phone']."\n\n";
$message .= 'Komentarze: '.$_POST['comments'];
```

6. Skrypt zgromadził już informacje potrzebne do wywołania funkcji `mail()`. Czwarty argument funkcji jest opcjonalny, więc możesz go użyć wedle swoich upodobań. Na stronie *www.ietf.org/rfc/rfc2076.txt* znajduje się lista poprawnych nagłówków e-mail. Choć jest ich tam sporo, najczęściej korzysta się tylko z dwóch. Dodaj następujące linie do kodu:

```
$additionalHeaders = "From: Flash feedback<feedback@example.com>\n";
$additionalHeaders .= 'Reply-To: '. $_POST['email'];
```

   Pierwsza linia dodaje adres e-mail, z którego nadeszła wiadomość. W przypadku funkcji `mail()`, to serwer Apache będzie wysyłał wiadomość, toteż może się zdarzyć, że w polu `From:` wyświetli się nieprzyjemnie wyglądający `nobody` (lub też dowolna inna nazwa, pod którą działa serwer). Dlatego też po słowie `From:` umieść nazwę, która ma się pojawiać we wiadomościach e-mail, a następnie między znakami mniejszości i większości (< i >) podaj swój adres e-mail. Zwróć uwagę, że cały ciąg znakowy znajduje się w cudzysłowie i kończy znakiem nowej linii `\n`. W ten sposób każdy nagłówek wiadomości rozpoczyna się od nowego wiersza.

   W drugiej linii do nagłówka `Reply-To` został przypisany adres e-mail. Jak się wkrótce przekonasz, okaże się to bardzo mądrym posunięciem.

7. Teraz można przystąpić do wysłania formularza za pomocą poczty e-mail. Ponownie można by po prostu umieścić wszystkie zmienne w nawiasach wewnątrz funkcji `mail()` i na tym zakończyć całe zadanie, jednak miło by było poinformować użytkownika o pomyślnym wysłaniu formularza. Dzięki wysłaniu odpowiednich danych z powrotem do pliku Flasha można uniknąć błędu niezdefiniowanej zmiennej, który możesz uzyskać podczas testowania programu. Funkcja `mail()` zwraca wartość `true`, jeśli wiadomość została wysłana pomyślnie, toteż musisz przechwycić jej wartość, a następnie na jej podstawie odesłać odpowiednie dane do pliku Flasha. W następnym rozdziale pokażę użycie instrukcji warunkowych, jednak dla Czytelników znających ActionScript, poniższy kod nie będzie zaskoczeniem.

Wpisz więc do pliku następujący fragment:

```
$OK = mail($to, $subject, $message, $additionalHeaders);
if ($OK) {
   echo 'sent=OK';
} else {
   echo 'sent=failed&reason='. urlencode('Prawdopodobnie nastąpił problem
   ➥z serwerem. Spróbuj wysłać wiadomość później. ');
}
```

Argumenty funkcji PHP (podobnie jak w kodzie ActionScript) są oddzielone od siebie przecinkami. Ponadto muszą też być wymienione we właściwej kolejności. Wartość `Boolean` zwrócona przez funkcję `mail()` jest przechowywana w zmiennej nazwanej `$OK` i to ona kontroluje wyjście skryptu. Jeśli wysłanie wiadomości będzie pomyślne, skrypt PHP wyśle ciąg znaków `sent=OK` do przeglądarki. Gdybyś tworzył zwykłą stronę WWW, właśnie taki tekst zostałby wyświetlony na ekranie. W tym jednak przypadku skrypt jest wywoływany przez plik filmowy Flasha. Po otrzymaniu danych, Flash wywoła zdarzenie `messageSent.onLoad`, które zostało oprogramowane w ActionScripcie.

Klasa `LoadVars` oczekuje, że dane będą przekazywane w postaci nazwa-wartość. Każda nazwa zmiennej jest oddzielona od jej wartości znakiem = (po obu stronach znaku równości nie powinno być żadnych spacji). Pary nazwa-wartość są z kolei oddzielone od siebie znakiem ampersand (&), ponownie bez żadnych spacji po obu stronach. Zmienne, które nie zawierają standardowych symboli alfanumerycznych (czyli zawierają znaki diakrytyczne) muszą zostać zakodowane w standardzie URL. PHP posiada bardzo przydatną funkcję nazwaną `urlencode()`, która może to wykonać. Do funkcji przekazuje się albo ciąg znakowy, albo też zmienną go przechowującą. W tym przypadku prościej będzie do zmiennej `reason` przekazać gotowy ciąg znakowy, jednak podczas lektury kolejnych rozdziałów dowiesz się, że lepiej korzystać ze zmiennych. Mimo potrzeby zakodowania wyjścia PHP w standardzie adresu URL, Flash dokona dekodowania automatycznie, toteż w zależności od otrzymanych wyników, plik filmowy albo wyświetli prostą informację o pomyślnym wysłaniu formularza, albo przedstawi wiadomość, że wysyłanie się nie powiodło i poda tego przyczynę. W celu odświeżenia pamięci umieszczam raz jeszcze całą funkcję przypisaną do zdarzenia `messageSent.onLoad`:

```
messageSent.onLoad = function () {
   if (this.sent == "OK") {
      gotoAndStop("acknowledge");
   } else {
      gotoAndStop("failure");
      failure_txt.text = this.reason;
   }
};
```

Zmienną `reason` zadeklarowałem jako dynamiczne pole tekstowe dlatego, że zamierzam powrócić do tej aplikacji w rozdziale 4., gdzie po dodaniu mechanizmu walidacji po stronie serwera komunikaty o błędach będą mogły się zmieniać.

8. Zapisz plik *feedback.php* i sprawdź swój skrypt PHP pod kątem zgodności z oryginalną wersją. Do zmiennej `$to` przypisz adresata wiadomości, natomiast w zmiennej `$additionalHeaders` umieść swój własny adres e-mail.

```
<?php
//inicjalizacja zmiennych w przypadku pól To oraz Subject
$to = 'david@example.com';
$subject = 'Formularz ze strony flash';
//treść wiadomości stwórz na podstawie zmiennych przechowywanych
//w tablicy POST
$message = 'Od: '.$_POST['from']."\n\n";
$message .= 'Email: '.$_POST['email']."\n\n";
$message .= 'adres: '.$_POST['snail']."\n\n";
$message .= 'telefon: '.$_POST['phone']."\n\n";
$message .= 'Komentarze: '.$_POST['comments'];

//dodatkowe nagłówki zawierające informacje zwrotne.
$additionalHeaders = "From: Flash feedback<<feedback@example.com>\n";
$additionalHeaders = "Reply-To: $_POST[email]";
//teraz wyślij wiadomość
$OK = mail($to, $subject, $message, $additionalHeaders);
//niech Flash dowie się o wynikach wysłania
if($OK) {
  echo 'sent=OK';
} else {
  echo 'sent=failed&reason=' . urlencode('Prawdopodobnie nastąpił problem
  ➥z serwerem. Spróbuj wysłać wiadomość później. ');
}
?>
```

9. Teraz nadeszła chwila prawdy. Przetestuj film Flasha albo w wewnętrznej przeglądarce studia MX, albo poprzez opublikowanie pliku SWF i załadowanie strony *biodiverse.html* w oknie przeglądarki (naciśnij w tym celu przycisk *F12*). Wpisz w formularzu dowolne informacje, a następnie kliknij przycisk *Wyślij*. Jeśli wszystko zostało wykonane właściwie, wyświetli się wiadomość o pomyślnie przesłanym formularzu:

10. Nie działa poprawnie? Jeśli wszystko już przetestowałeś, porównaj raz jeszcze swój skrypt z listingiem przedstawionym w 8. kroku lub też z kodem zawartym w materiałach źródłowych tego rozdziału. Sprawdź również poprawność działania serwera na komputerze i czy jesteś podłączony do internetu, a także czy ustawienia poczty są poprawne (użytkownicy Windows powinni zmienić ustawienia SMTP w pliku *php.ini*, tak jak zostało to opisane w rozdziale 1., natomiast na komputerze Mac OS X automatycznie powinny być wykorzystane domyślne ustawienia poczty wychodzącej). Jeśli aplikacja nadal nie działa poprawnie, opublikuj na serwerze swój plik filmowy razem z innymi plikami: *today2.php*, *feedback.php*, *biodiverse.html*, *biodiverse.swf*, a następnie tam dokonaj ich testowania.

> Pamiętaj, że zanim umieścisz swój plik SWF na zdalnym serwerze, musisz zmienić adres pliku PHP we wszystkich miejscach klasy `LoadVars`, gdzie się do niego odwołujesz. Adres `localhost` służy jedynie do celów testowych.

11. Zanim umieścisz plik SWF na zdalnym serwerze, zmień ścieżkę do pliku, tak jak pokazano to poniżej. Po zlokalizowaniu następującej linii w 1. klatce kodu ActionScript warstwy `actions`:

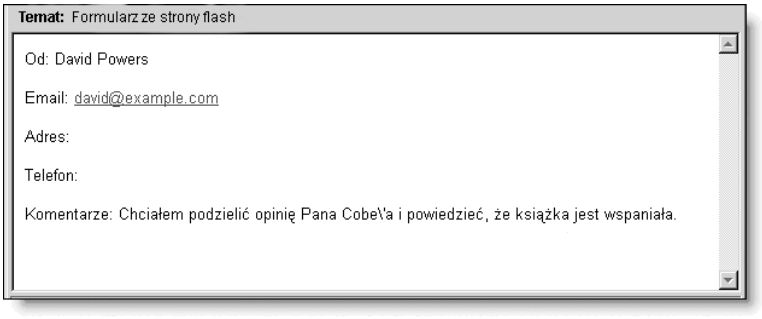
```
message.sendAndLoad("http://localhost/phpflash/ch02/feedback.php?ck="
➥+ new Date().getTime(), messageSent);
```

dokonaj jej zmiany na:

```
message.sendAndLoad("feedback.php?ck="+ new Date().getTime(), messageSent);
```

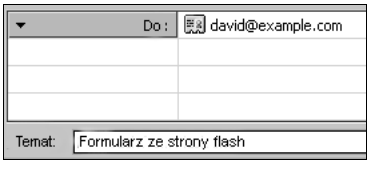
Zakłada się tutaj, że plik SWF oraz kod PHP zostanie umieszczony w tym samym folderze na zdalnym serwerze, toteż jeśli pragniesz umieścić skrypty w innym folderze, dostosuj adres do odpowiedniego katalogu.

12. Po wysłaniu testowej wiadomości sprawdź swoją pocztę. Twoja skrzynka pocztowa powinna teraz zawierać ładnie sformatowaną wiadomość, podobną do tej z rysunku 2.3. Byłoby wspaniale, gdyby nie apostrofy: są one poprzedzone lewymi ukośnikami, mimo że nie zapisano ich w ciągu znakowym kodu PHP. Dzieje się tak ze względu na ustawienie `magic_quotes_gpc`, znajdujące się w pliku *php.ini*. W poprzednim rozdziale zwróciłem uwagę użytkownikom Windows na to ustawienie i zasugerowałem, żeby na wszelki wypadek pozostawili je włączone. Użytkownicy systemu Mac OS, którzy instalowali pakiet PHP od Marca Liyanage'a, nie będą mieli problemów, gdyż ich wersja *php.ini* posiada zmienną `magic_quotes_gpc` ustawioną na `off`. W rzeczywistości niektóre serwery korzystają z tego ustawienia, inne nie. Do zagadnienia powrócę w rozdziale 4., gdzie pokażę, w jaki sposób należy postępować z apostrofami i znakami cudzysłowu w takich przypadkach.



Rysunek 2.3. *Wiadomość e-mail została ładnie sformatowana. Niektóre konfiguracje mechanizmu PHP automatycznie dodają lewe ukośniki przed znakami apostrofu*

13. Kliknij teraz przycisk *Odpowiedź*. Wiadomo już, dlaczego zasugerowałem użycie dodatkowego nagłówka `Reply-To:`. Mimo że ta akurat wiadomość e-mail pochodzi z adresu *feedback@example.com*, odpowiedź zostanie automatycznie przesłana do osoby, która wysłała ten formularz (zakładając oczywiście, że podano prawdziwy adres e-mail).



> Doświadczeni programiści PHP często zaglądają do dokumentacji tego języka. Warto, aby stało się to też Twoim nawykiem. Opisy elementów języka zawarte w dokumentacji są zazwyczaj krótkie i zawierają dużo użytecznych informacji. Więcej na temat funkcji `mail()` znajdziesz więc na stronie *www.php.net/manual/pl/ref.mail.php*.

Czego się dowiedziałeś do tej pory

Uff! Był to dość obszerny rozdział. Do tej pory nie tylko poznałeś podstawowe zasady rządzące PHP, ale także utworzyłeś przydatną aplikację, która połączyła możliwości PHP i Flasha. Ponadto dowiedziałeś się, w jaki sposób używać klasy `LoadVars` do wysyłania i otrzymywania danych oraz jak można sprawdzać, czy właściwe dane zostały przesłane.

Podsumujmy teraz w punktach najważniejsze spostrzeżenia wynikające z lektury rozdziału:

- Wszystkie strony zawierające skrypty PHP powinny posiadać rozszerzenie nazwy *.php* i powinny być obsługiwane przez serwer Apache.
- Skrypty PHP muszą być zawarte pomiędzy znacznikami PHP (najlepiej `<?php` oraz `?>`).
- W przypadku nazw zmiennych PHP wielkość liter jest istotna. Nazwy te muszą się rozpoczynać od znaku dolara, przykładowo, `$myVar`, `$myvar` oraz `$MYvaR` są trzema różnymi zmiennymi.
- Nazwy w PHP muszą się rozpoczynać od litery lub znaku podkreślenia i zawierają jedynie znaki alfanumeryczne oraz podkreślenia.
- Wszystkie instrukcje muszą być zakończone znakiem średnika.
- Wszelkie białe znaki poza ciągiem znakowym są ignorowane.
- PHP korzysta ze znaku kropki (`.`), a nie dodawania (+) do łączenia ciągów znakowych.
- PHP oprócz dwóch identycznych z ActionScriptem metod wstawiania komentarzy (`//` oraz `/* */`) stosuje jeszcze trzeci sposób (#).
- Zmienne, które zostały wysłane przez klasę `LoadVars`, można pobrać poprzez umieszczenie nazwy zmiennej między znakami apostrofu wewnątrz nawiasów kwadratowych metody `$_POST[]` (np. `$_POST['from']`).
- Zmienne pobierane przez klasę `LoadVars` muszą być przedstawione w postaci `nazwa=wartość`. Wszystkie takie pary powinny być oddzielone od siebie znakiem ampersanda (`&`), natomiast wszystkie znaki niealfanumeryczne muszą być najpierw zakodowane wewnątrz funkcji `urlencode()`.

Mimo że zaszliśmy już tak daleko, nadal pozostają pewne rzeczy do zrobienia. W następnym rozdziale dowiesz się czegoś więcej o operatorach arytmetycznych, funkcjach matematycznych oraz instrukcjach warunkowych PHP. Z pewnością ucieszysz się na wieść, że większość z nich będzie działać tak samo jak w ActionScripcie.